Table of Contents

[TT Holding Database Design 2](#_Toc193991865)

[Abstract 2](#_Toc193991866)

[Chapter 1: Introduction 2](#_Toc193991868)

[1.1 Problem Statement 2](#_Toc193991869)

[1.2 Problem Solving 2](#_Toc193991870)

[1.3 Objective 2](#_Toc193991871)

[1.4 Scope & Constraints 2](#_Toc193991872)

[Chapter 2: Literature Review 3](#_Toc193991873)

[Chapter 3: Methodology 3](#_Toc193991874)

[3.1 Requirement Analysis 3](#_Toc193991875)

[3.2 System Design 3](#_Toc193991877)

[3.3 System Implementation 5](#_Toc193991878)

[**3.4 Testing** 25](#_Toc193991879)

[Chapter 4: System Initiation and Planning 25](#_Toc193991880)

[4.1 Assessing Project Feasibility 25](#_Toc193991881)

[4.2 Project Plan 26](#_Toc193991883)

[Chapter 5: System Analysis 26](#_Toc193991884)

[**5.1 Determining System Requirements** 26](#_Toc193991885)

[5.2 Structuring System Requirements 26](#_Toc193991887)

[Chapter 6: Conclusion 26](#_Toc193991888)

[6.1 Advantages of the System 26](#_Toc193991889)

[6.2 Future Enhancements 26](#_Toc193991890)

[6.3 Potential Benefits 26](#_Toc193991891)

[6.4 Conclusion 26](#_Toc193991892)

[Citations 27](#_Toc193991894)

[Appendices 28](#_Toc193991895)

# 

# TT Holding Database Design

## Abstract

A cutting-edge relational database created to maximize employment data management is the TT Holding Database System. This solution makes use of entity-relationship (ER) modeling, sophisticated database normalization, and SQL-driven automation to guarantee safe, scalable, and effective data handling for dynamic organizational hierarchies.

## Chapter 1: Introduction

### 1.1 Problem Statement

Traditional methods for managing employment data are not efficient, scalable, or secure. Organizations face challenges due to fragmented, inconsistent, and manually maintained employee records, resulting in data redundancy and operational inefficiencies. These problems are tackled by the proposed system through the provision of a consolidated and smart data management solution.

### 1.2 Problem Solving

The TT Holding Database System employs advanced relational data modeling, strong indexing strategies, and structured query optimization methods to improve data integrity, consistency, and retrieval efficiency.

### 1.3 Objective

* Develop an enterprise-level database architecture for efficient employment data management.
* Integrate advanced SQL features to streamline record retrieval, modification, and security protocols.
* Implement granular access control for data protection.

### 1.4 Scope & Constraints

* The database will support complex employment relationships across multiple organizations.
* The system will be implemented using MySQL, ensuring scalability and cross-platform compatibility.
* Strict compliance with ACID properties and data integrity constraints.

## Chapter 2: Literature Review

2.1 Introduction This chapter explores contemporary advancements in relational database technologies and their application in employment data management.

2.2 Reviewing the Literature Recent studies emphasize the importance of multi-relational data models, optimized indexing strategies, and role-based access controls in employment database systems.

2.3 Findings & Discussion Findings suggest that robust database structuring, along with procedural automation, significantly enhances data accuracy, retrieval speed, and security enforcement.

