Database_Insights_and_SQL_Solutions

❖ Retrieve the nth Highest Salary?

Nth Highest Salary:

```
with cte as (select *,
DENSE_RANK() OVER (Order By Salary DESC) as Ranking
from Emp)
select * from cte where Ranking = n;
```

By Department:

```
with cte as (select *,
       DENSE_RANK() OVER (Partition By Dept_ID Order By Salary
DESC) as Ranking
       from Emp)
select * from cte where Ranking = n;
```

❖ Percentage distribution of total sales for each product?

```
with cte as (select Product_ID,
              SUM(Sales) as Total_Sale
              from Orders
              Group by Product_ID),
cte2 as (select Product_ID, Total_Sale,
              FORMAT(Total_Sale * 100.0/SUM(Total_Sale)
OVER(),'N2') as Percentage_Dist
              from cte)
select * from cte2
Order By Total_Sale DESC;
```

❖ Identify customers who have been active throughout the entire year, placing orders in every month. This can be useful for understanding customer loyalty and engagement?

```
select Customer_ID from Orders where YEAR(OrderDate) = 2023
Group By Customer_ID
Having COUNT(DISTINCT MONTH(OrderDate)) = 12;
```

❖ Calculate Cumulative Sales Percentage Contribution by Product?

```
with cte as (select Product_ID, SUM(Sales) as Total_Sale
              from Orders
              group By Product_ID),
cte2 as (select Product_ID, Total_Sale,
              Total_Sale * 100.0/sum(Total_Sale) over() as
Percentage_Dist
              from cte)
select Product_ID,Total_Sale,Percentage_Dist,
SUM(Percentage_Dist) over (Order By Total_Sale DESC)
as Cummilative_Sale from cte2;
```

❖ Identify employees whose salary is higher than their manager's salary?

```
with cte as (select Emp_ID, Salary as Manager_Sal from Employees
              where Role = 'Manager')
select e.Emp_ID, e.Emp_Name, e.Dept_ID, e.Salary, e.Role
from Employees e Join cte c
on e.Manger_ID = c.Emp_ID
where e.Salary > c.Manager_Sal;
```

Database_Insights_and_SQL_Solutions

❖ Identify employees whose salary is higher than the highest salary of managers within their department?

```
with cte as (select Dept_ID, MAX(Salary) as Manager_Sal from
Employees
              where Role = 'Manager'
              Group By Dept_ID)
select e.Emp_ID, e.Emp_Name, e.Dept_ID, e.Salary, e.Role
from Employees e Join cte c
on e.Dept_ID = c.Dept_ID
where e.Salary > c.Manager_Sal;
```

❖ Find duplicate records in a table?

```
with cte as (select *,
ROW_Number() Over (Partition By Customer_ID Order By Customer_ID)
as Duplicates
from Customer)
select * from cte where Duplicates > 1;
```

❖ Delete duplicate records in a table>

```
with cte as (select *,
       ROW_Number() Over (Partition By EmpID Order By EmpID) as
Duplicates
       from Employees)
delete from cte where Duplicates > 1;

select * from Employees
```

❖ Find Employees without department?

```
select Emp_ID, Emp_name from EMployees where Dept_ID is null;
```

- If data is in two different tables we can make use of Left Anti Join (Note: used as an example to show case use of Left Anti Join)

```
select e.Emp_ID as Emp_ID, e.Emp_Name as Emp_Name
from Employees e left join Departments d
on e.Emp_ID = d.Emp_ID
where d.Emp_ID is null;
```

❖ Calculate total revenue for product ?

```
select Product_ID,
       SUM(Quantity*Unit_Price) as Total_Revenue
from Orders
Group By Product_ID
Order By Total_Revenue DESC;
```

❖ Find the customers who made purchases but never returned products

```
select Customer_ID from
       (select o.Customer_ID
              from Orders Left Join Returns r
              on o.Order_ID = r.Order_ID
              where r.Order_ID is null) as Subquery;
```

Database_Insights_and_SQL_Solutions

❖ Temp Table with number of orders per customer - Customer Popularity?

```
select Customer_ID, COUNT(DISTINCT Order_ID) as Num_Orders into
#Popularity from Orders
Group By Customer_ID;

Select * from #Popularity
Order By Num_Orders Desc;
```

