**select \* from customers;**

**select \* from Suppliers;**

**select \* from order\_details;**

**select \* from Orders;**

**select \* from products;**

**select \* from Reviews;**

**create table Customers(customer\_ID text primary key, name text, age int, gender varchar(30), city varchar(30), state varchar(30),**

**country varchar(30), sign\_up\_date date, prime\_members boolean);**

**create table Suppliers (supplier\_ID text primary key, SupplierName text, ContactName text,**

**city text, state text);**

**create table Order\_details (orderID text, productID text, quantity int,**

**unitprice int, discount int);**

**create table Orders (OrderID text primary key, CustomerID text, orderDate date,**

**OrderAmt int, DeliveryFee int, DiscountApplied int);**

**create table Products (ProductID text primary key, ProductName text, Category text,**

**SubCategory text, PricePerUnit int, StockQnty int,**

**Supplier\_id text);**

**create table Reviews (ReviewID text primary key, ProductID text, CustomerID text,**

**Rating int, ReviewText text);**

**--Task 3: Write a query to: Retrieve all customers from a specific city.**

**SELECT name, city from customers**

**order by city;**

**--Fetch all products under the "Fruits" category.**

**select productname, category from products**

**where category = 'Fruits'**

**order by productname;**

**--Task 4: Write DDL statements to recreate the Customers table with the following constraints:**

**--CustomerID as the primary key.**

**--Ensure Age cannot be null and must be greater than 18.**

**--Add a unique constraint for Name.**

**create table Customers1(customer\_ID text primary key, name text unique, age int not null, gender varchar(30), city varchar(30), state varchar(30),**

**country varchar(30), sign\_up\_date date, prime\_members boolean);**

**select distinct(name), customer\_id, age from Customers**

**where age is not null**

**and age > 18 order by name;**

**--Task 5: Insert 3 new rows into the Products table using INSERT statements**

**select \* from products;**

**insert into products (productid, productname, category, subcategory, priceperunit, stockqnty, supplier\_id) values**

**('222222222', 'Serve Snack', 'Snacks', 'Sub-Snacks-1', 212, 333, '456789456789');**

**--Task 6: Update the stock quantity of a product**

**--where ProductID matches a specific ID.**

**select \* from Suppliers;**

**select \* from products;**

**select ProductName, stockqnty from Products**

**where productid is not null order by stockqnty desc;**

**--Task 7: Delete a supplier from the Suppliers table where**

**--their city matches a specific value.**

**select suppliername, supplier\_id from suppliers**

**where city = 'Chennai';**

**--Task 8: Use SQL constraints to:**

**--Add a CHECK constraint to ensure that ratings in the Reviews table are between 1 and 5.**

**select \* from reviews;**

**alter table reviews**

**add constraint rating check (rating between 1 and 5);**

**--Add a DEFAULT constraint for the PrimeMember column in the Customers table (default value: "No").**

**select \* from customers;**

**alter table customers**

**alter column prime\_members set Default 'No';**

**--Task 9: Write queries using:**

**--WHERE clause to find orders placed after 2024-01-01.**

**--HAVING clause to list products with average ratings greater than 4.**

**--GROUP BY and ORDER BY clauses to rank products by total sales.**

**select \* from products;**

**select \* from Reviews;**

**select \* from order\_details;**

**select \* from orders;**

**select orderid, customerid, orderdate from Orders**

**where orderdate > '2024-01-01';**

**select p.productname, p.category, avg(r.rating) from products p**

**left join reviews r on r.productid = p.productid**

**group by p.productname, p.category**

**having avg(r.rating) > '4';**

**select p.productname, sum(o.orderamt) from orders o**

**join order\_details od on od.orderid = o.orderid**

**join products p on p.productid = od.productid**

**group by p.productname**

**order by sum(o.orderamt);**

**select p.productname, sum(o.orderamt),**

**dense\_rank() over (partition by p.productname) from orders o**

**join order\_details od on od.orderid = o.orderid**

**join products p on p.productid = od.productid**

**group by p.productname, o.orderamt**

**order by sum(o.orderamt) desc;**

**--Task 10: Identifying High-Value Customers**

**--Scenario: Amazon Fresh wants to identify top customers based on their total spending.**

**--We will: Calculate each customer's total spending.**

**--Rank customers based on their spending.**

**--Identify customers who have spent more than ₹5,000.**

**select c.name, o.orderid, o.orderamt from customers c**

**join orders o on c.customer\_id = o.customerid**

**order by o.orderamt desc;**

**select c.name, o.orderamt,**

**dense\_rank() over (order by o.orderamt desc) from customers c**

**join orders o on c.customer\_id = o.customerid;**

**select c.name, o.orderid, o.orderamt from customers c**

**join orders o on c.customer\_id = o.customerid**

**where o.orderamt > '5000'**

**order by o.orderamt desc;**

**--Task 11: Use SQL to:**

**--Join the Orders and OrderDetails tables to calculate total revenue per order.**

**--Identify customers who placed the most orders in a specific time period.**

**--Find the supplier with the most products in stock.**

**select \* from order\_details;**

**select \* from orders;**

**select \* from customers;**

**select o.orderid, sum(o.orderamt) from orders o**

**join order\_details od on od.orderid = o.orderid**

**group by o.orderid**

**order by sum(o.orderamt) desc;**

**select c.name, count(c.name) as No\_of\_orders from customers c**

**left join orders o on o.customerid = c.customer\_id**

**group by c.name order by No\_of\_orders desc;**

**select \* from Suppliers;**

**select \* from products;**

**select s.suppliername, p.productname, p.stockqnty from suppliers s**

**left join products p on s.supplier\_id = p.supplier\_id;**

**order by s.suppliername;**

**--Task 12: Normalize the Products table to 3NF:**

**--Separate product categories and subcategories into a new table.**

**--Create foreign keys to maintain relationships.**

**create table ProductCat (**

**category text, subcategory text, productid text references products (productid)**

**);**

**select \* from productcat;**

**--Task 13: Write a subquery to:**

**--Identify the top 3 products based on sales revenue.**

**--Find customers who haven’t placed any orders yet.**

**select \* from products;**

**select \* from orders;**

**select \* from customers;**

**select \* from order\_details;**

**select p.productname, o.orderamt from products p**

**join order\_details od on od.productid = p.productid**

**join orders o on od.orderid = o.orderid**

**order by o.orderamt desc fetch first 3 rows only;**

**select c.name, o.orderid from customers c**

**left join orders o on c.customer\_id = o.customerid**

**where o.orderid is null;**

**--Task 14: Provide actionable insights:**

**--Which cities have the highest concentration of Prime members?**

**--What are the top 3 most frequently ordered categories?**

**select city, count(prime\_members) as Prime\_member\_count from customers**

**where prime\_members = 'true'**

**group by city order by prime\_member\_count desc;**

**select p.category, sum(od.quantity) as order from products p**

**join order\_details od on od.productid = p.productid**

**group by p.category order by sum(od.quantity) desc limit 3;**