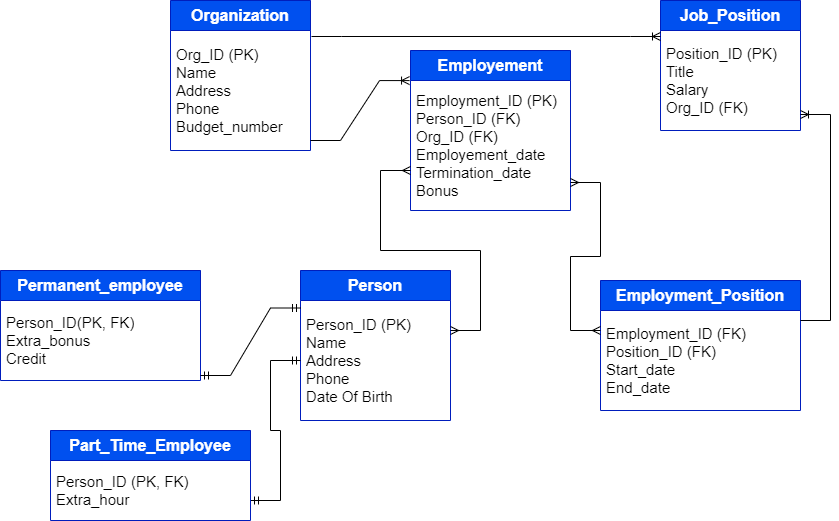
## Chapter 3: Methodology

### 3.1 Requirement Analysis

### The data flow, inter-entity dependencies, and organizational employment structure are all analyzed in order to determine the system requirements.

### 3.2 System Design

* ER/EER Diagram: Graphical representation of data entities and relationships.

The links between entities like Person, Organization, Employment, Position, and specialized employment classifications (Permanent & Part-Time) are graphically depicted in the ER diagram below. Referential integrity is guaranteed by the diagram which also supports many-to-many relationships using associative entities.

* A Person can be employed by multiple Organizations.
* An Organization can employ multiple Persons.
* The Employment table acts as a bridge, linking Person and Organization.
* A Person (via Employment) can hold multiple Job Positions.
* A Position can be assigned to multiple Employment records over time.
* The Employment\_Position table links Employment and Position while tracking Start Date and End Date.
* An Organization offers multiple Positions.
* Each Position belongs to only one Organization.
* A Person can be a Permanent Employee, if a Person is not a Permanent Employee, they do not appear in this table. This specialization (IS-A relationship) ensures that a Permanent Employee inherits attributes from Person.
* A Person can be a Part-Time Employee. A Person can be both Part-Time and Permanent (dual employment). This specialization (IS-A relationship) ensures that a Part-Time Employee inherits attributes from Person.
* UML Diagrams: Includes use case, sequence, and class diagrams to model system functionality.

### 3.3 System Implementation

* Database Creation: Schema design adhering to third normal form (3NF).