❖ Get the latest Order Placed by each customer?

```
with cte as (select Customer_ID, MAX(OrderDate) as Latest_Order
from Orders
Group By Customer_ID)
select a.Customer_ID as Customer_ID, a.Latest_Order as
Latest_Order, b.Order_ID
from cte a join Orders b
on a.Customer_ID = b.Customer_ID
where a.Latest_Order = b.OrderDate;
```

❖ Find products never sold?

```
select p.Product_ID
from Products p Left Join Orders o
on p.Product_ID = o.Product_ID
where o.Product_ID is Null;
```

❖ Count the number of customers who have placed more than 5 orders

```
select COUNT(*) as NumCust_AboveFiveorders from
       (select Customer_ID from Orders
       Group By Customer_ID
       Having COUNT(DISTINCT Order_ID) > 5) as Subquery;
```

❖ Retrieve Customers with orders above the average order value?

```
select Customer_ID, Order_ID, Sales from Orders
where Sales > (select AVG(Sales) from orders)
Order By Sales DESC;
```

❖ Find moving average of sales over the last 3 days?

```
select OrderDate,
       AVG(Sales) OVER (Order by OrderDate ROWS between 2
preceding and current row) as MovingAverage
from Orders
where DATEDIFF(Day, OrderDate, Getdate()) <= 3;
```

❖ Rolling 3-Month Total Sales for Each Product?

```
with cte as (select Product_ID,
             DATENAME(MONTH, OrderDate) as MonName,
             MONTH(OrderDate) as MonNum,
             SUM(Sales) as Total_Sale
from Orders
group By Product_ID, DATENAME(MONTH, OrderDate),MONTH(OrderDate))
select Product_ID, MonName, Total_Sale,
             SUM(Total_Sale) OVER (Partition By Product_ID Order
By MonNum rows between 2 preceding and current row) as
RollingThreeMonths
from cte
Order By Product_ID, MonNum;
```

Database_Insights_and_SQL_Solutions

- Analyse number of distinct orders placed by each customer on each order date, with the results sorted by the order date in descending order. This can help in understanding customer ordering patterns and trends over time?

```
select Customer_ID, OrderDate,
       COUNT(DISTINCT Order_ID) as Num_Orders
from Orders
Group By Customer_ID, OrderDate
Order By Num_Orders DESC;
```

- Find all Employees Hired on weekends

```
SELECT Emp_ID, Emp_Name
FROM Employees
WHERE DATENAME(weekday, Hire_date) IN ('Friday', 'Saturday',
'Sunday');
```

- Analyse order patterns of customers by calculating the time intervals between consecutive orders and summing these intervals for each customer?

```
with cte as (select Customer_ID,OrderDate,
       LAG(OrderDate) OVER (Partition By Customer_ID Order By
OrderDate) as PreviousOrder
from Orders)
select Customer_ID, SUM(DATEDIFF(Day,OrderDate,PreviousOrder))
       as Total_Difference
from cte where PreviousOrder is not null
Group By Customer_ID
Order By Total_Difference ASC;
```

- Identify products that are sold in every region?

```
select Product_ID from Orders
Group By Product_ID
Having COUNT(DISTINCT Region) = (select COUNT(Distinct Region)
from Orders)
```

- Rank Customers Based on Frequency of Purchases

```
with cte as (select Customer_ID,
             COUNT(DISTINCT Order_ID) as Num_Orders,
             SUM(Sales) as Total_Sale,
             DENSE_RANK() OVER (Order By COUNT(DISTINCT
Order_ID) DESC) as Ranking
from Orders
Group By Customer_ID)
select * from cte;
```

Database_Insights_and_SQL_Solutions

❖ Analyse monthly sales data, understand the distribution of sales across different months, and track the growth or decline in sales percentage from month to month?

```
with cte1 as (select DATENAME(Month, OrderDate) as MonName,
       Month(OrderDate) as MonNum, SUM(Sales) as Total_Sale
from Orders
Group By DATENAME(Month, OrderDate), Month(OrderDate)),
cte2 as (select MonName,MonNum, Total_Sale,
       Total_Sale*100.0/SUM(Total_Sale) OVER () as
Percentage_Dist
from cte1)
select MonName, Total_Sale, Percentage_Dist,
       Percentage_Dist-LAG(Percentage_Dist) OVER (Order By
MonNum) as Growth_Percentage
from cte2
Order By MonNum
```

