# 

# ****Project Report On****

**Company Management System (MySQL + PL/SQL Project)**

### ****SUBMITTED BY:****

### Mahamat Kher UID: 25MCD10068 CLASS: MCD (Data Science) SECTION: MCD-A 1

### ****SUBMITTED TO:****

**Dr. Gagandeep Kaur**  
Associate Professor

## ****Table of Content****

| **S. No.** | **Name** | **Page No.** |
| --- | --- | --- |
| 1 | Acknowledgement | 3 |
| 2 | Introduction / Background | 4 |
| 3 | Objective / Technologies Used | 4 |
| 4 | Technologies Used | 4 |
| 5 | Requirements and System Design | 5–7 |
| 6 | Source Code & Output | 8–15 |
| 7 | Functionalities / Conclusion / Future Work / Learning Outcomes | 16 |
| 8 | GitHub Link / Bibliography | 16 |

## ****1. Acknowledgement****

I would like to express my sincere gratitude to all those who guided and supported me in the successful completion of my project, **“Company Management System.”**

I am deeply thankful to my mentor, **Dr. Gagandeep Kaur**, Associate Professor, for her continuous guidance, encouragement, and valuable feedback throughout the project. Her expertise in database management and structured programming has been instrumental in shaping this project’s technical foundation.

I am equally grateful to my classmates and peers for their constructive discussions and ideas during the development process. Finally, I acknowledge the resources and open-source communities that have helped me learn and apply **MySQL** and **PL/SQL** effectively.

This project represents a small yet significant step in applying theoretical database concepts to real-world business scenarios.

## ****2. Introduction****

The **Company Management System** is a database-driven application designed to streamline the management of a company’s operations — including **managers, employees, products, customers, and sales transactions**.

The project focuses on creating a structured relational database that ensures **data integrity**, **automation through PL/SQL**, and **efficient reporting**. It simulates a real-world organizational structure where managers oversee employees who handle product sales to customers.

This system eliminates manual data redundancy, ensures consistency, and enables quick retrieval of critical business information.

## ****3. Objectives****

* To design a normalized relational database for company operations.
* To define relationships between **managers, employees, products, customers, and sales**.
* To apply **SQL and PL/SQL** concepts for automation (procedures and triggers).
* To generate insightful queries such as total sales per manager and top-selling product.
* To demonstrate database design principles and real-time data manipulation.

## ****4. Technologies Used****

| **Technology** | **Description** |
| --- | --- |
| **MySQL** | For database creation, manipulation, and relational integrity. |
| **PL/SQL** | For implementing stored procedures and triggers. |
| **SQL Queries** | For performing data operations and business analysis. |
| **MySQL Workbench / XAMPP** | For database testing and visualization. |

## ****5. Requirements and System Design****

### ****Hardware Requirements****

* Processor: Intel i3 or above
* RAM: 4 GB minimum
* Storage: 500 MB free space
* Operating System: Windows / Linux / macOS

### ****Software Requirements****

* MySQL Server or XAMPP
* MySQL Workbench
* Text Editor (VS Code / Notepad++) If you want to use UGI

### ****System Design****

**Database Name:** company\_db

**Tables:**

* Managers
* Employees
* Products
* Customers
* Sales
* Sale\_Details

#### ****Architecture Overview****

The project follows a **relational database model** with referential integrity through **primary and foreign keys**. It also includes **PL/SQL procedures** and **triggers** for automation.

#### ****Entity–Relationship Diagram****

Managers(Manager\_ID, Manager\_Name, Department, Phone, Salary)

|

|<-- one-to-many

Employees(Emp\_ID, Emp\_Name, Position, Salary, Phone, Manager\_ID)

|

|<-- one-to-many

Sales(Sale\_ID, Cust\_ID, Emp\_ID, Sale\_Date, Total\_Amount)

| |

| |-- many-to-one --> Customers(Cust\_ID, Cust\_Name, City, Phone)

|

|<-- one-to-many

Sale\_Details(Sale\_ID, Prod\_ID, Quantity, Subtotal)

|

|-- many-to-one --> Products(Prod\_ID, Prod\_Name, Category, Price, Stock)