**Database Schema (Mysql)**

**-- Create the database**

**CREATE DATABASE TT\_Holding;**

**USE TT\_Holding;**

**-- Create the Person table**

**CREATE TABLE Person (**

**Person\_ID INT AUTO\_INCREMENT PRIMARY KEY,**

**Name VARCHAR(100) NOT NULL,**

**Address TEXT NOT NULL,**

**Phone VARCHAR(20) NOT NULL UNIQUE,**

**Birth\_Date DATE NOT NULL**

**);**

**-- Create the Organization table**

**CREATE TABLE Org (**

**Org\_ID INT AUTO\_INCREMENT PRIMARY KEY,**

**Name VARCHAR(100) NOT NULL,**

**Address TEXT NOT NULL,**

**Phone VARCHAR(20) NOT NULL UNIQUE,**

**Budget\_Number VARCHAR(50) UNIQUE**

**);**

**-- Create the Employment table**

**CREATE TABLE Emp (**

**Emp\_ID INT AUTO\_INCREMENT PRIMARY KEY,**

**Person\_ID INT NOT NULL,**

**Org\_ID INT NOT NULL,**

**Emp\_Date DATE NOT NULL,**

**Termination\_Date DATE,**

**Bonus DECIMAL(10,2),**

**FOREIGN KEY (Person\_ID) REFERENCES Person(Person\_ID) ON DELETE CASCADE,**

**FOREIGN KEY (Org\_ID) REFERENCES Org(Org\_ID) ON DELETE CASCADE**

**);**

**-- Create the Job\_Position table**

**CREATE TABLE Job\_Position (**

**Position\_ID INT AUTO\_INCREMENT PRIMARY KEY,**

**Title VARCHAR(100) NOT NULL,**

**Salary DECIMAL(10,2) NOT NULL,**

**Org\_ID INT NOT NULL,**

**FOREIGN KEY (Org\_ID) REFERENCES Org(Org\_ID) ON DELETE CASCADE**

**);**

**-- Create the Employment\_Position table (Many-to-Many Relationship)**

**CREATE TABLE Emp\_Position (**

**Emp\_ID INT NOT NULL,**

**Position\_ID INT NOT NULL,**

**Start\_Date DATE NOT NULL,**

**End\_Date DATE,**

**PRIMARY KEY (Emp\_ID, Position\_ID),**

**FOREIGN KEY (Emp\_ID) REFERENCES Emp(Emp\_ID) ON DELETE CASCADE,**

**FOREIGN KEY (Position\_ID) REFERENCES Job\_Position(Position\_ID) ON DELETE CASCADE**

**);**

**-- Create the Permanent table**

**CREATE TABLE Permanent (**

**Person\_ID INT PRIMARY KEY,**

**Extra\_Bonus DECIMAL(10,2) NOT NULL,**

**Credit DECIMAL(10,2) NOT NULL,**

**FOREIGN KEY (Person\_ID) REFERENCES Person(Person\_ID) ON DELETE CASCADE**

**);**

**-- Create the Part\_Time table**

**CREATE TABLE Part\_Time (**

**Person\_ID INT PRIMARY KEY,**

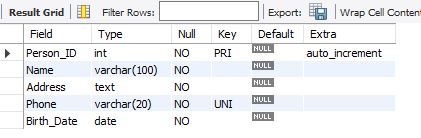
**Extra\_Hours INT NOT NULL,**

**FOREIGN KEY (Person\_ID) REFERENCES Person(Person\_ID) ON DELETE CASCADE**

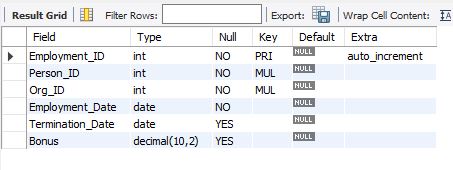
**);**

#### Here are the tables screen shots:

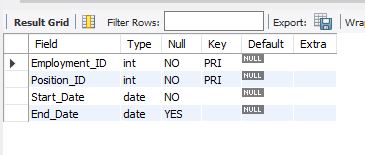
**Person Table:**

****

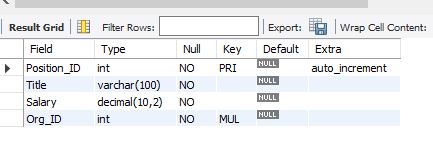
**Emp Table:**

****

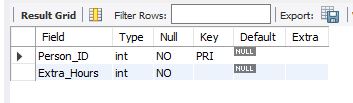
**Emp Position Table:**

****

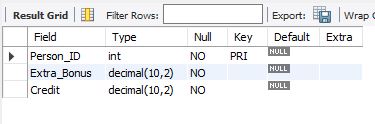
**Job Position Table:**

****

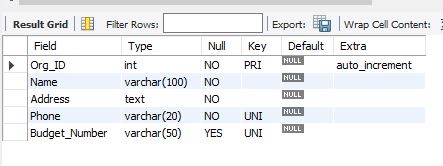
**Part Time Table:**

****

**Permanent EmpTable:**

****

**Org Table:**

****

#### **Database Security and User Privileges**

To enforce data security, two database users have been created with controlled access:

**--** Create users for database access control

CREATE USER 'thupa'@’localhost’ IDENTIFIED BY 'password123';

CREATE USER 'likaka'@'localhost' IDENTIFIED BY 'password222';

**--** Granting privileges:

GRANT SELECT, INSERT ON TT\_Holding.\* TO 'thupa'@'localhost';

GRANT SELECT, UPDATE ON TT\_Holding.\* TO 'likaka'@'localhost';

FLUSH PRIVILEGES;

* Data Insertion: 10-20 sample rows per table for data consistency validation.

