**UNIVERSITY INSTITUTE OF COMPUTING**

**CASE STUDY REPORT**

**ON**

**KFC MANAGEMENT SYSTEM**

Program Name: BCA

Subject Name/Code: Database Management System (23CAT-251)

**Submitted by: Submitted to:**

**Name: Shweta Kumari Name: Arvinder Singh**

**UID: 23BCA10813 Designation: Assist. Prof.**

**Section:** 4(A)

**Table of Contents**

**1.Introduction**

**2.Technique**

**3.System Configuration**

**4.Input**

**5. ER Diagram & Schema**

**6. Table Structures**

**7. SQL Implementation Code**

**8. Sample Query Outputs**

**9. Observations & Limitations**

**10. Objectives**

**11. Relationships**

**12. Conclusion**

**13.GITHub**

**https://github.com/shweta7-k/dbms.git**

ABSTRACT

* **Introduction:**

The KFC Employee Management System is a database-driven project designed to streamline the management of employee records, including contact details, performance tracking, and order handling within a KFC outlet. This system uses SQL to manage relational data effectively, providing valuable insights and control to management.

* **Technique:**

The system is built using the **Relational Database Management System (RDBMS)** approach. It employs normalization techniques to eliminate data redundancy and ensure data integrity. SQL commands are used to create, manipulate, and retrieve data using DDL, DML, and DQL operations. Relationships between tables are managed using **Primary Keys** and **Foreign Keys**.

* **System Configuration:**

 **Database**: MySQL 8.0

 **Interface**: MySQL Workbench

 **Operating System**: Windows 10 or higher

 **RAM**: Minimum 4 GB

 **Storage**: Minimum 1 GB free space for DBMS and project files

 **Diagram Tool**: Draw.io (for ER diagrams)

* **INPUT:**

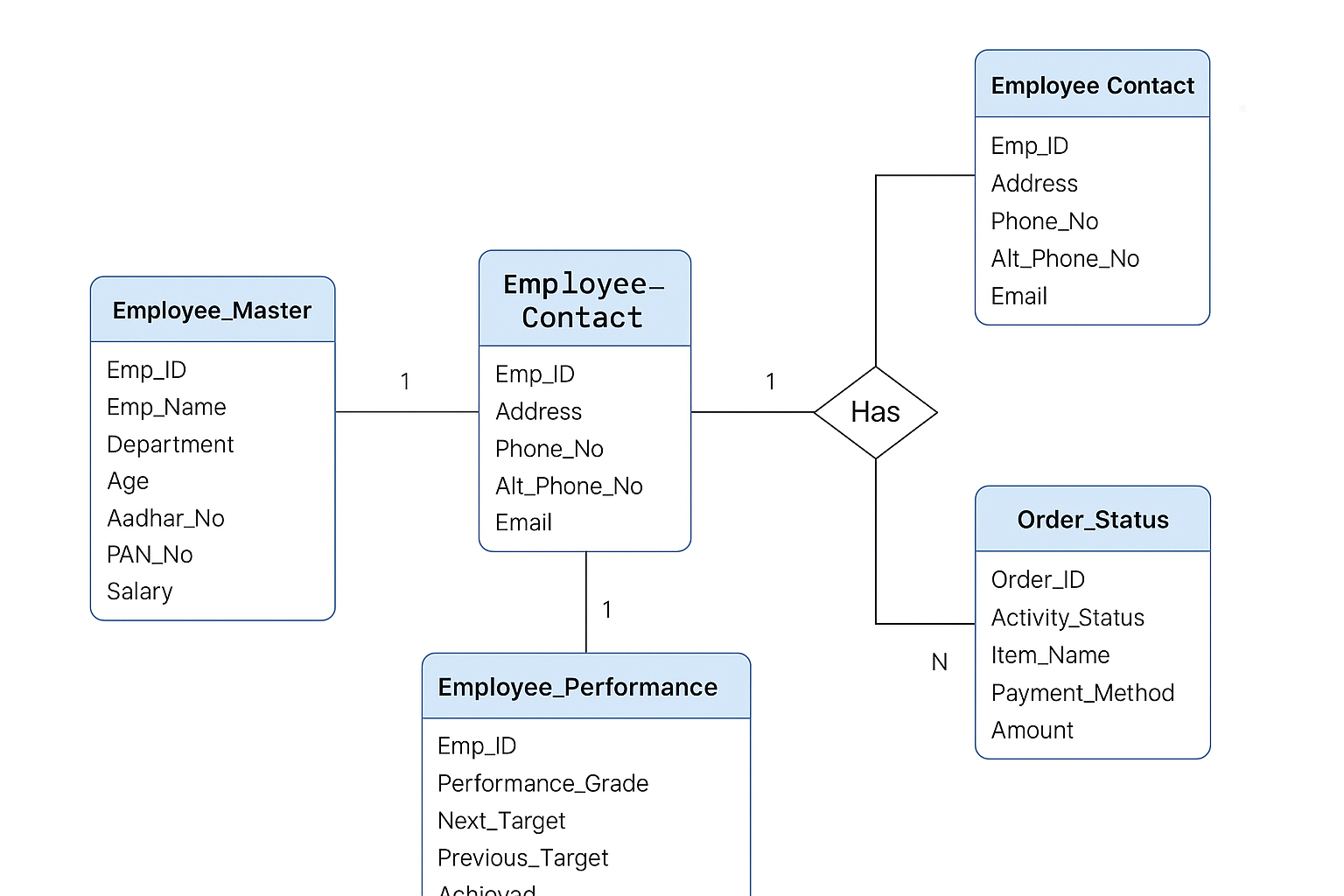
 **Employee details:** ID, name, department, age, salary, Aadhar, PAN

