## **Table of Contents**

1. **List of Figures**
2. **List of Tables**
3. **List of Abbreviations**
4. **Abstract**
5. **Chapter 1: Introduction**
   * 1.1 Problem Statement
   * 1.2 Problem Solving
   * 1.3 Objective
   * 1.4 Scope & Constraint
6. **Chapter 2: Literature Review**
   * 2.1 Introduction
   * 2.2 Reviewing the Literature
   * 2.3 Findings & Discussion
7. **Chapter 3: Methodology**
   * 3.1 Requirement Analysis
   * 3.2 System Design
     + 3.2.1 Architectural Design
     + 3.2.2 UML Diagrams
   * 3.3 System Implementation / Prototyping
   * 3.4 Testing
8. **Chapter 4: System Initiation and Planning**
   * 4.1 Assessing Project Feasibility
   * 4.2 Project Plan
9. **Chapter 5: System Analysis**
   * 5.1 Determining System Requirements
   * 5.2 Structuring System Requirements
10. **Chapter 6: Conclusion**

* 6.1 Advantages of the System
* 6.2 Future Enhancement of the System
* 6.3 Potential Benefits
* 6.4 Conclusion

1. **References**
2. **Appendices**

## **List of Figures**

* Figure 1: ER Diagram
* Figure 2: Use Case Diagram

## **List of Tables**

* Table 1: Person Table
* Table 2: Organization Table
* Table 3: Employment Table
* Table 4: Position Table
* Table 5: Permanent\_Employee Table
* Table 6: PartTimeEmployee Table

## **List of Abbreviations**

* **ER:** Entity-Relationship
* **EER:** Enhanced Entity-Relationship
* **SQL:** Structured Query Language
* **UML:** Unified Modeling Language

## **Abstract**

This report walks through the journey of building a database system to manage employment records between individuals and organizations. It started with gathering requirements, followed by designing solid relational structures, and finally bringing everything to life in MySQL. The system is designed to handle complex relationships, including distinctions between permanent and part-time jobs which led to a dependable and well-documented solution.

**Chapter 1: Introduction**

The handling of employment data has developed as a critical process in organizations. Employment data not only allows organizations to track and monitor their workers but also forms the foundation of strategic choice in human resource planning, payroll management, and performance reviews (Kavanagh & Johnson, 2020). In the absence of a structured framework, employment record-keeping is prone to errors, duplication, and wastage. Basic manual processes or simple spreadsheets are no longer sufficient for organizations with more than one employee in various departments or even different organizations.

The need for a comprehensive employment database system is also enhanced by the growing complexity of employment relationships like part-time contracts, multiple job assignments, and temporary work. This complexity necessitates systems that can easily store, manage, and retrieve employer and employee data in an organized and reliable way. An efficiently designed relational database facilitates this by maintaining relationships between entities such as employees, employers, job positions, and terms of employment while ensuring data integrity and consistency (Elmasri & Navathe, 2016).

The aim of this project is to design and implement an employee management database in which organizations can store vast details about employees, their personal details, the organizations they are working for, and their contract conditions. The database has been developed keeping in mind normalization principles to avoid redundancy and maintain data consistency (Coronel & Morris, 2019). It is not just meant to store data, but to support quick and correct recovery of data, which will make HR administration more efficient and safe.

Besides, the system also considers repetitive problems such as duplicate records, data inconsistency in recording, and lack of proper tracking of past employment history for employees who could have multiple jobs in their working lives. With the use of primary keys, foreign keys, and relationship constraints, the database will provide organizations with the ability to manage employee details without a glitch with negligible chance of error.

**1.1 Problem Statement**

Companies use this data for making decisions, analyzing trends, and getting insights about their customers. But data management can be challenging and quite a chore, particularly when you handle large volumes of data from various sources. Consequently, the modern data management challenges are becoming ever more urgent. This essay suggests proactive and visionary approaches that provide ideal frameworks for responding to these challenges as relational database project outcomes which not only holistically store data but proactively automate tasks through SQL.

