**UNIVERSITY INSTITUTE OF COMPUTING**

**CASE STUDY REPORT**

**ON**

**COCA-COLA MANAGEMENT SYSTEM**

Program Name: BCA

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

**Submitted by: Submitted to:**

**Name: Utsav Raj Name: Arvinder Singh**

**UID: 23BCA10816 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**

ABSTRACT

* Introduction:

The Coca-Cola Database Project is a structured representation of employee and operational information for a company-themed simulation. This database is designed to manage employee details, their addresses, performance tracking, and last recorded activity. It reflects how a real-world system may operate to organize data in a clean and accessible manner using SQL.

This project aims to provide a practical understanding of core database concepts like:

* Table creation with appropriate constraints
* Data integrity using primary and foreign keys
* Querying with conditions, joins, aggregations
* Efficient data insertion, retrieval, and management
* Technique:

This project employs **Relational Database Management System (RDBMS)** concepts, primarily using **Structured Query Language (SQL)** to design and manage the Coca-Cola database. The database is implemented using **MySQL**, one of the most popular and widely-used open-source RDBMS.

**Key Techniques Used:**

* + **Data Definition Language (DDL)**:  
    Used to define the structure of the database including creating tables, setting data types, and applying constraints such as PRIMARY KEY, FOREIGN KEY, UNIQUE, and CHECK.
  + **Data Manipulation Language (DML)**:  
    Used to insert, update, and delete records in the tables. Bulk INSERT statements are used for populating the data.
  + **Data Query Language (DQL)**:  
    Various SELECT queries are used to retrieve data using filters, joins, conditions, ordering, and aggregation.
  + **Relational Integrity**:  
    The database is structured with clear **relationships** between tables using **foreign keys** to maintain consistency.
  + **Normalization Principles**:  
    Data is stored in different tables to reduce redundancy and improve efficiency, following basic normalization rules.
  + **Join Operations**:  
    Inner Join, Left Join, Right Join, and Natural Join are used to fetch related data from multiple tables.
* System Configuration:

The Coca-Cola database project was developed and executed in a local development environment with the following system and software configuration:

**Hardware Configuration:**

* **Processor**: Intel Core i5 / AMD Ryzen 5 or equivalent
* **RAM**: 8 GB (minimum)
* **Storage**: 256 GB SSD / 500 GB HDD
* **Display**: 1366x768 resolution or higher

**Software Configuration:**

* **Operating System**: Windows 10 / 11 (64-bit)
* **Database Management System**: MySQL Server 8.0
* **MySQL Interface Tool**: MySQL Workbench
* **Text Editor**: VS Code
* **Document Tool**: Microsoft Word (for report preparation)
* INPUT:

The Coca-Cola database accepts structured input data related to employees, including their personal details, address, performance metrics, and last recorded business activity. The inputs were provided using **SQL INSERT statements** and are organized across four main tables.

**Types of Inputs:**

1. **Employee Information:**
   * Employee ID (E\_ID)
   * Name
   * Department
   * Age
   * Aadhar Number
   * PAN Number
   * Salary
2. **Employee Address:**
   * Employee ID (E\_ID)
   * Residential Address
   * Phone Number
   * Alternate Contact Number
   * Email Address
3. **Performance Metrics:**
   * Employee ID (E\_ID)
   * Current Performance Grade
   * Next Target
   * Previous Target
   * Achieved Target
4. **Last Status Activity:**
   * Employee ID (E\_ID)
   * Activity Status (YES/NO)
   * Order ID
   * Item Type (e.g., RGB, PET)
   * Payment Method (Cash, UPI, etc.)
   * Amount Paid