❖ Identify customers who have the longest periods of inactivity between their orders. This information can be useful for customer retention strategies, identifying at-risk customers, and understanding customer behaviour?

```
with cte as (select Customer_ID, OrderDate,
       LAG(OrderDate) Over (Partition By Customer_ID Order By
OrderDate) as Previous_Order
from Orders)
select Customer_ID, MAX(DATEDIFF(day, Previous_Order, OrderDate))
as Max_Inactivity
from cte where Previous_Order is not null
Group by Customer_ID
Order By Max_Inactivity DESC;
```

❖ Filter Orders with Most Recent Price for Each Product?

```
SELECT o.Order_ID, o.Customer_ID, o.Product_ID, o.Sale,
o.Unit_Price, p.change_date
FROM Orders o JOIN Products p ON o.Product_ID = p.Product_ID
WHERE DATEDIFF(day, p.change_date, GETDATE()) < 30;
```

❖ Calculate Total Sales of Top 10% Customers by Spend?

```
with cte as (select Customer_ID, SUM(Sales) as Total_Sale from
Orders
Group By Customer_ID),
cte2 as (select Customer_ID, Total_Sale ,
       PERCENTILE_CONT(0.9) within group (order By Total_Sale)
OVER () as NintythContribution
from cte)
select Customer_ID, Total_Sale from cte
where Total_Sale > (select MAX(NintythContribution) from cte2)
```

❖ Calculate the Median Sales Amount for Each Product?

```
select Product_ID, Sales,
       PERCENTILE_CONT(0.5) within group (Order By Sales) over
(Partition by Product_ID) as Median
from Orders
Order By Product_ID;
```

Database_Insights_and_SQL_Solutions

- Create a CTE to calculate the average Client_Income by Client_Marital_Status and Client_Gender. Filter out the clients who have defaulted on loan payments (`Default` = 1). Then, print only those groups where the average income is greater than $50k?

```
with cte as (select Client_Gender,
                    Client_Marital_Status,
                    AVG(Client_Income)  as Average_Income
from LoansData
Where [Default] <> 1
Group By Client_Gender,Client_Marital_Status
Having AVG(Client_Income) > 50000)
select * from cte;
```

- Write a CTE to find the clients who have both a car and bike (Car_Owned = 1 and Bike_Owned = 1) and Calculate the total credit amount (Credit_Amount) they have applied for. filterout clients with any active loans (Active_Loan = 1) ?

```
with cte as (select ID as Client_ID,
             SUM(Credit_Amount) as Total_Credit
from LoansData
Where Car_Owned = 1 AND Bike_Owned = 1 AND Active_Loan <> 1
Group By ID)
Select * from cte
Order By Total_Credit DESC;
```

- Create a temp table that should contain, clients' ID, their Age_Days converted to years, and Employed_Days, converted to years. Then, print only clients where the difference between their age and employment years is more than 15 years?

```
with cte as (select ID as Client_ID,
                      (Age_Days/365) as Age,
                     (Employed_Days/365) as Experience
from LoansData)
Select * into #Client_AgevsExperience from cte where (Age-
Experience) > 15;

Select * from #Client_AgevsExperience
Order By Age;
```

- Create a view to store ID, Client_Income, Client_Education, Client_Gender, and `Client_Occupation` and it contain the data based on the below filter conditions.
  - clients (ID's) who have provided all three types of phone numbers (Mobile_Tag, Homephone_Tag, and Workphone_Working all equal to 1), AND
  - Client have no active loans (Active_Loan = 0)?

```
Create View Potential_Clients as
select ID, Client_Income, Client_Education, Client_Gender,
Client_Occupation
from LoansData where Homephone_Tag = 1 AND Mobile_Tag = 1 AND
Workphone_Working = 1 AND Active_Loan <> 1;

Select * from Potential_Clients;
```

Database_Insights_and_SQL_Solutions

- Create a temporary table that should contain and that calculates the total number of loans (Credit_Amount) requested by clients grouped by Type_Organization. Then, print the top 3 organizations with the highest total loan amounts?

```
with cte as (select Type_Organization,
                    COUNT(*) as Num_Loans,
                    SUM(Credit_Amount) as Total_Credit
From LoansData
Group By Type_Organization)
select * into #Organization_Loans_Summary from cte;

Select Top 3 * from #Organization_Loans_Summary
Order By Total_Credit DESC;
```

