Analyzing Spatio-Temporal Behavior of Moving Object Trajectories using PostgreSQL and MobilityDB Operations: A Case Study of Pakistan Railways

By

Asif Nawaz

(2019-NUST-MS-GIS-320903)

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Sciences in Remote Sensing and GIS

Institute of Geographical Information Systems
School of Civil and Environmental Engineering
National University of Sciences & Technology
Islamabad, Pakistan

August 2023

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by **Asif Nawaz (Registration No. MSRSGIS 00000320903), of Session 2019 (Institute of Geographical Information systems)** has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulation, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: [illegible]

Dr. Muhammad Ali Tahir
Associate Professor
IGIS (SCEE)
NUST H-12 Campus
Islamabad.

Name of Supervisor: Dr. Muhammad Ali Tahir
Date: 23-8-23

Signature (HOD): [illegible]
Date: 23/08/2023

Dr. Javed Iqbal
Professor & HOD IGIS. SCEE (NUST)
H-12 Islamabad

Signature (Associate Dean): [illegible]
Date: 23-8-2023

Dr. Ejaz Hussain
Associate Dean IGIS. SCEE (NUST)
H-12, ISLAMABAD

Signature (Principal & Dean SCEE): [illegible]
Date: 24 AUG 2023

PROF DR MUHAMMAD IRFAN
Principal & Dean
SCEE, NUST

ACADEMIC THESIS: DECLARATION OF AUTHORSHIP

I, **Asif Nawaz**, declare that this thesis and the work presented in it are my own and have been generated by me as the result of my own original research.

Analyzing Spatio-Temporal Behavior of Moving Object Trajectories using PostgreSQL and MobilityDB Operations: A Case Study of Pakistan Railways

I confirm that:

1. This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text;
2. Wherever any part of this thesis has previously been submitted for a degree or any other qualification at this or any other institution, it has been clearly stated;
3. I have acknowledged all main sources of help;
4. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
5. None of this work has been published before submission.
6. This work is not plagiarized under the HEC plagiarism policy.

Signed: [illegible]

Date: 23-08-23

DEDICATION

This dissemination is dedicated to my Dearest Mother. By the virtue of whose prays, I have been able to reach at this high position. My Father whose support and believe in me made me stand in front of every obstacle in my life. My wife and sons whose support and courage after my parents made it possible to successfully continue this sacred path of knowledge. To my brothers for their unconditional support. May Allah bless them all with best reward.

ACKNOLEDGEMENTS

All praise to Allah Almighty, the most merciful, most forgiving and loves to forgive. To whom all the knowledge belongs for known and unknown. Who always blessed me with best of the best to achieve this knowledgeable landmark. This achievement was not possible without guidance under light of Prophet Mohammad (SAWA) saying “Seek knowledge from the cradle to the grave”.

Many thanks to my family who always believed in me and prayed for my success especially my father who is supporting me through think and thins. Enormous thanks to my mother, without her prayers, I was never able to complete this journey. A prayer follows every moment your thought crosses my heart. I wish you both be granted highest place in Jannah.

I would like to express my heartfelt appreciation to my supervisor Dr. Ali Tahir for patient guidance, advice and encouragement throughout this wonderful task. He always remained available to answer my queries promptly. I would also like to fortunately acknowledge to Dr. Salman Atif and Dr. M Tariq Saeed for their kind support and guidance which gave me new ideas throughout the course of the study. I would specially thank Dr. Ejaz Ahmed for continuously motivating me to work harder and complete this research in an easy and optimized way.

I am very thankful and pay my gratitude to all IGIS staff who always worked behind and arrange administrative support in time. I would like to acknowledge my fellow students Mr. Raees Ahmad, Mr. Faqir Hussain and Mr. Ala-Ud-Din Awan for creating healthy discussions on the subject.

Asif Nawaz

TABLE OF CONTENTS

CERTIFICATE i

ACADEMIC THESIS: DECLARATION OF AUTHORSHIP ii

DEDICATION iii

ACKNOLEDGEMENTS iv

TABLE OF CONTENTS v

LIST OF FIGURES vii

LIST OF TABLES viii

ABSTRACT ix

CHAPTER1. INTRODUCTION 1

1.1 Background information 2

1.2 Objectives 4

1.3 Scope of study 4

1.4 Pakistan railways 4

1.5 Literature review 6

1.5.1 Areas of application in Pakistan 12

CHAPTER 2. MATERIALS AND METHODS 15

2.1 Study area 15

2.2 Methodology 17

2.3 Data acquisition 18

2.3.1 PuTTY: terminal emulator 23

2.3.2 FileZilla: file transfer protocol 23

2.4 Environment building 25

2.5 Installations and configurations 26

2.6 Identification of missing footprints 28

CHAPTER 3. RESULTS AND DISCUSSIONS 31

3.1 Identification of idle and stopped trains 31

3.2 Trains speed analysis....................33
3.3 Over speeding at station crossings....................37
3.3.1 Query to filter over speeding events....................38
3.3.2 Summary of over speeding at station crossings....................39
3.4 Stop time analysis....................42
3.5 Stops frequency in trips....................44
3.6 Assessing signal visibility to trains: A comparative analysis of PostgreSQL and MobilityDB for efficient traffic management....................45
3.6.1 Utilizing PostgreSQL and PostGIS functions....................47
3.6.2 Utilizing MobilityDB functions....................52
CHAPTER 4. CONCLUSION AND RECOMMENDATIONS....................58
4.1 Conclusion....................58
4.2 Recommendations for further research....................59
REFERENCES....................60
APPENDICES....................66
Appendix-1. Python script to capture raw API responses....................67
Appendix-2. Python script to sort raw footprints....................69

LIST OF FIGURES

Figure 2.1. Study area map. 16

Figure 2.2. Methodological flowchart. 19

Figure 2.3. Trains footprints data collection. 21

Figure 2.4. Raw data structure of each API response. 22

Figure 2.5. Data migration sorting and pre-processing. 22

Figure 2.6. PuTTY configuration view. 24

Figure 2.7. Installation of MobilityDB. 24

Figure 2.8: Gap in coverage from Faisalabad to Toba Tek Singh. 29

Figure 2.9.Gap in coverage from Bahawalpur to Rahimyar Khan. 29

Figure 2.10. Identification of missing footprints. 29

Figure 3.1.Identification of idle and stopped trains. 34

Figure 3.2. Train streaming location in middle of trip. 34

Figure 3.3. Train streaming location on idle situation. 34

Figure 3.4. Speed summary on tracks. 35

Figure 3.5. Summary of speed on tracks. 36

Figure 3.6. Over speeding at station crossing. 40

Figure 3.7. Observed speed ranges. 40

Figure 3.8. Over speeding map at station crossings. 41

Figure 3.9. Trip stop times. 43

Figure 3.10. Top 5 trips with minimum stops. 43

Figure 3.11. Top 5 trips with maximum stops. 43

Figure 3.12. Map visualization of trains GPS point and signal. 50

Figure 3.13. Map visualization of PostgreSQL query. 50

Figure 3.14. Map visualization of MobilityDB query. 56

LIST OF TABLES

Table 3.1. Summary of speed on track. 35

Table 3.2. GCOR permissible trains speed at station crossings. 40

ABSTRACT

Recent advances in location tracking technologies have led to availability of the extensive spatio-temporal datasets. Extracting meaningful information from these datasets is a crucial for informed decision-making across the different industries. Objectives of the study encompass capturing the real-time trains trajectories, conducting a comprehensive analysis, and lastly visualizing results for the improved railway's monitoring efficiency. The study also focuses on leveraging the PostgreSQL and its extensions to manage the moving objects data efficiently, particularly in context of Pakistan Railways. In order to achieve these objectives, study collected the comprehensive inventory of trains, stations, and schedules data from Pakistan Railways. Also, dynamic train footprints were obtained using the custom developed Python scripts and web APIs. Demanding preprocessing ensured data accuracy and data consistency. The heart of study lies in leveraging the PostgreSQL's capabilities for executing complex spatio-temporal analyses that encompassed detailed speed assessment, stop time analysis, and evaluation of the signal visibility. The findings of the study highlighted hidden patterns and the insights within train trajectories. Stop time analysis highlighted the fact that about 12% of the trains time is being consumed during station stops. Speed analysis identified that about 40 km/h is the average speed of the trains. Signal visibility analysis performed and resulted that MobilityDB queries are computationally efficient and easy to understand compared to PostgreSQL queries. Study's outcomes are not only beneficial for the railway sector but also emphasize the broader potential of the data-driven decision-making in optimizing the operational processes across various

Chapter 1

INTRODUCTION

Over the past few decades, the need for easy and flexible management and visualization of temporal geographic data has become increasingly evident. As a result, numerous efforts have been made to develop computer systems capable of handling this type of data. GIS experts have proposed several spatio-temporal models based on three key components: space, time, and attributes. These models allow for the representation of state changes in space over time. One such model, known as the Temporal Map Sets (TMS) model, is an extension of the snapshot model and is designed to manage geographic data in a space-time environment. Many ideas and methods to use temporal data alongside relational information have been developed by GIS and computer scientists.

MobilityDB is advanced platform for managing, analysing, and visualising temporal datasets. With funding from the European Commission, Fonds de la Recherche Scientifique (FNRS), Belgium, and Innoviris, Belgium, the Computer & Decision Engineering Department of the Université Libre de Bruxelles (ULB) developed the MobilityDB platform, an extension of PostGIS and PostgreSQL for handling spatio-temporal data. Project was created by a group of programmers under the direction of the Project Steering Committee (PSC) members. In order to visualize mobility data, MobilityDB integrates PostgreSQL, PostGIS (a database add-on), and QGIS which is an open-source geographic information system (Zimányi, 2020). Additionally, Move is an open-source QGIS plugin for temporal mobility visualisation.

This study tries to explain the procedures that MobilityDB running after successful installations, use, and upgrade, then presents test scenarios with suggested solutions. In this regard, a case study involving the mobility datasets of Pakistan Railways will be examined in order to understand the effectiveness and usage of MobilityDB and PostgreSQL in Pakistan as well as its capacity to aid in the understanding of complex data in order to improve the monitoring effectiveness of Pakistan Railways and associated institutions. In this study, all workflow steps will be covered, including installation and configuration, data loading, preprocessing, explorations, spatio-temporal analysis, and trajectory design.

1.1 Background information

Pakistan Railways launched online train tracking system named as Pak Rail Live in year 2019. The development of OS apps, Android apps, product design workshops, back-end development, web app development, and UI/UX design were all handled by EASYWAYS[1]. Pak Rail Live is a free application that allows Pakistani citizens to track trains in real time via Pakistan Railways. Additionally, it offers pre-arrival notifications, schedule updates, and estimates for the arrival of the following stations. People who have access to Pak Rail Live can follow trains in real time. Any train's pre-arrival and expected arrival time can be announced to the public. Additionally, passengers can check for any railway delays or updates/news. By using GPS, Pakistan Railways' trains create a significant amount of multidimensional data every day. With this initiative from Pakistan Railways department, valuable analysis may be done on live stream data that is being streaming live publically using APIs into web and android

[1] http://easywayinnovations.com/bjsttl

applications to uncover information that could improve train surveillance effectiveness in addition to visualising real-time positions.

To analyse the data fast and effectively, this enormous data required a big data platform. Large spatial datasets can be easily accommodated by conventional database management systems like My SQL, Oracle, etc., but these systems still fall short when it comes to the temporal concept. The only database system available today that not only offers effective and optimised data storage but also assists in managing massive moving datasets with its advanced and constantly expanding temporal mobility functionalities and views is PostgreSQL and it extensions like MobilityDB which is a moving objects database management system. A developing platform called MobilityDB offers maximum database adapter support combined with support for the PostgreSQL platform. It offers datatypes to create trajectories from location and time and offer management of moving trajectories,

Thousands of studies on moving object databases were conducted during the previous ten years, but only a small fraction of them were highlighted by their prototypes, and even fewer of these prototypes were put to use in a commercial setting (Güting et al., 2010). The main prototype that uses abstract data types is MobilityDB, an extension to PostgreSQL and PostGIS. With its strong data extraction capabilities, MobilityDB develops indexes and aggregates, hence decreasing the storage capacity (Zimányi et al., 2020). The need for transferring object data and its application in the management of spatio-temporal databases is growing daily. Real-time data generation from mobility items necessitates an effective management system. Performance problems exist for PostgreSQL and PostGIS-configured applications with simple architecture.

1.2 Objectives

The study has following objectives:

a) To develop a mechanism for capturing and storing live streaming train trajectories into a database

b) To analyze trains footprints data to improve monitoring efficiency of trains using;

   - Stop time analysis
   - Speed analysis
   - Signal visibility analysis

1.3 Scope of study

The research presented here provides evidence that the analysis of GPS data can produce findings that are helpful in many different decision-making contexts. It displays the use of GPS data generated by roughly 44 active passenger trains each day during a period of 1.5 months, with each train broadcasting its location every 10 seconds. This data can be used to categories train behavior, track conditions, and track operations. Utilizing this information would enable Pakistan rails to more effectively monitor concerns relating to security, safety, and anticipation on the rails. Study describes the methods adopted for collecting and processing the raw footprints from each active train on track, till analysis. The footprints data of trains trajectory is streaming free for the general public of Pakistan.

1.4 Pakistan railways

Ministry of Railways is in a quest to upgrade their system based on new techniques and technologies. As they now have online bookings and purchasing system. Even they now have online land inventory management system and their lease managements are now monitoring by online portals. (Li et al., 2018) examines the current state of Pakistan Railways (PR) and its future prospects based on the industry life cycle theory. The results show that PR has been in the maturity stage of the industry life cycle since the 1990s and is facing various challenges including a lack of investment, outdated technology, and poor management. The study suggests that PR needs to adopt new technologies, improve its management, and increase investment to remain competitive and move towards the growth stage of the industry life cycle. The article provides valuable insights for policymakers, stakeholders, and researchers interested in the development of the railway industry in Pakistan. Pakistan Railways is investing and performing well currently and as per the Pakistan Railways yearbook 2020-2021, all business-critical services and applications, including e-ticketing, have been successfully moved to the Government Cloud Data Center (NTC), Islamabad, in a secure manner in accordance with industry standards.

Moreover, the central control center at Headquarters Lahore will have a video wall constructed as part of the ongoing railway tracking operation. Through a GIS system, this system would track the location of trains in real time. Wherever the transportation business is automating, this most recent technology is used. The goal of Pakistan Railways is to create a centralized control room for tracking all of the country's trains. All locomotives will have tracking devices installed, and a video wall will be put in the central control room so that everyone can see how each train is performing in terms of numerous criteria in real time (Ministry of Railways, 2021). Based on these initiative from Pakistan Railways it will be a good approach for respective authorities to adopt the top notch technology of spatio-temporal database management system that PostgreSQL offers do better deal mobility data also and get more depth insights of moving trains.

1.5 Literature review

A revolutionary solution to train monitoring and communication in the Egyptian National Railways was suggested by (Mohamed, 2014). The system seeks to deliver precise train location updates and effective communication between control centers and railway operators by integrating GSM and GPS technology. The system provides for continuous tracking of train positions as well as immediate transmission to a central web server, hence improving safety and operational efficiency. The proposed approach is applicable to both primary and secondary railway lines, considering the availability of infrastructure. The study gives a thorough examination of the system's advantages, illustrating how the extent of GPS technology penetration has a substantial impact on operational outcomes. As technology advances, train wait times decrease, resulting in a more efficient and user-friendly experience for passengers. The research paper proposes a paradigm shift in train monitoring and communication, emphasizing the potential of combining GSM and GPS technology to transform railway operations.

Imran (2009) described a public transportation in Pakistan as a critical overview. His paper also provides a comprehensive evaluation of the country's public transportation system. The report, which was published in the Journal of Public Transportation, evaluates the nation's public transportation system severely. It explores a number of topics, most likely including infrastructure, services, and difficulties. Imran describes the system's advantages and disadvantages through a thorough investigation, which contributes to a better comprehension of Pakistan's transportation environment.

In accordance with the notion of the industry life cycle, Pakistan Railways has gone through many stages and shown distinct patterns (Stripple et al., 2010, Irfan et al., 2012, Transport, 2015). Its founding in 1861 signaled the beginning of the pioneering age. Connectivity and economic development were bolstered by subsequent growth and expansion during the boom era (Karniouchina et al., 2013). Nevertheless, inefficiencies started to show up, which forced it into the maturity stage where problems like poor management and competition from the road transport sector occurred. Pakistan Railways needs to enter the revitalization phase by embracing innovation, modernizing its operations, and putting a strong emphasis on customer demands (Utterback et al., 1993). Pakistan Railways will be able to reenergize, adapt, and ensure a sustainable future by matching tactics with each phase's requirements.

In the context of Egyptian National Railways, this paper provides a proposed Radio Frequency (RF) system combined with GPS for tracking trains (Nahid et al., 2013). The model intends to improve the effectiveness, safety, and real-time monitoring of train operations while taking into account the unique requirements of the Egyptian railway network (Mohamed, 2014).