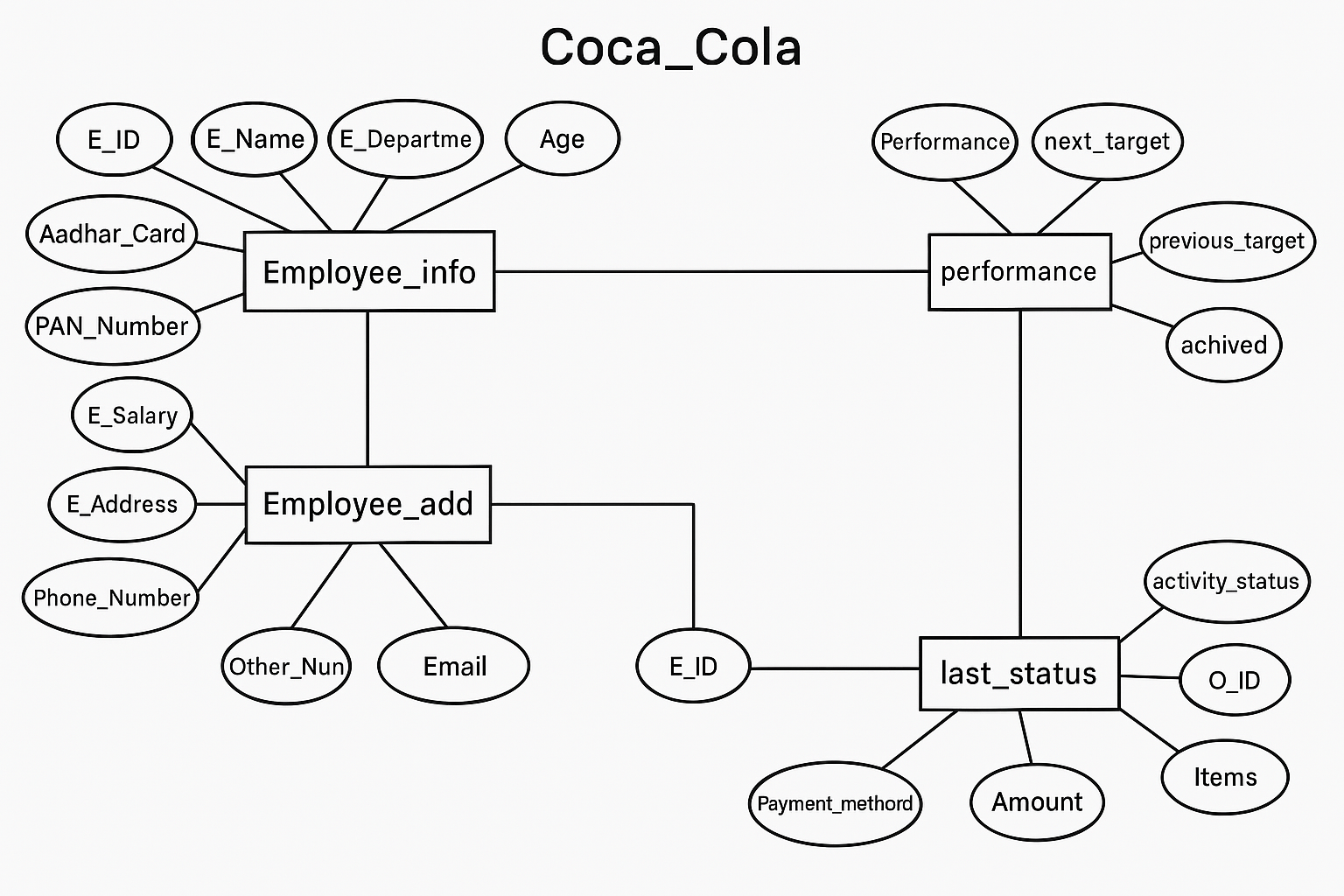
**Input Method:**

The data was inserted into the tables using SQL commands in the following format:

INSERT INTO table\_name (column1, column2, ...) VALUES (value1, value2, ...);

Each record entered into the system reflects a real-world scenario and helps simulate organizational data flow and integrity.

* ER DIAGRAM:



* ER DIAGRAM DESCRIPTION:

The ER diagram represents the structure of the Coca-Cola database, illustrating the relationships between different entities and their attributes. Below is a breakdown of the entities and their relationships:

Entities and attributes:

1. **Employee\_info**
   * **Attributes**:
     + E\_ID (Primary Key): Unique identifier for each employee.
     + E\_Name: Name of the employee.
     + E\_Department: Department to which the employee belongs (e.g., Spoke, HR, etc.).
     + Age: Age of the employee (with a constraint to ensure age is greater than 21).
     + Aadhar\_Card: Unique identifier (Aadhar number) for the employee.
     + PAN\_Number: Unique identifier (PAN number) for the employee.
     + E\_Salary: Employee’s salary.
2. **Employee\_add**
   * **Attributes**:
     + E\_ID (Foreign Key referencing Employee\_info): The employee ID linking to the Employee\_info table.
     + E\_Address: Address of the employee.
     + Phone\_Number: Primary phone number of the employee.
     + Other\_Number: Secondary phone number.
     + Email: Email address of the employee.
3. **Performance**
   * **Attributes**:
     + E\_ID (Foreign Key referencing Employee\_info): The employee ID linking to the Employee\_info table.
     + Performance: Performance rating (e.g., A+, B, etc.).
     + next\_target: The target set for the next period.
     + previous\_target: The target from the previous period.
     + achieved: The target achieved by the employee.
4. **Last\_status**
   * **Attributes**:
     + E\_ID (Foreign Key referencing Employee\_info): The employee ID linking to the Employee\_info table.
     + activity\_status: The status of the employee’s activity (e.g., YES or NO).
     + O\_ID: Unique order ID.
     + Items: Items associated with the activity.
     + Payment\_method: The method of payment used (e.g., CASH, UPI, etc.).
     + Amount: The amount associated with the activity.

Relationships:

1. **Employee\_info to Employee\_add**:
   * **Type**: One-to-One
   * **Description**: Each employee can have only one corresponding address and contact information in the Employee\_add table. The E\_ID from Employee\_info acts as a foreign key in the Employee\_add table, linking the two tables.
2. **Employee\_info to Performance**:
   * **Type**: One-to-One
   * **Description**: Each employee has one performance record. The E\_ID in the Performance table references the E\_ID in the Employee\_info table to link the employee's performance data.
3. **Employee\_info to Last\_status**:
   * **Type**: One-to-One
   * **Description**: Each employee has one activity status record in the Last\_status table. The E\_ID in Last\_status references the E\_ID in the Employee\_info table.