## ****6. Source Code & Output****

## **# Database Creation in MySQL**

## **CREATE DATABASE company\_db;**

## **USE company\_db;**

## **#Table Creation**

## **# Managers**

## **CREATE TABLE Managers (**

## **Manager\_ID INT PRIMARY KEY,**

## **Manager\_Name VARCHAR(50),**

## **Department VARCHAR(30),**

## **Phone VARCHAR(15),**

## **Salary DECIMAL(10,2)**

## **);**

## **# Employees**

## **CREATE TABLE Employees (**

## **Emp\_ID INT PRIMARY KEY,**

## **Emp\_Name VARCHAR(50),**

## **Position VARCHAR(30),**

## **Salary DECIMAL(10,2),**

## **Phone VARCHAR(15),**

## **Manager\_ID INT,**

## **FOREIGN KEY (Manager\_ID) REFERENCES Managers(Manager\_ID)**

## **);**

## **# Products**

## **CREATE TABLE Products (**

## **Prod\_ID INT PRIMARY KEY,**

## **Prod\_Name VARCHAR(50),**

## **Category VARCHAR(30),**

## **Price DECIMAL(10,2),**

## **Stock INT**

## **);**

## **# Customers**

## **CREATE TABLE Customers (**

## **Cust\_ID INT PRIMARY KEY,**

## **Cust\_Name VARCHAR(50),**

## **City VARCHAR(30),**

## **Phone VARCHAR(15)**

## **);**

## **# Sales**

## **CREATE TABLE Sales (**

## **Sale\_ID INT PRIMARY KEY,**

## **Cust\_ID INT,**

## **Emp\_ID INT,**

## **Sale\_Date DATE,**

## **Total\_Amount DECIMAL(10,2),**

## **FOREIGN KEY (Cust\_ID) REFERENCES Customers(Cust\_ID),**

## **FOREIGN KEY (Emp\_ID) REFERENCES Employees(Emp\_ID)**

## **);**

## **# Sale\_Details**

## **CREATE TABLE Sale\_Details (**

## **Sale\_ID INT,**

## **Prod\_ID INT,**

## **Quantity INT,**

## **Subtotal DECIMAL(10,2),**

## **FOREIGN KEY (Sale\_ID) REFERENCES Sales(Sale\_ID),**

## **FOREIGN KEY (Prod\_ID) REFERENCES Products(Prod\_ID)**

## **);**

## **# Insert Sample Data**

## **-- Managers**

## **INSERT INTO Managers VALUES**

## **(1, 'Ali Khan', 'Sales', '9876543210', 85000.00),**

## **(2, 'Sara Mehta', 'Production', '9876501234', 90000.00);**

## **-- Employees**

## **INSERT INTO Employees VALUES**

## **(101, 'Ahmed Hassan', 'Sales Executive', 45000.00, '9000011111', 1),**

## **(102, 'Fatima Noor', 'Sales Assistant', 35000.00, '9000022222', 1),**

## **(103, 'Omar Rahman', 'Production Officer', 40000.00, '9000033333', 2);**

## **-- Products**

## **INSERT INTO Products VALUES**

## **(201, 'Laptop', 'Electronics', 55000.00, 10),**

## **(202, 'Mouse', 'Accessories', 500.00, 50),**

## **(203, 'Keyboard', 'Accessories', 1000.00, 30);**

## **-- Customers**

## **INSERT INTO Customers VALUES**

## **(301, 'John Smith', 'Delhi', '7000011111'),**

## **(302, 'Priya Singh', 'Mumbai', '7000022222');**

## **-- Sales**

## **INSERT INTO Sales VALUES**

## **(401, 301, 101, '2025-10-01', 56000.00),**

## **(402, 302, 102, '2025-10-02', 1050.00);**

## **-- Sale Details**

## **INSERT INTO Sale\_Details VALUES**

## **(401, 201, 1, 55000.00),**

## **(401, 202, 2, 1000.00),**

## **(402, 203, 1, 1000.00),**

## **(402, 202, 1, 50.00);**

## **# Sample SQL Queries**

## **# View employees with their managers**

## **SELECT e.Emp\_Name, e.Position, m.Manager\_Name, m.Department**

## **FROM Employees e**

## **JOIN Managers m ON e.Manager\_ID = m.Manager\_ID;**

## **# Total sales by each manager**

## **SELECT m.Manager\_Name, SUM(s.Total\_Amount) AS Total\_Sales**

