

INFO 579 Week 16 Final Project Report

INFO 579: SQL/NoSQL Databases for Data and Information Sciences

Group Members: Sharad Talekar, Avi Kumar Talaviya, Ayesha Siddiqua Guffran, Prathamesh Deshpande, Ankur Singh

Name: INFO_579_SEC001_2254-1 – 4

- 1. Create tables using a database system. Insert data into the database tables. You must provide the DDL (CREATE TABLE statements), INSERT statements, and SELECT statements.**

```
CREATE TABLE CUSTOMERS (
    CustomerID INT PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    CustomerEmail VARCHAR(255) UNIQUE,
    Age INT,
    is_verified BIT,
    BirthDay INT,
    BirthMonth INT,
    BirthYear INT
);
```

```
mysql> CREATE TABLE CUSTOMERS (
    ->     CustomerID INT PRIMARY KEY,
    ->     FirstName VARCHAR(100),
    ->     LastName VARCHAR(100),
    ->     CustomerEmail VARCHAR(255) UNIQUE,
    ->     Age INT,
    ->     is_verified BIT,
    ->     BirthDay INT,
    ->     BirthMonth INT,
    ->     BirthYear INT
    -> );
Query OK, 0 rows affected (0.07 sec)
```

```
CREATE TABLE CATEGORIES (
    CategoryID INT PRIMARY KEY,
    CategoryName VARCHAR(100),
    CategoryDescription TEXT,
    ImageUrl VARCHAR(255)
);
```

```
mysql> CREATE TABLE CATEGORIES (
    ->     CategoryID INT PRIMARY KEY,
    ->     CategoryName VARCHAR(100),
    ->     CategoryDescription TEXT,
    ->     ImageUrl VARCHAR(255)
    -> );
Query OK, 0 rows affected (0.08 sec)
```

```
CREATE TABLE ADDRESSES (
    AddressID INT PRIMARY KEY,
    PinCode VARCHAR(20),
    Num_street VARCHAR(255),
    City VARCHAR(100),
    State VARCHAR(100),
    Country VARCHAR(100)
);
```

```
mysql> CREATE TABLE ADDRESSES (
    ->     AddressID INT PRIMARY KEY,
    ->     PinCode VARCHAR(20),
    ->     Num_street VARCHAR(255),
    ->     City VARCHAR(100),
    ->     State VARCHAR(100),
    ->     Country VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
CREATE TABLE SHOPPERS (
    DeliveryAgentID INT PRIMARY KEY,
    Name VARCHAR(100),
    Contact VARCHAR(50),
    CurrentLocation VARCHAR(255),
    DeliveryMode VARCHAR(50),
    Email VARCHAR(255) UNIQUE
);
```

```
mysql> CREATE TABLE SHOPPERS (
    ->     DeliveryAgentID INT PRIMARY KEY,
    ->     Name VARCHAR(100),
    ->     Contact VARCHAR(50),
    ->     CurrentLocation VARCHAR(255),
    ->     DeliveryMode VARCHAR(50),
    ->     Email VARCHAR(255) UNIQUE
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
CREATE TABLE PHONENUMBER (
    PhoneID INT PRIMARY KEY,
    CustomerID INT NOT NULL,
    PhoneNumber VARCHAR(20),
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
);
```

```
mysql> CREATE TABLE PHONENUMBER (
    ->     PhoneID INT PRIMARY KEY,
    ->     CustomerID INT NOT NULL,
    ->     PhoneNumber VARCHAR(20),
    ->     FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

```
CREATE TABLE CART (
    CartID INT PRIMARY KEY,
    Cart_Status VARCHAR(50),
    Timestamp DATETIME,
    CustomerID INT NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
);
```

```
mysql> CREATE TABLE CART (
    ->     CartID INT PRIMARY KEY,
    ->     Cart_Status VARCHAR(50),
    ->     Timestamp DATETIME,
    ->     CustomerID INT NOT NULL,
    ->     FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

```
CREATE TABLE PRODUCTS (
    ProductID INT PRIMARY KEY,
    Name VARCHAR(255),
    Description TEXT,
    Price DECIMAL(10, 2),
    CategoryID INT NOT NULL,
    FOREIGN KEY (CategoryID) REFERENCES CATEGORIES(CategoryID)
);
```

```
mysql> CREATE TABLE PRODUCTS (
    ->     ProductID INT PRIMARY KEY,
    ->     Name VARCHAR(255),
    ->     Description TEXT,
    ->     Price DECIMAL(10, 2),
    ->     CategoryID INT NOT NULL,
    ->     FOREIGN KEY (CategoryID) REFERENCES CATEGORIES(CategoryID)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
CREATE TABLE STORES (
    StoreID INT PRIMARY KEY,
    Location VARCHAR(255),
    StoreName VARCHAR(100),
    StoreDescription TEXT,
    AddressID INT NOT NULL,
    FOREIGN KEY (AddressID) REFERENCES ADDRESSES(AddressID)
);
mysql> CREATE TABLE STORES (
    ->     StoreID INT PRIMARY KEY,
    ->     Location VARCHAR(255),
    ->     StoreName VARCHAR(100),
    ->     StoreDescription TEXT,
    ->     AddressID INT NOT NULL,
    ->     FOREIGN KEY (AddressID) REFERENCES ADDRESSES(AddressID)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

