**Graded Assignment on Database Schema: Online Retail Store**

**Basic Queries:**

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

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

FROM Customers;

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

SELECT c.customer\_id, c.first\_name, c.last\_name, COUNT(o.order\_id) AS total\_orders

FROM Customers c

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

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

1. List all products along with their prices.

SELECT product\_id, product\_name, price

FROM Products;\

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

SELECT c.category\_id, c.category\_name, COUNT(p.product\_id) AS product\_count

FROM Categories c

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

GROUP BY c.category\_id, c.category\_name

ORDER BY product\_count DESC

LIMIT 1;

**Intermediate Queries:**

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

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

FROM Customers c

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

WHERE o.order\_id IS NULL;

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

-- Product with the highest price

SELECT product\_id, product\_name, price

FROM Products

ORDER BY price DESC

LIMIT 1;

-- Product with the lowest price

SELECT product\_id, product\_name, price

FROM Products

ORDER BY price ASC

LIMIT 1;

1. Calculate the average order amount for each customer.

SELECT c.customer\_id, c.first\_name, c.last\_name, AVG(o.total\_amount) AS average\_order\_amount

FROM Customers c

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

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

1. 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

FROM Customers c

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

INNER JOIN OrderDetails od ON o.order\_id = od.order\_id

INNER JOIN Products p ON od.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

FROM Customers c

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

INNER JOIN OrderDetails od ON o.order\_id = od.order\_id

INNER JOIN Products p ON od.product\_id = p.product\_id

INNER JOIN Categories cat ON p.category\_id = cat.category\_id

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

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

1. Find the products with the highest and lowest average customer ratings (if a rating table is available).

-- Product with the highest average rating

SELECT p.product\_id, p.product\_name, AVG(pr.rating) AS avg\_rating

FROM Products p

LEFT JOIN ProductRatings pr ON p.product\_id = pr.product\_id

GROUP BY p.product\_id, p.product\_name

ORDER BY avg\_rating DESC

LIMIT 1;

-- Product with the lowest average rating

SELECT p.product\_id, p.product\_name, AVG(pr.rating) AS avg\_rating

FROM Products p

LEFT JOIN ProductRatings pr ON p.product\_id = pr.product\_id

GROUP BY p.product\_id, p.product\_name

ORDER BY avg\_rating ASC

LIMIT 1;

1. Calculate the total revenue generated from each category.

SELECT c.category\_id, c.category\_name, SUM(od.unit\_price \* od.quantity) AS total\_revenue

FROM Categories c

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

LEFT JOIN OrderDetails od ON p.product\_id = od.product\_id

GROUP BY c.category\_id, c.category\_name;

**Complex Queries:**

1. Retrieve the names of customers who have placed orders in the last 30 days.

SELECT DISTINCT c.first\_name, c.last\_name

FROM Customers c

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

WHERE o.order\_date >= DATE\_SUB(CURDATE(), INTERVAL 30 DAY);

1. List the products that have been out of stock for more than 7 days.

SELECT p.product\_id, p.product\_name

FROM Products p

LEFT JOIN Inventory i ON p.product\_id = i.product\_id

WHERE i.quantity = 0

AND i.last\_stocked\_date <= DATE\_SUB(CURDATE(), INTERVAL 7 DAY);

1. Find the products that have the highest fluctuation in price (i.e., the products where the difference between the highest and lowest prices is the greatest).

SELECT p.product\_id, p.product\_name, MAX(p.price) AS highest\_price, MIN(p.price) AS lowest\_price

FROM Products p

GROUP BY p.product\_id, p.product\_name

ORDER BY (highest\_price - lowest\_price) DESC

LIMIT 1;

1. Create a summary report showing the category names, the total number of products in each category, and the total revenue generated from products in each category.

SELECT c.category\_name, COUNT(p.product\_id) AS total\_products, SUM(od.unit\_price \* od.quantity) AS total\_revenue

FROM Categories c

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

LEFT JOIN OrderDetails od ON p.product\_id = od.product\_id

GROUP BY c.category\_name;