

**By: Rana Ghazzi**

## Advanced Techniques for Business Solutions

- Objective: analyzing sales data and providing actionable insights to guide strategic decisions.
- I will utilize advanced SQL queries, including subqueries, multiple joins, set operations, and aggregate functions, to extract and interpret data

#1. find the top 5 customers with the highest total order amount.



The image shows a screenshot of a SQL IDE window. The window has a toolbar at the top with various icons for file operations, execution, and search. A dropdown menu is open, showing 'Limit to 200 rows'. The main area contains a SQL query with line numbers 1 through 12. The query is as follows:

```
1
2 • Select distinct
3   o.customerNumber,
4   c.customerName,
5   sum(od.priceEach* od.quantityOrdered) Over(Partition by o.customerNumber) as 'Total by Customer'
6 from
7   orders o
8 left join
9   orderdetails od
10  on od.orderNumber=o.orderNumber
11 left join customers c on o.customerNumber=c.customerNumber
12 order by 2 desc Limit 5;
```

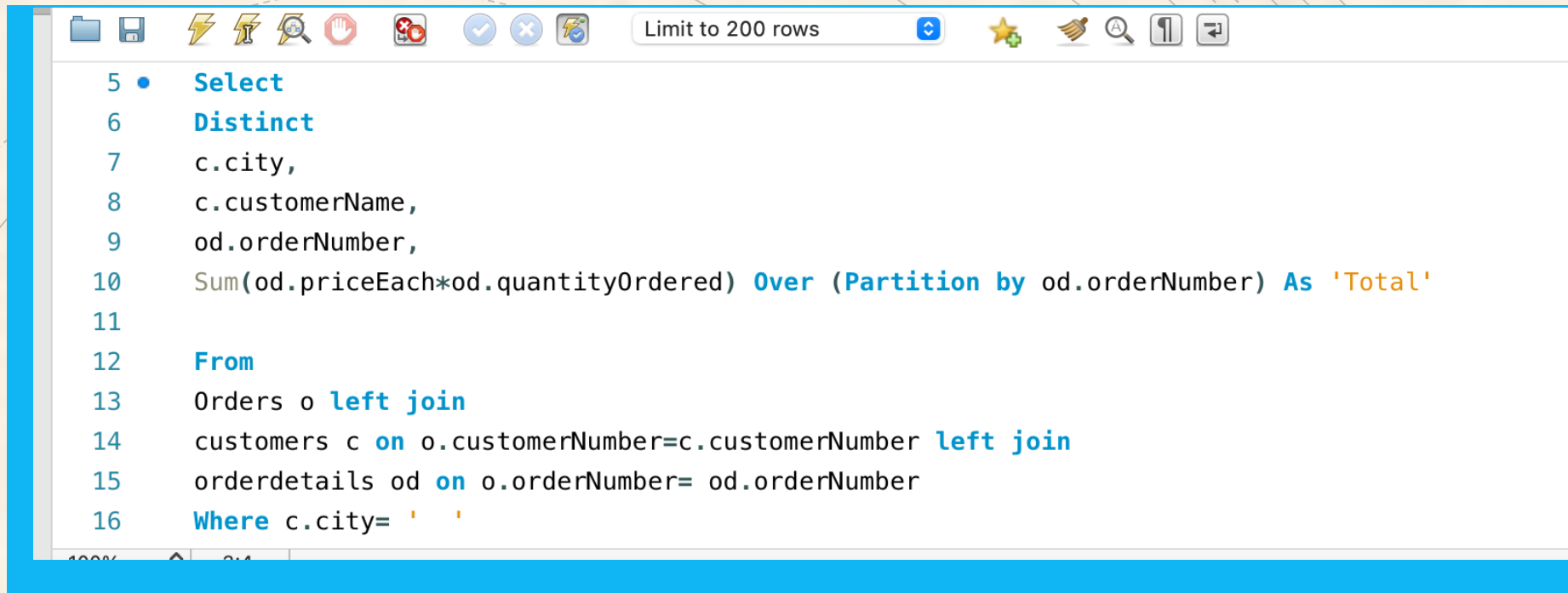
## 2.Retrieve the names of customers who have placed orders in the past 30 days.

```
4
5 • ⊖ With Rana as (
6   Select distinct
7
8   o.customerNumber,
9   max(o.orderDate) over(Partition by o.customerNumber) as 'Last_Order_Date'
10  from
11  orders o
12  )
13
14  select*
15  from
16  Rana left join
17  customers c ON Rana.customerNumber = c.customerNumber
18  Where
19  datediff( o.shippedDate ,Rana.Last_Order_Date) <=30
20  order by 1,2
```

# 3. Find the products that have been ordered at least three times.

```
3 • Select  
4 Distinct  
5   count(od.orderNumber) 'Ordered Products',  
6   od.productCode  
7 from  
8 orderdetails od  
9 Group by od.productCode  
10 Having count(od.orderNumber)>=3  
11  
12
```