 **Contact details:** Address, phone numbers, email

 **Performance metrics:** Grades, targets, achievements

 **Order records:** Items handled, payment methods, transaction amounts

* ER DIAGRAM:



* ER DIAGRAM DESCRIPTION

**Entities:**

* **Employee\_Master:** Stores core employee details (name, ID, department, salary, etc.)
* **Employee\_Contact:** Contains address, phone numbers, and email.
* **Employee\_Performance:** Tracks employee performance, targets, and grades.
* **Order\_Status:** Records item handling and order transactions linked to employees.

**Relationships:**

* One employee has one contact detail entry (1:1)
* One employee can have multiple performance entries (1:M)
* One employee can handle multiple orders (1:M)
* **TABLE RELATIONSHIPS:**
* **Employee\_Master (Emp\_ID)** → **Employee\_Contact (Emp\_ID)**
* **Employee\_Master (Emp\_ID)** → **Employee\_Performance (Emp\_ID)**
* **Employee\_Master (Emp\_ID)** → **Order\_Status (Emp\_ID)**
* **Order\_Status (Order\_ID)** is a unique identifier for each order
* **TABULAR FORMAT**

| **Table Name** | **Description** |
| --- | --- |
| **Employee\_Master** | Stores main employee data (ID, name, salary, etc.) |
| **Employee\_Contact** | Stores contact info (address, phone, email) |
| **Employee\_Performance** | Tracks performance metrics (targets, grades) |
| **Order\_Status** | Logs orders, payment method, items, and amount |

* **TABLE CREATION**

Each table is created using CREATE TABLE SQL statements.  
They include Primary Keys, Foreign Keys, and constraints for maintaining data integrity.