```
CREATE TABLE CUSTOMERADDRESS (
    CustomerID INT NOT NULL,
    AddressID INT NOT NULL,
    PRIMARY KEY (CustomerID, AddressID),
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID),
    FOREIGN KEY (AddressID) REFERENCES ADDRESSES(AddressID)
);
mysql> CREATE TABLE CUSTOMERADDRESS (
    ->     CustomerID INT NOT NULL,
    ->     AddressID INT NOT NULL,
    ->     PRIMARY KEY (CustomerID, AddressID),
    ->     FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID),
    ->     FOREIGN KEY (AddressID) REFERENCES ADDRESSES(AddressID)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

```
CREATE TABLE PRODUCTDETAILS (
    ProductID INT PRIMARY KEY,
    ProductWeight DECIMAL(10, 2),
    ProductHeight DECIMAL(10, 2),
    ProductLength DECIMAL(10, 2),
    ProductDiameter DECIMAL(10, 2),
    ProductInventory INT,
    FOREIGN KEY (ProductID) REFERENCES PRODUCTS(ProductID)
);
```

```
mysql> CREATE TABLE PRODUCTDETAILS (
    ->     ProductID INT PRIMARY KEY,
    ->     ProductWeight DECIMAL(10, 2),
    ->     ProductHeight DECIMAL(10, 2),
    ->     ProductLength DECIMAL(10, 2),
    ->     ProductDiameter DECIMAL(10, 2),
    ->     ProductInventory INT,
    ->     FOREIGN KEY (ProductID) REFERENCES PRODUCTS(ProductID)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
CREATE TABLE PRODUCTSTORE (
    ProductID INT NOT NULL,
    StoreID INT NOT NULL,
    PRIMARY KEY (ProductID, StoreID),
    FOREIGN KEY (ProductID) REFERENCES PRODUCTS(ProductID),
    FOREIGN KEY (StoreID) REFERENCES STORES(StoreID)
);
mysql> CREATE TABLE PRODUCTSTORE (
    ->     ProductID INT NOT NULL,
    ->     StoreID INT NOT NULL,
    ->     PRIMARY KEY (ProductID, StoreID),
    ->     FOREIGN KEY (ProductID) REFERENCES PRODUCTS(ProductID),
    ->     FOREIGN KEY (StoreID) REFERENCES STORES(StoreID)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
CREATE TABLE REVIEWS (
    ReviewID INT PRIMARY KEY,
    ProductID INT NOT NULL,
    CustomerID INT NOT NULL,
    Rating INT,
    Comments TEXT,
    Timestamp DATETIME,
    FOREIGN KEY (ProductID) REFERENCES PRODUCTS(ProductID),
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
);
mysql> CREATE TABLE REVIEWS (
    ->     ReviewID INT PRIMARY KEY,
    ->     ProductID INT NOT NULL,
    ->     CustomerID INT NOT NULL,
    ->     Rating INT,
    ->     Comments TEXT,
    ->     Timestamp DATETIME,
    ->     FOREIGN KEY (ProductID) REFERENCES PRODUCTS(ProductID),
    ->     FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

```
CREATE TABLE CARTITEMS (
    CartItemID INT PRIMARY KEY,
    CartID INT NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT,
```

```

        UnitPrice DECIMAL(10, 2),
        Timestamp DATETIME,
        FOREIGN KEY (CartID) REFERENCES CART(CartID),
        FOREIGN KEY (ProductID) REFERENCES PRODUCTS(ProductID)
    );
mysql> CREATE TABLE CARTITEMS (
        ->     CartItemID INT PRIMARY KEY,
        ->     CartID INT NOT NULL,
        ->     ProductID INT NOT NULL,
        ->     Quantity INT,
        ->     UnitPrice DECIMAL(10, 2),
        ->     Timestamp DATETIME,
        ->     FOREIGN KEY (CartID) REFERENCES CART(CartID),
        ->     FOREIGN KEY (ProductID) REFERENCES PRODUCTS(ProductID)
        -> );
Query OK, 0 rows affected (0.05 sec)

```

```

CREATE TABLE PAYMENTS (
    PaymentID INT PRIMARY KEY,
    Amount DECIMAL(10, 2),
    Currency VARCHAR(10),
    PaymentDay INT,
    PaymentMonth INT,
    PaymentYear INT,
    Quantity INT,
    ModeOfPayment VARCHAR(50),
    PaymentMethod VARCHAR(50),
    OrderID INT -- Foreign Key will be added *after* ORDERS table is created
);
mysql> CREATE TABLE PAYMENTS (
        ->     PaymentID INT PRIMARY KEY,
        ->     Amount DECIMAL(10, 2),
        ->     Currency VARCHAR(10),
        ->     PaymentDay INT,
        ->     PaymentMonth INT,
        ->     PaymentYear INT,
        ->     Quantity INT,
        ->     ModeOfPayment VARCHAR(50),
        ->     PaymentMethod VARCHAR(50),
        ->     OrderID INT -- Foreign Key will be added *after* ORDERS table is crea
ted
        -> );
Query OK, 0 rows affected (0.04 sec)

```

