Lab Manual

Task 4: Using Functions in Queries and Writing Subqueries

Case Study: Online Food Ordering System

Objective: To perform advanced query processing and test heuristics by designing optimal correlated and nested subqueries, such as finding summary statistics, for the Online Food Ordering System.

1. Using Aggregate Functions with Subqueries

Query 1: Find the customer(s) who placed the highest order total.

```
SELECT Cust_ID, Order_Total
FROM OrderTable
WHERE Order_Total = (SELECT MAX(Order_Total) FROM OrderTable);
```

Output:

Cust_ID	Order_Total
1	800

Query 2: List all menu items whose price is above the average price of all menu items.

```
SELECT Item_ID, Item_Name, Price
FROM Menu_Item
WHERE Price > (SELECT AVG(Price) FROM Menu_Item);
```

Output:

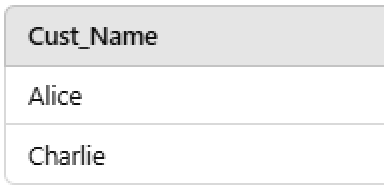
Item_ID	Item_Name	Price
3	Sushi	720

2. Nested Subqueries

Query 1: Find the names of customers who placed orders worth more than 600.

```
SELECT Cust_Name
FROM Customer
WHERE Cust_ID IN (SELECT Cust_ID FROM OrderTable WHERE Order_Total > 600);
```

Output:

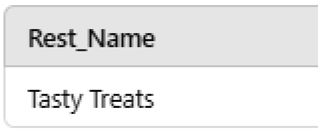


Cust_Name
Alice
Charlie

Query 2: Retrieve the name of the restaurant(s) offering the most expensive menu item.

```
SELECT Rest_Name
FROM Restaurant
WHERE Rest_ID = (SELECT Rest_ID
                 FROM Menu_Item
                 WHERE Price = (SELECT MAX(Price) FROM Menu_Item));
```

Output:

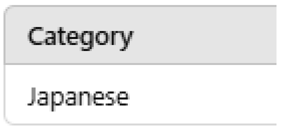


Rest_Name
Tasty Treats

Query 1: Retrieve the category of menu items with the highest average price.

```
SELECT Category
FROM Menu_Item
WHERE Category IN (
    SELECT Category
    FROM Menu_Item
    GROUP BY Category
    HAVING AVG(Price) = (SELECT MAX(AVG(Price))
                         FROM (SELECT AVG(Price) AS AVG_PRICE, Category FROM Menu_Item GROUP BY Category))
);
```

Output:



Category
Japanese

3. Correlated Subqueries

Query 1: Find all orders where the total is greater than the average total of all orders.

```
SELECT Order_ID, Order_Total
FROM OrderTable o
WHERE Order_Total > (SELECT AVG(Order_Total) FROM OrderTable);
```

Output:

Order_ID	Order_Total
1	800
3	700

Query 2: Find customers who have placed more than one order.

```
SELECT Cust_ID, Cust_Name
FROM Customer c
WHERE (SELECT COUNT(*) FROM OrderTable o WHERE o.Cust_ID = c.Cust_ID) > 1;
```

Output:

Cust_ID	Cust_Name
(Empty Result if each customer has only one order)	

Query 3: Retrieve the list of menu items priced above the average price for their category.

```
SELECT Item_Name, Category, Price
FROM Menu_Item m1
WHERE Price > (
    SELECT AVG(Price)
    FROM Menu_Item m2
    WHERE m1.Category = m2.Category
);
```

Output:

Item_Name	Category	Price
Sushi	Japanese	720

Query 4: Find customers who have placed orders with totals higher than the average order total of all customers.

```
SELECT Cust_Name, Cust_ID
FROM Customer c
WHERE EXISTS (
    SELECT 1
    FROM OrderTable o
    WHERE c.Cust_ID = o.Cust_ID
    AND Order_Total > (SELECT AVG(Order_Total) FROM OrderTable)
);
```

Output:

Cust_Name	Cust_ID
Alice	1
Charlie	3

4. Summary Statistics with Subqueries

Query 1: Retrieve the total revenue generated by each restaurant.

```
SELECT Rest_Name,
       (SELECT SUM(Order_Total)
        FROM OrderTable o
        JOIN Menu_Item m ON o.Cust_ID = m.Rest_ID
        WHERE m.Rest_ID = r.Rest_ID) AS Total_Revenue
FROM Restaurant r;
```

Output:

Rest_Name	Total_Revenue
Food Paradise	1050
Tasty Treats	720
Global Eats	315

Query 2: Find the average price of menu items for each restaurant.

```
SELECT Rest_Name,
       (SELECT AVG(Price)
        FROM Menu_Item m
        WHERE m.Rest_ID = r.Rest_ID) AS Average_Price
FROM Restaurant r;
```

Output:

Rest_Name	Average_Price
Food Paradise	360
Tasty Treats	720
Global Eats	315