#### 4.Retrieve the order details for orders placed by customers from a specific city.



```
Limit to 200 rows
5 • Select
6 Distinct
7 c.city,
8 c.customerName,
9 od.orderNumber,
10 Sum(od.priceEach*od.quantityOrdered) Over (Partition by od.orderNumber) As 'Total'
11
12 From
13 Orders o left join
14 customers c on o.customerNumber=c.customerNumber left join
15 orderdetails od on o.orderNumber= od.orderNumber
16 Where c.city= ' '
```

5. Write a query to find the customers who have placed orders for products with a price greater than \$100.

```
1 # 5. Write a query to find the customers who have placed orders for products
2 • select
3 c.customerNumber,
4 c.customerName
5 from
6 orderdetails od
7 left join orders o on od.orderNumber = o.orderNumber
8 left join customers c on o.customerNumber = c.customerNumber
9 where
10 od.priceEach > 100
```

## 6. Get the average order amount for each customer.

```
#6. Get the average order amount for each customer.
5 • with Total AS (
6   select
7   od.orderNumber
8   ,sum(od.priceEach*od.quantityOrdered) over(Partition by od.orderNumber) as 'Total Per Order'
9   from
10  orderdetails od )
11  Select
12  Distinct
13  c.CustomerNumber,
14  c.ContactName,
15  o.OrderNumber,
16  #Total.[Total Per Order],
17 ❌ avg(Total.[Total Per Order]) OVER(Partition by o.CustomerNumber) as 'Average per Customer'
18  from
19  Total
20  left join orders o on Total.orderNumber = o.orderNumber
21  left join customers c on o.customerNumber = c.customerNumber
22  order by 1
```

## 7. Find the products that have never been ordered.

```
2
3 • Select
4   p.productCode,
5   ifnull(Count(od.ProductCode),0) 'Total Orders of Products',
6   od.productCode
7 from
8   products p
9 left join
10  orderDetails od
11  on p.productCode = od.productCode
12  group by p.productCode
13  Having od.productCode is Null
14
```



## 8.Retrieve the names of customers who have placed orders on weekends (Saturday or Sunday).

```
Limit to 200 rows

1 • With All_Days as
2   ( Select
3     (CASE weekday( o.OrderDate)
4       WHEN 1 THEN 'SUNDAY'
5       WHEN 2 THEN 'MONDAY'
6       WHEN 3 THEN 'TUESDAY'
7       WHEN 4 THEN 'WEDNESDAY'
8       WHEN 5 THEN 'THURSDAY'
9       WHEN 6 THEN 'FRIDAY'
10      WHEN 7 THEN 'SATURDAY' END )as 'Day',
11     o.CustomerNumber,
12     CAST(o.OrderDate AS Date) 'Order Date'
13   From
14     orders o )
15
16   Select*
17   from
18   all_Days
19   left join
20   customers c on all_Days.customerNumber=c.customerNumber
21   Where
22   Day in ('SUNDAY', 'SATURDAY')
```

## 9. Get the total order amount for each month..

```
3 • with All_data as (  
4   Select  
5   SUM(od.Quantity * od.UnitPrice) As 'Total',  
6   od.OrderID,  
7   o.OrderDate,  
8   DATEPART(MONTH, OrderDate) As 'Month'  
9   from  
10  orderdetails od  
11  left join orders o on od.orderid = o.orderid  
12  Group by  
13  od.OrderID, o.OrderDate, DATEPART(MONTH, OrderDate)  
14  )  
15  Select  
16  Distinct  
17  All_data.OrderID,  
18  All_data.OrderDate,  
19  All_data.Total,  
20  All_data.Month,  
21  sum(Total) OVER(Partition by All_data.Month) as 'Total per  
22  Month' from  
23  All_data  
24  Order by 1
```

## 10. Write a query to find the customers who have placed orders for more than two different products.

```
1 #10. Write a query to find the customers who have placed orders for more than two different products.
2 • ○ With All_data as( Select
3   count(Distinct od.ProductID) as 'Count of Products',
4   OrderID
5   From
6   OrderDetails od
7   Group by
8   OrderID
9   Having
10  count(Distinct od.ProductID)> 1
11 )
12 Select
13  c.CustomerID, c.ContactName,
14  o.OrderID,
15  od.ProductID,
16  od.Quantity,
17  All_data.[Count of Products]
18  from
19  All_data
20  left join
21  OrderDetails od on All_data.OrderID = od.OrderID left join
22  Orders o on All_data.OrderID = o.OrderID
23  left join Customers c on o.CustomerID = c.CustomerID
24  Order by 2
```

**That's it!**

**Thank You.**



**By: Rana Ghazzi**