In the field of railway administration, the use of radio frequency (RF) equipment for train tracking has become a cutting-edge and effective technique. This strategy entails fusing RF technology with current systems to track and control train movements, improving operational performance across the board (Furman et al., 2001, Hofestadt, 1995, Ikeda. 1993).

The fundamental idea behind RF-based train tracking is to install a transponder-like RF transmitter and receiver system on each train. These gadgets talk to fixed RF base stations put in place all throughout the rail network. Real-time position updates are made possible by the RF transponder's periodic signal exchanges with the base stations as a train travels along its path. This technology has advantages such as accurate estimates of train arrival and departure times and exact train positioning and speed calculation. The capacity to quickly identify probable crashes or unauthorized entry onto rails enhances safety. Additionally, maintenance schedules can be made more efficient by looking at the train movement patterns and the wear-and-tear of the information (Bates, 1994). Implementing an RF-based train tracking system could aid in resolving operational issues with the Egyptian National Railways, such as delays and communication breakdowns. The railway network can improve its dependability, punctuality, and passenger experience by implementing this technology. However, infrastructural investment, technology integration, and the expert maintenance are necessary for the ultimately successful adoption (Mahalakshmi et al, 2013, Lancian, 1990).

Helland-Hansen and Hampson (2009) offers a thorough examination of trajectory analysis ideas and their real-world applications. The authors examine the theoretical underpinnings and methods of trajectory analysis, providing insights into its importance in a number of different domains. The article probably discusses methods for comprehending how processes and phenomena change through time, helping to enhance how geological, environmental, and spatial data are interpreted. The paper advances knowledge of dynamic processes in complex systems by highlighting the value of trajectory analysis.

Predictive analysis using multifaceted trajectories of fishing vessels in the Northern Adriatic Sea is the main topic of (Brandoli et al., 2022). Research explains multiple aspect trajectories to the predictive analysis which is a case study on fishing the vessels in northern Adriatic Sea. The study investigates the conversion of intricate trajectory data into prognostic information. This case study serves as an example of how trajectory analysis can be used in the real world to manage fisheries and conduct maritime operations in the area. The work contributes to better decision-making in marine situations by enhancing comprehension of trajectory-based predictive analysis techniques.

A ground-breaking research study conducted by (Das et al., 2009) that focuses on railway monitoring and its potential to revolutionize the process. The research describes a novel solution that combines satellite communication and global positioning system (GPS) technologies to collect accurate location data on trains and send it to a centralized server (Vershinin and Mustafina, 2021). This gives operators and authorities a comprehensive, real-time view of train positions, movements, and trajectories, allowing them to make informed decisions. The architectural architecture of the system, technological details, and potential hurdles to satellite communication reliability, data correctness, and smooth integration with pre-existing the railway frameworks are all discussed (Chacko, Basheer, & Kumar, 2015). The study presents a ground-breaking method that improves train tracking precision and efficiency, resulting in increased safety, fewer operational disturbances, and enhanced railway performance.

Graser and Dragaschnig (2020) in their research of Exploring the movement data in the notebook environments incorporated movement data into interactive digital notebooks. Their research, which was presented at the IEEE

VIS 2020 Workshop, highlights the value of geospatial data visualization in notebook scenarios and furthers the conversation about information visualization.

Another describes an innovative solution for real-time train tracking, railway crossing management, and emergency path establishment using D2D communication technology. The system allows for direct and seamless connection between trains and railway infrastructure, allowing for accurate tracking of train movements and management of railway crossings (Cubukcu et al., 2021). The system also includes emergency path establishment, which ensures safe travel in the event of a contingency or malfunction. The seamless integration of sophisticated technologies into the railway domain highlights D2D communication's ability to reduce operational issues and improve safety measures. The incorporation of this technology into railway infrastructure holds significant promise for revolutionizing train system efficiency, safety, and overall performance, solving critical challenges in current transportation management (Kumar and Ramesh, 2017).

A study investigates a significant area of concern in the field of database management: dealing with spatio-temporal pattern queries in the setting of moving objects. Databases must support the delicate interaction of spatial and temporal components in an increasingly linked world where things and entities are continuously on the move (Guedes et al., 2018). The writers recognize this difficulty and provide ideas on how to deal with its intricacies. They investigate novel ways for streamlining the querying process in circumstances with both spatial and temporal aspects. While the specifics of the techniques, experimental sets, and findings are not given in this brief description, the study advances database management in dynamic situations. By tackling the complexities of spatio-temporal pattern searches for moving objects, it provides a potential avenue for improving query processing efficiency and efficacy, therefore contributing to the broader field of data and software engineering (Yogiandra and Esterina Widagdo, 2019).

A study conducted by (D'Orso and Migliore, 2017) provide a GIS-based methodology for assessing the prospective demand for an integrated transport system in their research paper. The authors address the difficulty of measuring demand for such systems, which mix diverse forms of transportation, in an efficient manner. Geographic Information Systems (GIS) are used in the approach to analyze and visualize geographical data connected to transportation trends and population distribution (Sharma, Sharma, & Bundele, 2018). The suggested approach seeks to offer reliable projections of demand for the integrated transport system by integrating data on transport networks and demography. The study adds to transport planning by providing a methodical and technology-driven strategy for anticipating demand for complicated transport networks, assisting in optimal infrastructure construction and urban planning (Borodin et al., 2017).

The above discussed literature underscores with the transformative potential of the innovative solutions and technologies in enhancing railways operations and safety measures. Mohamed (2014) proposed a solution integrating GSM and GPS solutions for train monitoring that aligns with current study that focuses on spatio-temporal analysis by using PostgreSQL and the MobilityDB solutions. Das et al. (2009) proposed a groundbreaking study by combining satellite communication with GPS for real-time train tracking that resonates with this thesis emphasis on the accurate measurement of signal visibility. Similarly, exploration of D2D communication and its impact on trains tracking and safety measures reflect the integration of MobilityDB for the improved train monitoring operations (Kumar and Ramesh, 2017). Yogiandra et. al. (2019) performed investigation into spatio-temporal patterns queries that aligns with thesis aim to enhance the efficiency of query processing. Lastly, GIS-based methodology for transportation demand assessment aligns the thesis focus on the data-driven analysis for optimizing railway operations. These discussed studies collectively reinforce significance of the technological advancement in revolutionizing the railways safety, trains monitoring and its efficiency by echoing key themes that been explored in the thesis (Sharygin et al., 2017).

1.5.1 Areas of application in Pakistan

Moving objects like humans, vehicles, trains, ships and aircraft can be tracked, monitored and anticipated and these are one of the main applications of spatio-temporal databases. Live footprints datasets from moving objects like human, animals, trains, vehicles planes can be a useful resource to analyse the patterns in moving objects. There is a great potential in Pakistan in order to utilize the operations of moving object database management systems and utilize mobility data alongside GIS.

Analysis of speed and transportation infrastructure can not only produce helpful velocity maps but also influence Pakistani government officials to modernize the country's current rail and road networks. Travel time can be precisely calculated using route and trajectory analysis. Trip time forecasting can aid in the creation of more creative software tools that can aid in early planning and booking. Although the majority of large cities, including Karachi, Lahore, Faisalabad, Rawalpindi, and Multan, have experienced rapid population growth, their road infrastructure has not improved over time. The current traffic congestion problems in Pakistan's major cities can be helped by the use of datasets such as vehicle speed, school and office closing times, road conjoining, traffic lights, weather conditions, road infrastructure condition, and lane count along with mobility data (Syed, 2014). It is possible to measure traffic at various time periods, such as hourly, daily, or weekly. Using the tgeogpoint (inst) function, MobilityDB can quickly extract data on traffic volume and count for a specific time and location. Data from trips can be used to extract multi-dimensional information, such as mobility optimization, multimodal transportation, and commuter scenarios (Rovinelli, 2021).

When evaluating population or density, MobilityDB can be used to better understand pedestrian mobility patterns. With the aid of heat maps based on automobile, bicycle, and pedestrian crash incidences in densely populated cities like Karachi, Lahore, Faisalabad, Peshawar, and Multan, MobilityDB can assist reduce the crash ratio. In addition to saving lives, this will lighten the load on hospitals and emergency rooms like Rescue 1122. On busy roadways, MobilityDB can forecast collision hazards.

Based on demographic, socioeconomic, and population data, MobilityDB can anticipate resource equality. Velocity maps created with mobilityDB can also be used to evaluate equity. Based on the mobility loads, amenities, and services provided by various highways, MobilityDB can analyze and categorize them. Careem, Uber, Bykea, and other public transportation services in Pakistan can also be distinguished by their various socioeconomic effects on society. In Pakistan, there is currently no sophisticated system in place to upgrade the road system. The use of mobility data can help locate areas in need of further infrastructure restoration. Additionally, with the availability of mobility Data, places and points of interest like schools, hospitals, and grocery stores may be efficiently handled.

Mobility patterns can be a crucial factor in predicting criminal behavior, as an individual's movements can provide valuable insights into their activities and intentions. Therefore, it is no surprise that law enforcement agencies around the world have increasingly turned to mobility data analysis to prevent crime and maintain public safety. In Pakistan, the use of spatio Trajectory Analysis can be a promising tool in law and order management, especially given the country's growing crime rates. According to a report by the Pakistan Bureau of Statistics, there were over 1.1 million cases of crime reported in the country, with a sharp increase in the violent crime activities such as murder, robbery, and kidnapping (Statistics, 2017). With such alarming statistics, it is becoming more critical than ever for law enforcement to adopt innovative approaches to tackle crime.

Chapter 2

MATERIALS AND METHODS

2.1 Study area

This study focuses on the management of large datasets of live stream footprints of Pakistan Railways along with the trains stations details, railway network details and train schedule. Passenger trains are being consider for this research. Figure 2.1 shows the study area map along with railway network lines, stations location, and sample trains footprints. Overall study area is the whole railway network of Pakistan but live stream data from active trains does not cover most of the regions of Baluchistan and Khyber-Pakhtunkhwa.

The study area maps show around 296 mains stations out of 458 railway station excluding halts. Tracks of 11,881 KM of length are mapped as main line and other lines (Ministry of Railways, 2021).

The study area incorporates a diverse railway network that comprise railways stations, networks, and railways tracks. Railways stations serves as the key nodal point for the passenger and cargo transits. The railway network is categorized into three main distinct groups that is 1) main lines which represents core routes that connect major cities. 2) New lines that signifies recently developed railway tracks to accommodate the growing demands. 3) and existing lines that comprises the established railway tracks by connecting various region. The footprints on the tracks captures the spatial-temporal trajectories data of trains movements. This study investigates the spatio-temporal variables of these categories by leveraging advance technologies to analyze the train behaviors, and operational efficiency by contributing to into comprehensive understandings of the railway operations.

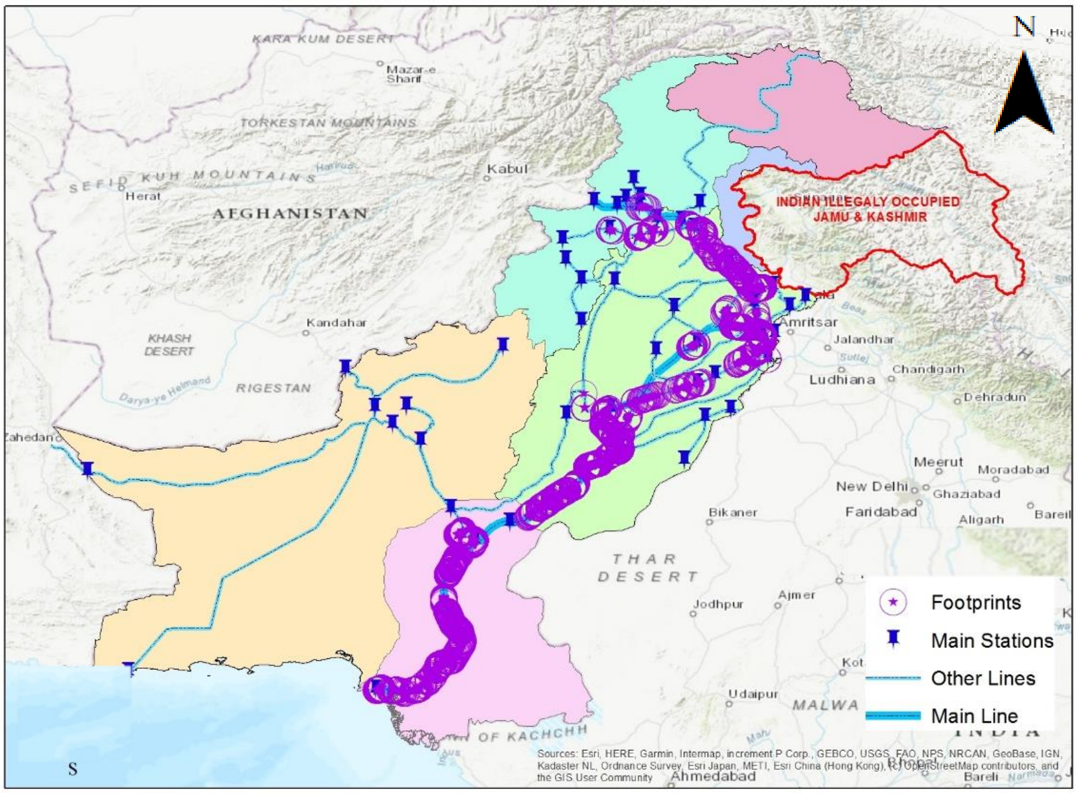


Figure 2.1. Study area map.

2.2 Methodology

The overall study is focused on analyzing spatio-temporal behavior of moving object trajectories by utilizing PostgreSQL and MobilityDB operations. In this regard study has been segregated into three main phases and the first phase was acquisition of Pakistan Railways datasets along with its acquisition and preprocessing and storing into a database for further analysis. The second phase was installations and configurations of the suitable version of MobilityDB to be used in this study. The third phase includes loading of Pakistan Railways datasets into MobilityDB environment, its upgradations, spatio-temporal analysis and extractions of useful information with the utilization of MobilityDB queries that makes it possible in more optimized and the simpler way. The phase wise segregation involves following steps;

1. Data acquisition and preprocessing
2. Installation and environment configurations
3. Spatio-temporal analysis and recommendations

Data was acquired by Pakistan Railways website and Pak Rail Live web APIs in raw JSON and HTML formats and then processed to reshape into readable formats. The process of data acquisition and preprocessing is further explained in data acquisition and pre-processing section.

Due to the limited help resources and lack of availability on the Windows platform, configuring MobilityDB is a difficult operation. In this case, MobilityDB was configured over Windows using the Docker (Linux environment simulation program for Windows) application. The initial and most difficult steps in the study include data input into MobilityDB, upgrading, and evaluation. Prior to visualizing temporal trajectories in QGIS, the MobilityDB data needed to be upgraded, which included creating points, temporal points, and trajectories. Generation of the spatio-temporal queries for analytical information extraction like stoppage of certain train at certain point in time, identification of train patterns at station crossings, comparison of scheduled and actual trip progress throughout the trip and anytime, maximum concurrent trips of trains, total travelled distances of trains, average duration and speed of each trip, shorted vs longest trips, histograms of trip lengths, identification of minimum distances between pair of trains on same tracks etc. Proposed methodology of the study is shown in Figure 2.2.

2.3 Data acquisition

Pakistan Railways datasets were obtained by utilizing the public information provided over Pakistan Railways website and the Pak Rail Live web application. List of datasets with its formats and sources are listed in Table 2.1.

In order to capture the trains list, stations list and trains schedule data, Pakistan Railways website was used to collect the data in JSON and hardcoded HTML pages. These data sets were then filtered sorted and stored in PostgreSQL database. To capture the raw footprints data of each active passenger train, AWS instance was developed and train footprints was acquired. Figure 2.3 shows the methodological flow chart of how footprints data was collected by using the Elastic Compute Cloud (EC2) instance of AWS system. Ubuntu 14.04 LTS version was installed over the instance and PostgreSQL version is 12.3 was installed so that captured data from API response should be captured and dumped into PostgreSQL Table. A Python script was developed as that enabled the data fetching from Pak Rail Live APIs. Python script to capture raw API responses is attached in Appendix – 1.