**Here’s a summary of key constraints used:**

* **Emp\_ID in Employee\_Master is the Primary Key.**
* **Emp\_ID in other tables is a Foreign Key referencing Employee\_Master.**
* **Order\_ID in Order\_Status is a unique primary key for identifying orders.**
* **Constraints like NOT NULL, UNIQUE, and CHECK (Age > 21) are used for validation.**
* **TABLE REALTION:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table Name | Primary Key | Foreign Key | Referenced Table | Constraints |
| Employee\_Master | Emp\_ID | - | - | Emp\_ID is UNIQUE and NOT NULL |
| Employee\_Contact | None | Emp\_ID | Employee\_Master | Emp\_ID is a FOREIGN KEY |
| Employee\_Performance | None | Emp\_ID | Employee\_Master | Emp\_ID is a FOREIGN KEY |
| Order\_Status | Order\_ID | Emp\_ID | Employee\_Master | Order\_ID is UNIQUE, Emp\_ID is a FOREIGN KEY |

* TABULAR FORMAT:

## Employee\_Master

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Constraints |
| Emp\_ID | INT | Primary Key |
| Emp\_Name | VARCHAR (50) |  |
| Department | VARCHAR (30) |  |
| Age | INT | CHECK (Age > 21) |
| Aadhar\_No | VARCHAR (15) | UNIQUE NOT NULL |
| PAN\_No | VARCHAR (15) |  |
| Salary | DECIMAL (10,2) |  |

## Employee\_Contact

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Constraints |
| Emp\_ID | INT | Foreign Key → Employee\_Master (Emp\_ID) |
| Address | VARCHAR (100) |  |
| Phone\_Number | VARCHAR (15) |  |
| Alternate\_Number | VARCHAR (15) |  |
| Email | VARCHAR (50) |  |

## Employee\_Performance

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Constraints |
| Emp\_ID | INT | Foreign Key → Employee\_Master (Emp\_ID) |
| Performance\_Grade | CHAR (2) |  |
| Next\_Target | INT |  |
| Previous\_Target | INT |  |
| Achieved | INT |  |

## Order\_Status

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Constraints |
| Emp\_ID | INT | Foreign Key → Employee\_Master (Emp\_ID) |
| Activity\_Status | CHAR (3) |  |
| Order\_ID | INT | UNIQUE |
| Item\_Name | VARCHAR (30) |  |
| Payment\_Method | VARCHAR (20) |  |
| Amount | INT |  |

* TABLE CREATION:

TABLE Employee\_Master

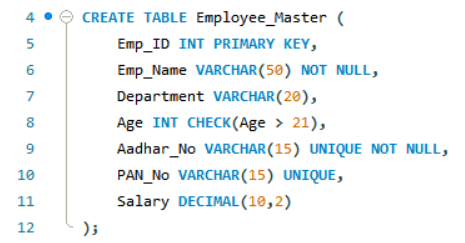


TABLE Employee\_Contact

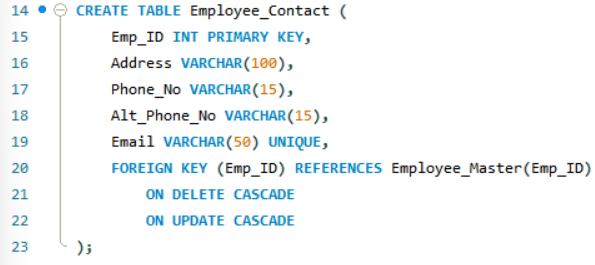


TABLE Employee\_Performance

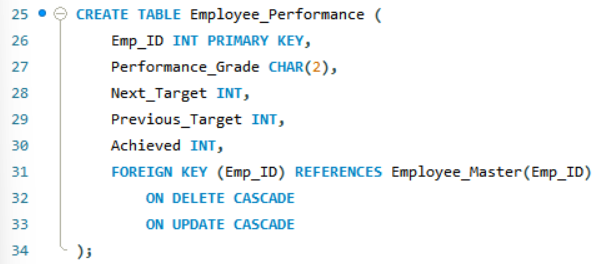
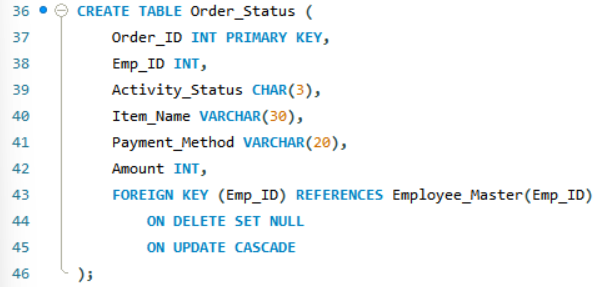