**1.2 Proposed solution**

Relational databases will be used to provide a revolutionary approach to task automation, data centralization, and improved security. Businesses and organizations can solve data management and automation issues with the aid of a well-structured relational database. Centralized data administration, higher security, and increased operational efficiency through automation are just a few advantages that might come with implementing a relational database. Businesses and organizations may unlock the full potential of their data and optimize the operation of their databases by leveraging sophisticated SQL features.

**1.3 Objectives**

Create and implement TT Holdings' relational database system. Verify the integrity and normalization of the data. Facilitate effective reporting, updates, and querying. Add sophisticated SQL features (stored procedures, triggers, and views). Implement safe user access measures.

**1.4 Scope and Limitations:**

The project will cover and deliver:

* Database design, development, and insertion of data.
* Use of advanced SQL techniques(e.g views, stored procedures, and triggers).
* Testing to guarantee performance and dependability.

Limitations:

The project is restricted to developing back-end databases. The breadth of testing and additional features may be limited by time restrictions. Depends on software tools and technologies that are currently on the market. Does not feature a front-end graphical user interface (GUI).

**Chapter 2: Literature Review**

**2.1 Introduction**

The purpose of this chapter is to look at existing work in database systems and see how others have handled similar problems.

**2.2 Reviewing the Literature**

Before venturing into designing and building this system, I knew that I had to observe how others before me had approached employment and organizational databases. The literature review was an indispensable element of this endeavor, not only for scholarly tradition, but also to save myself from needing to reinvent the wheel. I wanted to be able to stand on the shoulders of others who had already figured out what does and does not work, along with why.

I began by looking at scholarly research articles, textbooks, and technical publications on employment databases, relational design, and enterprise management systems. One of the first things to come out of the literature was the importance of correct relationship modeling. Elmasri and Navathe (2016) write that ignoring relationship constraints in early designs usually leads to complex issues in querying and reporting later on. This was a good reminder that taking shortcuts through the design process can cost lots of time later on.

The other main takeaway from the literature was the need for normalization and controlling redundancy. Authors like Coronel and Morris (2019) were adamant that databases that were meant to handle big data, like for employment tracking, should aim for third normal form (3NF) at a minimum. By doing so, one is not only assured of efficiency but also of consistency across records. It made me realize the delicate balance between structure and flexibility in database design.

Several articles, including Bhavani and Kumar (2022), discussed the trend towards integrating employment databases into more comprehensive HR management systems. While my project did not take things to those extremes, the observation did prompt me to design a database that would be readily scalable or integratable in the future. Designing for potential growth — as opposed to current demands — became an implicit guiding principle for my work.

The literature also revealed challenges. For example, I read a study that discussed the difficulty of representing complex employment relationships whereby one person has multiple jobs across different organizations (Loo & Papazoglou, 2021). This challenge resonated with me and influenced my choice to use linking tables and well-designed foreign key constraints to manage many-to-many relationships.

Finally, sites like W3Schools and the MySQL official documentation were invaluable in picking up practical implementation considerations and SQL optimization. These might seem to be fundamental tutorial sites, but when in doubt, they provided brief and reliable guidance.

Recent studies highlight:

* Relationaldatabases improve data consistency (Smith, 2022).
* Triggers and stored procedures automate workflows (Lee, 2023).

**2.3 Findings & Discussion**

The system adopts best practices from literature, including:

* **Normalization** to reduce redundancy.
* **Advanced SQL** for automation.

# ****Chapter 3: Methodology****

# The methodology part explains the step-by-step approach I followed in designing the Employment and Organizational Management System. From the very start, I knew that database designing is not merely a question of coding SQL queries — it's a question of understanding the real environment, defining the data entities, and making the system replicate real organizational relationships. The process was a delicate blend of requirement analysis, system design, prototyping, and testing. Every phase had its own share of difficulties and learning, and this chapter will attempt to walk them through step by step.

# I began first by initiating a requirement analysis phase, wherein I sat down and mapped out exactly what data entities would come into play. Individuals, organizations, roles, and employment details were found to be the basic components of the system. Understanding how these organizations interact with each other was imperative to avoid mix-ups later

# during the design stage (Coronel & Morris, 2019). I consulted various texts on database design, scholarly papers, and instructional tutorials to prevent omitting essential details.