(Or)

```
with cte as (select Type_Organization,
             COUNT(*) as Num_Loans,
             SUM(Credit_Amount) as Total_Credit,
             DENSE_RANK() OVER (Order By SUM(Credit_Amount)
DESC) as Ranking
from LoansData
Group By Type_Organization)
select * into #Organization_Loans_Summary from cte where
       Ranking <= 3;
```

- create a view that should contain Credit_Amount, Loan_Annuity, Loan_Contract_Type, Client_Income, Client_Gender, Client_Marital_Statuss and filer only ID's, who applied for loans on Fridays?

```
Create View Applications_on_Firday as
select ID,
Credit_Amount,Loan_Annuity,Loan_Contract_Type,Client_Income,
Client_Gender, Client_Marital_Status
from LoansData where Application_Process_Day = 5;

Select * from Applications_on_Firday;
```

- Create a join or CTE to Perform a self-join to identify clients who live in the same Client_Housing_Type and have the same Client_Marital_Status, but different (income type should be different) `Client_Income_Type`. and print only the clients' IDs and their corresponding income types?

```
with cte as (select ID, Client_Housing_Type,
             Client_Marital_Status,
             Client_Income_Type
from LoansData)
select DISTINCT a.ID as Client_ID, a.Client_Income_Type from cte
a join cte b
on a.Client_Housing_Type = b.Client_Housing_Type and
a.Client_Marital_Status = b.Client_Marital_Status
AND a.ID <> b.ID AND a.Client_Income_Type <>
b.Client_Income_Type;
```

- Write a query to find the maximum Loan_Annuity amount among clients who have at least 2 family members (Client_Family_Members >= 2)?

```
with cte as (select MAX(Loan_Annuity) as Max_Loan_Annuity
from LoansData where Client_Family_Members >= 2)
Select * from cte
```

Database_Insights_and_SQL_Solutions

- Print Credit_Amount, Loan annuity by Client_Housing_Type, Client occupation, Type Oraganization, and assign a distinct rank to every client_housing_type with in client occupation, and type, based on Credit Amount. Also, assign a rank without skipping the rank sequence every client_housing_type with in client occupation, and type, based on the Loan Ammunity ?

```
with cte as (select Client_Housing_Type, Type_Organization,
       Client_Occupation, SUM(Credit_Amount) as Total_Credit,
              SUM(Loan_Annuity) as Total_Annuity
       from LoansData
       Group By Client_Housing_Type, Type_Organization,
Client_Occupation),
cte2 as (select *,
       Row_Number() OVER (Partition By Type_Organization,
       Client_Occupation Order by Total_Credit DESC) as Distinct_Rank,
       DENSE_RANK() OVER (Partition By Type_Organization,
       Client_Occupation Order by Total_Credit DESC) as Sequential_Ranks
       from cte)

select * from cte2
ORDER BY Type_Organization, Client_Occupation, Distinct_Rank;
```

- Create a CTE to rank clients based on their Credit_Amount within each Client_Income_Type. Then, filter out only the top 3 clients with the highest Credit_Amount per income type. Include ID, Client_Income_Type, Credit_Amount, and the rank in the result ?

```
with cte as (select Client_Income_Type,
              ID,
              SUM(Credit_Amount) as Total_Credit,
              DENSE_RANK() OVER (Partition By Client_Income_Type Order
By SUM(Credit_Amount) DESC) as Ranking
from LoansData
Group By Client_Income_Type, ID)
select * from cte where Ranking <= 3;
```

- create temp table to store ID, Client_Income, Credit_Amount, Type_Organization, It should contain only Top 5 Client ID's within Type_Organization based on Client_income?

```
with cte as (select Type_Organization, ID,
       Credit_Amount,Client_Income,
       DENSE_RANK() OVER (Partition By Type_Organization Order By
Client_Income DESC) as Ranks
from LoansData)
select * into #Top5_Client_Income from cte where Ranks <= 5;

Select * from #Top5_Client_Income
Order By Type_Organization, Ranks;
```

