```
use shop;

create table customers(
CustomerID INT PRIMARY KEY auto_increment,
Name VARCHAR(100),
Email VARCHAR(100),
City VARCHAR(50),
SignupDate DATE
);

create table orders(
OrderID INT PRIMARY KEY,
CustomerID INT,
OrderDate DATE,
TotalAmount DECIMAL(10,2),
FOREIGN KEY (CustomerID) REFERENCES customers(CustomerID)
);

create table products(
ProductID INT PRIMARY KEY,
ProductName VARCHAR(100),
Category VARCHAR(50),
Price DECIMAL(10,2)
);
create table order_details(
OrderDetailID INT PRIMARY KEY,
OrderID INT,
ProductID INT,
Quantity INT,
Price DECIMAL(10,2),
FOREIGN KEY (OrderID) REFERENCES orders(OrderID),
FOREIGN KEY (ProductID) REFERENCES products(ProductID)
);

insert into customers values
(1,'Hari', 'hari@example.com', 'New York', '2023-01-15'),
(2,'Indrajith', 'indrajith@example.com', 'London', '2024-01-15'),
(3,'Adarsh', 'Adarsh@example.com', 'Delhi', '2025-01-15'),
(4,'Vaisakh', 'Vaisakh@example.com', 'Beijing', '2022-01-15');

insert into products values
(101, 'Laptop', 'Electronics', 75000.00),
(102, 'Mouse', 'Accessories', 500.00),
(201, 'Tomato', 'Vegetables', 50.00);
```

```
insert into orders values
(1111, 2, '2023-03-02', 75000.00),
(2222, 3, '2023-03-01', 500.00),
(3333, 4, '2023-03-03', 50.00);

insert into order_details values
(1234, 1111, 101, 1, 75000.00),
(5678, 2222, 102, 1, 500.00),
(9101, 3333, 201, 1, 50.00);


-- 1. Get the list of all customers.
select * from customers;

-- 2. Find all orders placed in the last 30 days.
select * from orders
where orderdate >= current_date - interval '30 days';

-- 3. Show product names and their prices.
select productname, price from products;

-- 4. Find the total number of products in each category.
select category, count(*) from products
group by category;

-- 5. Get all customers from the city 'Mumbai'.
select * from customers
where city = 'mumbai';

-- 6. Find orders with a total amount greater than ₹5000.
select * from orders
where totalamount > 5000;

-- 7. List customers who signed up after '2024-01-01'.
select * from customers
where signupdate > '2024-01-01';

-- 8. Show all orders along with the customer's name
select orders.orderid, orders.orderdate, orders.totalamount, customers.name
from orders
join customers on orders.customerid = customers.customerid;

-- 9. List products purchased in each order.
```

```
select orders.orderid, products.productname, order_details.quantity, order_details.price from
orders
join order_details on orders.orderid = order_details.orderid
join products on order_details.productid = products.productid;

-- 10. Find customers who have never placed an order.
select * from customers
left join orders on customers.customerid = orders.customerid
where orders.orderid is null;

-- 11. Find the total amount spent by each customer.
select customers.customerid, customers.name, sum(orders.totalamount) from customers
left join orders on customers.customerid = orders.customerid
group by customers.customerid, customers.name;

-- 12. Which product has been sold the most (by quantity)?
select products.productname, sum(order_details.quantity) as total from products
join order_details on products.productid = order_details.productid
group by products.productname
order by total desc
limit 1;

-- 13. Find the average order value for each customer.
select customers.name, avg(orders.totalamount) from customers
left join orders on customers.customerid = orders.customerid
group by customers.name;

-- 14. Total sales amount per product category.
select products.category, sum(order_details.quantity * order_details.price) from products
left join order_details on products.productid = order_details.productid
group by products.category;

-- 15. Find customers who spent more than the average spending.
select customerid, name from customers
where
  (select sum(totalamount) from orders
   where orders.customerid = customers.customerid) >
  (select avg(totalamount) from orders);

-- 16. List products that have never been ordered.
select * from products
where products.productid not in (select productid from order_details);

-- 17. Find the most recent order for each customer.
```

```
select customers.name, orders.orderid, orders.orderdate, orders.totalamount from customers
join orders on orders.customerid = customers.customerid
where orders.orderdate = (
    select max(orderdate)
    from orders
    where customers.customerid = orders.customerid
);

-- 18. Rank customers by total spending (highest first).
select customers.name, sum(orders.totalamount) as total_spent from customers
join orders on customers.customerid = orders.customerid
group by customers.name
order by total_spent desc;

-- 19. Get the top 3 customers based on the number of orders placed.
select customers.name, count(orderid) as total_orders from customers
join orders on customers.customerid = orders.customerid
group by customers.name
order by total_orders desc
limit 3;

-- 20. For each product, find how many unique customers have purchased it.
select products.productname, count(distinct orders.customerid) from products
join order_details on products.productid = order_details.productid
join orders on order_details.orderid = orders.orderid
group by products.productid, products.productname;
```