Key Constraints:

* **Primary Keys**:
  + E\_ID in Employee\_info, Employee\_add, Performance, and Last\_status tables ensures each entry is unique.
* **Foreign Keys**:
  + E\_ID in Employee\_add, Performance, and Last\_status ensures data consistency by referencing Employee\_info. This enforces referential integrity, meaning that all entries in Employee\_add, Performance, and Last\_status must relate to an existing employee in Employee\_info.
* **Check Constraints**:
  + The Age attribute in Employee\_info is constrained to values greater than 21.
* **Unique Constraints**:
  + Aadhar\_Card and PAN\_Number in Employee\_info are unique to each employee, ensuring no duplicates.

Relationship Cardinality:

1. **Employee\_info to Employee\_add**: One-to-One (Each employee has one address and contact information).
2. **Employee\_info to Performance**: One-to-One (Each employee has one performance record).
3. **Employee\_info to Last\_status**: One-to-One (Each employee has one activity status record).

* TABLE RELATIONSHIPS:

The Coca-Cola database is structured using **relational database principles**, connecting multiple tables via **primary and foreign keys**. These relationships ensure **data consistency**, **eliminate redundancy**, and **enable complex queries** across related tables.

**1. Employee\_info ⇄ Employee\_add**

* **Type**: One-to-One
* **Key Used**:
  + Employee\_info.E\_ID → Employee\_add.E\_ID (Foreign Key)
* **Explanation**: Each employee has one associated record containing their address and contact details.
* **Purpose**: To separate contact info from core employee data for modular design.

**2. Employee\_info ⇄ Performance**

* **Type**: One-to-One
* **Key Used**:
  + Employee\_info.E\_ID → Performance.E\_ID (Foreign Key)
* **Explanation**: Each employee has one performance record for tracking targets and achievements.
* **Purpose**: Keeps performance metrics distinct from personal/financial data.

**3. Employee\_info ⇄ Last\_status**

* **Type**: One-to-One
* **Key Used**:
  + Employee\_info.E\_ID → Last\_status.E\_ID (Foreign Key)
* **Explanation**: Each employee can be linked to one record of their last work activity, including items and payments.
* **Purpose**: Tracks the employee’s most recent transactional status and method of payment.
* TABLE REALTION:

**Primary & Foreign Key-Based Relations: -**

Each relation in the Coca-Cola database is built around the **E\_ID** field — which acts as the **Primary Key** in the main table (Employee\_info) and as a **Foreign Key** in the other tables.

**Detailed Table Relations**

1. **Employee\_info → Employee\_add**
   * **Relation Type**: One-to-One
   * **Primary Key**: Employee\_info.E\_ID
   * **Foreign Key**: Employee\_add.E\_ID
   * **Relation Description**: Each employee has exactly one address and contact detail.
2. **Employee\_info → Performance**
   * **Relation Type**: One-to-One
   * **Primary Key**: Employee\_info.E\_ID
   * **Foreign Key**: Performance.E\_ID
   * **Relation Description**: Each employee has a corresponding performance record, tracking their targets and achievements.
3. **Employee\_info → Last\_status**
   * **Relation Type**: One-to-One
   * **Primary Key**: Employee\_info.E\_ID
   * **Foreign Key**: Last\_status.E\_ID
   * **Relation Description**: Each employee is linked to one record of their last order or task, including payment and item details.

* TABULAR FORMAT:

**1. Employee\_info Table**

| **Column Name** |  | **Data Type** | **Constraints** |
| --- | --- | --- | --- |
| E\_ID |  | INT | Primary Key, NOT NULL |
| E\_Name |  | VARCHAR (20) |  |
| E\_Department |  | CHAR (10) |  |
| Age |  | INT | CHECK (Age > 21) |
| Aadhar\_Card |  | VARCHAR (15) | UNIQUE, NOT NULL |
| PAN\_Number |  | VARCHAR (15) |  |
| E\_Salary |  | DECIMAL (10,2) |  |

**2. Employee\_add Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| E\_ID | INT | Foreign Key → Employee\_info(E\_ID) |
| E\_Address | VARCHAR (20) |  |
| Phone\_Number | VARCHAR (15) |  |
| Other\_Number | VARCHAR (15) |  |
| Email | VARCHAR (30) |  |

**3. Performance Table**

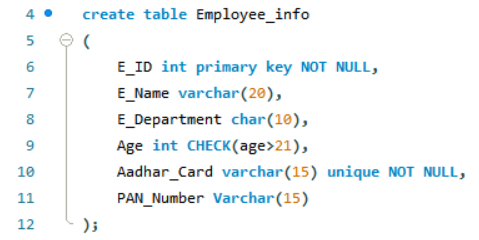
| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| E\_ID | INT | Foreign Key → Employee\_info(E\_ID) |
| Performance | CHAR (2) |  |
| next\_target | INT |  |
| previous\_target | INT |  |
| achived | INT |  |