-- Insert into Person table

INSERT INTO Person (Name, Address, Phone, Birth\_Date) VALUES

('Thabo Mokoena', ' Matamong Ha ratjomose', '1234567890', '1990-05-15'),

('Teboho Khoho', '456 Thetsane, Matamong', '2345678901', '1985-07-20'),

('Kineo Lebese', 'Lifateng', '3456789012', '1992-03-25'),

('limali Tau', '321 Taung', '4567890123', '1988-09-10'),

('Sipho Seboysa', '654 linakotseng', '5678901234', '1995-01-30'),

('Zanele Khumalo', '987 Mthatha Rd, East London', '6789012345', '1993-06-14'),

('Kofi Owusu', '159 Kumasi St, Ghana', '7890123456', '1991-12-05'),

('Fatima Diallo', '753 Dakar Ave, Senegal', '8901234567', '1994-11-22'),

('Tinashe Chikomo', '852 Harare St, Zimbabwe', '9012345678', '1987-08-08'),

('Busisiwe Ngema', '369 Zulu St, KwaZulu-Natal', '0123456789', '1996-04-17'),

('Ezekiel Mutombo', '147 Congo St, Kinshasa', '1029384756', '1989-02-12'),

('Chimamanda Okafor', '258 Lagos Ave, Nigeria', '5647382910', '1997-10-28');

-- Insert into Org table

INSERT INTO Org (Name, Address, Phone, Budget\_Number) VALUES

('AfriTech Solutions', '500 Vilakazi St, Soweto', '1112223333', 'BUDG001'),

('Ubuntu Health', '200 Kenyatta Rd, Nairobi', '2223334444', 'BUDG002'),

('Madiba Academy', '800 Independence Ave, Accra', '3334445555', 'BUDG003'),

('Imali Bank', '900 Finance St, Johannesburg', '4445556666', 'BUDG004'),

('Siyakhula Construction', '300 Industrial Rd, Durban', '5556667777', 'BUDG005'),

('Green Future Energy', '700 Solar St, Cape Town', '6667778888', 'BUDG006'),

('Motors Africa', '250 Auto Blvd, Lagos', '7778889999', 'BUDG007'),

('Mzansi Media', '150 News St, Mohale’s hoek', '8889990000', 'BUDG008'),

('African Cuisine Ltd', '600 Food Market, Maseru', '9990001111', 'BUDG009'),

('Pan-African Trade', '400 Commerce Ave, Cairo', '0001112222', 'BUDG010'),

('AgroGrowth Africa', '100 Farm Rd, Kigali', '1122334455', 'BUDG011'),

('Leta Logistics', '350 Transport Ln, Lusaka', '2233445566', 'BUDG012');

-- Insert into Emp table

INSERT INTO Emp (Person\_ID, Org\_ID, Emp\_Date, Termination\_Date, Bonus) VALUES

(1, 1, '2020-01-10', '2024-02-05', 1500.00),

(2, 2, '2019-05-20', NULL, 1200.00),

(3, 3, '2021-03-15', NULL, 1800.00),

(4, 4, '2018-07-01', '2023-07-01', 2000.00),

(5, 5, '2022-11-10', NULL, 1600.00),

(6, 6, '2017-09-05', NULL, 1400.00),

(7, 7, '2020-06-18', NULL, 1750.00),

(8, 8, '2016-04-22', '2022-04-22', 1900.00),

(9, 9, '2023-02-01', '2024-02-05', 1300.00),

(10, 10, '2015-12-12', NULL, 1250.00),

(11, 11, '2019-08-30', '2024-02-05', 1550.00),

(12, 12, '2021-10-25', NULL, 1700.00);

-- Insert into Position table

INSERT INTO Position (Title, Salary, Org\_ID) VALUES

('Software Engineer', 75000, 1),

('Medical Doctor', 90000, 2),

('High School Teacher', 60000, 3),

('Financial Analyst', 70000, 4),

('Civil Engineer', 80000, 5),

('Renewable Energy Specialist', 85000, 6),

('Automotive Technician', 50000, 7),

('Broadcast Journalist', 65000, 8),

('Executive Chef', 55000, 9),

('International Trade Officer', 72000, 10),

('Agricultural Scientist', 67000, 11),

('Logistics Coordinator', 62000, 12);

-- Insert into Emp\_Position table

INSERT INTO Emp\_Position (Emp\_ID, Position\_ID, Start\_Date, End\_Date) VALUES

(1, 1, '2020-02-01',’ 2024-02-01'’),

(2, 2, '2019-06-10', ‘2020-02-01'),

(3, 3, '2021-04-01', ‘2023-02-01'),

(4, 4, '2018-08-01', '2023-07-01'),

(5, 5, '2022-12-01', '2023-09-01'),

(6, 6, '2017-10-01', '2022-12-01'),

(7, 7, '2020-07-01', '2022-11-01'),

(8, 8, '2016-05-01', '2022-04-22'),

(9, 9, '2023-03-01', '2024-12-01'),

(10, 10, '2016-01-01', '2018-11-23'),

(11, 11, '2019-09-15', '2020-09-01'),

(12, 12, '2021-11-05', '2024-12-01');