```

CREATE TABLE DELIVERIES (
    DeliveryID INT PRIMARY KEY,
    Actual_Delivery_Time DATETIME,
    Estimated_Delivery_Time DATETIME,
    DeliveryDate DATETIME,
    DeliveryMonth INT,
    DeliveryYear INT,
    DeliveryInstructions TEXT,
    DeliveryAddress VARCHAR(255),
    Delivery_Status VARCHAR(50),
    AddressID INT NOT NULL,
    OrderID INT, -- Foreign Key will be added *after* ORDERS table is created
    FOREIGN KEY (AddressID) REFERENCES ADDRESSES(AddressID)
);

```

```

mysql> CREATE TABLE DELIVERIES (
->     DeliveryID INT PRIMARY KEY,
->     Actual_Delivery_Time DATETIME,
->     Estimated_Delivery_Time DATETIME,
->     DeliveryDate DATETIME,
->     DeliveryMonth INT,
->     DeliveryYear INT,
->     DeliveryInstructions TEXT,
->     DeliveryAddress VARCHAR(255),
->     Delivery_Status VARCHAR(50),
->     AddressID INT NOT NULL,
->     OrderID INT, -- Foreign Key will be added *after* ORDERS table is created
->     FOREIGN KEY (AddressID) REFERENCES ADDRESSES(AddressID)
-> );
Query OK, 0 rows affected (0.03 sec)

```

```

CREATE TABLE ORDERS (
    OrderID INT PRIMARY KEY,
    OrderDay INT,
    OrderMonth INT,
    OrderYear INT,
    OrderStatus VARCHAR(50),
    CreatedAt DATETIME,
    PaymentID INT NOT NULL,
    DeliveryID INT NOT NULL,
    CustomerID INT NOT NULL,
    FOREIGN KEY (PaymentID) REFERENCES PAYMENTS(PaymentID),
    FOREIGN KEY (DeliveryID) REFERENCES DELIVERIES(DeliveryID),
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
);

```

```

mysql> CREATE TABLE ORDERS (
->     OrderID INT PRIMARY KEY,
->     OrderDay INT,
->     OrderMonth INT,
->     OrderYear INT,
->     OrderStatus VARCHAR(50),
->     CreatedAt DATETIME,
->     PaymentID INT NOT NULL,
->     DeliveryID INT NOT NULL,
->     CustomerID INT NOT NULL,
->     FOREIGN KEY (PaymentID) REFERENCES PAYMENTS(PaymentID),
->     FOREIGN KEY (DeliveryID) REFERENCES DELIVERIES(DeliveryID),
->     FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
-> );
Query OK, 0 rows affected (0.07 sec)

```

```

CREATE TABLE CUSTOMERCARTPAY (
    CustomerID INT NOT NULL,
    PaymentID INT NOT NULL,
    CartID INT NOT NULL,
    PRIMARY KEY (CustomerID, PaymentID, CartID),
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID),
    FOREIGN KEY (PaymentID) REFERENCES PAYMENTS(PaymentID),
    FOREIGN KEY (CartID) REFERENCES CART(CartID)
);

```

```
mysql> CREATE TABLE CUSTOMERCARTPAY (
->     CustomerID INT NOT NULL,
->     PaymentID INT NOT NULL,
->     CartID INT NOT NULL,
->     PRIMARY KEY (CustomerID, PaymentID, CartID),
->     FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID),
->     FOREIGN KEY (PaymentID) REFERENCES PAYMENTS(PaymentID),
->     FOREIGN KEY (CartID) REFERENCES CART(CartID)
-> );
Query OK, 0 rows affected (0.06 sec)
```

```
CREATE TABLE DELIVERYSHOPPER (
    DeliveryID INT NOT NULL,
    DeliveryAgentID INT NOT NULL,
    PRIMARY KEY (DeliveryID, DeliveryAgentID),
    FOREIGN KEY (DeliveryID) REFERENCES DELIVERIES(DeliveryID),
    FOREIGN KEY (DeliveryAgentID) REFERENCES SHOPPERS(DeliveryAgentID)
);
mysql> CREATE TABLE DELIVERYSHOPPER (
->     DeliveryID INT NOT NULL,
->     DeliveryAgentID INT NOT NULL,
->     PRIMARY KEY (DeliveryID, DeliveryAgentID),
->     FOREIGN KEY (DeliveryID) REFERENCES DELIVERIES(DeliveryID),
->     FOREIGN KEY (DeliveryAgentID) REFERENCES SHOPPERS(DeliveryAgentID)
-> );
Query OK, 0 rows affected (0.04 sec)
```

```
CREATE TABLE ORDERITEMS (
    OrderItemID INT PRIMARY KEY,
    OrderID INT NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT,
    FOREIGN KEY (OrderID) REFERENCES ORDERS(OrderID),
    FOREIGN KEY (ProductID) REFERENCES PRODUCTS(ProductID)
);
mysql> CREATE TABLE ORDERITEMS (
->     OrderItemID INT PRIMARY KEY,
->     OrderID INT NOT NULL,
->     ProductID INT NOT NULL,
->     Quantity INT,
->     FOREIGN KEY (OrderID) REFERENCES ORDERS(OrderID),
->     FOREIGN KEY (ProductID) REFERENCES PRODUCTS(ProductID)
-> );
Query OK, 0 rows affected (0.04 sec)
```

(a) Columns, Primary Key (PK), Data Type and length, and NULL/NOT NULL need to be implemented.