**4. Last\_status Table**

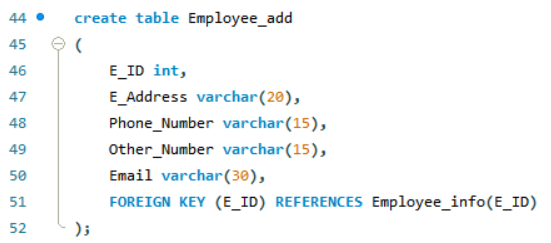
| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| E\_ID | INT | Foreign Key → Employee\_info(E\_ID) |
| activity\_status | CHAR (3) |  |
| O\_ID | INT | UNIQUE |
| Items | VARCHAR (10) |  |
| Payment\_methord | VARCHAR (10) |  |
| Amount | INT |  |

* TABLE CREATION:

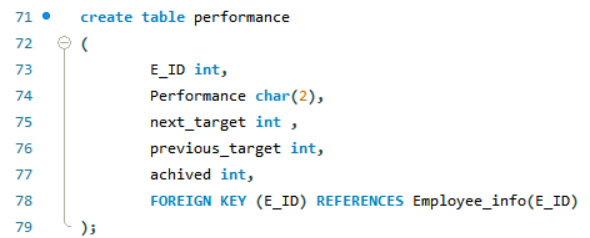
Employee\_info



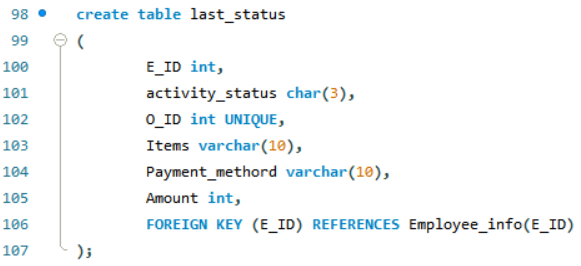
Employee\_add



Performance



last\_status



* SQL IMPLEMENTATION Code:

create database Coca\_Cola;

use Coca\_Cola;

create table Employee\_info

(

E\_ID int primary key NOT NULL,

E\_Name varchar (20),

E\_Department char (10),

Age int CHECK (age>21),

Aadhar\_Card varchar (15) unique NOT NULL,

PAN\_Number Varchar (15)

);

insert into Employee\_info(E\_ID, E\_Name, E\_Department, Age, Aadhar\_Card, PAN\_Number)

values

(101,'Aman','MGR',25,'185622458965','85lk6324'),

(102,'Bhabhuk','AMC',35,'285622458965','25BS8543');

alter table Employee\_info add E\_Salary decimal (10,2);

update employee\_info

set E\_Salary= 151536.85

where E\_id= 101;

update employee\_info

set E\_Salary= 15164.85

where E\_id= 102;

insert into Employee\_info(E\_ID, E\_Name, E\_Department, E\_Salary, Age, Aadhar\_Card, PAN\_Number)

values

(103,'Vimal','Spoke',45015.45,45,'385622458965','35CS8543'),

(104,'Hemal','Hub',35063.45,55,'485622458965','45DS8543'),

(105,'Shivam','AMC',85065.45,29,'585622458965','55ES8543'),

(106,'Saurav','HR',36520.45,28,'685622458965','65FS8543'),

(107,'Rishav','Hub',25410.45,36,'785622458965','75GS8543'),

(108,'Keshav','Spoke',96587.45,65,'885622458965','85HS8543'),

(109,'Sharma','HR',74589.45,48,'985622458965','95IS8543'),

(110,'Jatin','Spoke',78521.45,53,'085622458965','05JS8543'),

(111,'Harsh','Dealer',98523.45,32,'15622458965','10AA8543'),

(112,'Utsav','Wholesale',125630.45,43,'25622458965','20AB8543');

Select\* from Employee\_info;

create table Employee\_add

(

E\_ID int,

E\_Address varchar (20),

Phone\_Number varchar (15),

Other\_Number varchar (15),

Email varchar (30),

FOREIGN KEY (E\_ID) REFERENCES Employee\_info (E\_ID)

);

insert into Employee\_add (E\_ID, E\_Address, Phone\_Number, Other\_Number, Email)

values

(101,'Mumbai','1541030147','9632014530',' Aman@gmail.com'),

(102,'Pune','254130147','8632014530',' Bhabhuk@gmail.com'),

(103,'Patna','3541030147','7632014530',' vishal@gmail.com'),

(104,'Bihar','4541030147','6632014530',' hemal@gmail.com'),

(105,'Delhi','5541030147','5632014530',' shivam@gmail.com'),

(106,'Goa','6541030147','4632014530',' saurabh@gmail.com'),

(107,'Haryana','7541030147','3632014530',' rishav@gmail.com'),

(108,'Punjab','8541030147','2632014530',' keshav@gmail.com'),

(109,'Goa','9541030147','1632014530',' sharma@gmail.com'),

(110,'Mumbai','0541030147','0632014530',' jatin@gmail.com'),

(111,'Pune','5541030147','6632014530',' harsh@gmail.com'),

(112,'Patna','541830147','9639014530',' utsav@gmail.com');

Select\* from Employee\_add;

create table performance

(

E\_ID int,

Performance char (2),

next\_target int,

previous\_target int,

achived int,

FOREIGN KEY (E\_ID) REFERENCES Employee\_info(E\_ID)

