|  |  |  |
| --- | --- | --- |
|  | Pimpri Chinchwad Education Trust’s  **Pimpri Chinchwad College of Engineering**  An Autonomous Institute  (Permanently affiliated to Savitribai Phule Pune University) |  |
| SEMESTER-IV |
| Assignment 4 | | |

# Name – Vivek Painjane

# PRN – 123B1B280

# Batch – D3 Subject : DBMSL

**Assignment 4: SQL Computations in an E-Commerce Application**

1. **Customers :** Stores customer details.

CREATE TABLE Customers ( customer\_id INT PRIMARY KEY, name VARCHAR(50),

email VARCHAR(100), city VARCHAR(50),

age INT

);

# Orders :Stores order details placed by customers.

CREATE TABLE Orders (

order\_id INT PRIMARY KEY, customer\_id INT, order\_date DATE,

total\_amount DECIMAL(10,2), status VARCHAR(20),

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

# Products : Stores product details.

CREATE TABLE Products ( product\_id INT PRIMARY KEY, name VARCHAR(50),

category VARCHAR(50), price DECIMAL(10,2),

stock INT

);

1. Order\_Items : **Stores products included in an order.**

CREATE TABLE Order\_Items ( order\_item\_id INT PRIMARY KEY, order\_id INT,

product\_id INT, quantity INT,

subtotal DECIMAL(10,2),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id), FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);

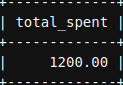
**Solve following queries :**

1. Find the total amount spent by customer ID 3 (sum of all their orders).

→ SELECT SUM(total\_amount) AS total\_spent

FROM Orders

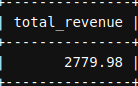
WHERE order\_id = 3;



1. Find the total revenue generated from all orders.

→ SELECT SUM(total\_amount) AS total\_revenue

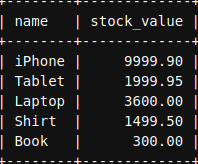
FROM Orders;



1. Calculate the stock value (price × stock) for each product.

→ SELECT name, (price \* stock) AS sock\_value

FROM Products;

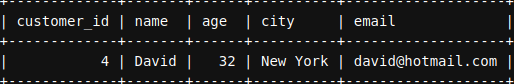


1. Retrieve customers who are above 30 years old and live in "New York".

→ SELECT age, city

FROM Customers

WHERE age > 30 AND city = ‘New York’



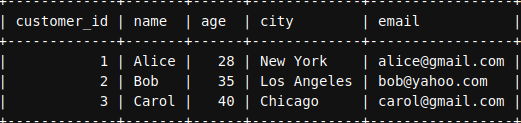
1. Find customers who either live in "Los Angeles" or have placed an order worth more than $500.

→ SELECT city, total\_amount

FROM Customers c

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

WHERE city = ‘Los Angeles’ OR total\_amount > 500;

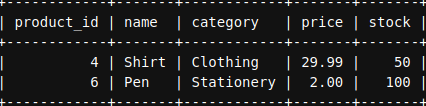


1. List products that are in stock but cost less than $50.

→ SELECT \*

FROM Products

WHERE stock > 0 AND price < 50;

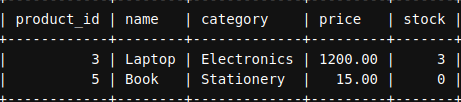


1. Find Products That Have Less Than 5 Items in Stock

→ SELECT \*

FROM Products

WHERE stock < 5;

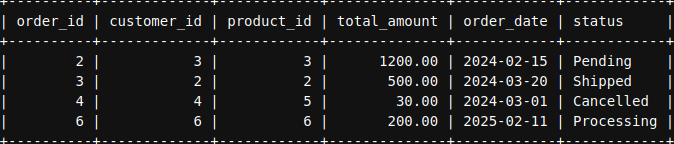


1. Find orders that are not completed (status is not 'Completed').

→ SELECT\*

FROM Orders

WHERE status <> ‘Completed’;

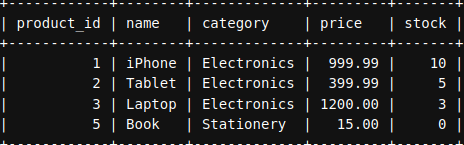


1. Find products that are out of stock or belong to the "Electronics" category.

→ SELECT \*

FROM Products

WHERE stock = 0 OR category = 'Electronics';