```
CREATE DATABASE instacart;
USE instacart;
CREATE TABLE PHONENUMBER (
    PhoneID INT PRIMARY KEY,
    CustomerID INT NOT NULL,
    PhoneNumber VARCHAR(20),
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID);
```

```
44 • CREATE DATABASE instacart;
45 • USE instacart;
46 • CREATE TABLE PHONENUMBER (
47     PhoneID INT PRIMARY KEY,
48     CustomerID INT NOT NULL,
49     PhoneNumber VARCHAR(20),
50     FOREIGN KEY (CustomerID) REFERENCES CUSTOMERS(CustomerID)
51 );
```

Action Output

#	Time	Action
10	12:11:06	SELECT * FROM PHONENUMBER ORDER BY PhoneID LIMIT 0, 1000

(b) Show the definition (DDL) that you implemented (not in a graphical view).

```
SELECT * FROM CUSTOMERS ORDER BY CustomerID;
SELECT * FROM CATEGORIES ORDER BY CategoryID;
SELECT * FROM ADDRESSES ORDER BY AddressID;
SELECT * FROM SHOPPERS ORDER BY DeliveryAgentID;
SELECT * FROM CART ORDER BY CartID;
SELECT * FROM PRODUCTS ORDER BY ProductID;
SELECT * FROM STORES ORDER BY StoreID;
SELECT * FROM PHONENUMBER ORDER BY PhoneID;
SELECT * FROM CUSTOMERADDRESS ORDER BY CustomerID, AddressID;
SELECT * FROM PRODUCTDETAILS
ORDER BY ProductID;
SELECT * FROM PRODUCTSTORE ORDER
BY ProductID, StoreID;
SELECT * FROM REVIEWS ORDER BY
ReviewID;
SELECT * FROM CARTITEMS ORDER BY
CartItemID;
SELECT * FROM PAYMENTS ORDER BY
PaymentID;
SELECT * FROM DELIVERIES ORDER BY
DeliveryID;
SELECT * FROM ORDERS ORDER BY OrderID;
SELECT * FROM CUSTOMERCARTPAY ORDER BY CustomerID, PaymentID, CartID;
SELECT * FROM DELIVERYSHOPPER ORDER BY DeliveryID, DeliveryAgentID;
SELECT * FROM ORDERITEMS ORDER BY OrderItemID;
```

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

CustomerID	FirstName	LastName	CustomerEmail	Age	is_verified	BirthDay	BirthMonth	BirthYear
1	Asha	Rao	asha.rao@example.com	29	1	12	5	1996
2	Vishnu	Panyam	vishnu.p@example.com	32	1	3	9	1993
3	Emily	Chen	emily.c@example.com	27	1	15	7	1998
4	Miguel	Santos	miguel.s@example.com	40	1	8	2	1985
5	Sara	Nair	sara.nair@example.com	23	1	10	10	2002
6	Reena	Singh	reena.singh@example.com	28	0	7	5	1997
7	Tom	Lee	tom.lee@example.com	34	1	21	4	1991
8	Priya	Sharma	priya.sharma@example.com	30	1	3	1	1995
9	David	Miller	david.miller@example.com	31	1	14	12	1993
10	Angela	Kaur	angela.kaur@example.com	26	0	2	6	1999

CUSTOMERS 30 X CATEGORIES 31 ADDRESSES 32 SHOPPERS 33 CART 34 PRODUCTS 35 STORES 36

(c) Insert the complete set of data that you have come up with and show the insert statements used.

```

222 • INSERT INTO CUSTOMERS (CustomerID, FirstName, LastName, CustomerEmail, Age, is_verified, BirthDay, BirthMonth, BirthYear) VALUES
223 (1, 'Asha', 'Rao', 'asha.rao@example.com', 29, 1, 12, 5, 1996),
224 (2, 'Vishnu', 'Panyam', 'vishnu.p@example.com', 32, 1, 3, 9, 1993),
225 (3, 'Emily', 'Chen', 'emily.c@example.com', 27, 1, 15, 7, 1998),
226 (4, 'Miguel', 'Santos', 'miguel.s@example.com', 40, 1, 8, 2, 1985),
227 (5, 'Sara', 'Nair', 'sara.nair@example.com', 23, 1, 10, 10, 2002),
228 (6, 'Reena', 'Singh', 'reena.singh@example.com', 28, 0, 7, 5, 1997),
229 (7, 'Tom', 'Lee', 'tom.lee@example.com', 34, 1, 21, 4, 1991),
230 (8, 'Priya', 'Sharma', 'priya.sharma@example.com', 30, 1, 3, 1, 1995),
231 (9, 'David', 'Miller', 'david.miller@example.com', 31, 1, 14, 12, 1993),
232 (10, 'Angela', 'Kaur', 'angela.kaur@example.com', 26, 0, 2, 6, 1999);
```