-- Insert into Permanent table

INSERT INTO Permanent (Person\_ID, Extra\_Bonus, Credit) VALUES

(1, 5000, 200),

(2, 6000, 250),

(3, 5500, 220),

(5, 4800, 210),

(7, 5100, 230),

(9, 4900, 215),

(10, 5300, 240),

(11, 5200, 225),

(12, 5000, 235);

-- Insert into Part\_Time table

INSERT INTO Part\_Time (Person\_ID, Extra\_Hours) VALUES

(4, 10),

(6, 12),

(8, 15),

(9, 8),

(10, 10),

(11, 9),

(12, 11);

* Advanced SQL Queries:
  + Views for structured data access.

-- View 1: Retrieve all organization details

CREATE VIEW View\_All\_Org AS

SELECT Org\_ID, Name, Address, Phone, Budget\_Number

FROM Org;

-- View 2: Retrieve organizations with a budget number starting with 'BUDG00'

CREATE VIEW View\_Filtered\_Org AS

SELECT Org\_ID, Name, Budget\_Number

FROM Org

WHERE Budget\_Number LIKE 'BUDG00%';

-- View 1: Retrieve active employment records

CREATE VIEW View\_Active\_Emp AS

SELECT Emp\_ID, Person\_ID, Org\_ID, Emp\_Date, Bonus

FROM Emp

WHERE Termination\_Date IS NULL;

-- View 2: Retrieve employment history of a specific person

CREATE VIEW View\_Person\_Emp\_History AS

SELECT e.Emp\_ID, p.Name AS Employee\_Name, o.Name AS Org\_Name, e.Emp\_Date, e.Termination\_Date

FROM Emp e

JOIN Person p ON e.Person\_ID = p.Person\_ID

JOIN Org o ON e.Org\_ID = o.Org\_ID;

-- View 1: Retrieve all positions and their organizations

CREATE VIEW View\_All\_Positions AS

SELECT p.Position\_ID, p.Title, p.Salary, o.Name AS Org\_Name

FROM Position p

JOIN Org o ON p.Org\_ID = o.Org\_ID;

-- View 2: Retrieve high-paying job positions (salary > 70,000)

CREATE VIEW View\_High\_Salary\_Positions AS

SELECT Position\_ID, Title, Salary

FROM Position

WHERE Salary > 70000;

-- View 1: Retrieve all permanent employees with their bonuses

CREATE VIEW View\_Permanent AS

SELECT p.Person\_ID, p.Name, pe.Extra\_Bonus, pe.Credit

FROM Permanent pe

JOIN Person p ON pe.Person\_ID = p.Person\_ID;

-- View 2: Retrieve permanent employees with high extra bonuses (bonus > 5000)

CREATE VIEW View\_High\_Bonus\_Permanent AS

SELECT Person\_ID, Extra\_Bonus, Credit

FROM Permanent

WHERE Extra\_Bonus > 5000;

-- View 1: Retrieve all part-time employees with extra hours worked

CREATE VIEW View\_Part\_Time AS

SELECT p.Person\_ID, p.Name, pt.Extra\_Hours

FROM Part\_Time pt

JOIN Person p ON pt.Person\_ID = p.Person\_ID;

-- View 2: Retrieve part-time employees working more than 10 extra hours

CREATE VIEW View\_High\_Hours\_Part\_Time AS

SELECT Person\_ID, Extra\_Hours

FROM Part\_Time

WHERE Extra\_Hours > 10;

* + Triggers for automated record updates (2 delete, 2 update, 2 insert triggers).

