CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

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:

Name: Utsav Raj

UID: 23BCA10816

Section: 4(A)

Submitted to:

Name: Arvinder Singh

Designation: Assist. Prof.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Table of Contents

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

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

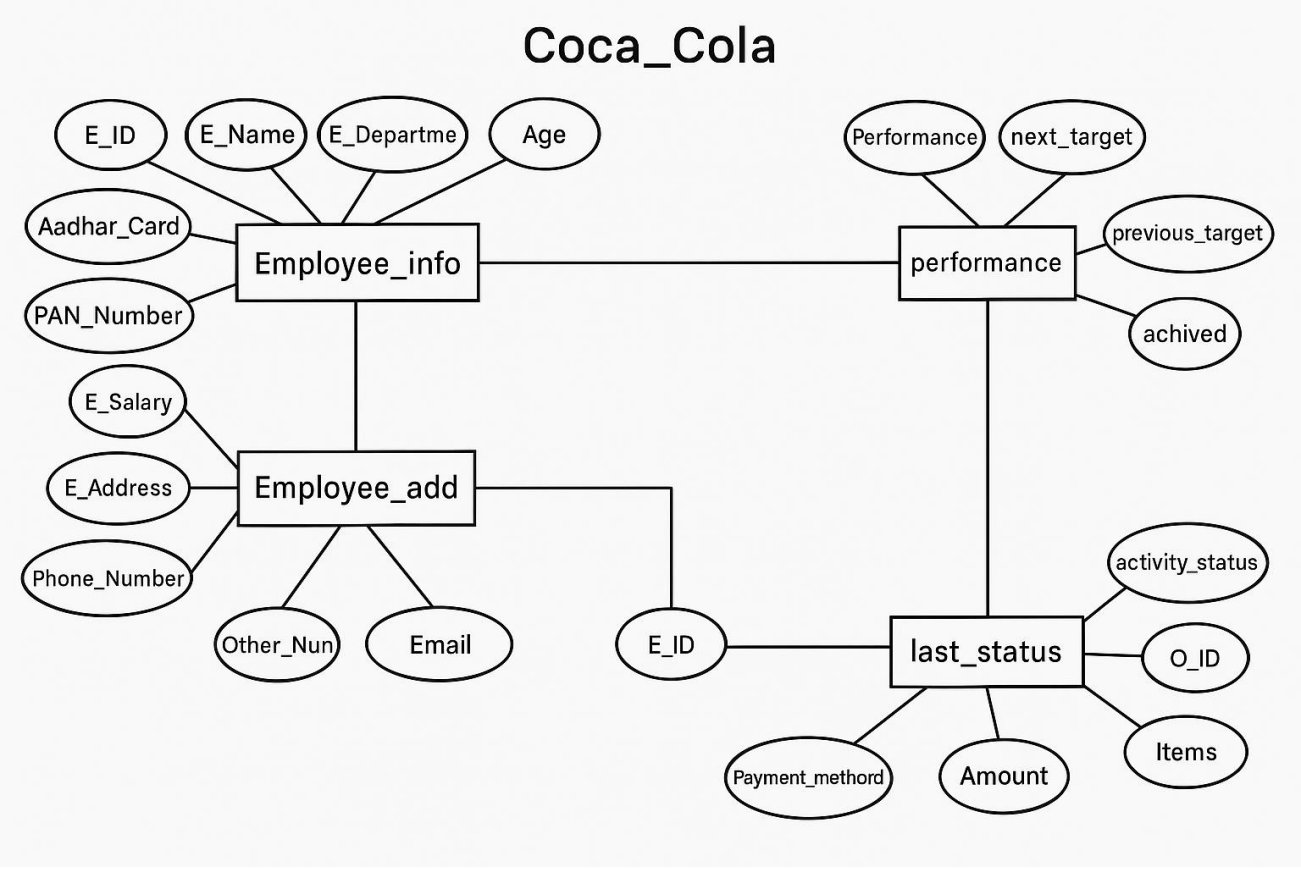
Input Method:

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

```
4  create table Employee_info
5  (
6      E_ID int primary key NOT NULL,
7      E_Name varchar(20),
8      E_Department char(10),
9      Age int CHECK(age>21),
10     Aadhar_Card varchar(15) unique NOT NULL,
11     PAN_Number Varchar(15)
12 );
```