TABLE Order\_Status



* SQL IMPLEMENTATION Code:

**CREATE DATABASE KFC;**

**USE KFC;**

**CREATE TABLE Employee\_Master (**

**Emp\_ID INT PRIMARY KEY,**

**Emp\_Name VARCHAR (50) NOT NULL,**

**Department VARCHAR (20),**

**Age INT CHECK (Age > 21),**

**Aadhar\_No VARCHAR (15) UNIQUE NOT NULL,**

**PAN\_No VARCHAR (15) UNIQUE,**

**Salary DECIMAL (10,2)**

**);**

**CREATE TABLE Employee\_Contact (**

**Emp\_ID INT PRIMARY KEY,**

**Address VARCHAR (100),**

**Phone\_No VARCHAR (15),**

**Alt\_Phone\_No VARCHAR (15),**

**Email VARCHAR (50) UNIQUE,**

**FOREIGN KEY (Emp\_ID) REFERENCES Employee\_Master(Emp\_ID)**

**ON DELETE CASCADE**

**ON UPDATE CASCADE**

**);**

**CREATE TABLE Employee\_Performance (**

**Emp\_ID INT PRIMARY KEY,**

**Performance\_Grade CHAR (2),**

**Next\_Target INT,**

**Previous\_Target INT,**

**Achieved INT,**

**FOREIGN KEY (Emp\_ID) REFERENCES Employee\_Master(Emp\_ID)**

**ON DELETE CASCADE**

**ON UPDATE CASCADE**

**);**

**CREATE TABLE Order\_Status (**

**Order\_ID INT PRIMARY KEY,**

**Emp\_ID INT,**

**Activity\_Status CHAR (3),**

**Item\_Name VARCHAR (30),**

**Payment\_Method VARCHAR (20),**

**Amount INT,**

**FOREIGN KEY (Emp\_ID) REFERENCES Employee\_Master(Emp\_ID)**

**ON DELETE SET NULL**

**ON UPDATE CASCADE**

**);**

**INSERT INTO Employee\_Master (Emp\_ID, Emp\_Name, Department, Age, Aadhar\_No, PAN\_No, Salary)**

**VALUES**

**(201, 'Arjun', 'Kitchen', 26, '111122223333', 'AKP1234P', 32000.75),**

**(202, 'Deepa', 'Delivery', 34, '222233334444', 'DPL4567Q', 29500.50),**

**(203, 'Sana', 'Service', 29, '333344445555', 'SNA7890R', 28000.00),**

**(204, 'Ravi', 'Kitchen', 40, '444455556666', 'RVI1234S', 30500.00),**

**(205, 'Meena', 'Cashier', 31, '555566667777', 'MNA4567T', 26500.00),**

**(206, 'Karan', 'Manager', 38, '666677778888', 'KRN7890U', 45500.00),**

**(207, 'Simran', 'Service', 27, '777788889999', 'SMR3210V', 27000.00),**

**(208, 'Tina', 'Delivery', 33, '888899990000', 'TNA6543W', 25000.00),**

**(209, 'Ayaan', 'Kitchen', 41, '999900001111', 'AYN0987X', 33500.00),**

**(210, 'Zara', 'Cashier', 36, '000011112222', 'ZRA5432Y', 31000.00),**

**(211, 'Nikhil', 'Service', 28, '123456789012', 'NKL1478Z', 