Database_Insights_and_SQL_Solutions

- Create a CTE to calculate the weighted average of Score_Source_1, Score_Source_2, and Score_Source_3 for each Client_Occupation. The weights should be 0.5, 0.3, and 0.2, respectively. Select the Client_Occupation, weighted average score, and the total number of clients in each occupation. Only include occupations with more than 10 clients ?

```
with cte as (select Client_Occupation,
COUNT(ID) as Num_Clients,
SUM(Score_Source_1 * 0.5 +
       Score_Source_2 * 0.3 +
       Score_Source_3 * 0.2) / COUNT(ID) AS Weighted_Average_Score
from LoansData
Group By Client_Occupation
Having COUNT(ID) > 10)
select * from cte
Order By Num_Clients DESC;
```

- Write a CTE statement to print calculate the average heart rate, temperature, and BMI for each medical condition. and return the condition, average heart rate, average temperature, and average BMI?

```
with cte as (select Condition,
              AVG(Heart_Rate) as Avg_HeartRate,
              AVG(BMI) as Avg_BMI,
              AVG(Temperature) as Avg_TempTemperature
from HealthcareData
Group By Condition)
Select * from cte;
```

- Create a view to store PatienID, Name, Age, Gender, Condition, BMI, it should contain data only for Age above 70, BMI above 30, and a condition should be Hypertension or Diabetes ?

```
create view Critcial_Patients as
select h.Patient_ID, p.Name, h.Age, h.Gender, h.Condition, h.BMI from
HealthcareData h join Patients p on h.Patient_ID = p.Patient_ID where
h.Age > 70 OR h.BMI > 30 OR h.CONDITION IN ('Hypertension', 'Diabetes');

Select * from Critcial_Patients
```

- Create a view to show Patient_ID, count of appointments, and interval days (find the different between Min and Max appointment date within the year, and print only patients who have more than one appointment?

```
create view AppointmentSummary
as
select Patient_ID, YEAR(Appointment_Date) as year,
              COUNT(Appointment_Date) as Num_Appiontments,
              DATEDIFF(day, MIN(Appointment_Date),
MAX(Appointment_Date)) Interval
from HealthcareData
Group By Patient_ID,YEAR(Appointment_Date)
Having COUNT(Appointment_Date) > 1;

select * from AppointmentSummary;
```

Database_Insights_and_SQL_Solutions

- Create a temp table to store a PatientID, BloodPressure, Classification (You need to extract Classification on the below conditions...

  - If Blood_Pressure 80 and 120 then 'Normal'
  - If Blood_Pressure 80 and 129 then 'Elevated'
  - If Blood_Pressure 89 and 139 then 'Hypertension Stage 1'
  - else 'Hypertension Stage 2' ???

```
with cte as (select Patient_ID, Blood_Pressure,
   Case when CAST(SUBSTRING(Blood_Pressure, 1,
CHARINDEX('/',Blood_Pressure)-1) as INT) < 120 AND
   CAST(SUBSTRING(Blood_Pressure,
CHARINDEX('/',Blood_Pressure)+1,LEN(Blood_Pressure)) as INT) < 80
   then 'Normal'
   when CAST(SUBSTRING(Blood_Pressure, 1, CHARINDEX('/',Blood_Pressure)-1)
as INT) Between 120 and 129 AND
   CAST(SUBSTRING(Blood_Pressure,
CHARINDEX('/',Blood_Pressure)+1,LEN(Blood_Pressure)) as INT) < 80
   Then 'Elevated'
   when CAST(SUBSTRING(Blood_Pressure, 1, CHARINDEX('/',Blood_Pressure)-1)
as INT) Between 130 and 139 AND
   CAST(SUBSTRING(Blood_Pressure,
CHARINDEX('/',Blood_Pressure)+1,LEN(Blood_Pressure)) as INT) between 80 AND
89 then 'Hypertension Stage 1'
   else 'Hypertension Stage 2' end as [Classification]
from HealthcareData)
select * into #Patient_Classification from cte;

Select * from #Patient_Classification
Order By [Classification];
```

- create a CTEs to calculate the percentage distribution of patients across different insurance providers. The query should return the insurance provider's name and the percentage of total patients they cover

```
with cte as (select Insurance_Provider, COUNT(Patient_ID) as
Num_Patients from HealthcareData
Group By Insurance_Provider), cte2 as
(select Insurance_Provider, Num_Patients,
      FORMAT(Num_Patients*100.0/SUM(Num_Patients) OVER(),'N2') as
Percentage_Dist
from cte)
select * from cte2
Order By Num_Patients DESC;
```