Employee_add

```
44 create table Employee_add
45 (
46     E_ID int,
47     E_Address varchar(20),
48     Phone_Number varchar(15),
49     Other_Number varchar(15),
50     Email varchar(30),
51     FOREIGN KEY (E_ID) REFERENCES Employee_info(E_ID)
52 );
```

Performance

```
71 •  create table performance
72    (
73            E_ID int,
74            Performance char(2),
75            next_target int ,
76            previous_target int,
77            achived int,
78            FOREIGN KEY (E_ID) REFERENCES Employee_info(E_ID)
79    );
```

last_status

```
98 •  create table last_status
99    (
100           E_ID int,
101           activity_status char(3),
102           O_ID int UNIQUE,
103           Items varchar(10),
104           Payment_methord varchar(10),
105           Amount int,
106           FOREIGN KEY (E_ID) REFERENCES Employee_info(E_ID)
107   );
```

- **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:**

```
146 •   select*from employee_info where E_Name='Utsav';
147
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary
112	Utsav	Wholesale	43	25622458965	20AB8543	125630.45
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
148 •   select*from employee_info where age>40;
149
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary
103	Vimal	Spoke	45	385622458965	35CS8543	45015.45
104	Hemal	Hub	55	485622458965	45DS8543	35063.45
108	Keshav	Spoke	65	885622458965	85HS8543	96587.45
109	Sharma	HR	48	985622458965	95IS8543	74589.45
110	Jatin	Spoke	53	085622458965	05JS8543	78521.45

```
150 •   SELECT COUNT(*) FROM employee_info;
151
```

Result Grid | Filter Rows: | Exp

COUNT(*)
12

```
152 •   SELECT * FROM employee_info
153     ORDER BY E_Name;
154
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary
101	Aman	MGR	25	185622458965	85lk6324	151536.85
102	Bhabhuk	AMC	35	285622458965	25BS8543	15164.85
111	Harsh	Dealer	32	15622458965	10AA8543	98523.45
104	Hemal	Hub	55	485622458965	45DS8543	35063.45
110	Jatin	Spoke	53	085622458965	05JS8543	78521.45

```
155 •  SELECT * FROM employee_info
156    WHERE E Name LIKE 'Aman%' AND Age > 20;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary
101	Aman	MGR	25	185622458965	85lk6324	151536.85
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
158 •  SELECT * FROM employee_info
159    WHERE Age BETWEEN 21 AND 25;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary
101	Aman	MGR	25	185622458965	85lk6324	151536.85
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
161 •  SELECT * FROM employee_info
162    WHERE E Department IN ('Spoke', 'Wholesale');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary
103	Vimal	Spoke	45	385622458965	35CS8543	45015.45
108	Keshav	Spoke	65	885622458965	85HS8543	96587.45
110	Jatin	Spoke	53	085622458965	05JS8543	78521.45
112	Utsav	Wholesale	43	25622458965	20AB8543	125630.45
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
164 •  SELECT * FROM employee_info
165    WHERE Age NOT IN (20, 21);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary
101	Aman	MGR	25	185622458965	85lk6324	151536.85
102	Bhabhuk	AMC	35	285622458965	25BS8543	15164.85
103	Vimal	Spoke	45	385622458965	35CS8543	45015.45
104	Hemal	Hub	55	485622458965	45DS8543	35063.45
105	Shivam	AMC	29	585622458965	55ES8543	85065.45

```
167 •  select * from employee_info
168    where Age > 21 or E Department = 'MGR';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary
101	Aman	MGR	25	185622458965	85lk6324	151536.85
102	Bhabhuk	AMC	35	285622458965	25BS8543	15164.85
103	Vimal	Spoke	45	385622458965	35CS8543	45015.45
104	Hemal	Hub	55	485622458965	45DS8543	35063.45
105	Shivam	AMC	29	585622458965	55ES8543	85065.45

```
170 •  select * from employee_info
171    where Age > 21 and E Department = 'MGR';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary
101	Aman	MGR	25	185622458965	85lk6324	151536.85
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
176 •  SELECT * FROM performance
177    where achived IS NULL;
178
```

Result Grid | Filter Rows: | Export: | Wr

E_ID	Performance	next_target	previous_target	achived

```
173 •  SELECT * FROM performance
174    WHERE achived IS NOT NULL;
175
```

Result Grid | Filter Rows: | Export:

E_ID	Performance	next_target	previous_target	achived
101	A+	854	630	530
102	B	965	930	430
103	A	632	630	330
105	B	1860	1030	259
106	D	986	830	230

```
179 •   SELECT COUNT(*) AS numberofemployee
180     FROM Employee info;
```