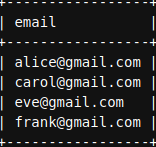


1. List all email addresses that belong to Gmail.

→ SELECT email

FROM Customers

WHERE email LIKE ['%@gmail.com](mailto:'%25@gmail.com)';



1. Find product names containing the word "Phone".

→ SELECT name

FROM Products

WHERE name LIKE '%Phone%';

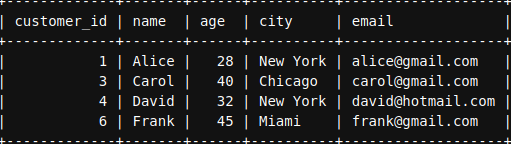


1. Find customers whose names have exactly 5 characters.

→ SELECT \*

FROM Customers

WHERE LENGTH(name) = 5;

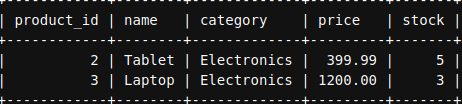


1. Find Products That Have "Laptop" or "Tablet" in Their Name

→ SELECT \*

FROM Products

WHERE name LIKE '%Laptop%' OR name LIKE '%Tablet%';

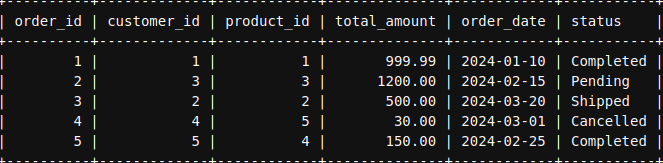


1. Retrieve orders placed in the year 2024.

→ SELECT \*

FROM Orders

WHERE YEAR(order\_date) = 2024;

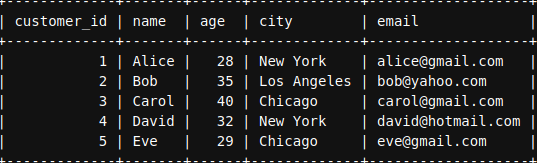


1. Find customers who live in "New York", "Los Angeles", or "Chicago".

→ SELECT \*

FROM Customers

WHERE city IN ('New York', 'Los Angeles', 'Chicago');

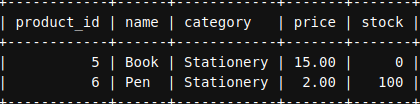


1. List products that do not belong to the 'Electronics' or 'Clothing' categories.

→ SELECT \*

FROM Products

WHERE category NOT IN ('Electronics', 'Clothing');

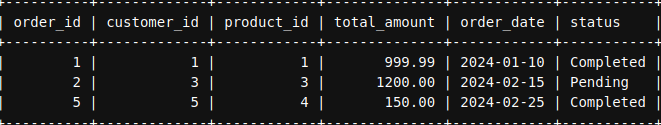


1. Find orders placed by customers with IDs 1, 3, or 5.

→ SELECT \*

FROM Orders

WHERE customer\_id IN (1, 3, 5);



1. Find customers who have not placed any orders.

→ SELECT \*

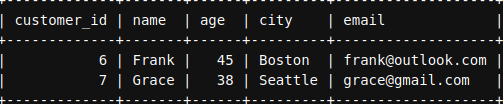
FROM Customers c

WHERE NOT EXISTS

(

SELECT 1 FROM Orders o WHERE o.customer\_id = c.customer\_id

);

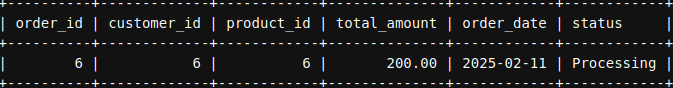


1. Show All Orders Placed in the Last 30 Days

→ SELECT \*

FROM Orders

WHERE order\_date >= CURDATE() - INTERVAL 30 DAY;



1. Find Orders Where the Total Amount is an Exact Multiple of 100

→ SELECT \*

FROM Orders

WHERE MOD(total\_amount, 100) = 0;