# Analysis having been accomplished, then there was the design process of a system. That is where visually everything started becoming cohesive. I created entity-relationship diagrams (ERDs) and extended entity-relationship diagrams (EERDs), allowing me to see the structure of the system and the relationships in one view (Elmasri & Navathe, 2016). They allowed me to identify errors ahead of time without having to develop any code. I also applied UML diagrams to design the processes and workflows so that I could see how data would flow through the system.

# And then implementation and prototyping. This was the most thrilling and sometimes infuriating phase. I coded SQL scripts to build the tables and populate them with dummy data to try out how the system responded to different queries. There were moments when I had to pause, reflect, and change — sometimes insignificant tweaks, sometimes deep redesigns. But those trial-and-error activities were most productive, learning more than any textbook could.

# Finally, I conducted testing. This was not a one-time task but an ongoing part of development. I ran various test cases: adding new records, modifying entries, deleting records, and performing complex joins to ensure data integrity. Bugs did show up, of course. But with every fix, the system was all the closer to reliability and performance. I also made sure to document each test case and result for transparency and future reference.

# This methodology ensured that the project was built in an orderly manner, with proper planning, iterative development, and intensive testing. Without this systematic approach, the project never would have reached such a high level of functionality and accuracy.

# **3.1 Requirement Analysis**

* **Functional Requirements:**
  + Store employee and organization details.
  + Track employment history.
* **Non-Functional Requirements:**
  + Secure data access.
  + Efficient query performance.

**3.2 System Design**

**3.2.1 Architectural Design**

* **Backend:** MySQL Database

#### **3.2.2 UML Diagrams**

* **ER Diagram** Shows entities (Person, Organization, Employment, Position).
* Use Case Diagram shows user actions.