29500.00),**

**(212, 'Utsav', 'Admin', 35, '234567890123', 'UTS7890A', 50000.00);**

**INSERT INTO Employee\_Contact (Emp\_ID, Address, Phone\_No, Alt\_Phone\_No, Email)**

**VALUES**

**(201, 'Delhi', '9876543210', '9123456780', 'arjun@kfc.com'),**

**(202, 'Mumbai', '8765432109', '9234567891', 'deepa@kfc.com'),**

**(203, 'Chennai', '7654321098', '9345678912', 'sana@kfc.com'),**

**(204, 'Hyderabad', '6543210987', '9456789123', 'ravi@kfc.com'),**

**(205, 'Kolkata', '5432109876', '9567891234', 'meena@kfc.com'),**

**(206, 'Pune', '4321098765', '9678912345', 'karan@kfc.com'),**

**(207, 'Ahmedabad', '3210987654', '9789123456', 'simran@kfc.com'),**

**(208, 'Bangalore', '2109876543', '9891234567', 'tina@kfc.com'),**

**(209, 'Lucknow', '1098765432', '9901234567', 'ayaan@kfc.com'),**

**(210, 'Patna', '9987654321', '9012345678', 'zara@kfc.com'),**

**(211, 'Indore', '8876543210', '8123456789', 'nikhil@kfc.com'),**

**(212, 'Jaipur', '7765432109', '7234567890', 'utsav@kfc.com');**

**INSERT INTO Employee\_Performance (Emp\_ID, Performance\_Grade, Next\_Target, Previous\_Target, Achieved)**

**VALUES**

**(201, 'A+', 120, 100, 110),**

**(202, 'B', 130, 115, 90),**

**(203, 'A', 140, 120, 100),**

**(204, 'C+', 150, 110, 80),**

**(205, 'B', 160, 130, 110),**

**(206, 'D', 100, 80, 60),**

**(207, 'B+', 125, 120, 115),**

**(208, 'C+', 105, 95, 70),**

**(209, 'C', 110, 100, 90),**

**(210, 'A+', 135, 125, 100),**

**(211, 'F', 95, 80, 30),**

**(212, 'F', 170, 160, 120);**

**INSERT INTO Order\_Status (Order\_ID, Emp\_ID, Activity\_Status, Item\_Name, Payment\_Method, Amount)**

**VALUES**

**(4001, 201, 'YES', 'Zinger', 'Cash', 250),**

**(4002, 202, 'YES', 'Popcorn', 'UPI', 300),**

**(4003, 203, 'NO', 'Bucket', 'Card', 800),**

**(4004, 204, 'YES', 'Fries', 'Cash', 150),**

**(4005, 205, 'YES', 'Burger', 'Card', 200),**

**(4006, 206, 'YES', 'Wrap', 'Cash', 220),**

**(4007, 207, 'YES', 'Krusher', 'Card', 180),**

**(4008, 208, 'YES', 'Zinger', 'Card', 250),**

**(4009, 209, 'YES', 'Chicken Wings', 'Card', 320),**

**(4010, 210, 'NO', 'Bucket', 'Cash', 700),**

**(4011, 211, 'YES', 'Rice Bowlz', 'UPI', 180),**

**(4012, 212, 'NO', 'Zinger', 'Card', 250);**

**SELECT \* FROM Employee\_Master;**

**SELECT Emp\_Name FROM Employee\_Master**

**WHERE Department = 'Kitchen';**

**SELECT \* FROM Employee\_Master**

**WHERE Age > 30;**

**SELECT Emp\_Name, Salary FROM Employee\_Master**

**WHERE Salary BETWEEN 35000 AND 50000;**

**SELECT \* FROM Employee\_Master**

**WHERE Emp\_Name LIKE 'S%';**

**UPDATE Employee\_Master SET Department = 'Logistics'**

**WHERE Emp\_Name = 'Amit';**

**UPDATE Employee\_Master**

**SET Salary = Salary \* 1.10**

**WHERE Department = 'Sales';**

**DELETE FROM Employee\_Master WHERE Emp\_ID = 206;**

**DELETE FROM Employee\_Master WHERE Department = 'Logistics';**

**SELECT em.Emp\_Name, ec.Email, ep.Performance\_Grade**

**FROM Employee\_Master em**

**JOIN Employee\_Contact ec ON em.Emp\_ID = ec.Emp\_ID**

**JOIN Employee\_Performance ep ON em.Emp\_ID = ep.Emp\_ID;**

**SELECT em.Emp\_Name, os.Item\_Name, os.Payment\_Method**

**FROM Employee\_Master em**

**JOIN Order\_Status os ON em.Emp\_ID = os.Emp\_ID;**

**SELECT Department, COUNT (\*) AS Num\_Employees**

**FROM Employee\_Master**

**GROUP BY Department;**

**SELECT AVG(Next\_Target) AS Avg\_Target**

**FROM Employee\_Performance;**

**SELECT MAX(Amount) AS**

**Max\_Amount FROM Order\_Status;**

**SELECT Payment\_Method, SUM(Amount) AS Revenue**

**FROM Order\_Status**

**GROUP BY Payment\_Method;**

**SELECT em.Emp\_Name, ep.Achieved**

**FROM Employee\_Master em**

**JOIN Employee\_Performance ep ON em.Emp\_ID = ep.Emp\_ID**

**WHERE ep.Achieved = (SELECT MAX(Achieved) FROM Employee\_Performance);**

**SELECT Emp\_Name, Salary**

**FROM Employee\_Master**

**WHERE Salary = (SELECT MAX(Salary) FROM Employee\_Master);**

**SELECT Emp\_Name, Salary**

**FROM Employee\_Master**

**ORDER BY Salary DESC**

**LIMIT 3;**

**SELECT AVG(Age) AS Average\_Age**

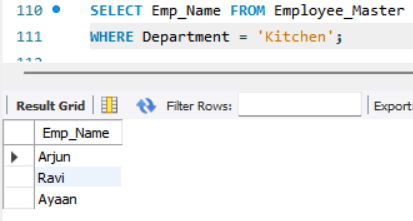
**FROM Employee\_Master;**

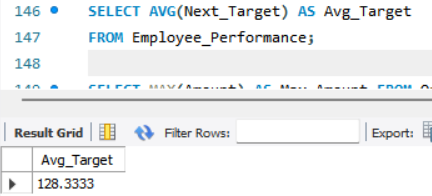
**SELECT em.Emp\_Name, em.Department, ec.Address**

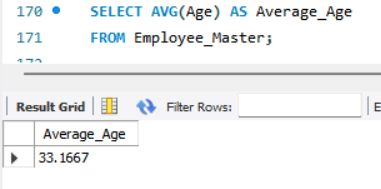
**FROM Employee\_Master em**

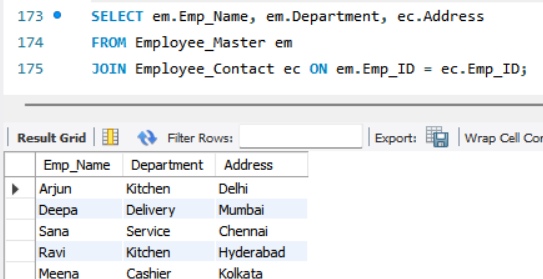
**JOIN Employee\_Contact ec ON em.Emp\_ID = ec.Emp\_ID;**

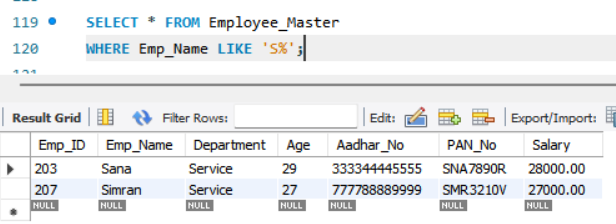
* SQL QUERIES WITH OUTPUT:

