Step-1 : create tables   
  
  
Table 1   
  
**- Customers table:**

- `customer\_id` (Primary Key)

- `first\_name'

- `last\_name`

- `email`

USE `online retail store`;

SQL **Queries :**

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

total\_amount DECIMAL(10, 2),

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

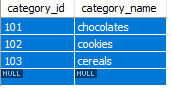


Table 2

**- Orders table:**

- `order\_id` (Primary Key)

- `customer\_id` (Foreign Key)

- `order\_date`

- `total\_amount`

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100)

);

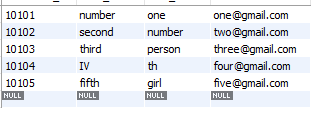


Table 3

**- Products table:**

- `product\_id` (Primary Key)

- `product\_name`

- `category\_id` (Foreign Key)

- `price`

CREATE TABLE Products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(255),

category\_id INT,

price DECIMAL(10, 2),

FOREIGN KEY (category\_id) REFERENCES Categories(category\_id)

);

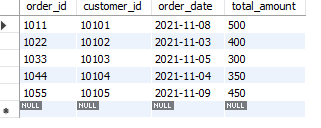


Table 4

**- Categories table:**

- `category\_id` (Primary Key)

- `category\_name`

CREATE TABLE Categories (

category\_id INT PRIMARY KEY,

category\_name VARCHAR(100)

);

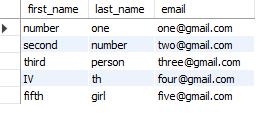


SECTION 1:

**Basic Queries:**

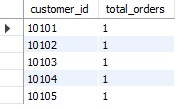
1. Query: Retrieve a list of all customers along with their email addresses.

SELECT first\_name, last\_name, email FROM Customers;



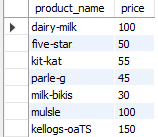
1. Query: Find the total number of orders placed by each customer.

SELECT customer\_id, COUNT(\*) AS total\_orders FROM Orders GROUP BY customer\_id;



1. Query: List all products along with their prices.

SELECT product\_name, price FROM Products;



1. Query: Retrieve the category with the highest number of products.

SELECT category\_id, COUNT(\*) AS product\_count FROM Products GROUP BY category\_id ORDER BY product\_count DESC LIMIT 1;



**Intermediate Queries**

1. Query: Find all customers who have not placed any orders.

SELECT c.customer\_id, c.first\_name, c.last\_name, c.email

FROM Customers c

LEFT JOIN Orders o ON c.customer\_id = o.customer\_id

WHERE o.order\_id IS NULL;



1. Query: List the products with the highest and lowest prices.

SELECT \* FROM Products ORDER BY price DESC LIMIT 1;

SELECT \* FROM Products ORDER BY price ASC LIMIT 1;

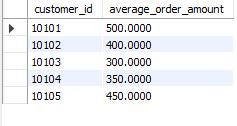


1. Query: Calculate the average order amount for each customer.

SELECT customer\_id, AVG(total\_amount) AS average\_order\_amount

FROM Orders

GROUP BY customer\_id;



1. Query: Find the categories that do not have any products.

SELECT c.category\_id, c.category\_name

FROM Categories c

LEFT JOIN Products p ON c.category\_id = p.category\_id

WHERE p.product\_id IS NULL;

****

**Advanced Queries**

1. Retrieve a list of customers who have placed orders for products with a price higher than $100.

SELECT DISTINCT c.customer\_id, c.first\_name, c.last\_name, c.email

FROM Customers c

JOIN Orders o ON c.customer\_id = o.customer\_id

JOIN Products p ON o.product\_id = p.product\_id

WHERE p.price > 100;

1. List the customers who have placed orders for products from at least three different categories.

SELECT c.customer\_id, c.first\_name, c.last\_name, c.email

FROM Customers c

JOIN Orders o ON c.customer\_id = o.customer\_id

JOIN Products p ON o.product\_id = p.product\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name, c.email

HAVING COUNT(DISTINCT p.category\_id) >= 3;

1. Find the products with the highest and lowest average customer ratings (if a rating table is available).
2. Calculate the total revenue generated from each category.

SELECT c.category\_id, c.category\_name, SUM(p.price) AS total\_revenue

FROM Categories c

JOIN Products p ON c.category\_id = p.category\_i