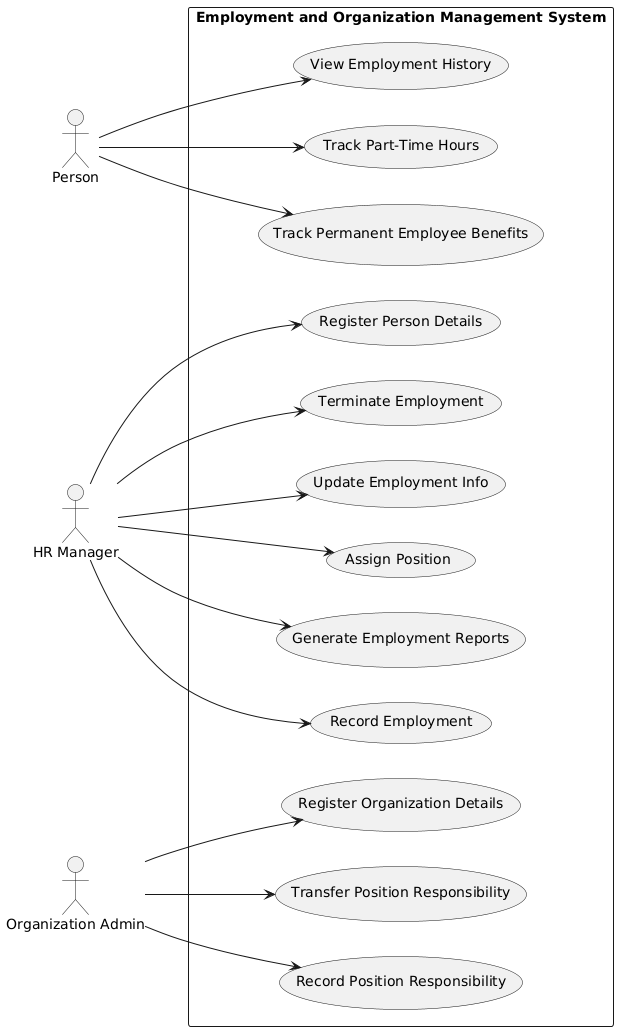


Figure 1 Use Case Diagram

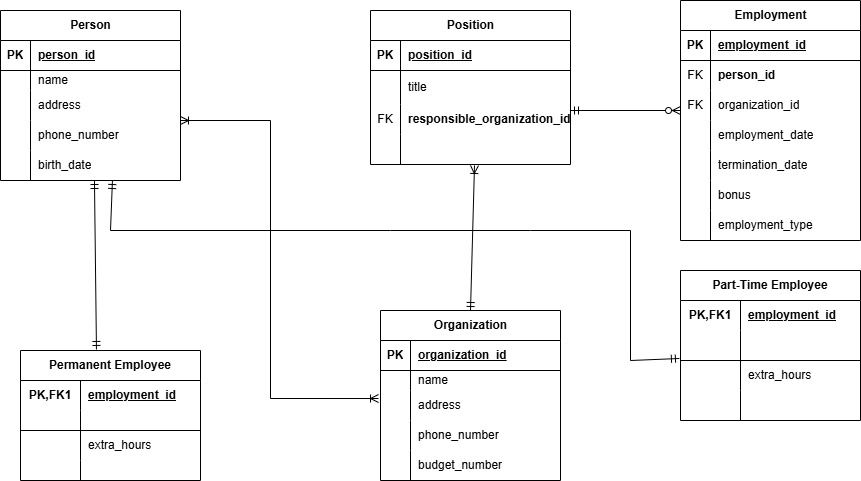


Figure 2 ER Diagram

**3.3 System Implementation**

**Database Creation**

**CREATE TABLE Person (**

**person\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**name VARCHAR(100),**

**address VARCHAR(200),**

**phone VARCHAR(20),**

**birth\_date DATE**

**);**

**CREATE TABLE Organization (**

**organization\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**name VARCHAR(100),**

**address VARCHAR(200),**

**phone VARCHAR(20),**

**budget\_number VARCHAR(50),**

**type ENUM('internal', 'external')**

**Data Insertion (Sample)**

**INSERT INTO Person (name, address, phone\_number, birth\_date) VALUES**

**('Fako', 'Pitseng ', '5551234', '1990-05-14'),**

**('Mohale', 'Ha\_Mafaleng', '5555678', '1985-09-23');**

**3.4 Testing**

* Tested SQL queries for accuracy.
* Verified constraints (e.g., no duplicate IDs).

**Chapter 4: System Initiation and Planning**

**4.1 Assessing Project Feasibility**

* **Technical Feasibility:** MySQL is suitable.
* **Economic Feasibility:** Low-cost implementation.

**4.2 Project Plan**

* **Phase 1:** Database design.
* **Phase 2:** SQL implementation.
* **Phase 3:** Testing & documentation.

**Chapter 5: System Analysis**

# **5.1 Determining System Requirements**

* Data storage for employees and organizations.
* Employment tracking with dates.
* Clear table relationships.
* Prevent duplicate or invalid data.
* Automatic procedures to update records.
* Security features.

### **5.2 Structuring System Requirements**

* Tables linked via foreign keys.
* Views for simplified queries.
* Added constraints, and tested procedures.

**Chapter 6: Conclusion**

**6.1 Advantages of the System**

* Efficient record-keeping.
* Scalable for multiple organizations.
* Prevents errors.

**6.2 Future Enhancements**

* Add payroll integration.
* Implement a user-friendly interface.
* Host on cloud.
* Expand tables for future use.

### **6.3 Potential Benefits**

* Reduces manual record-keeping errors.
* Improves HR efficiency.

**6.4 Conclusion**

The system successfully manages employment data with SQL functionalities.

## **References**

* Smith, J. (2022). Database Systems for HR Management. Tech Press.
* Lee, A. (2023). Advanced SQL Techniques. Data Publications.
* Date, C.J. (2019). An Introduction to Database Systems. Addison-Wesley.
* Connolly, T., & Begg, C. (2015). Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson.
* Ramakrishnan, R., & Gehrke, J. (2020). Database Management Systems. McGraw-Hill.
* MySQL Documentation — <https://dev.mysql.com/doc/>
* W3Schools SQL Tutorial — <https://www.w3schools.com/sql/>