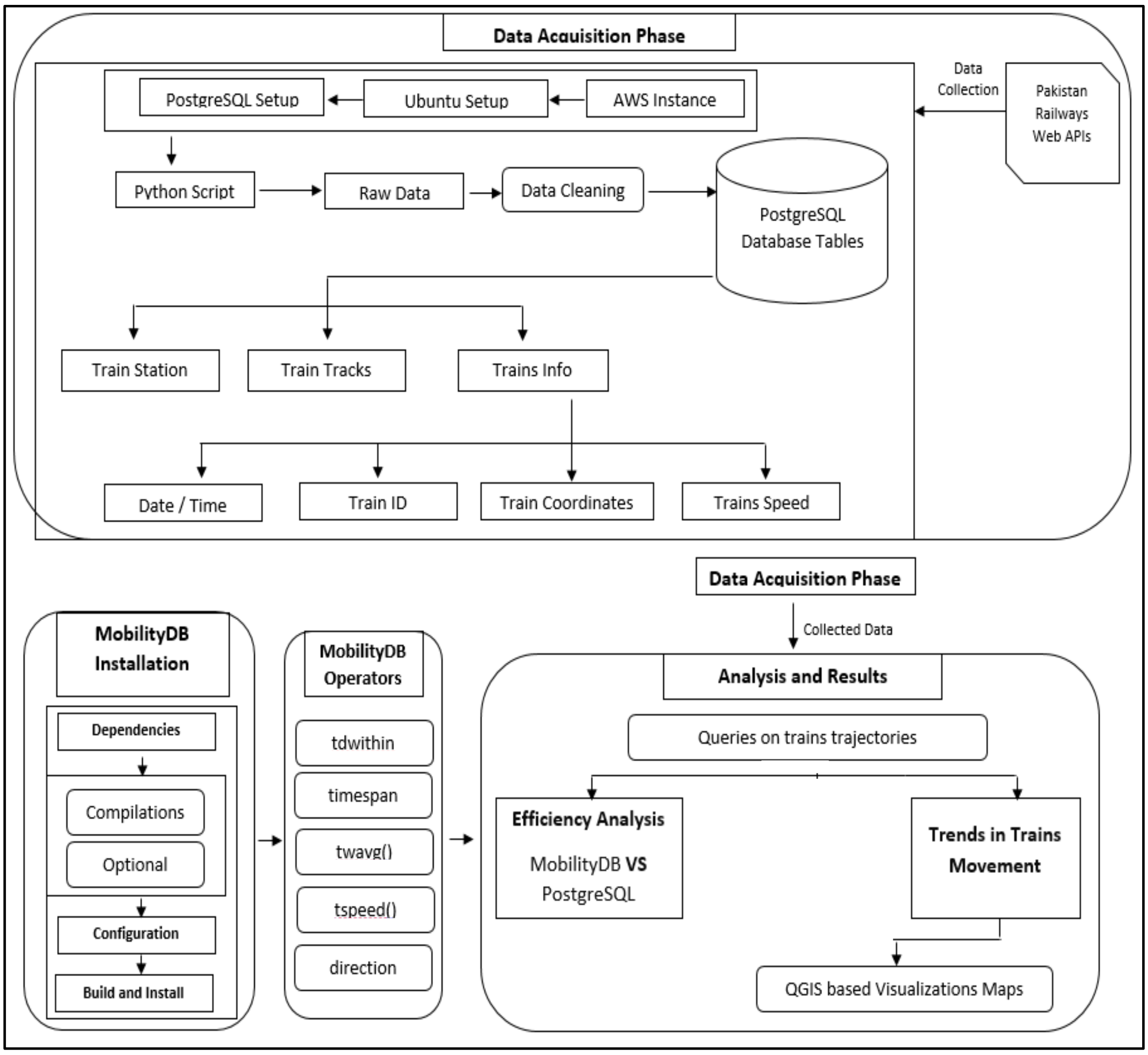


Figure 2.2. Methodological flowchart.

The duration of acquired footprints is about 1.5 month of the duration starting from 27-06-2022 till 15-08-2022. Raw footprints data captured from about 44 active passenger trains but this number slightly varies depending upon the nature of train to be active and is in middle of its trip or there could be chance of train to be inactive and have to trip. For each active train on track, about 298,656 API responses was captured with every 10 second interval where the APIs are streaming location after every 5 second interval. In order to capture and store record without delay in internet coverage and to reduce the density of data, 10 second interval was opted which still gives ample of information about trains activity and trips trajectories can be easily developed by using this interval.

Now each API is set in such a way that possess the location of all active trains in its single response, so with each API response a complex nested JSON is received at the end with relevant information from all the trains in it. Figure 2.4 shows the data structure of raw response from a single API response out the about 300,000 collected responses. Now, from about 44 passenger trains, the 300,000 API responses generated a total of about 13 Million footprints record that been captured from all active trains.

Data migration of collected data was initiated that includes backup of database, transferring backup file to local storage and then restoring backup into local PostgreSQL system as shown in Figure 2.5. Footprints table was then sorted by converting nested JSON strings into a PostgreSQL table where records of each footprint contained data columns as are mentioned in Figure 2.5. Whereas, Appendix – 2 contained the Python script was used to sort data into readable format.

Table 2.1. Datasets.

Sr#	**Dataset**	**Description**	**File Type**	**Source**
1	Railway Tracks	Railway Tracks Network of Pakistan	JSON	Pak Rail Live APIs / OSM
2	Train Stations	Coordinates of train stations with station names.	JSON/HTML	Pakistan Railways Website / OSM
3	Trains Schedule	Train ID	JSON/HTML	Pakistan Railways Website
		Stops		
		Arrival Time		
4	Trains Trajectories	Train ID	APIs	Pak Rail Live APIs
		Train Coordinates		
		Date/Time		

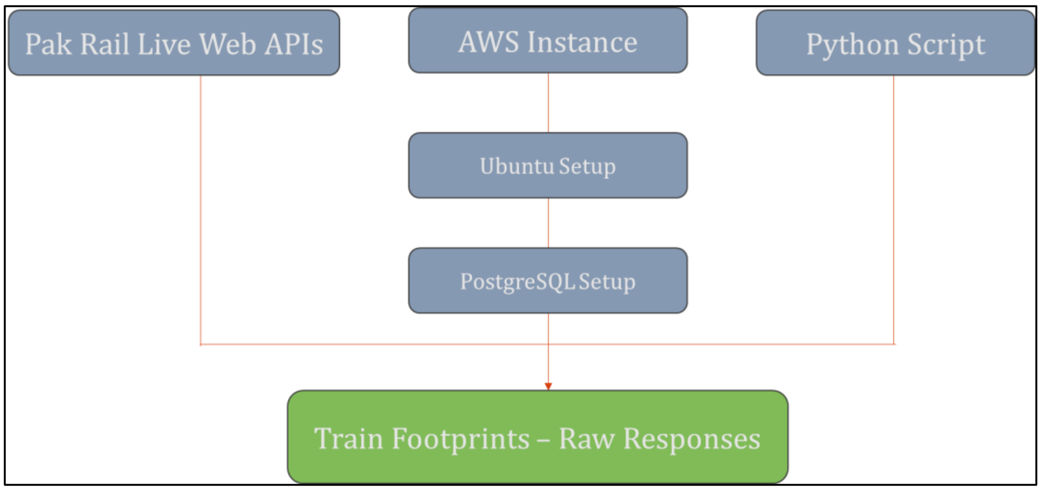


Figure 2.3. Trains footprints data collection.

```
156:42["102906",{"lat":"28.3233016666667","lon":"70.1578616666667","last_updated":
"1656523101","late_by":"47","next_st":"38","prev_st":"19","sp":"68","st":"A"}]

156:42["342906",{"lat":"29.5356783333333","lon":"71.6242783333333","last_updated":
"1656523101","late_by":"21","next_st":"17","prev_st":"16","sp":"69","st":"A"}]

141:42["172906",{"lat":"26.572025","lon":"68.30283","last_updated":"1656523101",
"late_by":"46","next_st":"74","prev_st":"23","sp":"52","st":"A"}]

149:42["252906",{"lat":"25.975265","lon":"68.6013483333333","last_updated":
"1656523101","late_by":"11","next_st":"23","prev_st":"24","sp":"82","st":"A"}]

156:42["432906",{"lat":"25.3848716666667","lon":"68.3832416666667","last_updated":
"1656523101","late_by":"32","next_st":"20","prev_st":"26","sp":"19","st":"A"}]

153:42["4022906",{"lat":"25.0135683333333","lon":"68.01244","last_updated":
"1656523102","late_by":"1163","next_st":"73","prev_st":"1227","sp":"23","st":"A"}]
```

Figure 2.4. Raw data structure of each API response.

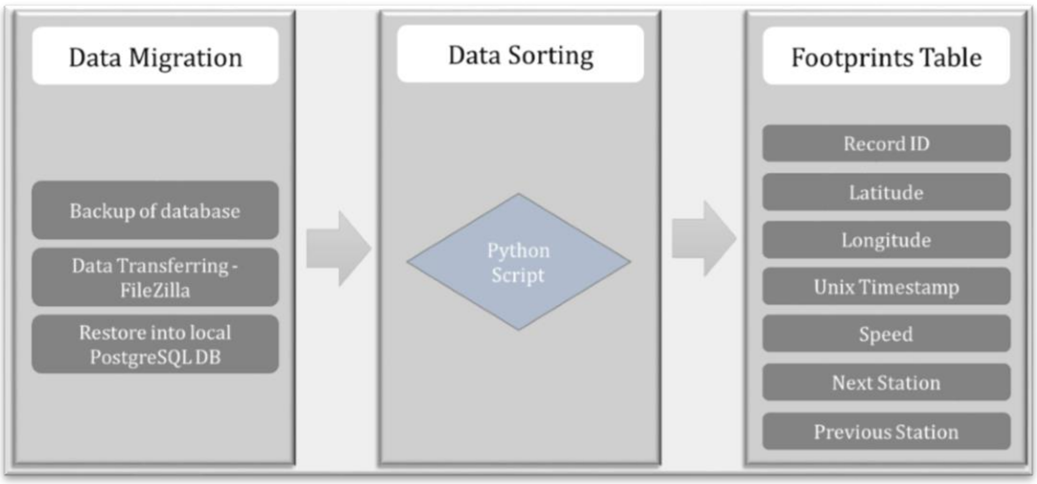


Figure 2.5. Data migration sorting and pre-processing.

2.3.1 PuTTY: terminal emulator

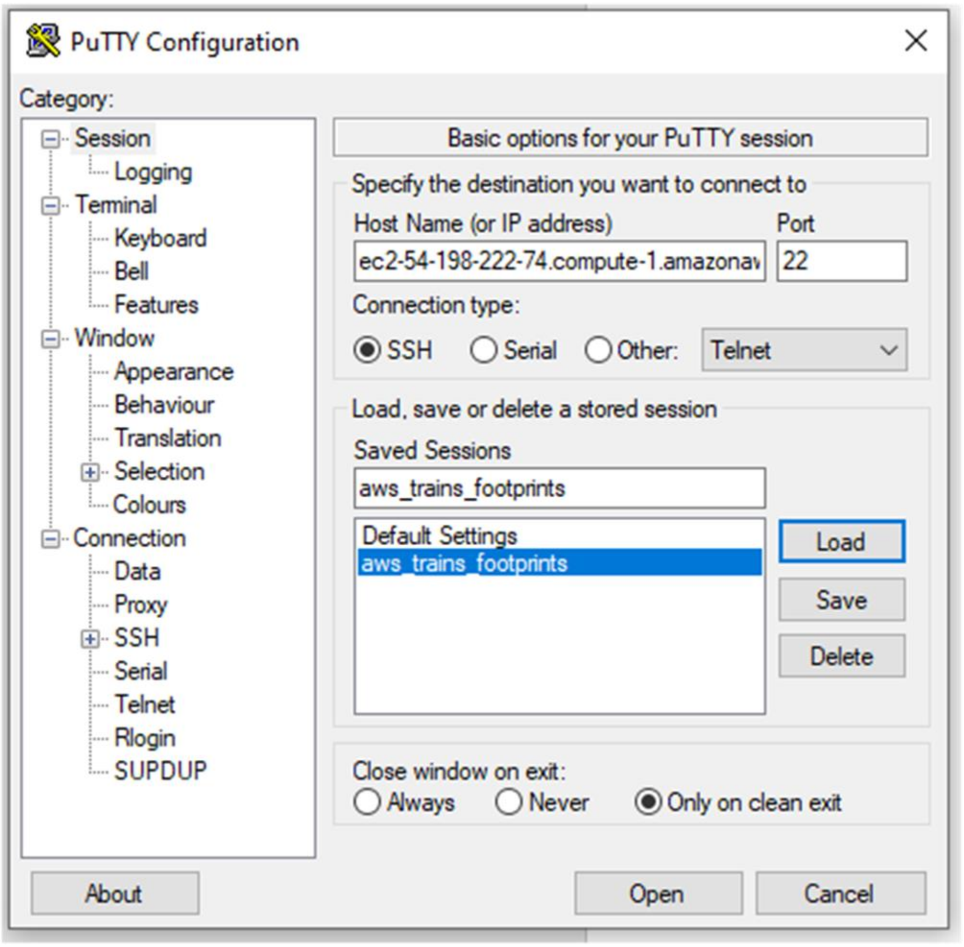
PuTTY was used to communicate with the AWS instance where installations, configurations and Python script development and implementations was performed through terminal based window. As PuTTY provides faceless view of the AWS Instance and Ubuntu. Once Python script was finalized and tested on local system, on cloud implementation of script was performed where it was running 24/7 to capture the active trains footprints data. Figure 2.6 shows the configuration view to access the AWS instance.

PuTTY is a free and open-source terminal emulator, serial console, and network file transfer application. It supports various network protocols, including SSH, Telnet, rlogin, and SCP, as well as raw socket connection. PuTTY is commonly used on Windows operating systems to establish a secure remote connection to a server or other computer over the internet. It is also popular for its lightweight and customizable interface, as well as its ability to save session settings for easy access to frequently-used connections.

2.3.2 FileZilla: file transfer protocol

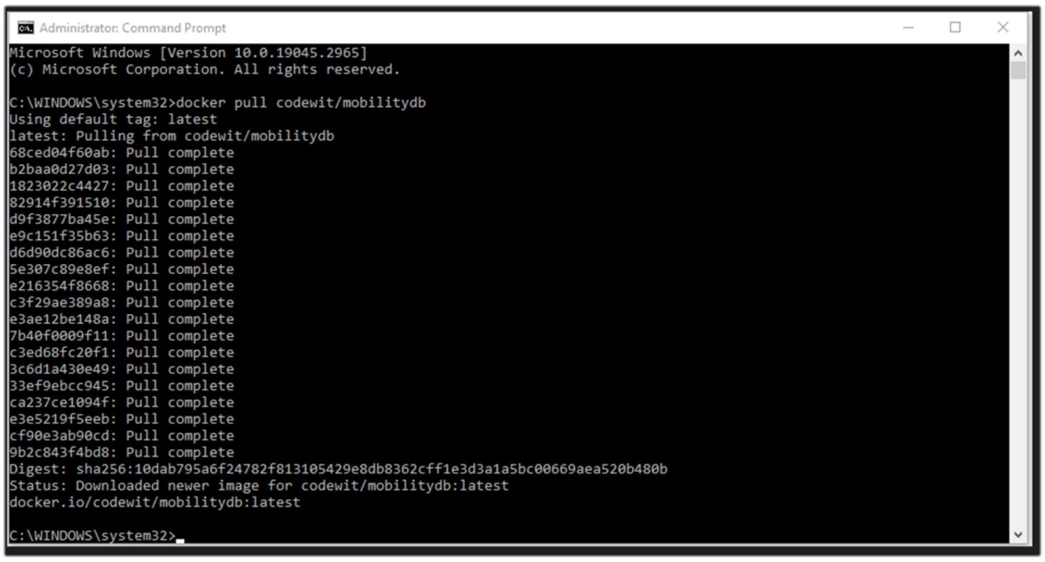
Raw captured API responses were backed up from AWS based PostgreSQL in .sql format and was transferred to local storage by using FileZilla.

FileZilla is a free, open-source, cross-platform FTP (File Transfer Protocol), FTPS, and SFTP client. It is widely used by website administrators and developers to upload, download, and manage files on remote servers. FileZilla provides a user-friendly interface for transferring files between the local machine and a remote server, allowing users to manage files and folders seamlessly.



Specify the destination you want to connect to
Host Name (or IP address): ec2-54-198-222-74.compute-1.amazonav
Port: 22
Connection type: [✓ SSH] Serial Other: Telnet
Load, save or delete a stored session
Saved Sessions: aws_trains_footprints
Default Settings
aws_trains_footprints
Close window on exit: Always Never [✓ Only on clean exit]

Figure 2.6. PuTTY configuration view.



```
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>docker pull codewit/mobilitydb
Using default tag: latest
latest: Pulling from codewit/mobilitydb
68ced04f60ab: Pull complete
b2baa0d27d03: Pull complete
1823022c4427: Pull complete
82914f391510: Pull complete
d9f3877ba45e: Pull complete
e9c151f35b63: Pull complete
d6d90dc86ac6: Pull complete
5e307c89e8ef: Pull complete
e216354f8668: Pull complete
c3f29ae389a8: Pull complete
e3ae12be148a: Pull complete
7b40f0009f11: Pull complete
c3ed68fc20f1: Pull complete
3c6d1a430e49: Pull complete
33ef9ebcc945: Pull complete
ca237ce1094f: Pull complete
e3e5219f5eeb: Pull complete
cf90e3ab90cd: Pull complete
9b2c843f4bd8: Pull complete
Digest: sha256:10dab795a6f24782f813105429e8db8362cff1e3d3a1a5bc00669aea520b480b
Status: Downloaded newer image for codewit/mobilitydb:latest
docker.io/codewit/mobilitydb:latest

C:\WINDOWS\system32>
```

Figure 2.7. Installation of MobilityDB.