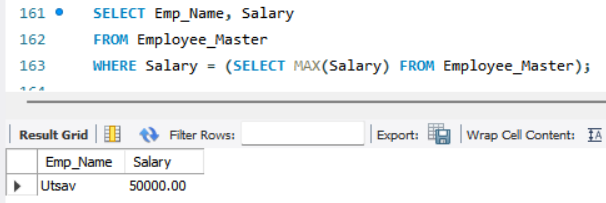


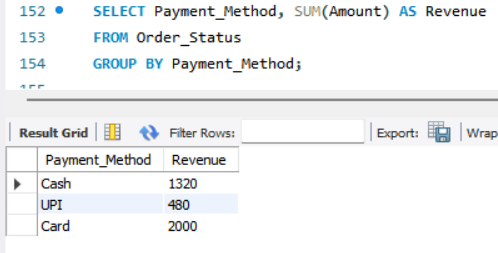


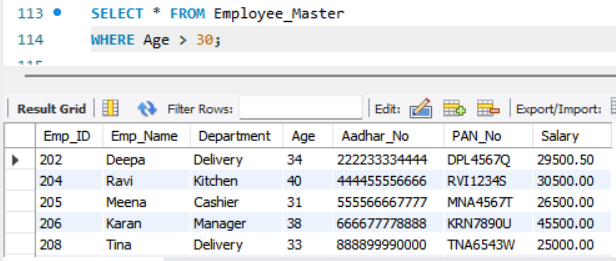


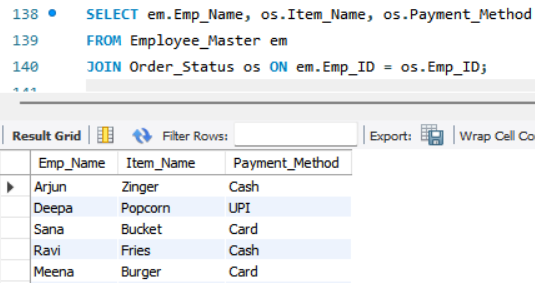


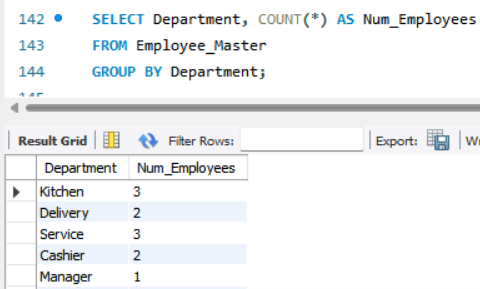


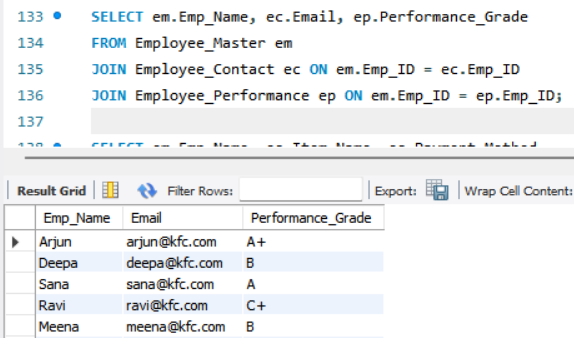


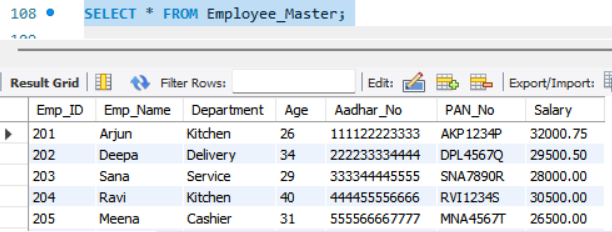


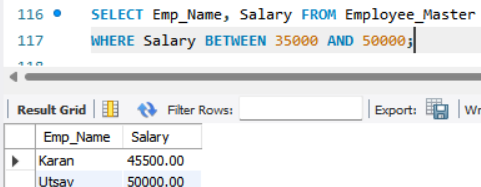












* SUMMARY:

The **KFC Employee Management System** is a comprehensive **database management mini-project** that simulates the core operations of a KFC branch from a backend perspective. It primarily focuses on storing, managing, and retrieving data related to employees, their performance, contact details, and their involvement in order processing and payments.