);

insert into performance (E\_ID, Performance, next\_target, previous\_target, achived)

values

(101,'A+',854,630,530),

(102,'B',965,930,430),

(103,'A',632,630,330),

(104,'C+',986,630,430),

(105,'B',1860,1030,590),

(106,'D',986,830,230),

(107,'B+',963,1030,930),

(108,'C+',365,230,230),

(109,'C',965,600,500),

(110,'A+',963,650,360),

(111,'F',632,206,53),

(112,'F',9860,8630,5530);

select \*from performance;

create table last\_status

(

E\_ID int,

activity\_status char (3),

O\_ID int UNIQUE,

Items varchar(10),

Payment\_methord varchar (10),

Amount int,

FOREIGN KEY (E\_ID) REFERENCES Employee\_info(E\_ID)

);

insert into last\_status (E\_ID, activity\_status, O\_ID, Items, Payment\_methord, Amount)

values

(101,'YES',6532,'RGB','CASH',589),

(102,'YES',6502,'PET','UPI',16589),

(103,'NO',1532,'RGB','CHEQUE',26589),

(104,'YES',7532,'Water','CASH',36589),

(105,'YES',9532,'SPARKLING','CC',5589),

(106,'YES',3532,'RGB','CASH',6810),

(107,'YES',6732,'SPARKLING','CC',1089),

(108,'YES',6932,'RGB','DC',96589),

(109,'YES',6572,'PET','CC',8900),

(110,'NO',6531,'RGB','CASH',6500),

(111,'YES',6539,'RGB','UPI',659),

(112,'NO',655,'WATER','DC',891);

select \*from last\_status;

select\* from employee\_info;

select\* from employee\_add;

select\* from performance;

select \*from last\_status;

drop table employee\_info;

drop table employee\_add;

drop table performance;

drop table last\_status;

desc employee\_add;

desc employee\_info;

desc performance;

desc last\_status;

truncate table employee\_add;

truncate table employee\_info;

truncate table performance;

truncate table last\_status;

select\*from employee\_info where E\_Name='Utsav';

select\*from employee\_info where age>40;

SELECT COUNT (\*) FROM employee\_info;

SELECT \* FROM employee\_info

ORDER BY E\_Name;

SELECT \* FROM employee\_info

WHERE E\_Name LIKE 'Aman%' AND Age > 20;

SELECT \* FROM employee\_info

WHERE Age BETWEEN 21 AND 25;

SELECT \* FROM employee\_info

WHERE E\_Department IN ('Spoke', 'Wholesale');

SELECT \* FROM employee\_info

WHERE Age NOT IN (20, 21);

select \* from employee\_info

where Age > 21 or E\_Department = 'MGR';

select \* from employee\_info

where Age > 21 and E\_Department = 'MGR';

SELECT \* FROM performance

WHERE achived IS NULL;

SELECT \* FROM performance

where achived IS NOT NULL;

SELECT COUNT (\*) AS numberofemployee

FROM Employee\_info;

SELECT AVG(E\_salary) AS Averagesalary

FROM Employee\_info;

SELECT MAX(E\_salary) AS Max\_salary

FROM Employee\_info;

SELECT COUNT (\*) AS NumberOfdepartment

FROM Employee\_info

GROUP BY E\_department;

SELECT Employee\_info.E\_ID, Employee\_info.E\_Name, Employee\_add.E\_Address AS Address

FROM Employee\_info

INNER JOIN Employee\_add ON Employee\_info.E\_ID = Employee\_add.E\_ID;

SELECT Employee\_info.E\_ID,Employee\_info.E\_Name,Employee\_add.Email

FROM Employee\_add

LEFT JOIN Employee\_info ON Employee\_info.E\_ID = Employee\_add.E\_ID;

SELECT Employee\_info.E\_ID,Employee\_info.E\_Name,Employee\_add.Email

FROM Employee\_add

Right JOIN Employee\_info ON Employee\_info.E\_ID = Employee\_add.E\_ID;

SELECT \* FROM Employee\_info

WHERE age > 50

UNION

SELECT \* FROM Employee\_info

WHERE E\_Salary < 45000;

SELECT EI.E\_ID, EI.E\_Name, P.Performance

FROM Employee\_info EI

LEFT JOIN performance P ON EI.E\_ID = P.E\_ID

UNION

SELECT P.E\_ID, NULL AS E\_Name, P.Performance

FROM performance P

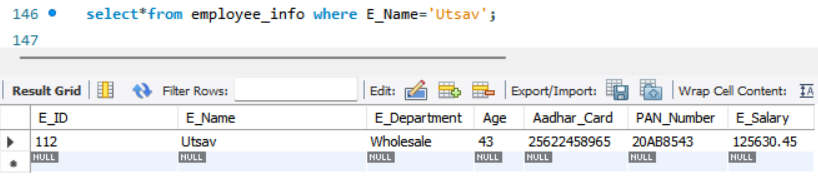
LEFT JOIN Employee\_info EI ON EI.E\_ID = P.E\_ID;