Result Grid | Filter Rows: | Expor

numberofemployee
12

```
182 •   SELECT AVG(E_salary) AS Averagesalary
183     FROM Employee info;
```

Result Grid | Filter Rows: | Export:

Averagesalary
72302.433333

```
204 •   SELECT * FROM Employee_info
205     WHERE age > 50
206     UNION
207     SELECT * FROM Employee_info
208     WHERE E_Salary < 45000 ;
209
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary
104	Hemal	Hub	55	485622458965	45DS8543	35063.45
108	Keshav	Spoke	65	885622458965	85HS8543	96587.45
110	Jatin	Spoke	53	085622458965	05JS8543	78521.45
102	Bhabhuk	AMC	35	285622458965	25BS8543	15164.85
106	Saurav	HR	28	685622458965	65FS8543	36520.45

```
188 •  SELECT COUNT(*) AS NumberOfdepartment
189    FROM Employee_info
190    GROUP BY E_department;
191
```

Result Grid | Filter Rows: | Export

NumberOfdepartment
1
2
3
2
2

```
185 •  SELECT MAX(E_salary) AS Max_salary
186    FROM Employee_info;
187
```

Result Grid | Filter Rows: | Exp

Max_salary
151536.85

```
196 •  SELECT Employee_info.E_ID,Employee_info.E_Name,Employee_add.Email
197    FROM Employee_add
198    LEFT JOIN Employee_info ON Employee_info.E_ID = Employee_add.E_ID;
199
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

E_ID	E_Name	Email
101	Aman	Aman@gmail.com
102	Bhabhuk	Bhabhuk@gmail.com
103	Vimal	vishal@gmail.com
104	Hemal	hemal@gmail.com
105	Shivam	shivam@gmail.com
106	Saurav	saurabh@gmail.com
107	Rishav	rishav@gmail.com

```
210 •   SELECT EI.E_ID, EI.E_Name, P.Performance
211     FROM Employee_info EI
212     LEFT JOIN performance P ON EI.E_ID = P.E_ID
213     UNION
214     SELECT P.E_ID, NULL AS E_Name, P.Performance
215     FROM performance P
216     LEFT JOIN Employee_info EI ON EI.E_ID = P.E_ID;
217
```

Result Grid | Filter Rows: | Export: | Wrap Cel

E_ID	E_Name	Performance
101	Aman	A+
102	Bhabhuk	B
103	Vimal	A
104	Hemal	NULL
105	Shivam	B

```
218 •   SELECT *
219     FROM Employee_info
220     NATURAL JOIN performance;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

E_ID	E_Name	E_Department	Age	Aadhar_Card	PAN_Number	E_Salary	Performance	next_target	previous_ta
101	Aman	MGR	25	185622458965	85lk6324	151536.85	A+	854	630
102	Bhabhuk	AMC	35	285622458965	25BS8543	15164.85	B	965	930
103	Vimal	Spoke	45	385622458965	35CS8543	45015.45	A	632	630
105	Shivam	AMC	29	585622458965	55ES8543	85065.45	B	1860	1030
106	Saurav	HR	28	685622458965	65FS8543	36520.45	D	986	830
107	Rishav	Hub	36	785622458965	75GS8543	25410.45	B+	963	1030

```
192 •  SELECT Employee_info.E_ID, Employee_info.E_Name, Employee_add.E_Address AS Address
193    FROM Employee_info
194    INNER JOIN Employee_add ON Employee_info.E_ID = Employee_add.E_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

E_ID	E_Name	Address
101	Aman	Mumbai
102	Bhabhuk	Pune
103	Vimal	Patna
104	Hemal	Bihar
105	Shivam	Delhi
106	Saurav	Goa
107	Rishav	Haryana

```
200 •  SELECT Employee_info.E_ID,Employee_info.E_Name,Employee_add.Email
201    FROM Employee_add
202    Right JOIN Employee_info ON Employee_info.E_ID = Employee_add.E_ID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

E_ID	E_Name	Email
101	Aman	Aman@gmail.com
102	Bhabhuk	Bhabhuk@gmail.com
103	Vimal	vishal@gmail.com
104	Hemal	hemal@gmail.com
105	Shivam	shivam@gmail.com
106	Saurav	saurabh@gmail.com
107	Rishav	rishav@gmail.com

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

- **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.