SQL SAAIGNMENT – BASIC

```
CREATE DATABASE retaildb_1;
USE retaildb_1;

CREATE TABLE Customers (
customer_id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100),
email VARCHAR(150) UNIQUE,
city VARCHAR(50),
signup_date DATE
);
CREATE TABLE Products (
product_id INT AUTO_INCREMENT PRIMARY KEY,
product_name VARCHAR(100),
category VARCHAR(50),
price DECIMAL(10,2)
);
CREATE TABLE Orders (
order_id INT AUTO_INCREMENT PRIMARY KEY,
customer_id INT,
product_id INT,
order_date DATE,
quantity INT,
total_amount DECIMAL(10,2),
payment_mode VARCHAR(50),
FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
```

```sql
FOREIGN KEY (product_id) REFERENCES Products(product_id)
);

select * from Customers;
select * from Products;
select * from Orders;
```

Task: Solve the below mentioned questions by writing SQL queries.

1. Fetch all customers from the database.

```sql
select * from Customers;
```

2. Show only the customer names and their cities.

```sql
SELECT name, city FROM Customers;
```

3. Find customers who live in Mumbai.

```sql
select name, city from Customers
where city = "Mumbai";
```

4. Get all orders placed after 1st August 2024.

```sql
select * from Orders
where order_date > '2024-08-01';
```

5. List all products priced greater than ₹5000.

```sql
select * from Products
```

```
where price > 5000;
```

6. Count how many customers exist in the system.

```
select count(*) AS Customer_count
 from customers;
```

7. Update a customer’s city (e.g., change Rohit Kumar’s city to Hyderabad).

```
update  Customers
set city = "Hyderabad"
where name = "Rohit Kumar";

set SQL_SAFE_UPDATES=0;
```

8. Delete an order (e.g., remove order with ID = 5).

```
DELETE FROM Orders
WHERE order_id = 5;
```

9. Display product names with their original price and price increased by 10%.

```
SELECT product_name, price,
     (price + price*0.10) AS increased_price_by_10_percent
FROM Products;
```

10. Show only the unique cities where customers live.

```
SELECT DISTINCT(city) FROM Customers;
```

11. Get the first 3 customers who signed up.

```
SELECT * FROM Customers
ORDER BY signup_date ASC
LIMIT 3;
```

12. Skip the first 2 customers and fetch the next 3 customers.

```
SELECT * FROM Customers
LIMIT 3
OFFSET 2;
```

13. Find products with prices between ₹2000 and ₹6000.

```
SELECT * FROM Products
WHERE price BETWEEN 2000 AND 6000;
```

14. Find customers who are from Mumbai OR Chennai.

```
SELECT * FROM Customers
WHERE city = 'Mumbai' OR city = 'Chennai';
```

15. Find customers who are NOT from Delhi.

```
SELECT * FROM Customers
WHERE city != 'Delhi';
```

16. Find orders that are NOT paid by UPI.

```
SELECT * FROM Orders
WHERE payment_mode <> 'UPI';
```

17. Get the average order amount across all orders.

```
SELECT AVG(total_amount) AS average_order_amount
FROM Orders;
```

18. Show the highest order amount.

```
SELECT * FROM Orders
ORDER BY total_amount DESC
LIMIT 1;
```

19. Show the lowest product price.

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

20. Find the total money spent across all orders.

```
SELECT SUM(total_amount) AS total_money_spent
FROM Orders;
```