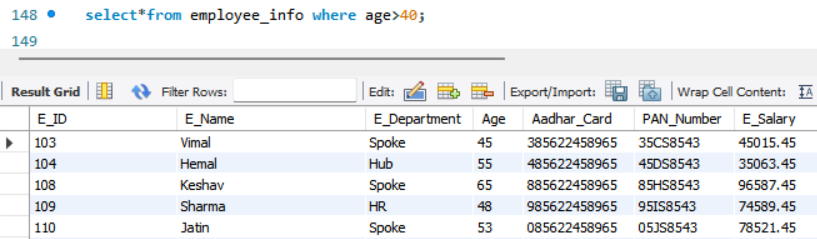
SELECT \*

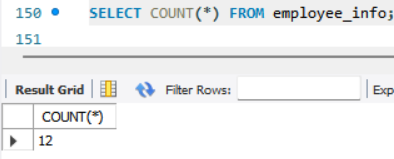
FROM Employee\_info

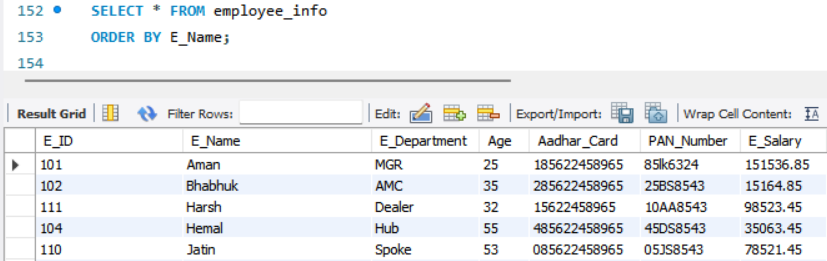
NATURAL JOIN performance;

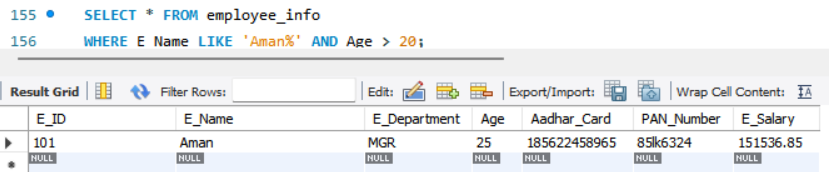
* SQL QUERIES WITH OUTPUT:

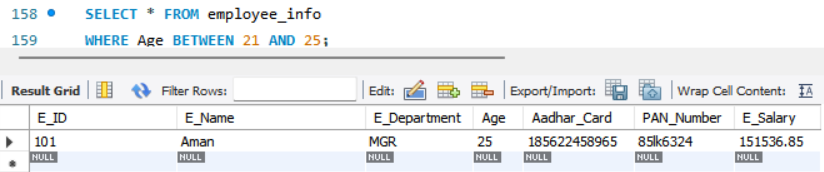


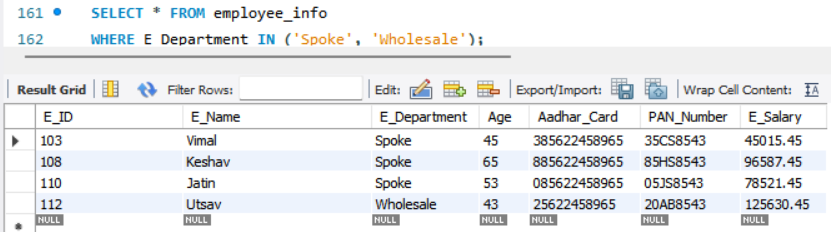


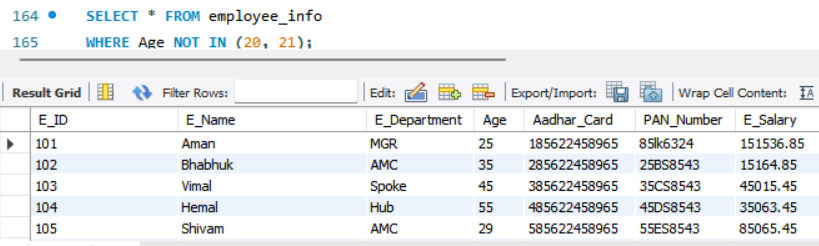


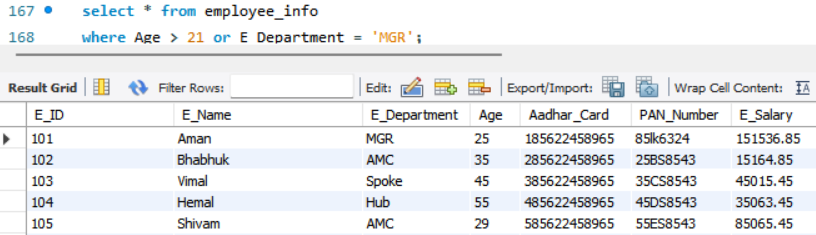


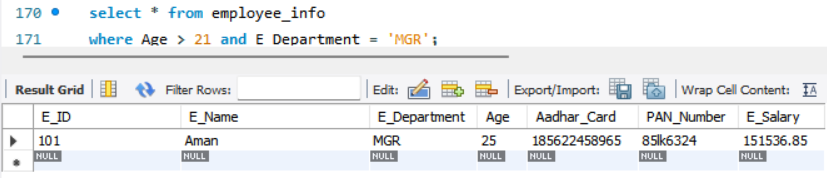


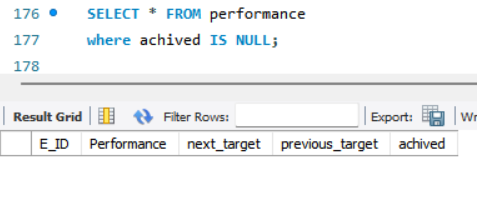


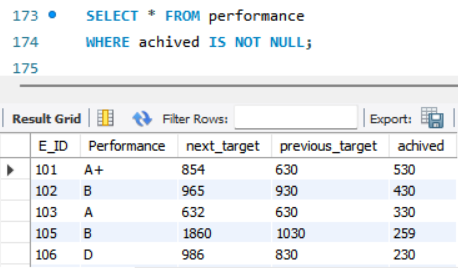


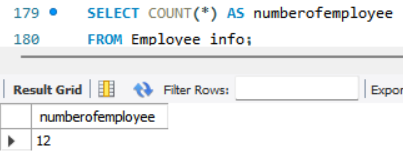


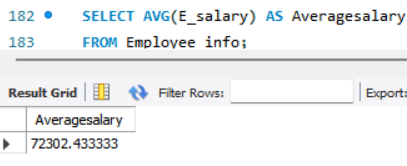


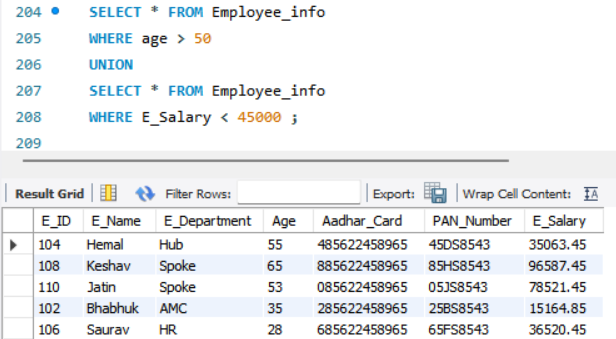


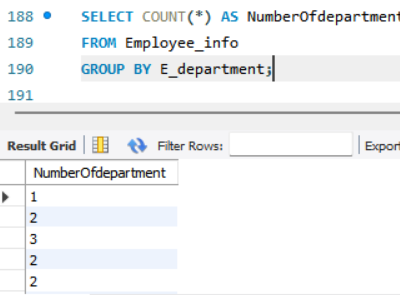


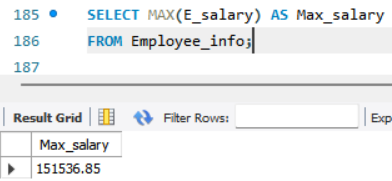


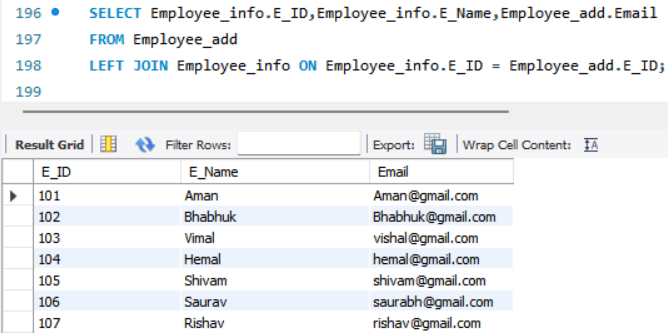


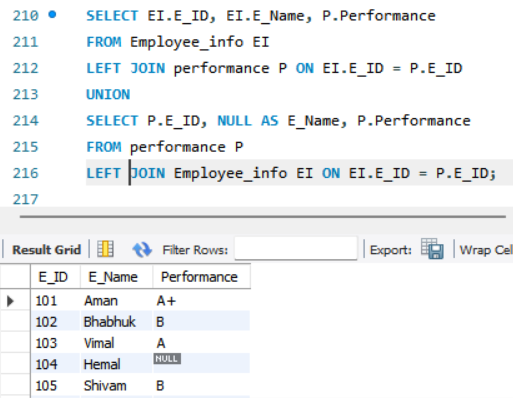


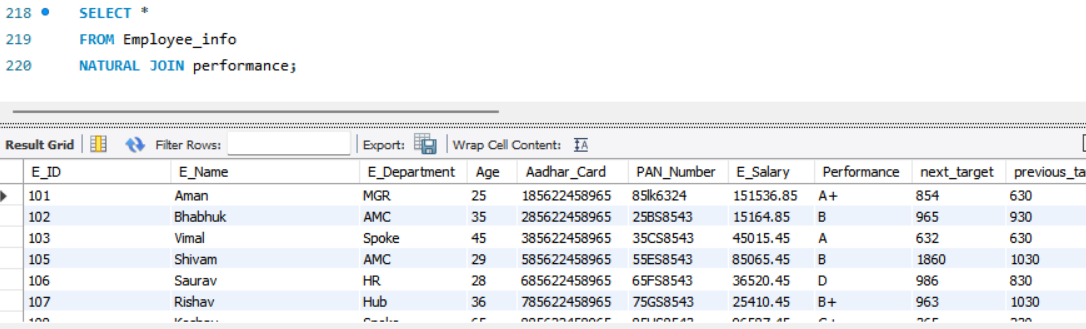


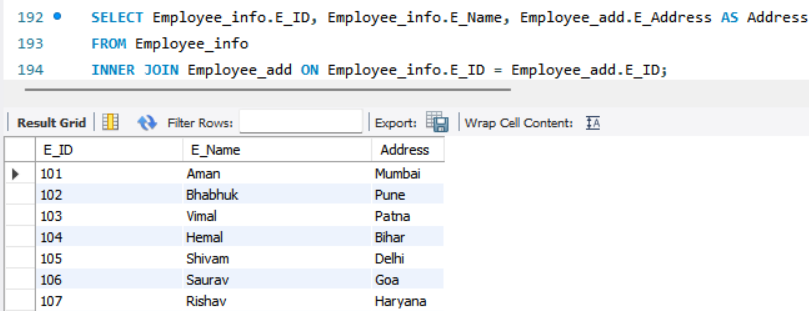


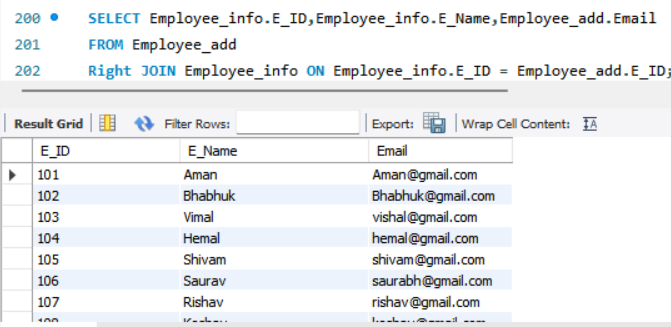












* SUMMARY:

**Key Highlights:**

* A company-themed database named **Coca\_Cola** was successfully designed and implemented using SQL.
* The system revolves around employee data management, covering personal details, contact information, performance tracking, and recent activities.
* The database includes **four interrelated tables**:
  + Employee\_info
  + Employee\_add
  + Performance
  + Last\_status
* Strong relational integrity was ensured using **primary and foreign keys**.
* Real-world SQL operations like **joins, aggregations, filtering, ordering, and unions** were applied effectively.

**Modular Table Setup**:

* Each table serves a **specific modular purpose**, enhancing readability and maintainability.
  + Employee\_info: Core employee details and salary.
  + Employee\_add: Contact and address data.
  + Performance: Evaluation metrics and targets.
  + Last\_status: Tracks final activities and transactions.

### **Learning Outcomes:**

* Gained hands-on experience in:
  + **Creating and managing relational databases**.
  + Writing and optimizing **SQL queries**.
  + **Understanding relationships** using keys and constraints.
  + Applying **data validation** using CHECK, UNIQUE, and NOT NULL.

### **Project Application:**

* This structure can be used in real-world company systems for:
  + HR management
  + Sales tracking
  + Performance analytics
  + Employee lifecycle monitoring

Objectives:

* ✅ Created a normalized, efficient database.