```

Output  
Action Output  
# Time Action Message Duration / Fetch  
55 16:08:13 INSERT INTO CUSTOMERS (CustomerID, FirstName, LastName, CustomerEmail, ... 10 rows(s) affected Records: 10 Duplicates: 0 Warnings: 0 0.000 sec

INSERT INTO CUSTOMERS (CustomerID, FirstName, LastName, CustomerEmail, Age, is\_verified, BirthDay, BirthMonth, BirthYear) VALUES  
(1, 'Asha', 'Rao', 'asha.rao@example.com', 29, 1, 12, 5, 1996),  
(2, 'Vishnu', 'Panyam', 'vishnu.p@example.com', 32, 1, 3, 9, 1993),  
(3, 'Emily', 'Chen', 'emily.c@example.com', 27, 1, 15, 7, 1998),  
(4, 'Miguel', 'Santos', 'miguel.s@example.com', 40, 1, 8, 2, 1985),  
(5, 'Sara', 'Nair', 'sara.nair@example.com', 23, 1, 10, 10, 2002),  
(6, 'Reena', 'Singh', 'reena.singh@example.com', 28, 0, 7, 5, 1997),  
(7, 'Tom', 'Lee', 'tom.lee@example.com', 34, 1, 21, 4, 1991),  
(8, 'Priya', 'Sharma', 'priya.sharma@example.com', 30, 1, 3, 1, 1995),  
(9, 'David', 'Miller', 'david.miller@example.com', 31, 1, 14, 12, 1993),  
(10, 'Angela', 'Kaur', 'angela.kaur@example.com', 26, 0, 2, 6, 1999);

**2. Create a variety of SQL queries to retrieve data from one or many tables:**

**1. Retrieve the data from each table by using the SELECT \* statement and order by PK column(s). Show the output. Make sure you show the print screen of the complete set of rows and columns. The rows must be ordered by PK column(s).**

USE instacart;  
SELECT \* FROM ORDERS  
ORDER BY OrderID;

```

1 • USE instacart;
2
3 # Query 1: List all orders in the DB in order they were created.
4 #
5
6 • SELECT * FROM ORDERS
7 ORDER BY OrderID;

```

| OrderID | OrderDay | OrderMonth | OrderYear | OrderStatus | CreatedAt           | Paid | DeliveryID | CustomerID |
|---------|----------|------------|-----------|-------------|---------------------|------|------------|------------|
| 1       | 28       | 10         | 2025      | Paid        | 2025-11-05 12:27:08 | 1    | 1          | 2          |
| 2       | 29       | 10         | 2025      | Shipped     | 2025-11-05 12:27:08 | 2    | 2          | 3          |
| 3       | 30       | 10         | 2025      | Pending     | 2025-11-05 12:27:08 | 3    | 3          | 1          |
| 4       | 25       | 10         | 2025      | Delivered   | 2025-11-05 12:27:08 | 4    | 4          | 4          |
| 5       | 27       | 10         | 2025      | Canceled    | 2025-11-05 12:27:08 | 5    | 5          | 5          |
| 6       | 26       | 10         | 2025      | Paid        | 2025-11-05 12:27:08 | 6    | 6          | 6          |
| 7       | 24       | 10         | 2025      | Refunded    | 2025-11-05 12:27:08 | 7    | 7          | 7          |
| 8       | 23       | 10         | 2025      | Paid        | 2025-11-05 12:27:08 | 8    | 8          | 8          |
| 9       | 22       | 10         | 2025      | Shipped     | 2025-11-05 12:27:08 | 9    | 9          | 9          |
| 10      | 21       | 10         | 2025      | Pending     | 2025-11-05 12:27:08 | 10   | 10         | 10         |

## 2. Write an SQL involving the junction table and two other related tables. You must use the INNER JOIN to connect with all three tables.

```
SELECT ps.ProductID, p.Name AS ProductName, ps.StoreID, s.StoreName, s.Location
FROM PRODUCTSTORE ps JOIN PRODUCTS p USING(ProductID)
JOIN STORES s
USING(StoreID)
ORDER BY ps.StoreID, ps.ProductID;
```

```
9 # Query 2: Show the Store details for each product sold
10 #
11
12 • SELECT ps.ProductID, p.Name AS ProductName, ps.StoreID, s.StoreName, s.Location
13 FROM PRODUCTSTORE ps JOIN PRODUCTS p USING(ProductID)
14 JOIN STORES s USING(StoreID)
15 ORDER BY ps.StoreID, ps.ProductID;
16
```

|    | ProductID     | ProductName | StoreID             | StoreName      | Location |
|----|---------------|-------------|---------------------|----------------|----------|
| 1  | Smartphone X  | 1           | BayTech Electronics | Phoenix, AZ    |          |
| 2  | Laptop Pro    | 2           | SmartMart           | Tempe, AZ      |          |
| 3  | Air Fryer     | 2           | SmartMart           | Tempe, AZ      |          |
| 4  | Yoga Mat      | 3           | Gotham Fitness      | Scottsdale, AZ |          |
| 6  | Running Shoes | 3           | Gotham Fitness      | Scottsdale, AZ |          |
| 5  | Organic Rice  | 4           | Daily Grains        | Mesa, AZ       |          |
| 7  | Classic Novel | 5           | FabReads            | Glendale, AZ   |          |
| 8  | T-shirt       | 6           | Fashion House       | Chandler, AZ   |          |
| 9  | Remote Car    | 7           | Toyland             | Gilbert, AZ    |          |
| 10 | Face Wash     | 8           | Glow Beauty         | Peoria, AZ     |          |

