|  |  |  |
| --- | --- | --- |
|  | Pimpri Chinchwad Education Trust’s  **Pimpri Chinchwad College of Engineering**  An Autonomous Institute  (Permanently affiliated to Savitribai Phule Pune University) |  |
| SEMESTER-IV |
| Assignment 6 | | |

Assignment 6

Consider the following database schema for an E-commerce system:

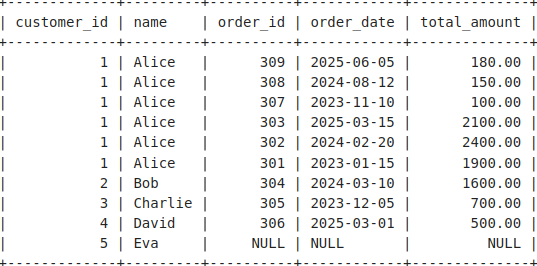
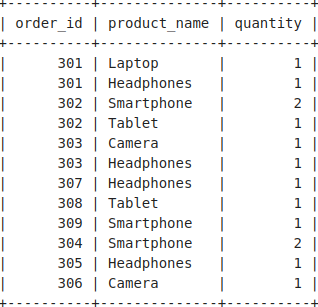
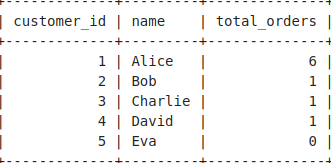
**Tables:**

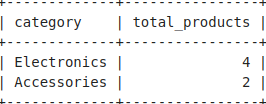
Customers (customer\_id, name, email, city)

Orders (order\_id, customer\_id, order\_date, total\_amount) Products (product\_id, product\_name, category, price)

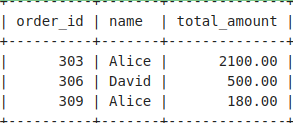
Order\_Items (order\_item\_id, order\_id, product\_id, quantity, subtotal) Sellers (seller\_id, seller\_name, city)

**Solve following SQL Queries:**

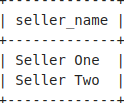
1. Retrieve all customers along with their corresponding orders (including customers who haven't ordered).  
     
   SQL Query:  
   SELECT c.customer\_id, c.name, o.order\_id, o.order\_date, o.total\_amount  
   FROM Customers\_180 c  
   LEFT JOIN Orders\_180 o ON c.customer\_id = o.customer\_id;  
     
   
2. List all orders along with the product names and their quantities.  
     
   SQL Query:  
   SELECT o.order\_id, p.product\_name, oi.quantity  
   FROM Orders\_180 o  
   JOIN Order\_Items\_180 oi ON o.order\_id = oi.order\_id  
   JOIN Products\_180 p ON oi.product\_id = p.product\_id;  
     
   
3. Find the total number of orders placed by each customer.  
     
   SQL Query:  
   SELECT c.customer\_id, c.name, COUNT(o.order\_id) AS total\_orders  
   FROM Customers\_180 c  
   LEFT JOIN Orders\_180 o ON c.customer\_id = o.customer\_id  
   GROUP BY c.customer\_id, c.name;  
     
   
4. Find the total number of products available in each category.  
     
   SQL Query:  
   SELECT category, COUNT(product\_id) AS total\_products  
   FROM Products\_180  
   GROUP BY category;



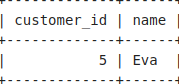
1. Retrieve the order details, including customer name and total amount, for all orders placed in the last 30 days.  
     
   SQL Query:  
   SELECT o.order\_id, c.name, o.total\_amount  
   FROM Orders\_180 o  
   JOIN Customers\_180 c ON o.customer\_id = c.customer\_id  
   WHERE o.order\_date >= CURRENT\_DATE - INTERVAL '30 days';

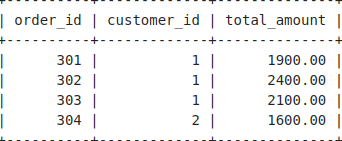
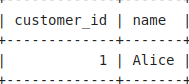


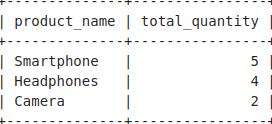
1. Retrieve the names of sellers who sell a specific product (e.g., "Laptop").  
     