- create a temp tables to find the top 5 most frequently prescribed medications across all conditions. The query should return the medication name, the count of prescriptions, and the conditions for which they were prescribed ?

```
SELECT Top 5 Medication,
    COUNT(Medication) AS Num_Prescribed,
    STRING_AGG(CAST(Condition AS VARCHAR(256)), ',') AS
Prescribed_For
       into #Medication_Trends
from HealthcareData
GROUP BY Medication
Order By Num_Prescribed DESC;

Select * from #Medication_Trends
Order By Num_Prescribed DESC;
```

Database_Insights_and_SQL_Solutions

- Create a view that predicts future appointment needs based on the current patients' conditions. The prediction logic should assume that patients with chronic conditions like 'Diabetes' or 'Hypertension' will need a follow-up every 3 months. The view should include the patient ID, condition, last appointment date, and the predicted next appointment date ?

```
Create view Patient_Appointments
as
select Patient_ID, Condition,
       MAX(Appointment_Date) as Last_Appointment,
       DATEADD(month, 3, MAX(Appointment_Date)) as
Next_Appointment
from HealthcareData where Condition in
('Diabetes','Hypertension')
Group By Patient_ID, Condition;

select * from Patient_Appointments
Order By Next_Appointment;
```

- Write a CTE statement to find clients who have active loans (`status = 'A'`) and their total loan amount. it should print the client's name, loan status, and total loan amount.

```
with cte as (
select Client_Name, [Status] as Loan_Status, SUM(amount) as
Total_Loan_Amount
from completedloan where [Status] = 'A'
Group By Client_Name, [Status])
Select * from cte
Order By Total_Loan_Amount DESC;
```

- Create a view that provides a summary of loans by status (`A` or `C`), including the total number of loans, the total loan amount, and the average loan duration for each status.

```
Create view LoanStatusSummary
as
select [Status], COUNT(*) Num_Loans,
             SUM(amount) as Total_LoanAmount,
             AVG(duration) as Avg_Duration
from completedloan where [Status] in ('A','C')
Group By [Status];

select * from LoanStatusSummary;
```

- Create a temp table to identify the top 10 transactions with the highest amount for each year. The query should include the transaction ID, account ID, amount, and rank of the transaction for that year ?

```
with cte as (select YEAR,trans_id, account_id, amount,
       DENSE_RANK() OVER (Partition By YEAR Order By Amount DESC) as
Ranks
from CompTrans where trans_id is not null)
select * into #top10TransCustomers from cte where Ranks <= 10;

select * from #top10TransCustomers
Order By Year DESC;
```

Database_Insights_and_SQL_Solutions

- Use an anti-join to find all clients who do not have any loans. The query should display the client's ID, name, and email?

```
select c.Account_ID as Client_ID,
              c.Name as Client_Name,
              c.Email as Client_Email
from completedclient c Left Join completedloan l
on c.Account_ID = l.Account_ID
where l.account_id is null;
```

- Create a view that calculates the average transaction amount for each client, grouped by account ID. Include the client’s ID, name, and the calculated average transaction amount ?

```
Create view Average_transactions
as
select t.account_id as Client_ID,
       c.Name as Client_Name,
       AVG(t.amount) as Average_Transaction
from CompTrans t join completedclient c
on t.account_id = c.account_id
Group By t.account_id, c.Name;

select * from Average_transactions
Order By Average_Transaction DESC;
```

- Create a stored procedure to categorize clients into different age groups (e.g., `<30`, `30-50`, `>50`) and count the number of clients in each group. The procedure should accept an age range as input ?

```
Create PROC Age_Trends (@Min_Age INT, @Max_Age INT)
as
select
       Case when Age < 30 then '<30'
       when Age between 30 and 50 then '30-50'
       else '>50' end as Age_Group,
       COUNT(*) as Num_Clients
from completedclient where Age Between @Min_Age and @Max_Age
Group By Case when Age < 30 then '<30'
       when Age between 30 and 50 then '30-50'
       else '>50' end
Order By Num_Clients DESC;

Exec Age_Trends 20, 50;
```