It supports various transfer modes, including ASCII, binary, and auto, and can be customized according to user preferences. Additionally, FileZilla offers features such as site manager, transfer queue, drag and drop support, and remote file editing, making it a popular choice for website management and maintenance.

2.4 Environment building

The MobilityDB team is diligently working to deploy the MobilityDB extension on PostGIS in Windows, but as of right now, MobilityDB is only accessible on Linux / Debian operating systems. Users can use MobilityDB's two primary branches, Master and Development, in either research or production environments. Only the number of functions each database instance carries distinguishes the two. Prior to being published on GIT or Docker hubs, MobilityDB team thoroughly tested every function in the master copy, which is the most recent release (Zimányi, 2020). However, the MobilityDB development team is still validating the development branch, which has greater functionality than the master branch. In order to configure the MobilityDB extension into PostgreSQL that is installed on Linux following are the mandatory requirements:

- The following criteria must be met: PostgreSQL version greater than 10 verified using the simple SQL command # SELECT version ();
- The GNU Scientific Library (GSL) is a free C and C++ library for numerical computations.
- PostGIS version greater than 2.4 10 -- verified using the simple SQL statement # SELECT PostGIS_Version();
- CMake version greater than 3.6 -- verified using the batch command # cmake --version;

- JSON-C is an implementation of the RCO (Reference Counting Object) model that makes it simple to create JSON objects in C and convert them to JSON representations of the objects
- PostgreSQL, PostGIS/liblwgeom, JSON-C, and PROJ development files

2.5 Installations and configurations

The procedure of installing MobilityDB on Linux from scratch involves configuring all necessary components to the same or compatible specifications. Pre-compiled images have been made available on Docker Hub by the MobilityDB team to simplify this laborious process. Using these images will make it simple to install and upgrade MobilityDB on Windows, Linux, and Debian-based operating systems. For creating, sharing, and running applications, Docker is a free and open platform. While in a loosely linked environment, Docker containers help run many applications simultaneously.

Applications that are packaged with all essential flaws and configurations are portable artifacts. When a group of developers is necessary to install the majority of the operating system's services locally on each development workstation while working on an application, Docker containers streamline the development process. This kind of development architecture is frequently complicated and fraught with mistakes when it comes to configuring and versioning the services that are essential to the host operating system. Depending on the complexity of the program, this method of setting up a new environment can be a laborious task. In our scenario, completely configuring MobilityDB on a distant client. Each remote client needs the MobilityDB components that were previously specified.

MobilityDB containers, which have their own distinct OS layer mounted by Linux-based images, replace this form of laborious installation of MobilityDB because they are not intended to be deployed on the local operating system. Every requirement of MobilityDB with a specific version is packaged with a configuration in the start script inside of one container, and everything is here bundled in one isolated environment. MobilityDB is available in a variety of Docker images on Docker Hub, a public repository for Docker that is open to everyone. Each image has a distinct PostGIS extension or function that corresponds to a different MobilityDB release or branch. However, for large and scalable datasets, the codewit repository is the most reliable with the most extensions / functions and dependencies.

The Docker engine (a docker runtime), which can be installed on Mac, Linux, and Windows, pulls and runs Docker images. When MobilityDB is implemented using Docker, it is possible to run many branch MobilityDB containers at once on various host ports.

Listed below is the installation processes of MobilityDB from Docker hub using its Codewit repository also can be shown in Figure 2.7;

A. Docker simply needs one command to get a container, irrespective of the host machine's operating system.
    - docker pull codewit/mobilitydb
B. Although the MobilityDB container can function without a Docker-volume, the following command is used to establish a Docker Volume with the name "mobility_data" in order to maintain the MobilityDB files outside of the container.
    - Docker volume creation with sudo mobility_data

C. In order for any pulled pictures to be operational, they must all be executed. The following command maps the Postgres default port of 5432 to the host port of 25432 to avoid any conflicts with any Postgres instances currently operating on the local machine:

   - mobilitydb_data:/var/lib/postgresql codewit/mobilitydb docker run --name "mobilitydb_codewit" -d -p 25432:5432

Upon successful execution of above three commands, MobilityDB docker image is now up and running and is ready to accept connections. We can connect newly configured MobilityDB container by using any version of PgAdmin4 Client (a web-based GUI tool used to communicate with Postgress database both locally and remotely) with following parameters: -

- **Name**: mobilitydb_image
- **Host**: Container IP address
- **Maintenance Database**: postgres_database
- **Username & Password**: docker (it is the default ID & Password of Docker images for MobilityDB)

2.6 Identification of missing footprints

Now after migrating and converting raw footprints of trains into the Postgres environment, data were analyzed to apply some pre-processing over it that includes identification of missing footprints as shown in Figure 2.8 and Figure 2.9. Gaps in the coverage of the train's location footprint were discovered throughout the data processing. These pauses ranged in length from a couple of hours to the entire journey.

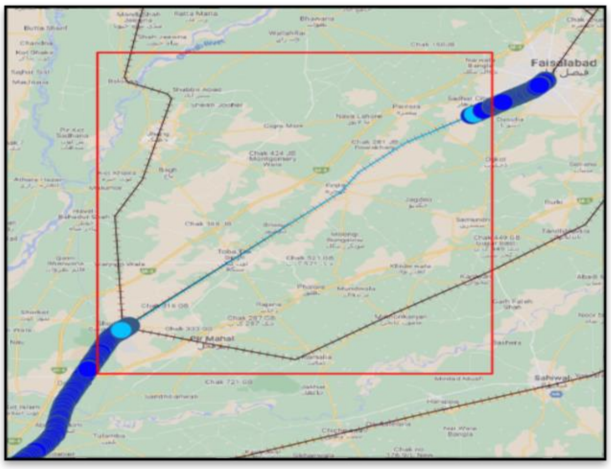


Figure 2.8: Gap in coverage from Faisalabad to Toba Tek Singh.

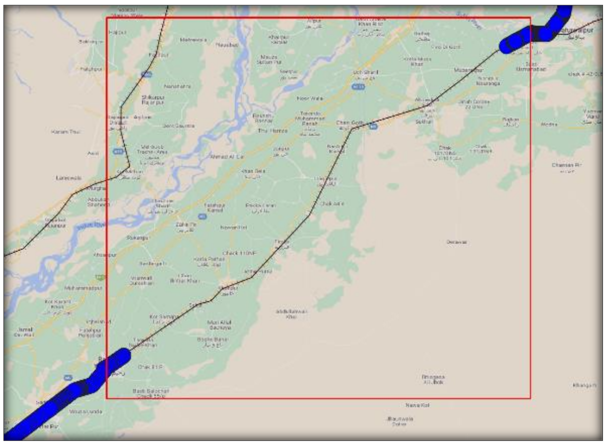
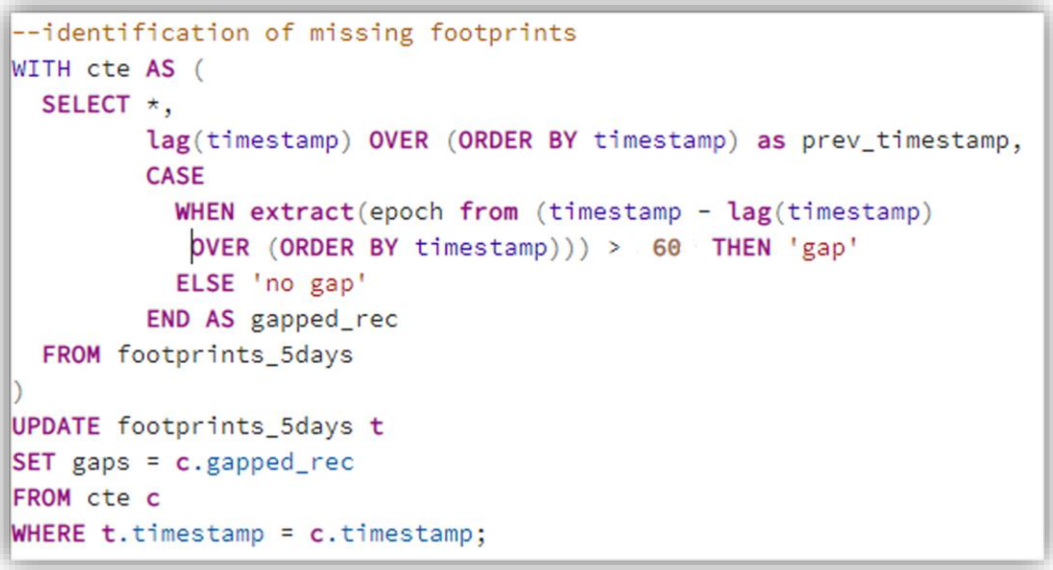


Figure 2.9.Gap in coverage from Bahawalpur to Rahimyar Khan.



```
--identification of missing footprints
WITH cte AS (
  SELECT *,
         lag(timestamp) OVER (ORDER BY timestamp) as prev_timestamp,
         CASE
           WHEN extract(epoch from (timestamp - lag(timestamp)
             OVER (ORDER BY timestamp))) >  60  THEN 'gap'
           ELSE 'no gap'
         END AS gapped_rec
  FROM footprints_5days
)
UPDATE footprints_5days t
SET gaps = c.gapped_rec
FROM cte c
WHERE t.timestamp = c.timestamp;
```

Figure 2.10. Identification of missing footprints.

The train started its journey from Faisalabad and originally transmitted its location information. The train, however, eventually stopped transmitting updates, which caused a void in the location data that had previously been logged. This gap was recorded as beginning at 12:44 and lasting until14:08 on 27th June 2022.

The train stopped streaming its location data while it was between Bahawalpur to Rahimyar Khan. The recorded footprint coverage had a gap as a result of this disruption. This gap was recorded as beginning at 22:49 and lasting until 23:19 on 05th July 2022.

These noted footprint coverage gaps represent times when the train's location information was either unavailable or not recorded. These gaps were taken into account while examining how the train moved as well as when evaluating the correctness and completeness of the entire data set.

Therefore, PostgreSQL query was made to handle the problem of finding and excluding trips with data gaps. This query aids in classifying each record's status as "gap" or "no gap" based on a certain circumstance. The test involves determining whether the time gap between two successive footprints is larger than one hour. Figure 2.10 shows the query.

This query can be used to quickly filter out all footprint entries with the "gap" state. This makes it possible to exclude certain records from additional analysis and reducing their potential influence.

Chapter 3

RESULTS AND DISCUSSIONS

3.1 Identification of idle and stopped trains

Two conditions were taken into account in order to recognize and categorize trains as stopped or idle. First, speed of trains had to be 0, indicating that it wasn't moving and second, no station should be within a 1-kilometer radius of any footprints (that is the train location data).

This logic was implemented utilizing the "st_contains" function of Postgres. This tool aids in determining whether a footprint is contained within the limits of a particular station. It is possible to tell whether a train is close to a station or not by using this function. Figure 3.1 shows the query used to identify and filter the dataset with status as idle and not idle.

The goal of the query is to update the idle column status under the assumption that the train is moving at a standstill. By utilizing the st_contains method to see if any footprints fall within the selected buffer table, it also makes sure that another condition is not true.

In order to determine whether one spatial object is included within another, the st_contains function is frequently employed. It looks to see if any footprints are present in the buffer zone in this instance. But in this case, the query is made to prevent footprints inside the buffer area from being regarded as idle. As a result, they won't be classified as idle if the st_contains function detects any footprints inside the buffer.

The footprints that met the aforementioned requirements were tagged as "yes" for the idle status in order to update the status of the idle column in the footprints table. This update makes it simple to recognize the footprints of stopped or idle trains. This method's goal is to correctly detect and categorize stationary trains so that they can be left out of any future research or calculations where their presence might create bias or errors. The analysis can concentrate on the pertinent and active train movements by putting these conditions into place and updating the idle status accordingly.

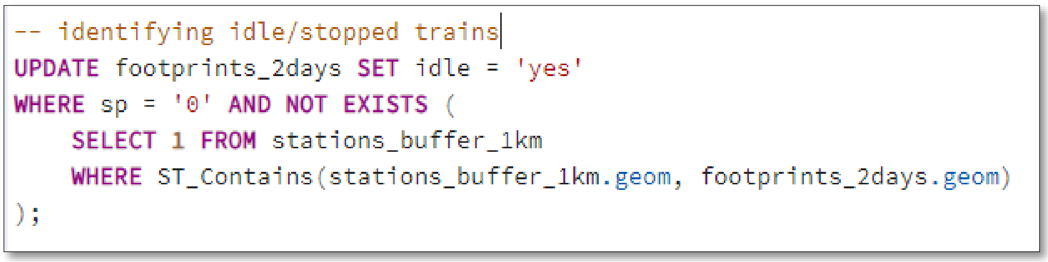
Figure 3.2 reveals a distinct cluster that denotes a circumstance in which the train is inactive or stopped. In this instance, the cluster on the map designates a particular location where the train's motion has stopped. The proximity of the imprints in this group indicates that the train was stopped for a considerable amount of time. This observation of the cluster's features can help with additional analysis and decision-making about train operations and scheduling as well as analyzing and evaluating the operational behavior of the train.

It is clear from Figure 3.3 that the train in the illustration has no scheduled route but is nonetheless transmitting location information. This case raises a number of plausible possibilities. The location streaming option may have accidentally been left on by the train's driver, to name one possibility. Due to human error or oversight, this can happen, resulting in the transmission of location data even when there isn't a scheduled trip or the train isn't in motion. Another option is that the train may have had a problem or been involved in an accident, which forced an unforeseen stop in the middle of the journey. In these scenarios, the train's location streaming might still be going strong despite the unplanned break in its route.

For successful train operations management and monitoring, these situations must be recognized. It enables prompt investigation of any problems, guaranteeing that suitable responses may be given to any malfunctions, mishaps, or operational mistakes that may have happened.

3.2 Trains speed analysis

The railway tracks' speed data was analyzed and condensed using a PostgreSQL query. The summary contains data on the maximum, average, and minimum track speeds. EL-1, EL-2, EL-3, and line L-1, L-2, L-3 are two other subcategories of the Pakistan railways networks. A speed summary table was created and it offers a thorough breakdown of the track speeds. By removing footprint records that are within a 2-kilometer radius of a station, the minimum speed was calculated. As the rains usually slowdown in these buffer zones as they approach stations, therefore this exclusion was necessary. A graph shown in Figure 3.4 was made utilizing the search results to illustrate the speed data. The greatest speed is shown on the graph as a blue bar, the average speed is shown as an orange bar, and the minimum speed is shown as a gray bar. The graph's minimal speed regions show places other than stations. Additionally, maps and tables as shown in Figure 3.5 and Table 3.1 summarize the track speed are included in the query results.



```
-- identifying idle/stopped trains
UPDATE footprints_2days SET idle = 'yes'
WHERE sp = '0' AND NOT EXISTS (
    SELECT 1 FROM stations_buffer_1km
    WHERE ST_Contains(stations_buffer_1km.geom, footprints_2days.geom)
);
```

Figure 3.1.Identification of idle and stopped trains.

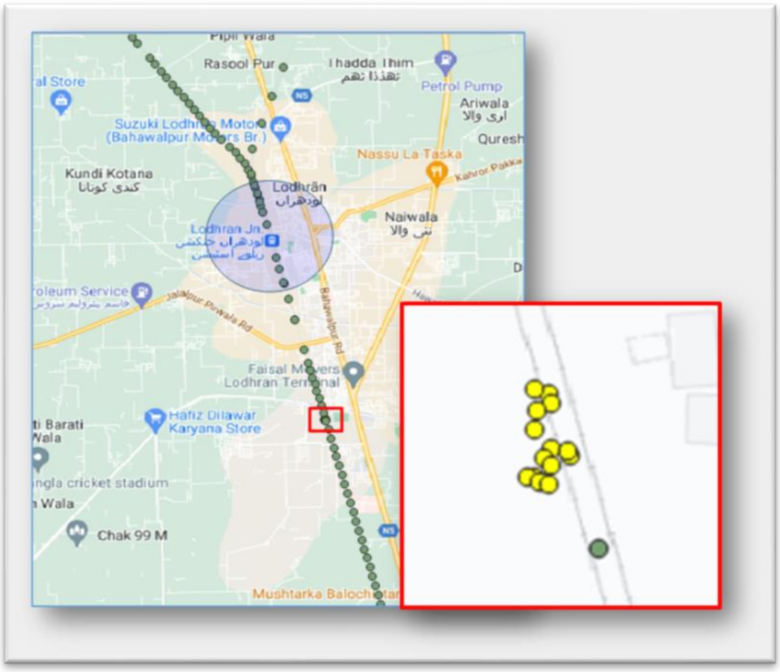
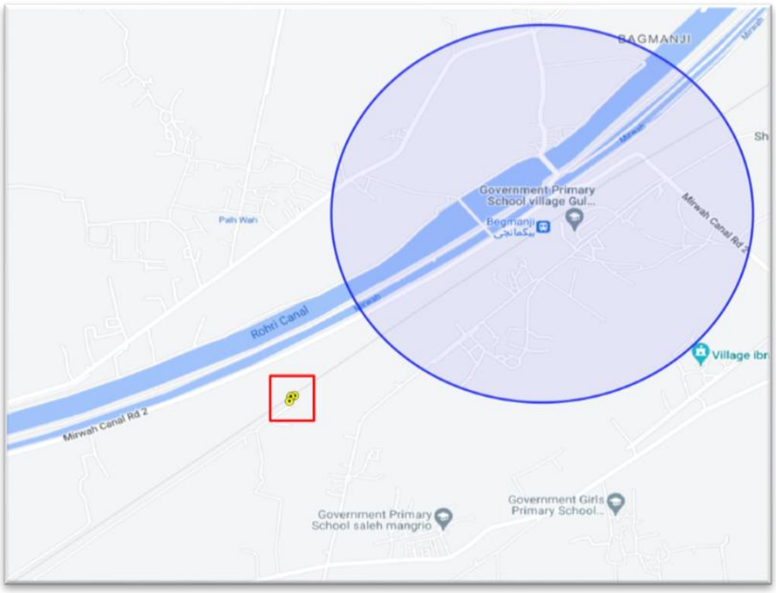


Figure 3.2. Train streaming location in middle of trip.