   SQL Query:  
   SELECT DISTINCT s.seller\_name  
   FROM Sellers\_180 s  
   JOIN Product\_Sellers\_180 ps ON s.seller\_id = ps.seller\_id  
   JOIN Products\_180 p ON ps.product\_id = p.product\_id  
   WHERE p.product\_name = 'Laptop';



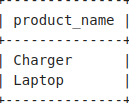
1. Show all customers who have never placed an order.  
     
   SQL Query:  
   SELECT c.customer\_id, c.name  
   FROM Customers\_180 c  
   LEFT JOIN Orders\_180 o ON c.customer\_id = o.customer\_id  
   WHERE o.order\_id IS NULL;



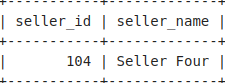
1. Retrieve details of orders where the total amount is greater than the average order total.( solve suing subquery)  
     
   SQL Query:  
   SELECT order\_id, customer\_id, total\_amount  
   FROM Orders\_180  
   WHERE total\_amount > (SELECT AVG(total\_amount) FROM Orders\_180);
2. Find customers who have placed at least two orders.  
     
   SQL Query:  
   SELECT c.customer\_id, c.name  
   FROM Customers\_180 c  
   JOIN Orders\_180 o ON c.customer\_id = o.customer\_id  
   GROUP BY c.customer\_id, c.name  
   HAVING COUNT(o.order\_id) >= 2;
3. Find the top 3 most ordered products based on quantity sold.  
     
   SQL Query:  
   SELECT p.product\_name, SUM(oi.quantity) AS total\_quantity  
   FROM Products\_180 p  
   JOIN Order\_Items\_180 oi ON p.product\_id = oi.product\_id  
   GROUP BY p.product\_name  
   ORDER BY total\_quantity DESC  
   LIMIT 3;



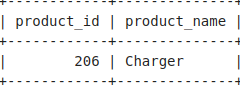
1. Display the product names that are sold by multiple sellers.  
     
   SQL Query:  
   SELECT p.product\_name  
   FROM Products\_180 p  
   JOIN Product\_Sellers\_180 ps ON p.product\_id = ps.product\_id  
   GROUP BY p.product\_name  
   HAVING COUNT(DISTINCT ps.seller\_id) > 1;



1. Retrieve sellers who have never sold any product.  
     
   SQL Query:  
   SELECT s.seller\_id, s.seller\_name  
   FROM Sellers\_180 s  
   LEFT JOIN Product\_Sellers\_180 ps ON s.seller\_id = ps.seller\_id  
   WHERE ps.product\_id IS NULL;



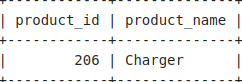
1. Find all products that have never been ordered.  
     
   SQL Query:  
   SELECT p.product\_id, p.product\_name  
   FROM Products\_180 p  
   LEFT JOIN Order\_Items\_180 oi ON p.product\_id = oi.product\_id  
   WHERE oi.order\_item\_id IS NULL;



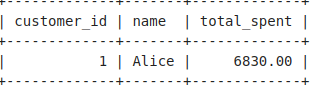
14. Retrieve the names of customers who have placed the highest number of orders.  
  
SQL Query:  
SELECT c.name  
FROM Customers\_180 c  
JOIN Orders\_180 o ON c.customer\_id = o.customer\_id  
GROUP BY c.customer\_id, c.name  
HAVING COUNT(o.order\_id) = (  
 SELECT MAX(order\_count)  
 FROM (  
 SELECT COUNT(order\_id) AS order\_count  
 FROM Orders\_180  
 GROUP BY customer\_id



15. Find all customers who have ordered more than 5 different products.  
  
SQL Query:  
SELECT c.customer\_id, c.name  
FROM Customers\_180 c  
JOIN Orders\_180 o ON c.customer\_id = o.customer\_id  
JOIN Order\_Items\_180 oi ON o.order\_id = oi.order\_id  
GROUP BY c.customer\_id, c.name  
HAVING COUNT(DISTINCT oi.product\_id) > 5;

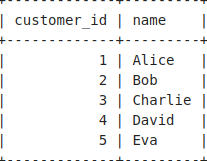


16. Find products that are sold by at least two different sellers but have never been ordered.  
  
SQL Query:  
SELECT p.product\_id, p.product\_name  
FROM Products\_180 p  
JOIN Product\_Sellers\_180 ps ON p.product\_id = ps.product\_id  
GROUP BY p.product\_id, p.product\_name  
HAVING COUNT(DISTINCT ps.seller\_id) > 1  
AND NOT EXISTS (  
 SELECT 1  
 FROM Order\_Items\_180 oi  
 WHERE oi.product\_id = p.product\_id  
);



17.Find the customer who has spent the most money overall.