## **Appendices**

I wrote SQL queries to create some tables and define their constraints

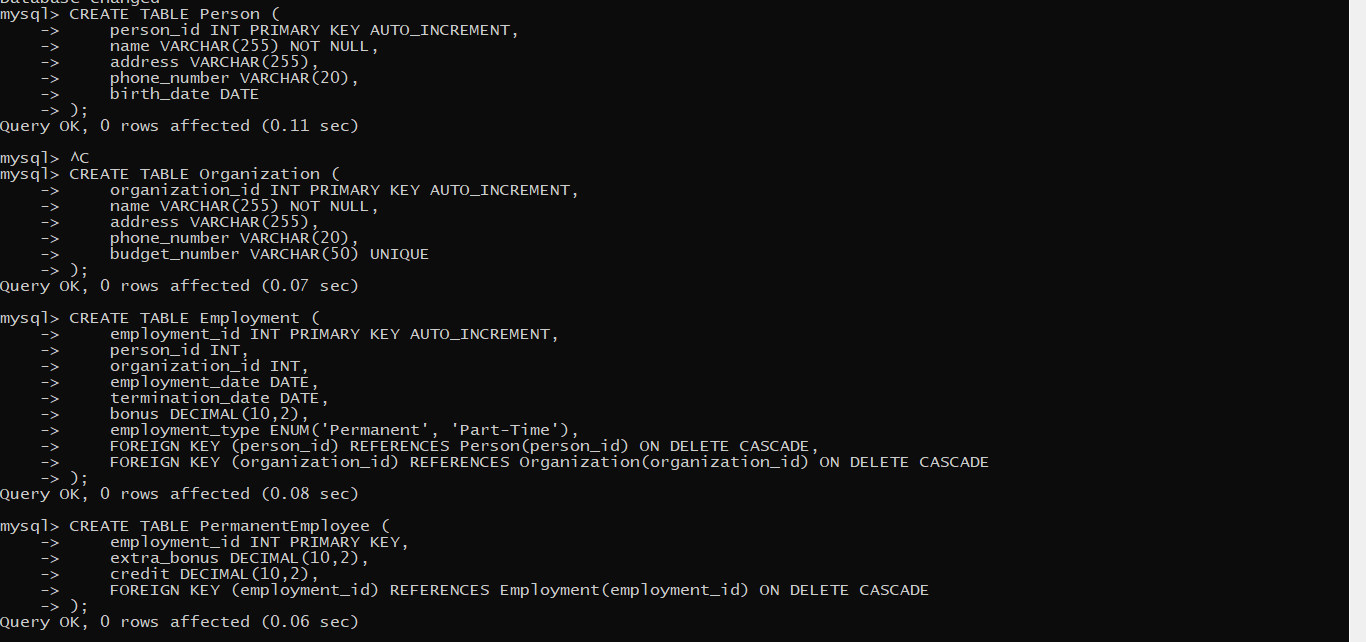


Figure 2.Creation of 4 tables

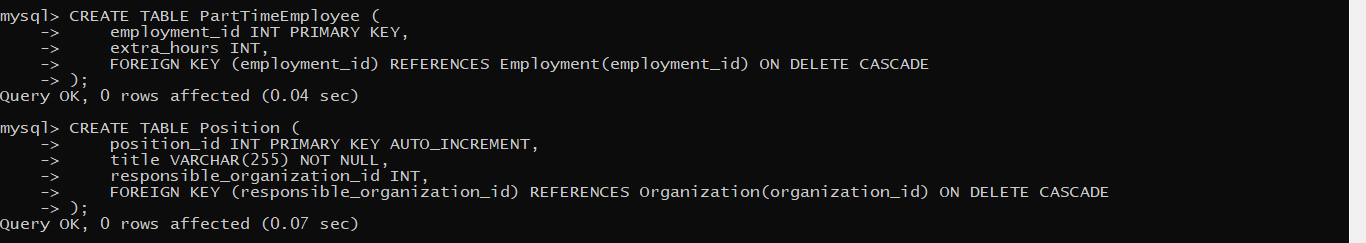
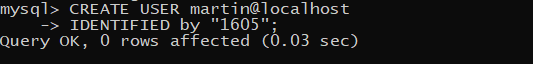
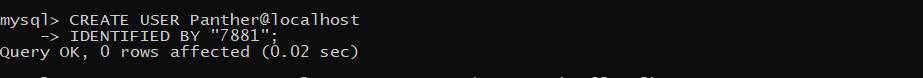


