SQL – ASSIGNMENT MEDIUM

```
USE RetailDB_2;


1. Customers
CREATE TABLE Customers (
customer_id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100),
email VARCHAR(100),
city VARCHAR(50),
signup_date DATE
);

 2. Suppliers
CREATE TABLE Suppliers (
supplier_id INT AUTO_INCREMENT PRIMARY KEY,
supplier_name VARCHAR(100),
contact_email VARCHAR(100),
city VARCHAR(50)
);

 3. Products
CREATE TABLE Products (
product_id INT AUTO_INCREMENT PRIMARY KEY,
product_name VARCHAR(100),
category VARCHAR(50),
price DECIMAL(10,2),
stock_qty INT,
supplier_id INT,
```

```
FOREIGN KEY (supplier_id) REFERENCES Suppliers(supplier_id)
);
```

4. Orders

```
CREATE TABLE Orders (
order_id INT AUTO_INCREMENT PRIMARY KEY,
customer_id INT,
order_date DATE,
total_amount DECIMAL(10,2),
payment_mode VARCHAR(50),
FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

5. Order_Items

```
CREATE TABLE Order_Items (
order_item_id INT AUTO_INCREMENT PRIMARY KEY,
order_id INT,
product_id INT,
quantity INT,
price_each DECIMAL(10,2),
FOREIGN KEY (order_id) REFERENCES Orders(order_id),
FOREIGN KEY (product_id) REFERENCES Products(product_id)
);

SELECT * FROM Customers;
SELECT * FROM Suppliers;
SELECT * FROM Products;
SELECT * FROM Orders;
SELECT * FROM Order_Items;
```

Task: Solve all the below mentioned problems by writing the SQL queries.

a) Normal Queries

1. Fetch all products along with their supplier name (INNER JOIN).

```
SELECT p.product_name, s.supplier_name
FROM Products p
INNER JOIN Suppliers s
ON p.supplier_id = s.supplier_id;
```

2. Find all customers and their orders, even if they have not placed any (LEFT JOIN).

```
SELECT c.customer_id, c.name, c.city,
    o.order_id, o.order_date, o.total_amount, o.payment_mode
FROM Customers c
LEFT JOIN Orders o
ON c.customer_id = o.customer_id;
```

3. Get all suppliers and the products they supply, even if no products exist for a supplier (RIGHT JOIN).

```
SELECT s.supplier_id, s.supplier_name, s.city,
 p.product_id, p.product_name, p.category, p.price, p.stock_qty, p.supplier_id
FROM Products p
RIGHT JOIN Suppliers s
ON p.supplier_id = s.supplier_id;
```

4. Show all customers and all orders (FULL OUTER JOIN simulation using UNION).

```
SELECT c.customer_id,c.name AS customer_name,o.order_id,o.order_date
FROM customers c
LEFT JOIN orders o
 ON c.customer_id = o.customer_id
UNION
SELECT c.customer_id,c.name AS customer_name,o.order_id,o.order_date
FROM customers c
RIGHT JOIN orders o
ON c.customer_id = o.customer_id;
```

5. List all products priced between ₹5000 and ₹50,000 and supplied from "Mumbai".b) Aggregations & Group By

```
SELECT p.product_id,p.product_name ,p.price,s.supplier_name ,s.city
FROM products p
JOIN suppliers s
ON p.supplier_id = s.supplier_id
WHERE p.price BETWEEN 5000 AND 50000 AND s.city = 'Mumbai';
```

6. Find the total number of orders placed by each customer and show only those who placed more than 2 (GROUP BY + HAVING).

```
SELECT c.name, count(order_id) AS Total_orders
FROM Orders o
JOIN Customers c
ON o.customer_id = c.customer_id
GROUP BY c.name
```

```
HAVING Total_orders > 2;
```

7. Show each supplier's total sales value (sum of quantity × price_each).