## 3. Write an SQL by including two or more tables and using the LEFT OUTER JOIN. Show the results and sort the results by key field(s).

```
SELECT c.CustomerID, c.FirstName, c.LastName, COUNT(o.OrderID) AS OrderCount
FROM CUSTOMERS c LEFT JOIN ORDERS o USING(CustomerID)
GROUP BY c.CustomerID, c.FirstName, c.LastName
ORDER BY c.CustomerID;
Query that answers the Question
SELECT c.CustomerID, c.FirstName,
c.LastName, o.OrderID, o.OrderStatus,
o.CreatedAt
FROM CUSTOMERS c LEFT JOIN
ORDERS o USING(CustomerID)
WHERE o.CreatedAt = (SELECT
MAX(o2.CreatedAt) FROM ORDERS o2
WHERE o2.CustomerID =
c.CustomerID)
ORDER BY c.CustomerID;
```

```
19 # Query 3: Show all customers and their last order, including ones that have not ordered.
20 # To check if there is a delivery complain regarding last order (Refund or Cancel)
21 #
22 # gets order count of all customers (not related to query)
23 • SELECT c.CustomerID, c.FirstName, c.LastName, COUNT(o.OrderID) AS OrderCount
24 FROM CUSTOMERS c LEFT JOIN ORDERS o USING(CustomerID)
25 GROUP BY c.CustomerID, c.FirstName, c.LastName
26 ORDER BY c.CustomerID;
27
28 # Query that answers the Question
29 • SELECT c.CustomerID, c.FirstName, c.LastName, o.OrderID, o.OrderStatus, o.CreatedAt
30 FROM CUSTOMERS c LEFT JOIN ORDERS o USING(CustomerID)
31 WHERE o.CreatedAt = (SELECT MAX(o2.CreatedAt) FROM ORDERS o2
32 WHERE o2.CustomerID = c.CustomerID)
33 ORDER BY c.CustomerID;
34
```

| CustomerID | FirstName | LastName | OrderID | OrderStatus | Createdat           |
|------------|-----------|----------|---------|-------------|---------------------|
| 1          | Asha      | Rao      | 3       | PENDING     | 2025-11-05 12:27:08 |
| 2          | Vishnu    | Panyam   | 1       | PAID        | 2025-11-05 12:27:08 |
| 3          | Emily     | Chen     | 2       | SHIPPED     | 2025-11-05 12:27:08 |
| 4          | Miguel    | Santos   | 4       | DELIVERED   | 2025-11-05 12:27:08 |
| 5          | Sara      | Nair     | 5       | CANCELLED   | 2025-11-05 12:27:08 |
| 6          | Reena     | Singh    | 6       | PAID        | 2025-11-05 12:27:08 |

#### 4. Write a single-row subquery. Show the results and sort the results by key field(s).

```
SELECT * FROM ORDERS;
SELECT o.OrderID, o.CustomerID, p.Amount,
o.OrderStatus
FROM ORDERS o JOIN PAYMENTS p
USING(PaymentID)
ORDER BY o.OrderID;
```

# Query that answers the Question

```
SELECT o.OrderID, o.CustomerID, p.Amount,
o.OrderStatus
FROM ORDERS o JOIN PAYMENTS p USING(PaymentID)
WHERE p.Amount = (SELECT MAX(Amount) FROM PAYMENTS)
ORDER BY o.OrderID;
```

```
35 # Query 4: Find the Order with highest total amount
36 # -----
37 • SELECT * FROM ORDERS;
38
39 • SELECT o.OrderID, o.CustomerID, p.Amount, o.OrderStatus
40 FROM ORDERS o JOIN PAYMENTS p USING(PaymentID)
41 ORDER BY o.OrderID;
42
43 # Query that answers the Question
44 • SELECT o.OrderID, o.CustomerID, p.Amount, o.OrderStatus
45 FROM ORDERS o JOIN PAYMENTS p USING(PaymentID)
46 WHERE p.Amount = (SELECT MAX(Amount) FROM PAYMENTS)
47 ORDER BY o.OrderID;
48
```

| Result Grid |            |         |             |
|-------------|------------|---------|-------------|
| OrderID     | CustomerID | Amount  | OrderStatus |
| 1           | 2          | 1428.99 | PAID        |