### **Key Highlights:**

* **Real-World Case Study**: Inspired by a real-life fast-food chain scenario (KFC), making the system highly relatable and practical.
* **Normalization & Design**: The database is normalized to avoid redundancy, with clear separation of concerns across multiple tables.
* **Relational Structure**: Relationships are well-defined using **primary keys**, **foreign keys**, and **referential integrity constraints**.
* **Modular Table Setup**:
  + Employee\_Master: Handles core employee data including salary and identification.
  + Employee\_Contact: Stores address, contact numbers, and email.
  + Employee\_Performance: Records performance stats such as targets and achievements.
  + Order\_Status: Tracks orders, items sold, payment method, and amount.

### **Learning Outcomes:**

* Practical understanding of **SQL table creation**, **data insertion**, **modification**, and **data integrity**.
* Real-time execution of **20+ SQL queries** demonstrating all essential operations:
  + **Data Retrieval (SELECT)**
  + **Data Filtering (WHERE, LIKE, IN, BETWEEN)**
  + **Data Manipulation (INSERT, UPDATE, DELETE)**
  + **Joins and Subqueries**
  + **Aggregate Functions (AVG, MAX, COUNT, SUM)**
* Use of **ER Diagram** for better visualization of entity relationships.
* Emphasis on **query optimization** and **structured reporting**.

### **Project Application:**

This system can be applied or expanded into:

* A real-time **HR system** for franchise restaurants.
* Integration with a **sales dashboard** or **inventory system**.
* A part of a **Payroll and Performance Appraisal System**.

### **Technologies Used:**

* **MySQL** for database management.
* **DB Browser / MySQL Workbench** for query execution and schema visualization.
* **Microsoft Word / Notepad** for documentation and screenshots.

**Objectives:**

To understand and implement real-world database design.

To create and manipulate tables using SQL.

To apply constraints for data integrity.

To perform queries for data analysis, reporting, and updates.

To visualize entity relationships through ER diagrams.

**Relationships**:

One-to-many relationship from Employee\_Master to all other tables via Emp\_ID.

Represented in:

ER Diagram (attached in report)

Tabular relationship format (showing PK, FK, and constraints)

* CONCLUSION:

The **KFC Employee Management System** project successfully demonstrates the core concepts and practical applications of **Database Management Systems (DBMS)** using a real-world business scenario. Through the creation and management of multiple relational tables, this project illustrates how structured data can be stored, manipulated, and retrieved efficiently.

By implementing essential DBMS functionalities—such as table creation, insertion, updating, deletion, joins, and aggregations—we were able to design a system that not only meets operational needs but also maintains data integrity and consistency through appropriate constraints and keys.

The **ER diagram**, **schema design**, **tabular relationships**, and **SQL queries** serve as a strong foundation for understanding how back-end systems operate in organizations like KFC. Moreover, the hands-on execution of various queries provided in-depth insights into how data flows and interacts across multiple tables.

### **Observations:**

1. The database effectively captures all relevant employee-related information such as personal details, contact information, performance metrics, and order status.
2. Use of **relational integrity** ensures data consistency and enforces correct referencing between tables via **foreign keys**.
3. The **SQL queries** demonstrate a strong grasp of basic to advanced query operations including subqueries, joins, and aggregate functions.
4. Proper normalization has reduced redundancy and improved query efficiency.
5. The design supports scalability — more tables such as Inventory, Payroll, or Attendance could easily be added in the future.

### **Limitations:**

1. **Limited Scope**: The current project covers only core employee-related operations and does not include other essential business aspects such as inventory, customer data, or shift management.
2. **Security Aspects Not Covered**: No implementation of user roles or access controls (e.g., admin vs. employee access).
3. **Lack of Frontend Interface**: It is a backend-only project. There’s no graphical user interface (GUI) or web frontend for user interaction.
4. **Manual Data Entry**: All data is manually inserted. No automated data import or API integration for real-time updates.
5. **Performance Metrics**: While performance data is captured, it is quite basic. There is no advanced analytics, graphical reporting, or trend tracking.

In conclusion, this mini-project helped in building a strong foundation in relational database concepts and SQL, while offering a scalable blueprint that can be further enhanced for large-scale business systems such as HR management, order tracking, and performance evaluation.