BAGMANJI
Government Primary School village Gul...
Begmanji
Rohri Canal
Mirwah Canal Rd 2
Village ibr
Government Primary School saleh mangrio
Government Girls Primary School...

Figure 3.3. Train streaming location on idle situation.

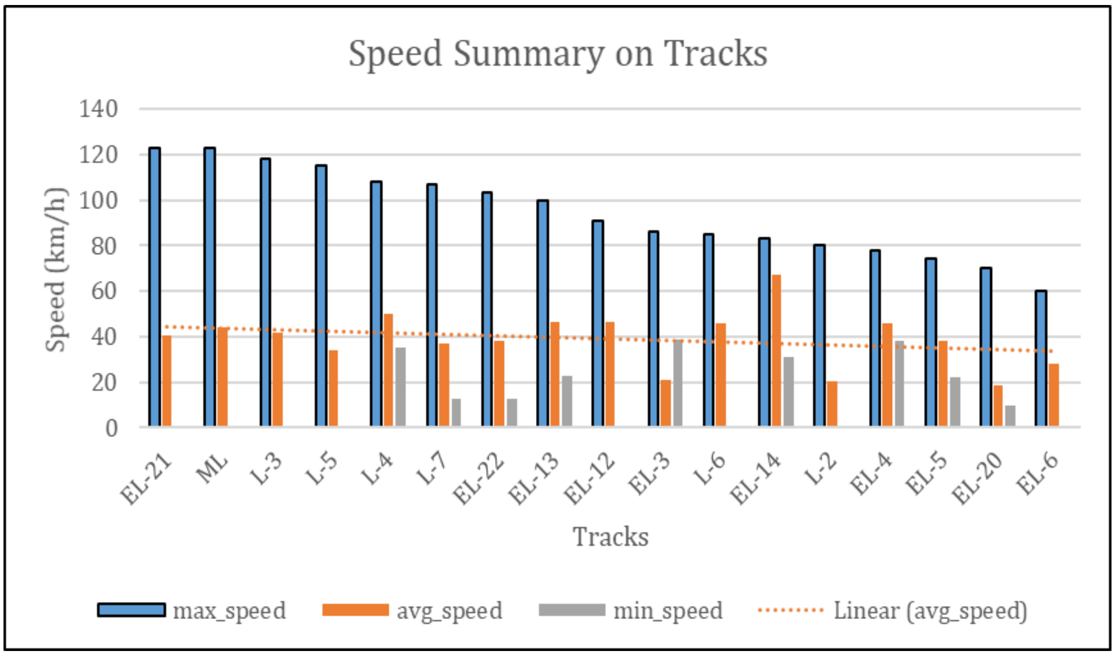


Figure 3.4. Speed summary on tracks.

Table 3.1. Summary of speed on track.

Track ID	**Track Name**	**Route**	**Speed of Top 5 Tracks (Km/h)**	
			Max	Avg.
ML	Main Line	Peshawar - Karachi	123	44
EL-21	Existing Line 21	Attock - Multan	123	41
L-3	Line 3	Lahore - Multan	118	40
L-5	Line 5	Khanewal – Lodhran	115	34
L-4	Line 4	Sheikhupura - Shorkot	108	49

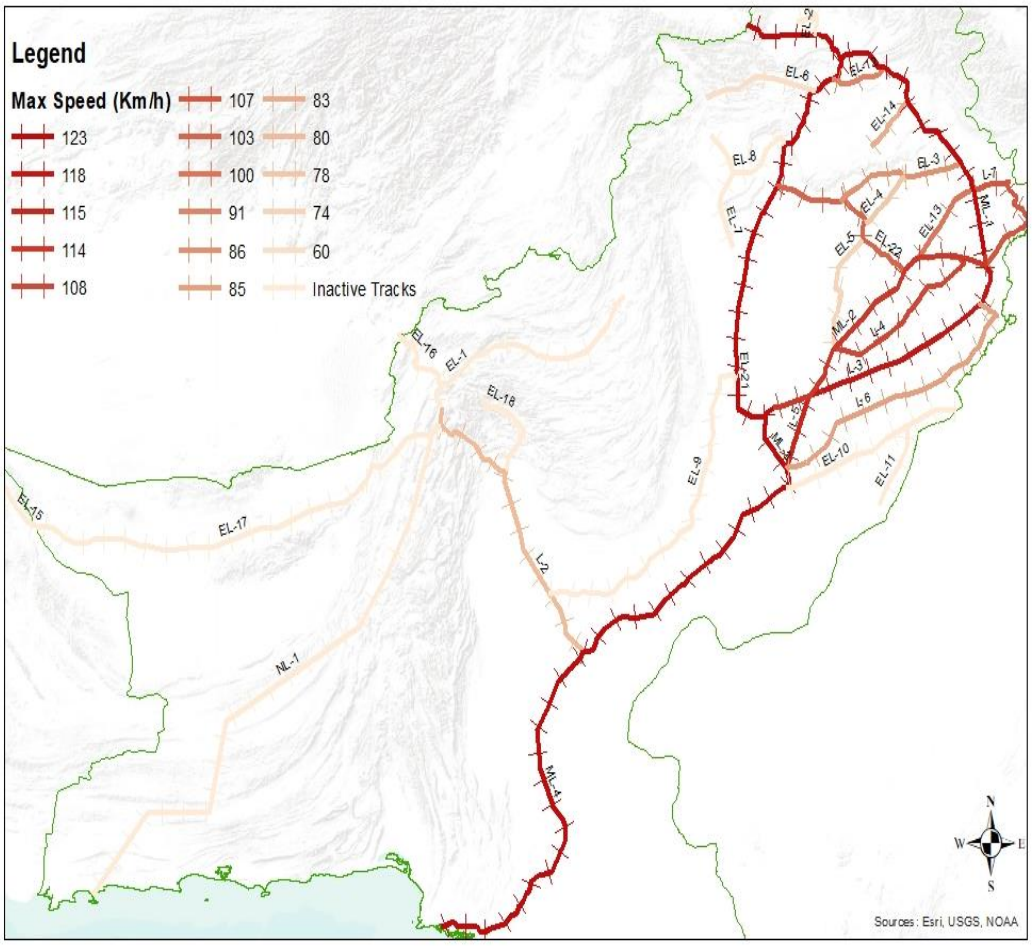


Figure 3.5. Summary of speed on tracks.

The quickest tracks are ML (Main Line) and EL-21, with a recorded speed of 123 km/h on these lines were observed, as can be seen from the graph and table. While the starting point for the ML track is Peshawar to Karachi and the stretch of Existing Line 21 (EL-21) is from Multan to Attock. The analysis also identifies tracks where no operational trains were discovered in the query data.

3.3 Over speeding at station crossings

It has been noted that many passenger trains frequently exceed the posted speed limitations. There have been numerous reports of trains speeding through stations at speeds of up to 110 km/h. This pattern appears to be constant regardless of place, province, or level of station activity. Concerns are raised by these rapid speeds when crossing stations, especially when the station is shut off to the public and passengers. The authorities need to pay attention to this issue to handle the over speeding situation. The Pakistan Railway's General Code of Operating Rules (GCOR) specifies the maximum permissible speeds, which train operators and other personnel must follow. These speed restrictions have been put in place to promote safety and guard against mishaps or occurrences inside station boundaries. Table 3.2 shows the permissible speeds at station crossings.

Speed of trains can be maintained with proper boundary walls and steel barriers at the railway tracks and specifically at the areas where tracks are crossing the slums, villages and cities. This way speed can be maintained to improve the efficiency and new policy can be implemented with new speed limits based on locations, train and type of tracks. Moreover, in such scenarios, slop will also be a crucial factor to be considered during trains movements.

3.3.1 Query to filter over speeding events

To filter and identify over speeding incidents, particularly at station crossings, a query was created. The goal of the search was to locate footprints within a 1-kilometer radius of stations where the train was traveling at a speed more than or equal to 40 km/h as the trains with speed less than 40KM were expected to stop at terminal so they were filtered out. A new table called "jul2_fp_1km_st" was made in order to carry out this filtering operation where the information was preserved against columns of Rec_id (record ID), rec_time (record time), sp (speed), and geom (geographical coordinates). The "footprints_2july" table is used as the source of the query's data. Only footprints with a speed more than or equal to 40 km/h are included in the new table, according to the requirement stated in the WHERE clause.

```
-- speed at station crossings-footprints within 1km of Stations
CREATE TABLE jul2_fp_1km_st AS
SELECT rec_id, rec_time, sp, geom
FROM footprints_2july
WHERE sp >= 40;
```

By running this query, all the footprints that indicate over speeding incidents at station crossings and fulfill the required criteria are saved in the "jul2_fp_1km_st" table for additional analysis or research. Figure 3.6 shows the representations of over speeding events captured from the trains footprints dataset throughout the Pakistan Railways network. Speed parameter was also being provided along with each streaming point, by utilizing these parameters, over speeding was mapped for all trains.

It has been noted that the tendency to drive too fast at station crossings seems unrelated to the station's location, province, or even its level of activity. This shows that the problem of excessive speeding within the railway system is a general one and is not limited to certain regions or stations. When taking into account the prospect of a station being shut down with no public or passenger access, the ramifications of such high speeds during station crossings become even more crucial. In such circumstances, the dangers of speeding are increased because there are no safeguards in place to safeguard people or avoid accidents.

3.3.2 Summary of over speeding at station crossings

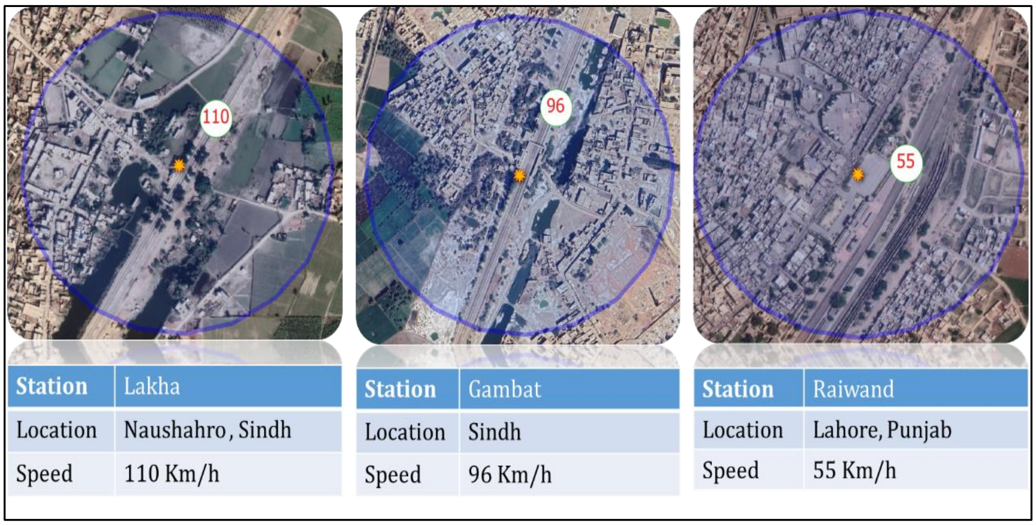
The observed speed ranges of a single day over numerous stations were compiled in a graph in Figure 3.7, taking into account that the maximum permissible speed within station boundaries is 25 km/h for passenger trains. According to the data, 20 sites recorded with speeds greater than 100 km/h, and about 50 stations reported speed ranges between 40 and 50 km/h.

The speed ranges at station crossings were also visually represented on a map in Figure 3.8. The map shows that the over speeding trend is constant independent of the station's location, province, or degree of commerce. This demonstrates the importance of thorough and uniform enforcement of speed limits at all station crossings throughout the entire railway network.

These findings emphasize the significance of addressing the problem of excessive speeding and putting more stringent measures in place to ensure adherence to established speed limits. The safety of travelers, railroad employees, and the general integrity of the railroad system can all be improved by encouraging adherence to these rules.

Table 3.2. GCOR permissible trains speed at station crossings.

Train Type	**Allowed Speed (km/h)**	**Allowed Speed (mph)**
Passenger trains	25 km/h	15.5 mph
Freight trains	15 km/h	9.3 mph

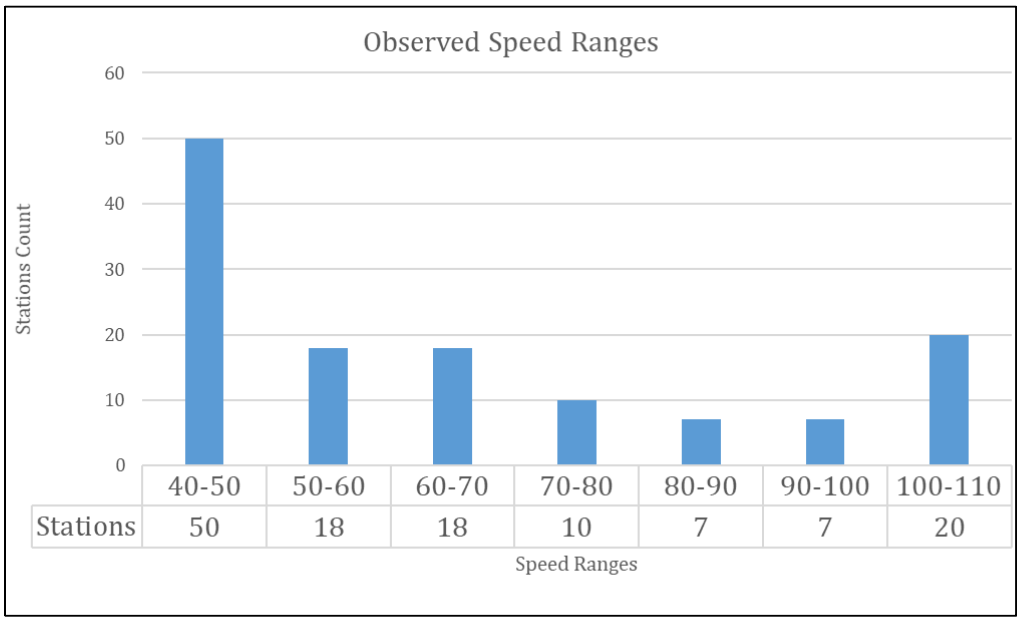


Station	Lakha
Location	Naushahro, Sindh
Speed	110 Km/h

Station	Gambat
Location	Sindh
Speed	96 Km/h

Station	Raiwand
Location	Lahore, Punjab
Speed	55 Km/h

Figure 3.6. Over speeding at station crossing.



	40-50	50-60	60-70	70-80	80-90	90-100	100-110
Stations	50	18	18	10	7	7	20

Figure 3.7. Observed speed ranges.