#### 5. Write a multiple-row subquery. Show the results and sort the results by key field(s).

```
SELECT c.CustomerID, c.FirstName,
c.LastName, c.CustomerEmail,
COUNT(ReviewID) as ReviewCount
FROM CUSTOMERS c LEFT JOIN REVIEWS r
ON c.CustomerID = r.CustomerID
GROUP BY
```

```
49 # Query 5: List all Customers with reviews
50 # -----
51 # see number of reviews per customer
52 • SELECT c.CustomerID, c.FirstName, c.LastName, c.CustomerEmail, COUNT(ReviewID) as ReviewCount
53 FROM CUSTOMERS c LEFT JOIN REVIEWS r ON c.CustomerID = r.CustomerID
54 GROUP BY c.CustomerID,c.FirstName,c.LastName,c.CustomerEmail
55 ORDER BY c.CustomerID;
56
57 # Query that answers the question
58 • SELECT c.CustomerID, c.FirstName, c.LastName, c.CustomerEmail, COUNT(ReviewID) as ReviewCount
59 FROM CUSTOMERS c
60 WHERE c.CustomerID IN (SELECT DISTINCT CustomerID FROM REVIEWS)
61 ORDER BY c.CustomerID;
```

| Result Grid |           |          |                         |             |
|-------------|-----------|----------|-------------------------|-------------|
| CustomerID  | FirstName | LastName | CustomerEmail           | ReviewCount |
| 1           | Asha      | Rao      | asha.rao@example.com    | 1           |
| 2           | Vishnu    | Panyam   | vishnu.p@example.com    | 1           |
| 3           | Emily     | Chen     | emily.c@example.com     | 1           |
| 4           | Miguel    | Santos   | miguel.s@example.com    | 1           |
| 5           | Sara      | Nair     | sara.nair@example.com   | 1           |
| 6           | Reena     | Singh    | reena.singh@example.com | 1           |
| 7           | Tom       | Lee      | tom.lee@example.com     | 1           |

c.CustomerID,c.FirstName,c.LastName,c.CustomerEmail

ORDER BY c.CustomerID;

# Query that answers the question

```
SELECT c.CustomerID, c.FirstName, c.LastName, c.CustomerEmail,
COUNT(reviewID) as ReviewCount
FROM CUSTOMERS c
WHERE c.CustomerID IN (SELECT DISTINCT CustomerID FROM REVIEWS)
ORDER BY c.CustomerID;
```

## 6. Write an SQL to aggregate the results by using multiple columns in the SELECT clause.

```
SELECT * FROM ORDERITEMS;
```

```
SELECT * FROM ORDERS;
```

# note that order 10 has a failed order,  
since there was no item added, hence its  
in Pending state

# but contains payment and other  
details

```
SELECT oi.OrderID,oi.ProductID,p.Name
AS ProductName,oi.Quantity
FROM ORDERITEMS oi JOIN PRODUCTS p USING(ProductID)
WHERE oi.OrderID IN (1, 10);
```

# Query that answers the question

```
SELECT p.ProductID, p.Name AS ProductName,
 COUNT(oi.OrderItemID) AS NumOrderLines, SUM(oi.Quantity) AS
TotalUnitsSold
FROM PRODUCTS p LEFT JOIN ORDERITEMS oi USING(ProductID)
GROUP BY p.ProductID, p.Name
ORDER BY p.ProductID;
```

```
63 # Query 6: For each product, how many times it was added to orders and what is the total number sold.
64 # -----
65 • SELECT * FROM ORDERITEMS;
66 • SELECT * FROM ORDERS;
67
68 # note that order 10 has a failed order, since there was no item added, hence its in Pending state
69 # but contains payment and other details
70 • SELECT ol.OrderID,oi.ProductID,p.Name AS ProductName,oi.Quantity
71 FROM ORDERITEMS ol JOIN PRODUCTS p USING(ProductID)
72 WHERE ol.OrderID IN (1, 10);
73
74 # Query that answers the question
75 • SELECT p.ProductID, p.Name AS ProductName,
76 COUNT(ol.OrderItemID) AS NumOrderLines, SUM(ol.Quantity) AS TotalUnitsSold
77 FROM PRODUCTS p LEFT JOIN ORDERITEMS ol USING(ProductID)
78 GROUP BY p.ProductID, p.Name
79 ORDER BY p.ProductID;
```

| ProductID | ProductName   | NumOrderLines | TotalUnitsSold |
|-----------|---------------|---------------|----------------|
| 1         | Smartphone X  | 1             | 1              |
| 2         | Laptop Pro    | 1             | 1              |
| 3         | Air Fryer     | 1             | 1              |
| 4         | Yoga Mat      | 1             | 2              |
| 5         | Organic Rice  | 1             | 3              |
| 6         | Running Shoes | 1             | 1              |

## 7. Write a subquery using the NOT IN operator. Show the results and sort the results by key field(s).

```
SELECT c.CustomerID,c.FirstName,c.LastName,c.CustomerEmail
FROM CUSTOMERS c
WHERE c.CustomerID NOT IN (SELECT
DISTINCT CustomerID FROM CART)
ORDER BY c.CustomerID;
```

```
82 # Query 7: Find customers who have never created a cart (for further marketing targets)
83 # -----
84
85 • SELECT c.CustomerID,c.FirstName,c.LastName,c.CustomerEmail
86 FROM CUSTOMERS c
87 WHERE c.CustomerID NOT IN (SELECT DISTINCT CustomerID FROM CART)
88 ORDER BY c.CustomerID;
89
90 # We do not have such customers yet
91
```

| CustomerID | FirstName | LastName | CustomerEmail |
|------------|-----------|----------|---------------|
| *          | NULL      | NULL     | NULL          |