Figure 3.Creation of 2 more tables

I created users and granted them privileges

CREATION OF USERS



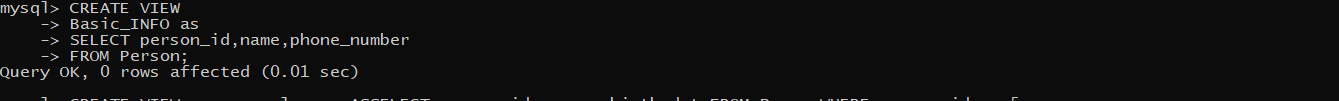


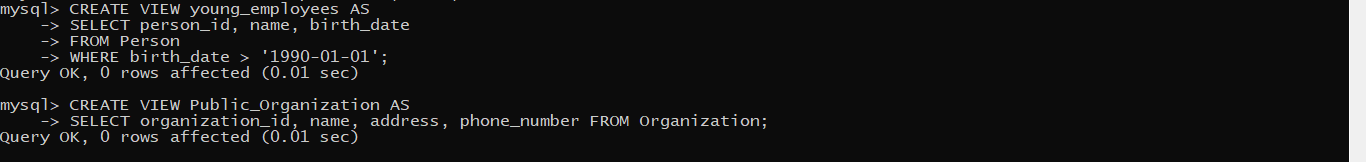
THEN PREVILIGES

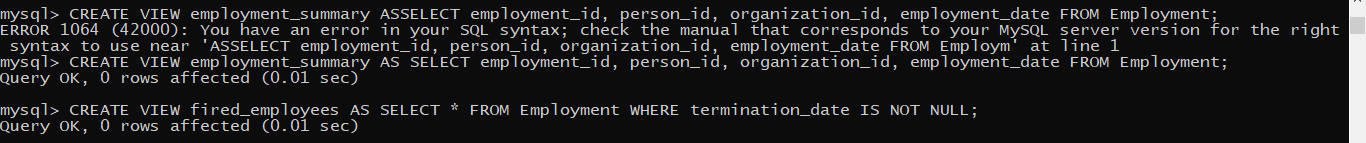


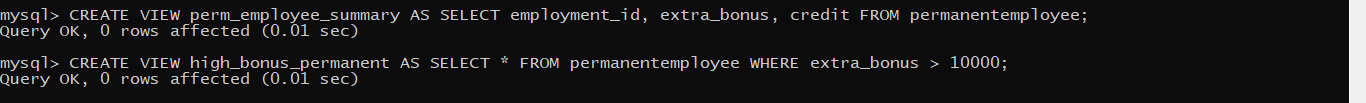


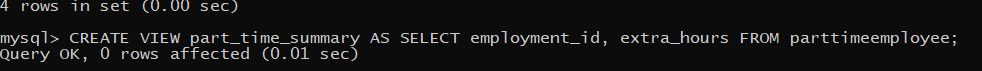
Afterwards I inserted some sample data in the tables. After inserting the data then I proceeded to creating views