* ✅ Ensured data consistency using constraints.
* ✅ Successfully implemented CRUD operations and advanced queries.
* ✅ Demonstrated relationship modeling through joins and unions.

Relationships:

* One-to-one relationships connect Employee\_info with all other tables using E\_ID as the foreign key.
* Proper use of SQL joins allows multi-table analysis and insights.
* CONCLUSION:

### **Observations:**

* The Coca\_Cola database project demonstrates a clear understanding of **relational database concepts** and practical implementation using SQL.
* The design emphasizes **data normalization, integrity**, and **real-world applicability** in managing organizational data such as employee records, performance tracking, and operational activity.
* By creating structured, interlinked tables, this project ensures **efficiency, scalability**, and **clarity** in data retrieval and manipulation.

### **Limitations:**

* The current system does not implement **advanced security measures** such as role-based access control.
* The project focuses on **static data**; integration with real-time updates or dynamic front-end interfaces could further enhance its usability.
* **Data redundancy** was minimized but could be optimized further with stored procedures and triggers.

**In conclusion,** this DBMS mini project on the Coca\_Cola company database has successfully demonstrated the practical implementation of database design principles using SQL. It involved the creation of multiple interrelated tables to manage employee information, contact details, performance tracking, and transaction history.

* Through this project, a strong foundation in concepts such as **primary and foreign keys, normalization, data integrity, and query operations** was established. The system supports a variety of SQL operations including **joins, filters, aggregation, ordering, and conditions**, reflecting real-world usage scenarios.
* This project not only enhanced technical skills but also highlighted the importance of **structured and efficient data management** in modern businesses. With some future improvements like automation, front-end integration, and security implementation, this database could serve as a core backend for an actual corporate system.