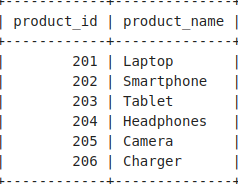
SQL Query:  
SELECT c.customer\_id, c.name, SUM(o.total\_amount) AS total\_spent  
FROM Customers\_180 c  
JOIN Orders\_180 o ON c.customer\_id = o.customer\_id  
GROUP BY c.customer\_id, c.name  
ORDER BY total\_spent DESC  
LIMIT 1;



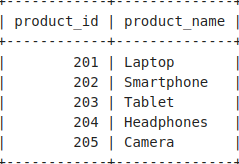
18. Find all customers who have either placed an order or live in the same city as a seller.

(Hint: Use UNION to combine customers who placed orders and those in seller cities.)  
  
SQL Query:  
SELECT DISTINCT customer\_id, name  
FROM Customers\_180  
WHERE customer\_id IN (SELECT DISTINCT customer\_id FROM Orders\_180)

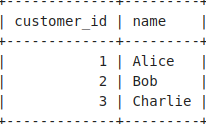
UNION  
SELECT DISTINCT customer\_id, name  
FROM Customers\_180  
WHERE city IN (SELECT DISTINCT city FROM Sellers\_180);



19. Retrieve all products that are either in stock with at least one seller or have been ordered at least once.  
  
SQL Query:  
SELECT DISTINCT p.product\_id, p.product\_name  
FROM Products\_180 p  
WHERE p.product\_id IN (SELECT DISTINCT product\_id FROM Product\_Sellers\_180)  
UNION  
SELECT DISTINCT p.product\_id, p.product\_name  
FROM Products\_180 p  
WHERE p.product\_id IN (SELECT DISTINCT product\_id FROM Order\_Items\_180);

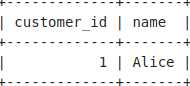


20. Retrieve products that have been both ordered and are currently in stock.  
  
SQL Query:  
SELECT p.product\_id, p.product\_name  
FROM Products\_180 p  
WHERE p.product\_id IN (SELECT product\_id FROM Order\_Items\_180)  
INTERSECT  
SELECT p.product\_id, p.product\_name  
FROM Products\_180 p  
WHERE p.product\_id IN (SELECT product\_id FROM Product\_Sellers\_180);



21. Find customers who have both placed an order and live in a city where a seller is located.  
  
SQL Query:  
SELECT c.customer\_id, c.name  
FROM Customers\_180 c  
WHERE c.customer\_id IN (SELECT DISTINCT customer\_id FROM Orders\_180)  
INTERSECT

SELECT c.customer\_id, c.name  
FROM Customers\_180 c  
WHERE c.city IN (SELECT DISTINCT city FROM Sellers\_180);



22. Retrieve all customers who have placed at least one order in each year available in the database.  
  
SQL Query:  
SELECT customer\_id, name  
FROM Customers\_180 c  
WHERE NOT EXISTS (  
 SELECT DISTINCT EXTRACT(YEAR FROM o.order\_date) AS year  
 FROM Orders\_180 o  
 WHERE NOT EXISTS (  
 SELECT 1  
 FROM Orders\_180 o2  
 WHERE o2.customer\_id = c.customer\_id   
 AND EXTRACT(YEAR FROM o2.order\_date) = EXTRACT(YEAR FROM o.order\_date)  
 )  
);

→ Query 22 returns no output because no single customer has placed orders in every year present in the Orders table (2023, 2024, and 2025). The NOT EXISTS condition filters out any customer missing even one year, leading to an empty result set.