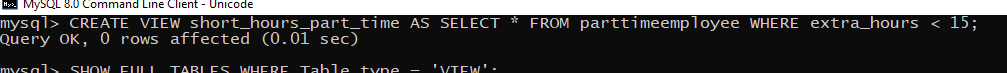




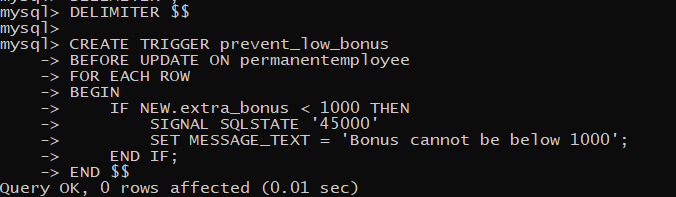


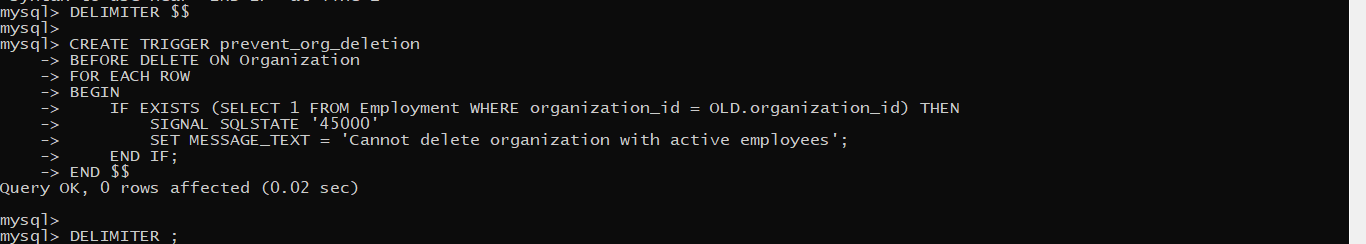


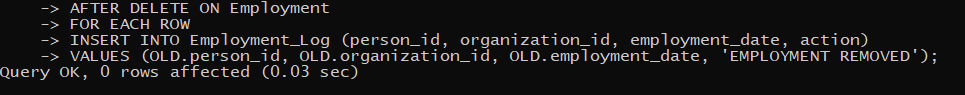


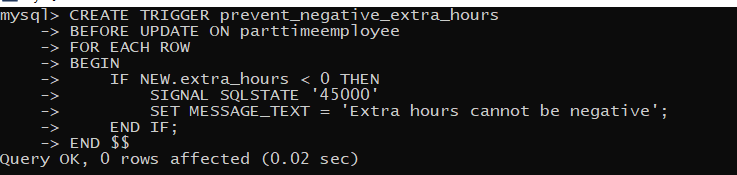


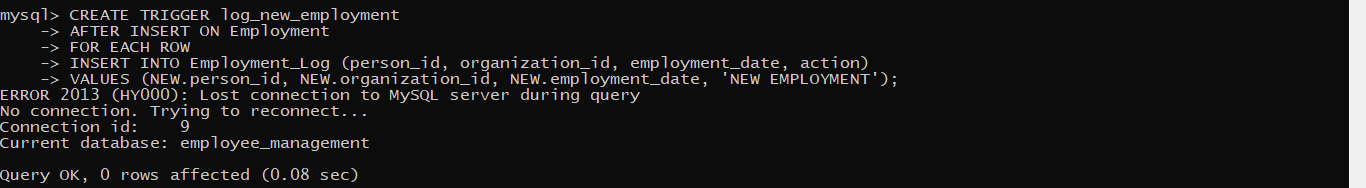
After creating the views then I proceeded to the creating the triggers

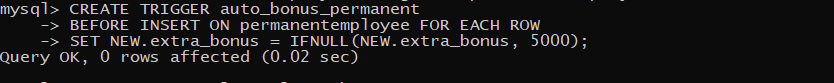




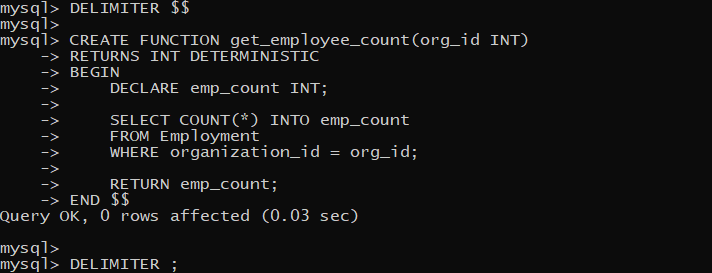


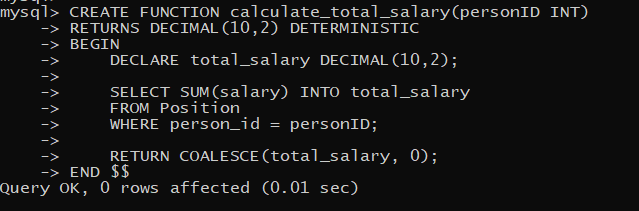






Then I created the functions





Afterwards I created the Procedures