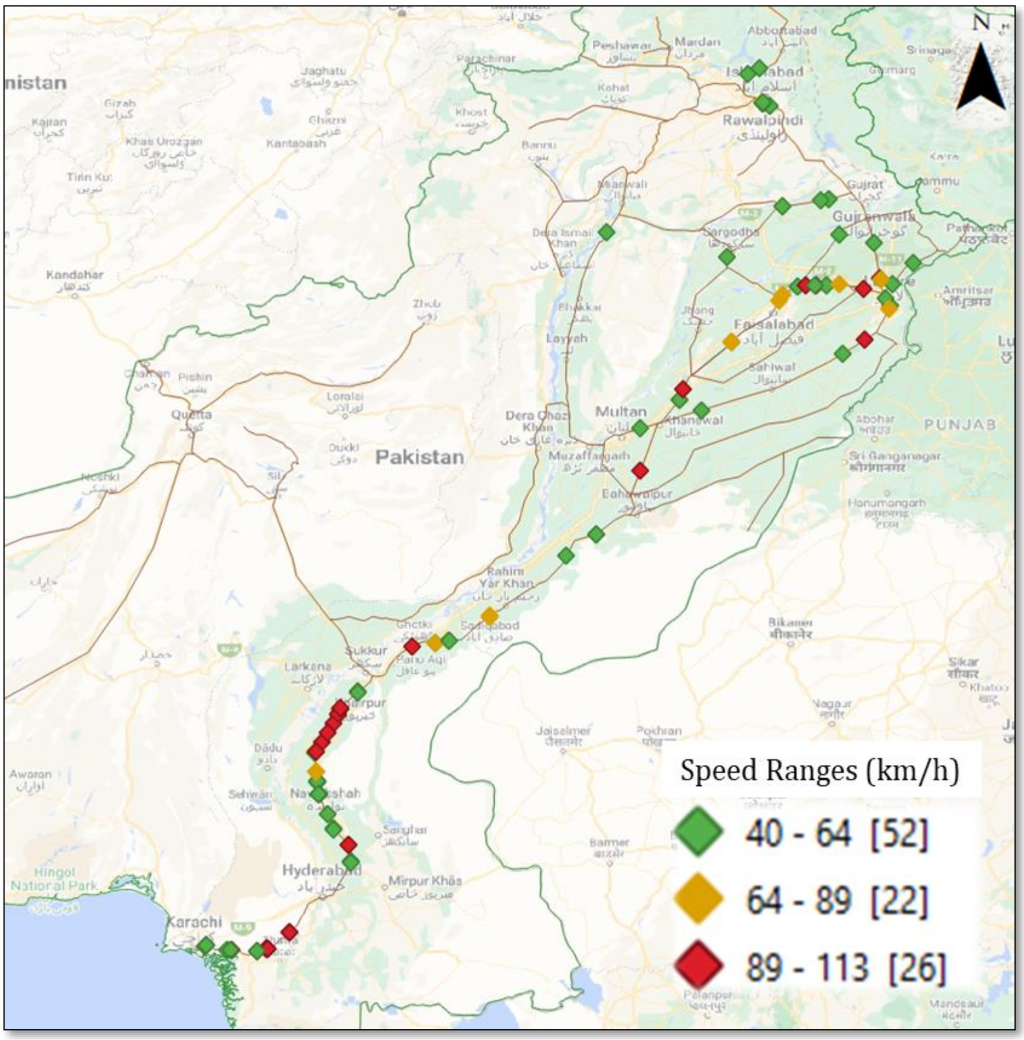


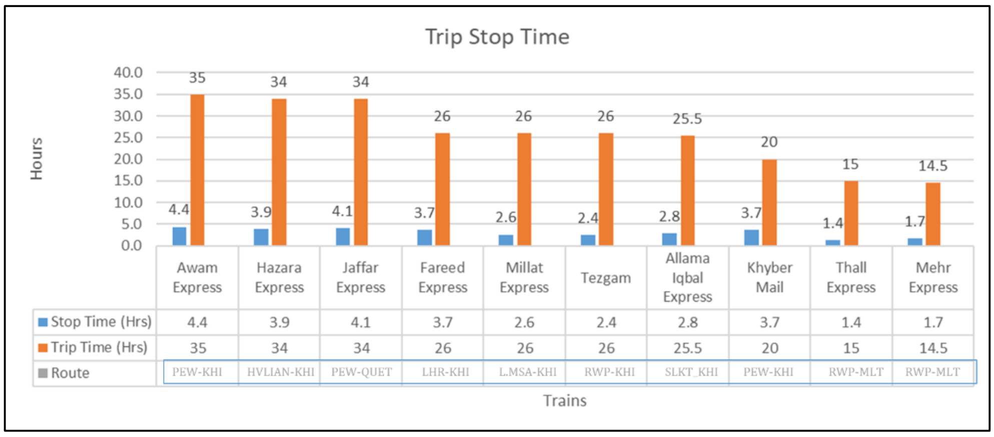
Figure 3.8. Over speeding map at station crossings.

3.4 Stop time analysis

To evaluate the length of the entire journey and the amount of time spent at each station stop, train schedules were examined. For the most popular routes, the graph in Figure 3.9 shows the total journey time as an orange bar and the time spent at station stops as a blue bar to illustrate this comparison.

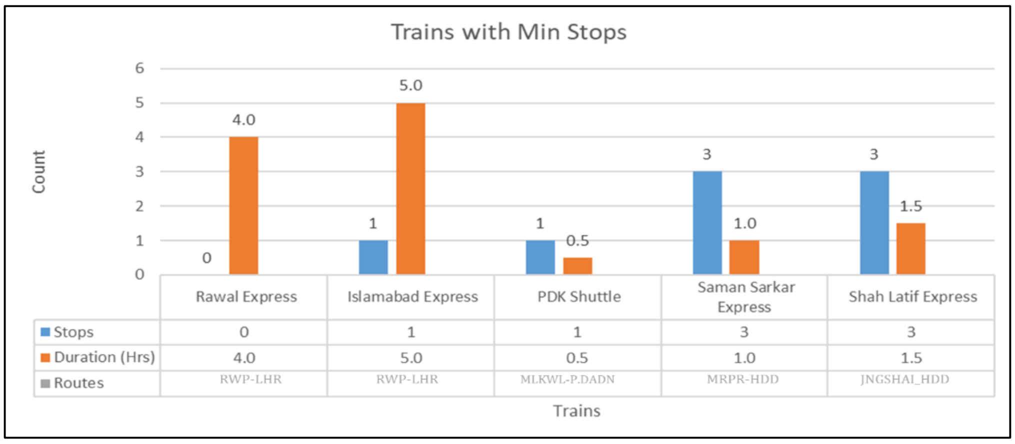
The data shows that stopping at stations takes up a substantial amount of time. For instance, the Awam Express, which runs between Peshawar and Karachi, completes the journey in about 35 hours. However, it takes about 4.4 hours just for station stops, which makes up a sizable chunk of the overall travel time. From Rawalpindi (RWP) to Multan, the Thall Express and Mehr Express follow the same route. Although they stop for around 1.5 hours each along the way at stations, the trains take nearly 15 hours to make the trip. With both trains at same track, the introduction of at least a direct train with no stops might be one way to shorten the overall trip duration given the significant time spent at station pauses. By doing this, waiting times at different stations would be eliminated, possibly resulting in shorter travel times for passengers.

Station pauses account for about 12% of the average trip's total time. This suggests that stopping at numerous stations along the way takes up a considerable amount of the travel time. Improvements can be made to train timetables, cut down on unneeded delays, and increase the overall effectiveness of the railway system by looking at the data and identifying places where a lot of time is spent at station pauses.



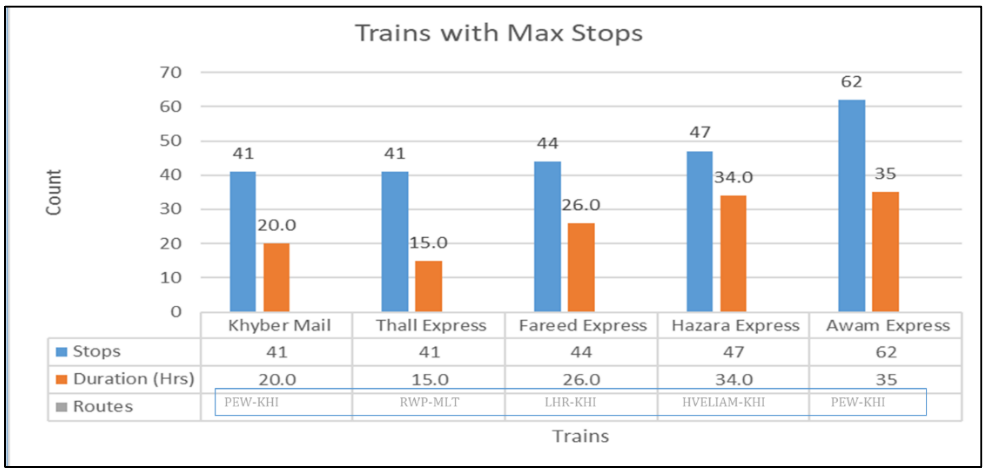
	Awam Express	Hazara Express	Jaffar Express	Fareed Express	Millat Express	Tezgam	Allama Iqbal Express	Khyber Mail	Thall Express	Mehr Express
🟦 Stop Time (Hrs)	4.4	3.9	4.1	3.7	2.6	2.4	2.8	3.7	1.4	1.7
🟧 Trip Time (Hrs)	35	34	34	26	26	26	25.5	20	15	14.5
⬜ Route	PEW-KHI	HVLIAN-KHI	PEW-QUET	LHR-KHI	L.MSA-KHI	RWP-KHI	SLKT_KHI	PEW-KHI	RWP-MLT	RWP-MLT

Figure 3.9. Trip stop times.



	Rawal Express	Islamabad Express	PDK Shuttle	Saman Sarkar Express	Shah Latif Express
🟦 Stops	0	1	1	3	3
🟧 Duration (Hrs)	4.0	5.0	0.5	1.0	1.5
⬜ Routes	RWP-LHR	RWP-LHR	MLKWL-P.DADN	MRPR-HDD	JNGSHAI_HDD

Figure 3.10. Top 5 trips with minimum stops.



	Khyber Mail	Thall Express	Fareed Express	Hazara Express	Awam Express
🟦 Stops	41	41	44	47	62
🟧 Duration (Hrs)	20.0	15.0	26.0	34.0	35
⬜ Routes	PEW-KHI	RWP-MLT	LHR-KHI	HVELIAM-KHI	PEW-KHI

Figure 3.11. Top 5 trips with maximum stops.

3.5 Stops frequency in trips

Stops frequency in trains trips analyse the count and duration of stops in various train journeys The graph in Figure 3.10 shows the summary of top 5 minimum stops on trip where y-axis in the graph reflects the count of stops observed from all trips in the database, and the x-axis represents the duration of stops in hours.

The graph shows the frequency of stops in journeys. As an illustration, the Rawal Express shows zero stops on its route from Rawalpindi (RWP) to Lahore (LHR), showing that it is a non-stop or direct service between these two locations. The Islamabad Express, in contrast, only makes one stop along the way. Passengers now have alternatives to select their favorite train based on elements like cost and schedule thanks to this information. Others may not mind making a few stops if it fits their timetable or has a more reasonable ticket. Some passengers may choose a non-stop service for faster travel. The graph helps passengers to choose a train that best suits their unique needs and preferences by outlining the number of stops and their durations.

The top five journeys with the most stops are shown on the graph in Figure 3.11. The y-axis displays the total number of stops observed throughout all journeys in the database, while the x-axis displays the stop time in hours and routes.

For instance, the Awan Express makes 62 stops along the way as it travels from Peshawar to Karachi, covering the distance in around 35 hours. This impressive number of stops is remarkable since it enables the connection of numerous Pakistani cities along a single rail route. However, providing alternate options such as direct trains between important cities would provide customers more freedom to select according to their preferences for cost and travel time.

3.6 Assessing signal visibility to trains: A comparative analysis of PostgreSQL and MobilityDB for efficient traffic management

In this thesis research, investigation on how to measure the visibility of signals for the moving trains was performed by using a database query that was performed by implementing the strategy[2]. In order to analyze GPS trajectories and determine the visibility of signals for the best traffic management, this study compares MobilityDB against traditional PostgreSQL to show how efficiently and effectively it does these tasks. Railway authorities can improve the shift from manual operations to a computerized approach and arrange the visibility of specific signal indications to individual train drivers by extracting useful data.

Analysis started by building a streamlined PostGIS database to represent the described scenario. The next step was to progressively create a SQL query to assess if signals are visible to train passengers. In order to use spatial data in the database, the PostGIS extension is first created. As part of the database setup, two tables gpspoints and signals were created. GPS point information, including timestamps and geographical coordinates, was be kept in the gpspoints table. The signals table, on the other hand, contained details on the locations of the signals, including their names and related coordinates. Within the SQL code block given

[2] https://techcommunity.microsoft.com/t5/azure-database-for-postgresql/analyzing-gps-trajectories-at-scale-with-postgres-mobilitydb-amp/ba-p/1859278#fn-1

below, first PostGIS extension was created. And then created two tables: gpspoints and signals with listed columns and datatypes. A representation of the database is shown in the map, see Figure 3.12. Green points, which represent the location of the train in the gpspoints table. Database was created with following queries .

```
-- filtering gpspointss and signals
CREATE TABLE gpspoints (tripID int, pointID int, t timestamp, geom
geometry(Point, 32643));
CREATE TABLE signals(signalID int, geom geometry(Point, 32643));
INSERT INTO gpspoints Values
(1, 1, '2022-07-21T08:37:27.000', 'SRID=32643;POINT(308124.109890976
3728273.58762612)'), (1, 2, '2022-07-21T08:37:37.000',
'SRID=32643;POINT(308227.027802868 3728292.46531692)'), (1, 3, '2022-07-
21T08:37:47.000', 'SRID=32643;POINT(308334.174048068
3728335.19043466)'), (1, 4, '2022-07-21T08:37:57.000',
'SRID=32643;POINT(308408.640584836 3728369.59647611)'), (1, 5, '2022-07-
21T08:38:07.000', 'SRID=32643;POINT(308461.9169791 3728407.41833811)'),
(1, 6, '2022-07-21T08:38:17.000', 'SRID=32643;POINT(308551.381967878
3728441.52537676)'), (1, 7, '2022-07-21T08:38:27.000',
'SRID=32643;POINT(308663.644254388 3728490.1344151)'), (1, 8, '2022-07-
21T08:38:37.000', 'SRID=32643;POINT(308767.68893195
3728503.01244136)'), (1, 9, '2022-07-21T08:38:47.000',
'SRID=32643;POINT(308871.271302941 3728495.83161316)'), (1, 10, '2022-
07-21T08:38:57.000', 'SRID=32643;POINT(308948.63331003
3728484.44304838)'), (1, 11, '2022-07-21T08:39:07.000',
'SRID=32643;POINT(309024.462819388 3728461.99059971)'), (1, 12, '2022-
07-21T08:39:17.000', 'SRID=32643;POINT(309133.341037885
3728403.73139497)'), (1, 13, '2022-07-21T08:39:27.000',
'SRID=32643;POINT(309245.657120396 3728298.29116713)'),
(1, 14, '2022-07-21T08:39:37.000', 'SRID=32643;POINT(309315.104214897
3728206.41934101)'), (1, 15, '2022-07-21T08:39:47.000',
'SRID=32643;POINT(309373.581720467 3728113.08386572)'), (1, 16, '2022-
07-21T08:39:57.000', 'SRID=32643;POINT(309461.975318551
3727972.88083162)'), (1, 17, '2022-07-21T08:40:07.000',
'SRID=32643;POINT(309530.497389953 3727860.55748382)'), (1, 18, '2022-
07-21T08:40:17.000', 'SRID=32643;POINT(309574.593020876
3727751.24417646)'), (1, 19, '2022-07-21T08:40:27.000',
'SRID=32643;POINT(309657.650540303 3727620.21726585)'), (1, 20, '2022-
07-21T08:40:37.000', 'SRID=32643;POINT(309729.083467153
3727506.36102802)'), (1, 21, '2022-07-21T08:40:47.000',
'SRID=32643;POINT(309785.946987735 3727421.36138047)');

INSERT INTO signals Values
(1, 'SRID=32643;POINT(309245.657120396 3728298.29116713)')
```

3.6.1 Utilizing PostgreSQL and PostGIS functions

Each points record of the table has timestamp, enabling a temporal examination of the train's motion. A red diamond icon on the map also designates the location of a signal that is located shortly before the Golra railway station in Islamabad. For determining visibility and examining how the train and the signal interact, this signal is an essential point of reference. It is possible to learn a lot about how visible the signal is to drivers on a moving train by analyzing the spatial distribution of the gpspoints data in conjunction with the signal location. This visualization helps with situation comprehension and provides a framework for additional investigation and query creation.

The next step was to locate the points where a train is within a 300-meter radius of a signal and to calculate the times when it is in this vicinity. This technique seeks to evaluate the period of time when the train and signal are close to each other, indicating probable passenger visibility.

```
CREATE TABLE signal_psql_300 AS
SELECT a.tripid, a.pointid, a.t, a.geom, b.signalid
FROM gpspoint1 a, signals b
WHERE ST_DWithin(ST_Transform(a.geom, 32643), ST_Transform(b.geom,
32643), 300);
```

While the preceding the above PostGIS query identified GPS locations within a 300-meter radius of a signal, it did not address the question of how long this event lasted. Additionally, if a particular GPS point was absent, it would produce null values and ignore the bus trip's continuity and moving trajectory.