## **FROM Sales s**

## **JOIN Employees e ON s.Emp\_ID = e.Emp\_ID**

## **JOIN Managers m ON e.Manager\_ID = m.Manager\_ID**

## **GROUP BY m.Manager\_Name;**

## **# Best-selling product**

## **SELECT p.Prod\_Name, SUM(sd.Quantity) AS Total\_Sold**

## **FROM Sale\_Details sd**

## **JOIN Products p ON sd.Prod\_ID = p.Prod\_ID**

## **GROUP BY p.Prod\_Name**

## **ORDER BY Total\_Sold DESC**

## **LIMIT 1;**

## **# Check Table Data Anytime**

## **SELECT \* FROM Managers;**

## **SELECT \* FROM Employees;**

## **SELECT \* FROM Products;**

## **SELECT \* FROM Customers;**

## **SELECT \* FROM Sales;**

## **SELECT \* FROM Sale\_Details;**

## **# PL/SQL Extensions (for PL/SQL Lab)**

## **# Stored Procedure – Calculate Manager’s Total Sales**

## **DELIMITER //**

## **CREATE PROCEDURE GetManagerSales(IN managerId INT)**

## **BEGIN**

## **SELECT m.Manager\_Name, SUM(s.Total\_Amount) AS Total\_Sales**

## **FROM Sales s**

## **JOIN Employees e ON s.Emp\_ID = e.Emp\_ID**

## **JOIN Managers m ON e.Manager\_ID = m.Manager\_ID**

## **WHERE m.Manager\_ID = managerId**

## **GROUP BY m.Manager\_Name;**

## **END //**

## **DELIMITER ;**

## **#Execute:**

## **CALL GetManagerSales(1);**

## **#Trigger – Update Stock After Sale**

## **DELIMITER //**

## **CREATE TRIGGER UpdateStockAfterSale**

## **AFTER INSERT ON Sale\_Details**

## **FOR EACH ROW**

## **BEGIN**

## **UPDATE Products**

## **SET Stock = Stock - NEW.Quantity**

## **WHERE Prod\_ID = NEW.Prod\_ID;**

## **END //**

## **DELIMITER ;**

## ****Fig -1 for the all the table****

## 

## ****7. Functionalities****

* Add and manage **Managers**, **Employees**, **Products**, and **Customers**.
* Record and track **Sales** and **Sale\_Details**.
* **Automatically update stock** after sales using triggers.
* Generate **reports** such as total sales by manager or best-selling products.
* Demonstrate **stored procedures** and **PL/SQL automation**.

## ****8. Conclusion / Future Work / Learning Outcomes****

### ****Conclusion****

The **Company Management System** successfully integrates multiple entities within a company into a single relational database. It automates repetitive tasks, improves data accuracy, and demonstrates the practical power of SQL and PL/SQL in business data management.

### ****Future Work****

* Develop a **front-end interface** using Python (Tkinter) or Java (JDBC).
* Add **user authentication and roles** (Admin, Employee).
* Generate **visual reports** for analytics and decision-making.
* Integrate **backup and recovery features**.

### ****Learning Outcomes****

* Mastery of **database schema design** and **normalization**.
* Practical experience in **SQL joins, aggregates, and constraints**.
* Understanding of **stored procedures** and **triggers** for automation.
* Improved skills in **systematic project documentation** and testing.

### ****GitHub Link (Optional)****

(Add your repository link if available)  
👉 GitHub Repository – Company Management System

### ****Bibliography****

* MySQL Documentation – <https://www.mysql.com/>
* W3Schools SQL Tutorial – https://www.w3schools.com/sql/
* Oracle PL/SQL Reference – https://docs.oracle.com/cd/B10501\_01/appdev.920/a96624/toc.htm