```
SELECT s.supplier_id,s.supplier_name ,SUM(oi.quantity * oi.price_each) AS total_sales
FROM suppliers s
JOIN products p
ON s.supplier_id = p.supplier_id
JOIN order_items oi ON p.product_id = oi.product_id
GROUP BY s.supplier_id, s.supplier_name;
```

8. Find the average, highest, and lowest price of products in each category.

```
SELECT category, AVG(price) AS average_price,
    MAX(price) AS max_price,
    MIN(price) AS min_price
FROM products
GROUP BY category;
```

9. Find the top 5 customers by total spending (ORDER BY SUM(total_amount) DESC LIMIT 5).

```
SELECT c.customer_id, c.name , SUM(o.total_amount) AS total_spent
FROM Customers c
JOIN Orders o
ON c.customer_id = o.customer_id
GROUP BY c.customer_id, c.name
ORDER BY total_spent DESC
LIMIT 5;
```

10. Show the number of unique products ordered by each customer.

c) Subqueries

```
SELECT DISTINCT(p.product_name)
FROM Customers
SELECT c.customer_id,c.name AS customer_name,COUNT(DISTINCT oi.product_id) AS
unique_products_ordered
FROM customers c
JOIN orders o
ON c.customer_id = o.customer_id
JOIN order_items oi
 ON o.order_id = oi.order_id
GROUP BY c.customer_id, c.name;
```

11. Find customers who placed an order with an amount greater than the average

order amount (subquery).

```
SELECT c.customer_id, o.total_amount
FROM Customers c
JOIN Orders o
ON c.customer_id = o.customer_id
WHERE o.total_amount > (SELECT AVG(total_amount) AS average_amount FROM Orders);
```

12. Find products that have never been ordered (subquery with NOT IN).

```
SELECT p.product_id,p.product_name,p.price
FROM products p
WHERE p.product_id NOT IN (
   SELECT DISTINCT product_id FROM order_items);
```

13. List customers who ordered at least one product from the "Electronics" category.

```
SELECT DISTINCT c.customer_id,c.name AS customer_name
FROM customers c
JOIN orders o
ON c.customer_id = o.customer_id
JOIN order_items oi
ON o.order_id = oi.order_id
JOIN products p
ON oi.product_id = p.product_id
WHERE p.category = 'Electronics';
```

14. Get suppliers who provide products that have been ordered more than 100 times in total.

```
SELECT s.supplier_id,s.supplier_name,SUM(oi.quantity) AS total_ordered
FROM suppliers s
JOIN products p
ON s.supplier_id = p.supplier_id
JOIN order_items oi
ON p.product_id = oi.product_id
GROUP BY s.supplier_id, s.supplier_name
HAVING SUM(oi.quantity) > 100;
```

15. Find the most expensive product(s) using a subquery with MAX().

d) Advanced Filters

```
SELECT product_id,product_name,price
FROM products
```

```
WHERE price = (SELECT MAX(price) FROM products);
```

16. Show orders placed by customers who live in either Mumbai, Delhi, or Bengaluru (IN operator).

```
SELECT o.order_id,c.name AS customer_name,c.city,o.order_date,o.total_amount
FROM orders o
JOIN customers c
ON o.customer_id = c.customer_id
WHERE c.city IN ('Mumbai', 'Delhi', 'Bengaluru');
```

17. Show orders where payment mode is NOT UPI or Credit Card (NOT IN).

```
SELECT order_id,customer_id,order_date,total_amount,payment_mode
FROM orders
WHERE payment_mode NOT IN ('UPI', 'Credit Card');
```

18. Find customers who have no email address recorded (IS NULL).

```
SELECT customer_id,name AS customer_name,city
FROM customers
WHERE email IS NULL;
```

19. Show suppliers who are not from the same city as any customer (NOT IN subquery).

```
SELECT supplier_id,supplier_name,city
FROM suppliers
WHERE city NOT IN (SELECT DISTINCT city FROM customers WHERE city IS NOT NULL);
```

20. Get the latest 3 orders placed, skipping the first 2 (ORDER BY + LIMIT + OFFSET).

```
SELECT * FROM Orders
ORDER BY order_date DESC
LIMIT 3
OFFSET 2;
```