PostGIS query was needed that creates a continuous movement trajectory from the provided GPS coordinates in order to get around these restrictions. This query computes how long the signal will be visible to train while they are traveling, in addition to identifying the locations where the signal is visible. This improved query will give a more accurate assessment of signal visibility and the associated duration it is visible to the train by taking into account the movement trajectory and the timestamps of GPS sites. This method guarantees a thorough study that captures both the precise visibility of the signal along the trajectory and the continuity of the bus ride. Following query was developed by utilizing Postgres, PostGIS and windows operations and functions.

```
WITH pointPair AS (
  SELECT
    tripID, pointID AS p1, t AS t1, geom AS geom1,
    lead(pointID, 1) OVER (PARTITION BY tripID ORDER BY pointID) p2,
    lead(t, 1) OVER (PARTITION BY tripID ORDER BY pointID) t2,
    lead(geom, 1) OVER (PARTITION BY tripID ORDER BY pointID) geom2
  FROM gpspoints ), segment AS ( SELECT
    tripID, p1, p2, t1, t2, st_makeline(geom1, geom2) geom
  FROM pointPair
  WHERE p2 IS NOT NULL ), approach AS (
  SELECT
    tripID, p1, p2, t1, t2, a.geom, st_closestpoint(a.geom, b.geom)
visibilityTogglePoint
  FROM segment a, signal b WHERE st_dwithin(a.geom, b.geom, 300)
)
SELECT
  tripID, p1, p2, t1, t2, geom,  visibilityTogglePoint,
  (st_lineLocatePoint(geom, visibilityTogglePoint) * (t2 - t1)) + t1
visibilityToggleTime
FROM approach;
```

Illustrated above query seems though fulfilling the objective, seemed to be quite complex and many Common Table Expressions (CTEs) have been used to structure the complex PostGIS query, making it easier to interpret and write SQL queries with many phases. The window function "lead" is used by the first CTE, "pointPairs" in lines 1–7, to pair up consecutive points that are part of the same bus trip. The data prepared for the subsequent CTE in this stage.

The second CTE, referred to as "segment" in lines 7–12, draws a line segment joining the two locations in each pair. The path between each pair of GPS locations can be thought of as being linearly interpolated in this way. The outcome is a collection of line segments that depict the bus's motion trajectory. Understanding the logic and flow of the actions being carried out is made simpler by decomposing the query into various CTEs. With a modular design, the query is easier to read and maintain, which makes it more manageable for intricate spatial analysis combining GPS points and trajectory interpolation. Result of above discussed query can be visualized as in Figure 3.13.

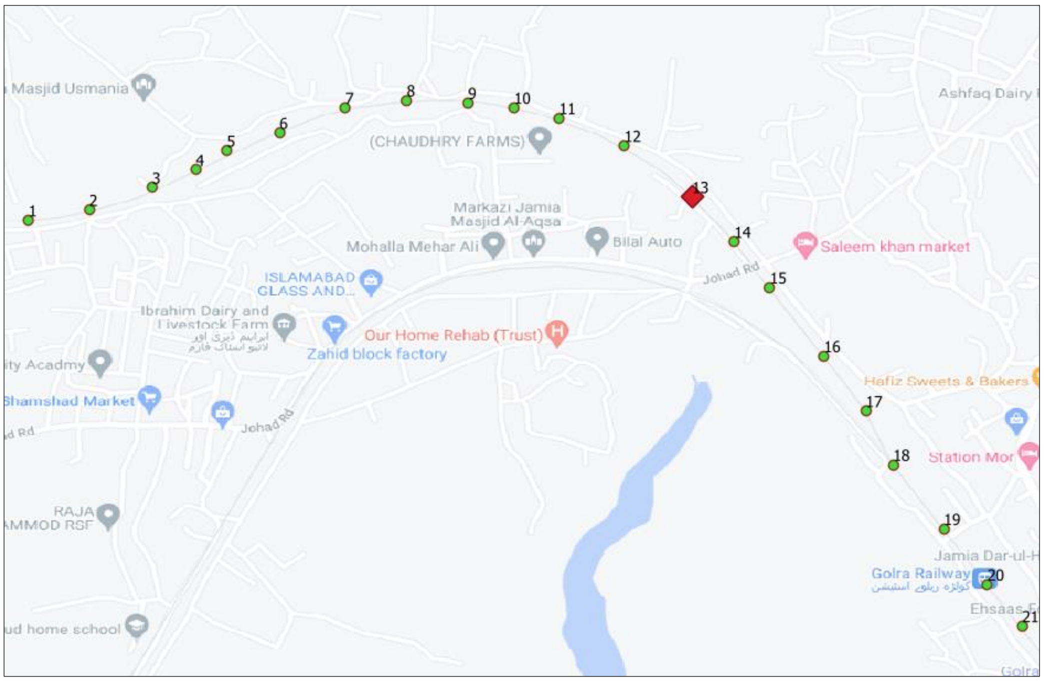


Figure 3.12. Map visualization of trains GPS point and signal.

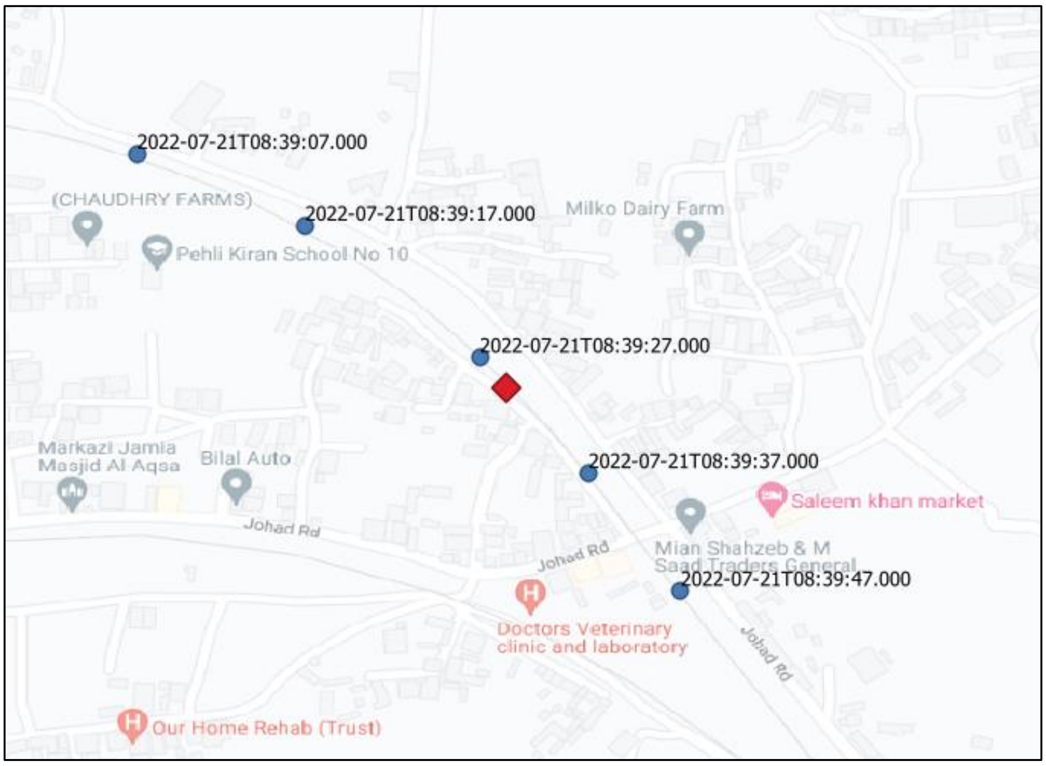


Figure 3.13. Map visualization of PostgreSQL query.

The third CTE, referred to as "approach" in lines 12–18, is concerned with locating the points where the train departs from or arrives at within a 300-meter range of the signal. To do this, a 30-meter-diameter ring is formed around the placement of the signal. The line segments that depict the train trajectory are then intersected with this ring. These points reflect the places where the train passes the signal within 300 meters, suggesting probable driver visibility of the signal. In order to examine the spatial link between the bus route segments and the circular ring surrounding the signal, this method makes use of geometric operations in PostGIS. It enables us to identify the precise locations along the trajectory where the bus is close to the signal, allowing us to estimate how visible the signal may be to the drivers while they are traveling.

The time at the two points identified in the "approach" CTE are computed using linear referencing in the final phase of the preceding PostGIS query at lines 19–22. Along the train trajectory, linear referencing implies a constant speed each segment. Based on the distance traveled along the trajectory, linear referencing can be used to estimate the amount of time spent at each point. The amount of time needed to go from the train starting point to each destination can be determined by assuming a constant speed every segment.

This computation enables us to determine the amount of time that drivers may potentially see the signal while traveling. An estimation of the time spent near the signal can be obtained by taking into account the distance covered along the trajectory and the anticipated constant speed. This method offers information on the length of visibility, enabling a more thorough investigation of the signal’s impact and potential exposure to train drivers.

The query was that much complex due to two non-trivial concepts that were mused in development of the PostGIS query:

1. The GPS data that is currently accessible is discrete and made up of separate points. Nevertheless, the goal of the query was to create a continuous travel trajectory from these separate sites. This required joining adjacent dots to create a seamless trail that represented the train's journey.
2. Spatial-temporal proximity: The train's continuous movement trajectory was then used to pinpoint the precise locations and moments when it was less than 300 meters from the signal. This methodology evaluated the distance between the train and the signal at various times throughout its path, taking into account both spatial and temporal variables.

3.6.2 Utilizing MobilityDB functions

The analysis of such movement trajectories is made easier by MobilityDB. As explained in previous chapters that MobilityDB is an extension of PostgreSQL and PostGIS. By adding unique types and methods to the PostgreSQL foundation, MobilityDB integrates spatio-temporal notions. This integration makes it possible to analyze complex spatio-temporal data more quickly and effectively, making it easier to evaluate movement trajectories.

MobilityDB can be used to implement the same PostgreSQL query to achieve the same results with quite reduced code effort and with easy to understand logic. The transition of the previous PostGIS query might look something like this in MobilityDB.

```
create table signal_mdb_300_raw as
SELECT astext(atperiodset(trip, getTime(atValue(tdwithin(a.trip, b.geom,
300), TRUE))))
FROM trainTrip a, signals b
WHERE dwithin(a.trip, b.geom, 300)
```

This is as simple in the MobilityDB as describe in above query block. The supplied query uses the outcomes of a SELECT operation to create a new table called signal_mdb_300_raw. Here is a description of the problem:

The trainTrip table and the signals table are queried using the SELECT statement to retrieve the spatio-temporal data. The tdwithin function is used to determine whether the train's trajectory is 300 meters or less from a signal position. The getTime and atValue procedures extract the precise time periods of intersection, whereas the atperiodset and atperiodset functions combine the matching time periods of the train trajectory and signal proximity The outcome is then transformed into a text representation using the astext function. The CREATE TABLE statement is used to store the whole result set in the signal_mdb_300_raw table.

Now in order to map this query in QGIS, text column was divided into geom and time columns as explain in following code block.

```
-- mapped query --
-- Add new columns to the existing table
ALTER TABLE signal_mdb_300
ADD COLUMN geom geometry(Point, 32643),
ADD COLUMN time timestamp;

-- Update the new columns with the extracted information
UPDATE signal_mdb_300
```

```
            SET geom = ST_SetSRID(ST_MakePoint(
                                  CAST(split_part(SUBSTRING(astext
   FROM 'POINT\((.*?)\)'), ' ', 1)
   AS DOUBLE PRECISION),
                                  CAST(split_part(SUBSTRING(astext
   FROM 'POINT\((.*?)\)'), ' ', 2)
   AS DOUBLE PRECISION)
                        ), 32643),
                        time = to_timestamp(SUBSTRING(astext
   FROM '@(.*?)\)') || '+00', 'YYYY-MM-DD HH24:MI:SS.US+00');
```

The given code snippet runs the following commands on the'signal_mdb_300' database to make it changes:

1. It adds the two columns "geom" of type "geometry(Point, 32643)" and "time" of type "timestamp" to the table. The extracted geographic and temporal information will be kept in these columns, respectively.
2. The newly added columns in the table are updated by the 'UPDATE' statement using the data that was previously taken from the 'astext' column.
    a. The 'ST_SetSRID' and 'ST_MakePoint' functions are used to create a 'Point' geometry in order to update the 'geom' column. Using string manipulation functions, the coordinates for the point are extracted from the 'astext' column.
    b. The 'time' column is updated by removing the timestamp data from the 'astext' column using string manipulation techniques and converting it to a 'timestamp' data type.

trainTrip table creation was performed with following query.

```
CREATE TABLE trainTrip(tripID, trip) AS
  SELECT  tripID,tgeompointseq(array_agg(tgeompointinst(geom,  t)  ORDER
BY t))
FROM gpspoints
GROUP BY tripID;
```

The provided above code creates one new table that is called trainTrip using the results obtained from the SELECT statement. Here is the explanation of the query:

1. The SELECT statement pulls information from the gpspoints table and aggregates it by tripID to group the points. The points are combined into an array using the array_agg function, and temporal geometry points are created using the tgeompointinst function for each point and its corresponding time (t).
2. The array of temporal geometry points is then converted into a temporal sequence of points using the tgeompointseq function. The time sequence that results depicts each journey's path.
3. The CREATE database statement generates the trainTrip database with two columns: tripID and trip, where tripID holds the temporal sequence.

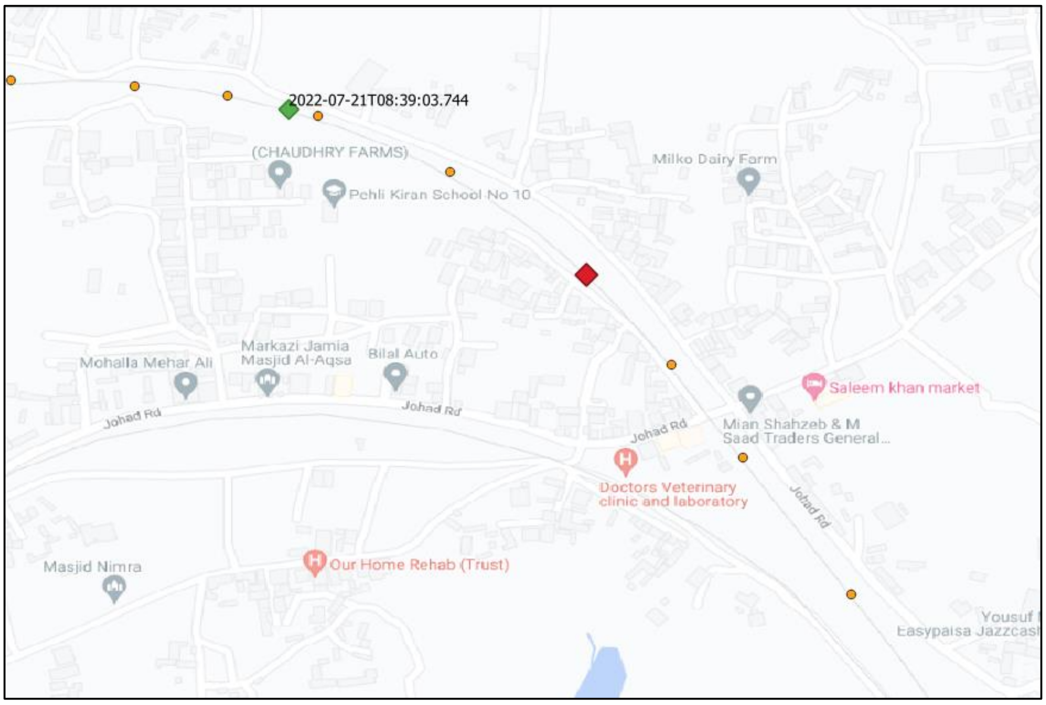


Figure 3.14. Map visualization of MobilityDB query.

One thing to note, even though no GPS coordinates or timestamps were explicitly provided, the result of the given query using MobilityDB displays the beginning point where the train is anticipated to be within 300 meters of a particular location and that point is marked with green diamond symbol and the orange points are the GPS points of the trip section where red diamond symbol is the signal as shown on map in Figure 3.14.

Based on the information at hand, MobilityDB has the capacity to internally calculate and rebuild the train's trajectory. It determines the movement trajectory by using the spatial and temporal data contained in the trainTrip table. The query can calculate the time when the train will be close by using the temporal geometry functions and types offered by MobilityDB.

The query works by taking into account the temporal order of points stored in the trainTrip table's journey column. The starting point where the train is anticipated to be within 300 meters of a signal position is determined from this sequence. Based on the trajectory's internal calculation, the time at this location is inferred.

Because of its temporal capabilities, MobilityDB can analyze and forecast movement trajectories even when explicit GPS coordinates or timestamps are not given. In situations where complete data may be unavailable, it makes use of the existing information to reconstruct and evaluate the trajectory, enabling more thorough spatio-temporal analysis.

Chapter 4

CONCLUSION AND RECOMMENDATIONS

4.1 Conclusion

Research conducted through the thesis addressed the problem of enhancing monitoring efficiency of the Pakistan Railways by exploring capabilities of MobilityDB, PostgreSQL and PostGIS for analyzing the train trajectories. The main goal was to provide a valuable insight into spatio-temporal behavior of moving object and to offer suggestions for the optimization of railway operations. Various tools and techniques were also utilized to achieve the research objectives. A mechanism was also developed to capture and to store live streaming train trajectories data in a PostgreSQL database. The data preprocessing and the cloud collected data migration was conducted by utilizing customized Python scripts.