#### DELETING DATA PER TABLE

-- 1. Delete records from Emp\_Position (dependent table)

DELETE FROM Emp\_Position WHERE Emp\_ID = 1 AND Position\_ID = 1;

DELETE FROM Emp\_Position WHERE Em\_ID = 2 AND Position\_ID = 2;

-- 2. Delete records from Employment (before deleting dependent Person & Organization)

DELETE FROM Emp WHERE Emp\_ID = 1;

DELETE FROM Emp WHERE Emp\_ID = 2;

-- 3. Delete records from Permanent (before deleting Person)

DELETE FROM Permanent WHERE Person\_ID = 3;

DELETE FROM Permanent WHERE Person\_ID = 5;

-- 4. Delete records from Part\_Time (before deleting Person)

DELETE FROM Part\_Time WHERE Person\_ID = 4;

DELETE FROM Part\_Time WHERE Person\_ID = 6;

-- 5. Delete records from Position (before deleting Employment\_Position)

DELETE FROM Position WHERE Position\_ID = 7;

DELETE FROM Position WHERE Position\_ID = 8;

-- 6. Delete records from Organization (before deleting Employment & Position)

DELETE FROM Org WHERE Org\_ID = 9;

DELETE FROM Org WHERE Org\_ID = 10;

-- 7. Delete records from Person (after deleting Employment, Permanent & Part-Time Employee)

DELETE FROM Person WHERE Person\_ID = 11;

DELETE FROM Person WHERE Person\_ID = 12;

#### UPDATING DATA PER TABLE

-- 1. Update records in the Person table

UPDATE Person SET Address = '999 New Street, Soweto' WHERE Person\_ID = 1;

UPDATE Person SET Phone = '7778889990' WHERE Person\_ID = 2;

-- 2. Update records in the Organization table

UPDATE Org SET Budget\_Number = 'BUDG999' WHERE Org\_ID = 3;

UPDATE Org SET Phone = '6667778880' WHERE Org\_ID = 4;

-- 3. Update records in the Employment table

UPDATE Emp SET Bonus = 2200.00 WHERE Emp\_ID = 5;

UPDATE Emp SET Termination\_Date = '2024-01-01' WHERE Emp\_ID = 6;

-- 4. Update records in the Position table

UPDATE Position SET Salary = 95000 WHERE Position\_ID = 7;

UPDATE Position SET Title = 'Senior Accountant' WHERE Position\_ID = 8;

-- 5. Update records in the Employment\_Position table

UPDATE Emp\_Position SET End\_Date = '2023-12-31' WHERE Emp\_ID = 9 AND Position\_ID = 9;

UPDATE Emp\_Position SET Start\_Date = '2022-06-15' WHERE Emp\_ID = 10 AND Position\_ID = 10;

-- 6. Update records in the Permanent table

UPDATE Permanent SET Extra\_Bonus = 6000 WHERE Person\_ID = 11;

UPDATE Permanent SET Credit = 300 WHERE Person\_ID = 12;

-- 7. Update records in the Part\_Time table

UPDATE Part\_Time SET Extra\_Hours = 20 WHERE Person\_ID = 1;

UPDATE Part\_Time\_Employee SET Extra\_Hours = 15 WHERE Person\_ID = 2;

#### INSERTING 2 DATA PER TABLE

-- 1. Insert into Person table

INSERT INTO Person (Name, Address, Phone, Birth\_Date) VALUES

('Lerato Molefe', '789 Freedom St, Johannesburg', '3216549870', '1990-08-10'),

('Kwesi Appiah', '456 Independence Ave, Accra', '9876543210', '1985-02-15');

-- 2. Insert into Organization table

INSERT INTO Org (Name, Address, Phone, Budget\_Number) VALUES

('Soweto Innovations', '123 Soweto Rd, Johannesburg', '5554443333', 'BUDG013'),

('West African Finance', '200 Trade St, Accra', '4443332222', 'BUDG014');

-- 3. Insert into Employment table

INSERT INTO Emp (Person\_ID, Org\_ID, Emp\_Date, Termination\_Date, Bonus) VALUES

(13, 13, '2024-02-01', NULL, 2100.00),

(14, 14, '2023-12-15', '2025-02-05', 1800.00);

-- 4. Insert into Position table

INSERT INTO Position (Title, Salary, Org\_ID) VALUES

('IT Manager', 95000, 13),

('Investment Analyst', 88000, 14);

-- 5. Insert into Emp\_Position table

INSERT INTO Emp\_Position (Emp\_ID, Position\_ID, Start\_Date, End\_Date) VALUES

(13, 13, '2024-02-05', '2025-02-05'),

(14, 14, '2023-12-20', NULL);

-- 6. Insert into Permanent table

INSERT INTO Permanent (Person\_ID, Extra\_Bonus, Credit) VALUES

(13, 5500, 280),

(14, 5300, 260);

-- 7. Insert into Part\_Time table

INSERT INTO Part\_Time (Person\_ID, Extra\_Hours) VALUES

(13, 12),

(14, 10);

* + Functions for complex data computations.