## 8. Write a query using a UNION statement. Show the results and sort the results by key field(s).

```
WITH CitySource AS (
 SELECT DISTINCT a.City,
 'CUSTOMER_ADDRESS' AS Source
 FROM ADDRESSES a JOIN
 CUSTOMERADDRESS ca
 USING(AddressID)
 UNION
 SELECT DISTINCT a.City, 'STORE_ADDRESS' AS Source
 FROM ADDRESSES a JOIN STORES s USING(AddressID)
)
SELECT City FROM CitySource
GROUP BY City
ORDER BY City;
```

```
92 # Query 8: Get a list of cities covered by operations of instacart2 based on customer and store locations
93 #
94 WITH CitySource AS (
95 SELECT DISTINCT a.City, 'CUSTOMER_ADDRESS' AS Source
96 FROM ADDRESSES a JOIN CUSTOMERADDRESS ca USING(AddressID)
97 UNION
98 SELECT DISTINCT a.City, 'STORE_ADDRESS' AS Source
99 FROM ADDRESSES a JOIN STORES s USING(AddressID)
100)
101 SELECT City FROM CitySource
102 GROUP BY City
103 ORDER BY City;
```

| City     |
|----------|
| Chandler |
| Gilbert  |
| Glendale |
| Mesa     |
| Peoria   |
| Phoenix  |

## 9. Write a query using the NOT EXISTS operator. Show the results and sort the results by key field(s).

```
SELECT * FROM SHOPPERS;
Query that answers the question
SELECT s.DeliveryAgentID, s.Name,
s.Contact, s.CurrentLocation
FROM SHOPPERS s
WHERE NOT EXISTS (SELECT 1 FROM
DELIVERYSHOPPER ds
WHERE ds.DeliveryAgentID =
s.DeliveryAgentID
)
ORDER BY s.DeliveryAgentID;
```

```
105 # Query 9: Find shoppers who currently have no deliveries assigned (Capacity planning)
106 #
107 SELECT * FROM SHOPPERS;
108
109 # Query that answers the question
110 SELECT s.DeliveryAgentID, s.Name, s.Contact, s.CurrentLocation
111 FROM SHOPPERS s
112 WHERE NOT EXISTS (SELECT 1 FROM DELIVERYSHOPPER ds
113 WHERE ds.DeliveryAgentID = s.DeliveryAgentID
114)
115 ORDER BY s.DeliveryAgentID;
116
117 # report all our shoppers have deliveries assigned (Full Capacity)
118
```

| DeliveryAgentID | Name | Contact | CurrentLocation |
|-----------------|------|---------|-----------------|
| NULL            | NULL | NULL    | NULL            |

## 10. Write a subquery using the NOT NULL operator in the inner query. Show the results and sort the results by key field(s).

```
SELECT * FROM REVIEWS; # Every customer has a review
```

```
Query that answers the question.
```

```
SELECT DISTINCT o.OrderID, o.CustomerID,
```

```
o.OrderStatus, o.CreatedAt
```

```
FROM ORDERS o
```

```
WHERE o.OrderID IN (
```

```
SELECT oi.OrderID FROM ORDERITEMS oi
```

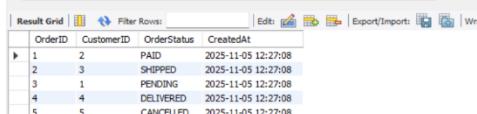
```
JOIN REVIEWS r USING(ProductID)
```

```
WHERE r.Rating IS NOT NULL
```

```
)
```

```
ORDER BY o.OrderID;
```

```
119 # Query 10: List all orders that have at least one reviewed product
120 # -----
121 • SELECT * FROM REVIEWS; # Every customer has a review
122
123
124 # Query that answers the question.
125 • SELECT DISTINCT o.OrderID, o.CustomerID, o.OrderStatus, o.CreatedAt
126 FROM ORDERS o
127 WHERE o.OrderID IN (
128 SELECT oi.OrderID FROM ORDERITEMS oi
129 JOIN REVIEWS r USING(ProductID)
130 WHERE r.Rating IS NOT NULL
131)
132 ORDER BY o.OrderID;
133
```



| OrderID | CustomerID | OrderStatus | CreatedAt           |
|---------|------------|-------------|---------------------|
| 1       | 2          | PAID        | 2025-11-05 12:27:08 |
| 2       | 3          | SHIPPED     | 2025-11-05 12:27:08 |
| 3       | 1          | PENDING     | 2025-11-05 12:27:08 |
| 4       | 4          | DELIVERED   | 2025-11-05 12:27:08 |
| 5       | 5          | CANCELLED   | 2025-11-05 12:27:08 |