Throughout the research, several key aspects of Pakistan Railways were identified and analyzed. Idle and stopped trains were analyzed and successfully identified using the speed and location datasets, enabling efficient monitoring and the resource allocation. Speed patterns were also analyzed to detect the instances of over speeding and suggested the importance of adhering to speed limits. A comprehensive analysis of trains speed, track speed, stop times stop frequency were performed that provided insights to optimize schedules and to reduce the unnecessary delays in trains trips. The visibility of the signals to trains were explored by using MobilityDB and PostgreSQL, enabling assessment of signal visibility duration and the spatial coverage. This analysis then offered insights to optimize the traffic management and ensure the passengers safety.

Also, the research outcomes demonstrate effectiveness of using MobilityDB over PostGIS and PostgreSQL for analyzing spatio-temporal behavior of transportation systems. By leveraging the spatio-temporal analysis, the resource allocations can be optimized and passenger experiences can be enhanced and safe and reliable journeys can be ensured. In conclusion, this research highlights the significance of the analysis of moving object trajectories using MobilityDB and PostgreSQL operations, particularly in context of Pakistan Railways systems.

4.2 Recommendations for further research

This study mainly focuses on captured archived footprints of trains but a real time dataset can be used to perform spatio-temporal analysis to look into the creation of real-time monitoring and decision support systems for transportation networks. To enable prompt decision-making, optimize resource allocation, and boost overall system efficiency, this could entail integrating live streaming data, real-time analytics, and visualization approaches.

Godfrid (2022) used the real time GTFS data to develop a system to monitor trajectories in real time but during its research period, no live feeds were available and thus got no responses. Same was the case with this thesis research where Pak Rail Live APIs got down after August 2019 due to portal maintenance issue and the researched was focused to only archived data. Now based on the availability of live stream trajectory data, new dimension of implementing MobilityDB queries over real time data can be explored to further extend this research.

REFERENCES

1. Brandoli, B., Raffaetà, A., Simeoni, M., Adibi, P., Bappee, F. K., Pranovi, F., & Matwin, S. (2022). From multiple aspect trajectories to predictive analysis: a case study on fishing vessels in the Northern Adriatic sea. GeoInformatica, 26(4), 551-579.

2. Bates, G. A. (1994, February). The use of GPS in a mobile data acquisition system. In Developments in the Use of Global Positioning Systems (pp. 2-1). IET.

3. Borodin, A., Mirvoda, S., Kulikov, I., & Porshnev, S. (2017). Optimization of memory operations in generalized search trees of PostgreSQL. In Beyond Databases, Architectures and Structures. Towards Efficient Solutions for Data Analysis and Knowledge Representation: 13th International Conference, BDAS 2017, Ustroń, Poland, May 30-June 2, 2017, Springer International Publishing. Proceedings 13 (pp. 224-232).

4. Chacko, A. M., Basheer, A. M., & Kumar, S. M. (2015, December). Capturing provenance for big data analytics done using SQL interface. In 2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON) (pp. 1-6). IEEE.

5. Cubukcu, U., Erdogan, O., Pathak, S., Sannakkayala, S., & Slot, M. (2021, June). Citus: Distributed postgresql for data-intensive applications. In Proceedings of the 2021 International Conference on Management of Data (pp. 2490-2502).

6. D’Orso, G., & Migliore, M. (2017). A GIS-based methodology to estimate the potential demand of an integrated transport system. In Computational Science and Its Applications–ICCSA 2017: 17th International Conference, Trieste, Italy, July 3-6, 2017, Springer International Publishing. Proceedings, Part IV 17 (pp. 525-540).

7. Das, N. K., Das, C. K., Mozumder, R., & Bhowmik, J. C. (2009). Satellite based train monitoring system. Journal of Electrical Engineering the Institution of Engineers, Bangladesh Vol. EE, 36.

8. Furman, E., Lampe, S., & Immel, E. (2001). Keeping Track of RF High-speed trains will use GPS in their data communications platform to automate selections of radio channels, improve safety, and coordinate operations with the train dispatch center. A pilot projects tests this system along tracks in the Pacific Northwest. GPS WORLD, 12(2), 16-23.

9. Godfrid, J., Radnic, P., Vaisman, A., & Zimányi, E. (2022). Analyzing public transport in the city of Buenos Aires with MobilityDB. Public Transport, 14(2), 287-321.

10. Graser, A., & Dragaschnig, M. (2020). Exploring movement data in notebook environments. In IEEE VIS 2020 Workshop on Information Visualization of Geospatial Networks, Flows and Movement (MoVis). 2020.

11. Georgiou, H., Karagiorgou, S., Kontoulis, Y., Pelekis, N., Petrou, P., Scarlatti, D., & Theodoridis, Y. (2018). Moving objects analytics: Survey on future location & trajectory prediction methods. arXiv preprint arXiv:1807.04639.

12. Güting, R. H., Behr, T., & Xu, J. (2010). Efficient k-nearest neighbor search on moving object trajectories. The VLDB Journal, 19, 687-714.

13. Guedes, T., Silva, V., Mattoso, M., Bedo, M. V., & de Oliveira, D. (2018, November). A practical roadmap for provenance capture and data analysis in spark-based scientific workflows. In 2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS) (pp. 31-41). IEEE.

14. Hofestadt, H. (1995, March). GSM-R: global system for mobile radio communications for railways. In 1995 International Conference on Electric Railways in a United Europe (pp. 111-115). IET.

15. Helland-Hansen, W., & Hampson, G. J. (2009). Trajectory analysis: concepts and applications. Basin Research, 21(5), 454-483.

16. Ikeda, M. (1993). Characteristic of position detection and method of position correction by rotating axle. Railway Technical Research Institute, Quarterly Reports, 34(4).

17. Imran, M. (2009). Public transport in Pakistan: a critical overview. Journal of Public Transportation, 12(2), 53-83.

18. Irfan, S. M., Kee, D. M. H., & Shahbaz, S. (2012). Service quality and rail transport in Pakistan: A passenger perspective. World Applied Sciences Journal, 18(3), 361-369.

19. Kumar, G. H., & Ramesh, G. P. (2017, February). Intelligent gateway for real time train tracking and railway crossing including emergency path using D2D communication. In 2017 International Conference on Information Communication and Embedded Systems (ICICES) (pp. 1-4). IEEE.

20. Karniouchina, E. V., Carson, S. J., Short, J. C., & Ketchen Jr, D. J. (2013). Extending the firm vs. industry debate: Does industry life cycle stage matter?. Strategic management journal, 34(8), 1010-1018.

21. Li, X., Alam, K. M., & Wang, S. (2018). Trend analysis of Pakistan railways based on industry life cycle theory. Journal of Advanced Transportation, 2018.

22. Lancien, D. (1990). Full-Scale ASTREE Tests Planned. IRJ November, 90.

23. Mohamed, H. A. R. (2014). A Proposed Model for Radio Frequency Systems to Tracking Trains via GPS (The Study for Egyptian National Railways). International Journal of Intelligent Systems and Applications, 6(4), 76.

24. Ministry of Railways (Ed.). (2021). Year Book 2020-2021 [Https://www.pakrail.gov.pk/]. Ministry of Railways. https://www.pakrail.gov.pk/images/yearbook/yearbook2020_21.pdf

25. Mahalakshmi, V., & Joseph, K. O. (2013). GPS based railway track survey system. IJCAES, 3.

26. Mohamed, H. A. R. (2014). A Proposed Model for Radio Frequency Systems to Tracking Trains via GPS (The Study for Egyptian National Railways). International Journal of Intelligent Systems and Applications, 6(4), 76.

27. Nahid, S., Padala, S., & Kumar, V. S. D. (2013). Design and development of train tracking system in south central railways. International Journal of Science and Modern Engineering, Vol1 (12), 60-64.

28. Statistics, F. B. (2017). Provisional summary results of 6th population and housing census. Islamabad: Pakistan Bureau Of Statistics, Ministry Of Statistics, Islamabad, Islamic Republic Of Pakistan.

29. Syed, W. H., Yasar, A., Janssens, D., & Wets, G. (2014). Analyzing the real time factors: which causing the traffic congestions and proposing the solution for Pakistani City. Procedia Computer Science, 32, 413-420.

30. Sharma, M., Sharma, V. D., & Bundele, M. M. (2018, November). Performance analysis of RDBMS and no SQL databases: PostgreSQL, MongoDB and Neo4j. In 2018 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE) (pp. 1-5). IEEE.

31. Sharygin, E. Y., Buchatskiy, R. A., Zhuykov, R. A., & Sher, A. R. (2017). Query compilation in PostgreSQL by specialization of the DBMS source code. Programming and Computer Software, 43, 353-365.

32. Stripple, H., & Uppenberg, S. (2010). Life cycle assessment of railways and rail transports-Application in environmental product declarations (EPDs) for the Bothnia Line.

33. Transport, R. (2015). Environment Facts & Figures. UIC/CER, Paris, 1-68.

34. Utterback, J. M., & Suárez, F. F. (1993). Innovation, competition, and industry structure. Research policy, 22(1), 1-21.

35. Vaisman, A., & Zimányi, E. (2019). Mobility data warehouses. ISPRS International Journal of Geo-Information, 8(4), 170.

36. Vershinin, I. S., & Mustafina, A. R. (2021, September). Performance analysis of PostgreSQL, MySQL, microsoft SQL server systems based on

TPC-H tests. In 2021 International Russian Automation Conference (RusAutoCon) (pp. 683-687). IEEE.

37. Yogiandra, A. F., & Widagdo, T. E. (2019, November). Handling of Spatio-Temporal Pattern Queries in Moving Object Database. In 2019 International Conference on Data and Software Engineering (ICoDSE) (pp. 1-6). IEEE.

38. Zimányi, E., Sakr, M., & Lesuisse, A. (2020). MobilityDB: A mobility database based on PostgreSQL and PostGIS. ACM Transactions on Database Systems (TODS), 45(4), 1-42.

39. Zimányi, E., Sakr, M., Lesuisse, A., & Bakli, M. (2019, August). Mobilitydb: A mainstream moving object database system. In Proceedings of the 16th International Symposium on Spatial and Temporal Databases (pp. 206-209).

40. Zimányi, E., Sakr, M., Bakli, M., Schomans, M., Tsesmelis, D., & Choquet, R. (2021, November). MobilityDB: hands on tutorial on managing and visualizing geospatial trajectories in SQL. In Proceedings of the 3rd ACM SIGSPATIAL International Workshop on APIs and Libraries for Geospatial Data Science (pp. 1-2).

Appendices

Appendix-1. Python script to capture raw API responses.

```
import urllib.request
from urllib.request import urlopen
from datetime import datetime
import csv
import pytz
import psycopg2
import psycopg2.extras
import time



#login credentials
hostname = 'localhost'
database = 'Dump_APIs_Resp'
username = 'postgres'
pwd = 'postgres'
port_id = 5432

conn = None

Iteration=0

#Fetchning Fresh SID
url =
'https://socket.pakraillive.com/socket.io/?EIO=3&transport=polling&t='
with urlopen(url) as response:
   html_response = response.read()
   encoding = response.headers.get_content_charset('utf-8')
   decoded_html = html_response.decode(encoding)
   sid = decoded_html[12:32]

#login to postgres databse

try:
   while True:

      try:
         with psycopg2.connect(
                     host = hostname,
                     dbname = database,
                     user = username,
                     password = pwd,
                     port = port_id) as conn:
            with conn.cursor(cursor_factory=psycopg2.extras.DictCursor) as
cur:

               cur.execute(
                  CREATE TABLE IF NOT EXISTS footprints (
```

Continue….

```
                    id serial not null,
                    time timestamp default current_timestamp,
                    train_id  VARCHAR ( 50 ),
                    lat  VARCHAR ( 50 ),
                    long  VARCHAR ( 50 ),
                    from_station  VARCHAR ( 50 ),
                    to_station VARCHAR ( 50 ),
                    last_updated  VARCHAR ( 50 ),
                    late_by  VARCHAR ( 50 ),
                    station  VARCHAR ( 50 ),
                    sp  VARCHAR ( 50 )
                    )
                 insert_script  = 'INSERT INTO footprints (sid, url, data)
VALUES (%s, %s, %s)'
             #while len(sid)==20:
             Time = datetime.now(pytz.timezone('Asia/Karachi'))
             Iteration = Iteration+1
             t = 'O4Qc14S'
             url =
'https://socket.pakraillive.com/socket.io/?EIO=3&transport=polling&t='+t+'
&sid='+sid
             API_Response = urllib.request.urlopen(url)
             data = API_Response.read(20000).decode('utf-8')

             print ("Iteration:", Iteration)
             print("Time =", t)
             print("sid: ", sid)
             #print("API Resp:", data)

             if len(data) < 20000:
                insert_values = [sid, url, data]
                cur.execute(insert_script, insert_values)
             time.sleep(5)
          #Fetchning Fresh SID for Next Iteration
             url =
'https://socket.pakraillive.com/socket.io/?EIO=3&transport=polling&t='
             with urlopen(url) as response:
                html_response = response.read()
                encoding = response.headers.get_content_charset('utf-8')
                decoded_html = html_response.decode(encoding)
                sid = decoded_html[12:32]
      except Exception as error:
         print(error)
      finally:
         if conn is not None:
            conn.close()
except KeyboardInterrupt:
   pass
print ("Script Terminated")
```

Appendix-2. Python script to sort raw footprints.

```
    import json
import psycopg2
import psycopg2.extras
import time
import winsound

frequency = 2500  # Set Frequency To 2500 Hertz
duration = 1000  # Set Duration To 1000 ms == 1 second



start_time = time.time()
"main()"



try:
    connection = psycopg2.connect(user="postgres",
                                  password="postgres",
                                  host="localhost",
                                  port="5432",
                                  database="trains_footprints")
    cursor = connection.cursor()
    postgreSQL_select_Query = "select * from footprints where id >= 31261
and id<= 100000 order by id asc"

    cursor.execute(postgreSQL_select_Query)
    print("Selecting rows from footprints table")
    mobile_records = cursor.fetchall()
    print("Fetched records from footprints table")
    #print("raw rows:",mobile_records )
except (Exception, psycopg2.Error) as error:
    print("Con 1: Error while fetching data from PostgreSQL", error)

finally:
    # closing database connection.
    if connection:
        cursor.close()
        connection.close()
        print("PostgreSQL connection is closed")

    ########Opening new database connection########
try:
    connection = psycopg2.connect(user="postgres",
                                  password="postgres",
                                  host="localhost",
                                  port="5432",
                                  database="trains_footprints")
    cursor = connection.cursor()
```

Continue….

```
  postgres_insert_query = """ INSERT INTO TrainWiseFootprints (rec_id,
rec_time, rec_sid, trip_id, lat, long, prev_st, next_st, station, sp, lst_updated,
lateby_m) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""


  for row in mobile_records:
     print("@@@@@@@@ Record:", row[0]," @@@@@@@@")
     #print("rec_id = ", row[0], )
     #print("rec_timestamp = ", row[1])
     #print("rec_sid  = ", row[2])
     m = row[4]
     #print("rec_rawdata  = ", m, "\n")

     t=1

     while "[" in m:
        start = m.index('[')
        end = m.index(']',start+1)
        substring = m[start:end+1]
        #print("this: ",substring)
        m=m[end+1:]
        data_json=json.loads(substring)
        print()
        print("//////////// Train#", t, " in Record ID: ",row[0]," ////////////")


        #print ("trip_id : " + data_json[0])
        jsd=data_json[1]
        if jsd is not None:
           #print("lat : " + jsd["lat"])
           #print("long : " + jsd["lon"])
           #print("last_updated : " + jsd["last_updated"])
           #print("late_by_min : " + jsd["late_by"])
           #print("next_st : " + jsd["next_st"])
           #print("Prev_st : " + jsd["prev_st"])
           #print("sp : " + jsd["sp"])
           #print("station : " + jsd["st"])
           t=t+1

           record_to_insert = (row[0], row[1], row[2], data_json[0],
jsd["lat"], jsd["lon"], jsd["prev_st"], jsd["next_st"], jsd["st"], jsd["sp"],
jsd["last_updated"], jsd["late_by"])
           cursor.execute(postgres_insert_query, record_to_insert)

           connection.commit()
           count = cursor.rowcount
           #print(count, "Record inserted successfully into mobile table")
```

Continue….

```
except (Exception, psycopg2.Error) as error:
    print(" Con 2: Error while fetching data from PostgreSQL", error)

finally:
    # closing database connection.
    if connection:
        cursor.close()
        connection.close()
        print("PostgreSQL connection is closed")
print("--- %s minutes consumed in filtering records ---" % ((time.time() -
start_time)/60))
winsound.Beep(frequency, duration)
```