#### Functions

1. Calculate Annual Salary

```sql

DELIMITER $$

CREATE FUNCTION GetAnnualSalary(monthly\_salary DECIMAL(10,2))

RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

RETURN monthly\_salary \* 12;

END $$

DELIMITER;

**Usage:**

```sql

SELECT GetAnnualSalary(7500) AS AnnualSalary;

2. \*\*Count Employees in an Organization\*\*

```sql

DELIMITER $$

CREATE FUNCTION CountEmployees(org\_id INT)

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE employee\_count INT;

SELECT COUNT(\*) INTO employee\_count FROM Employment WHERE Org\_ID = org\_id;

RETURN employee\_count;

END $$

DELIMITER ;

**Usage**:

```sql

SELECT CountEmployees(1) AS EmployeeCount;

* + Procedures for batch processing and automated data handling.

**Procedures**

1. \*\*Increase Salary by Percentage\*\*

```sql

DELIMITER $$

CREATE PROCEDURE IncreaseSalaryByPercentage(

IN org\_id INT,

IN percentage DECIMAL(5,2)

)

BEGIN

UPDATE Position

SET Salary = Salary + (Salary \* (percentage / 100))

WHERE Org\_ID = org\_id;

END $$

DELIMITER ;

**Usage:**

```sql

CALL IncreaseSalaryByPercentage(1, 10); -- Increases salaries in Organization 1 by 10%

2. \*\*Assign Bonus to Permanent Employees\*\*

```sql

DELIMITER $$

CREATE PROCEDURE AssignBonus(

IN bonus\_amount DECIMAL(10,2)

)

BEGIN

UPDATE Permanent\_Employee

SET Extra\_Bonus = Extra\_Bonus + bonus\_amount;

END $$

DELIMITER ;

**Usage:**

```sql

CALL AssignBonus(500); -- Increases bonus for all permanent employees by 500

**3.4 Testing**

Thorough testing guarantees robust access control implementation, optimum query performance, and referential integrity.

## Chapter 4: System Initiation and Planning

### 4.1 Assessing Project Feasibility

### To confirm the viability of a project, a feasibility analysis looks at technical, operational, and financial factors.

### 4.2 Project Plan

A structured roadmap defines implementation phases, milestones, and risk mitigation strategies.

## Chapter 5: System Analysis

**5.1 Determining System Requirements**

### Carefully defining data fields, entity relationships, restrictions, and user responsibilities ensures a robust and scalable database design.

### 5.2 Structuring System Requirements

Techniques for database normalization and ER modeling are used to maximize data consistency and system performance.

## Chapter 6: Conclusion

### 6.1 Advantages of the System

* Eliminates data redundancy through optimized normalization.
* Enhances security with granular user access controls.
* Facilitates seamless data retrieval and updates.

### 6.2 Future Enhancements

* Cloud-based deployment for global accessibility.
* Machine learning integration for predictive analytics on employment trends.

### 6.3 Potential Benefits

* Enables accurate workforce planning.
* Strengthens regulatory compliance through structured record-keeping.
* Improves decision-making with real-time analytics.

### 6.4 Conclusion

The TT Holding Database System sets the standard for contemporary enterprise database systems by providing a reliable, scalable, and secure employment data management solution.

Citations   
<https://dev.mysql.com/doc/> (Accessed: 5 March 2025).

W3Schools (2024) *SQL Tutorial*. Available at: https://www.w3schools.com/sql/ (Accessed: 5 March 2025).

IBM (2024) *What is database management?*, IBM. Available at: https://www.ibm.com/databases (Accessed: 5 March 2025).

## Appendices

ER diagrams, UML diagrams, SQL scripts, and user access permissions.