- Create a query using a temporary table to find the top 3 most frequent loan purposes. Include the purpose, count of loans, and percentage of total loans for each purpose ?

```
with cte as (select Purpose, COUNT(Loan_ID) as Num_Loans,
       DENSE_RANK() OVER (Order By COUNT(Loan_ID) DESC) as Ranking from
completedloan
       Group by Purpose),
cte2 as (select *,
       FORMAT(Num_Loans*100.0/SUM(Num_Loans) OVER(),'N2') as
Percentage_Dist from cte)
select Purpose, Num_Loans, Percentage_Dist into #Top3_Purpose from cte2
where Ranking <= 3;

Select * from #Top3_Purpose;
```

Database_Insights_and_SQL_Solutions

❖ Create a CTE to find all clients whose last payment date (from `CompTrans`) is more than 30 days before the current date. Include the client ID, name, and the date of the last payment ?

```
with cte as (select t.Account_id as Client_ID, c.Name as Client_Name,
                    MAX(t.fulldate) as Last_Payment_Date
from CompTrans t join completedclient c on t.account_id = c.account_id
Group By t.Account_id, c.Name
Having DATEDIFF(day, MAX(t.fulldate), GETDATE()) > 30)
Select * from cte
Order By Last_Payment_Date;
```

❖ Write SQL query using window functions to find the highest loan amount for each location. The query should include the location, loan ID, and the loan amount ?

```
with cte as (select [Location], Loan_id, Amount as Loan_Amount,
       DENSE_RANK() OVER (Partition By [Location] Order By Amount DESC)
as Ranking
from completedloan)
Select * from cte where Ranking = 1;
```

❖ Create a view to identify clients with more than 5 transactions in a single month. The view should include the client ID, name, and the number of transactions?

```
Create view MonthlyTransTrends as
select t.account_id as Client_ID, c.Name as Client_Name, DATENAME(Month,
t.fulldate) as Month_Name,
       Month(t.fulldate) as MonthNum,COUNT(t.trans_id) as
Num_Transactions
from CompTrans t join completedclient c
on t.account_id = c.account_id
Group By t.account_id, c.Name, MONTH(t.fulldate), DATENAME(Month,
t.fulldate)
Having COUNT(t.trans_id) > 5;

Select * from MonthlyTransTrends
Order By MonthNum;
```

❖ Write a SQL query using a window function to rank clients based on the number of loans they have taken. The query should display the client ID, name, total loans, and their rank?

```
select l.account_id as Client_ID, c.Name as Client_Name,COUNT(l.loan_id)
as Num_Loans,
       DENSE_RANK() over (Order By COUNT(l.loan_id) DESC) as Ranking
from completedloan l join completedclient c
on l.account_id = c.account_id
Group By l.account_id, c.Name
Order by Ranking;
```

Database_Insights_and_SQL_Solutions

- create a CTE to calculate the average loan payment amount and identify clients who pay above the average. The query should include the client ID, name, loan ID, and payment amount ?

```
with cte as (select l.Account_id as Client_ID,
              c.Name as Client_Name,
              l.Loan_ID  as Loan_ID,
              l.Payments as Payment_Amount
from completedloan l Join completedclient c
ON l.account_id = c.account_id
where l.Payments > (select AVG(payments) from completedloan))
select * from cte;
```

- create a stored procedure that accepts a transaction type (e.g., "Credit") and returns a summary of all transactions of that type, including the total amount, average amount, and the number of transactions ?

```
Create PROC TransactionTypeSummary (@Type VARCHAR(50))
as
select [Type],
       SUM(amount) as Total_Amount,
       AVG(amount) as Avg_Amount,
       COUNT(*) as Num_Transactions
from CompTrans where [Type] = @Type
Group By [Type];

Exec TransactionTypeSummary 'Credit'
```

- Create joins to find all clients who do not have any transactions. The query should display the client ID, name, and email ?

```
select c.Client_ID as Client_ID,
              c.Name as Client_Name,
              c.Email as Client_Email
from completedclient c Left JOIN CompTrans t
on c.account_id = t.account_id
Where t.account_id is NULL;
```

- Create a view that shows the complete loan payment history for each client, including the loan ID, client name, payment amount, and payment date ?

```
Create VIEW LoanPaymentSummary
as
select l.loan_id as Loan_ID,
              c.Name as Client_Name,
              l.Payments as Payment_Amount,
              l.fulldate as Payment_date
from completedloan l Join completedclient c
on l.account_id = c.account_id;

Select * from LoanPaymentSummary
Order By Payment_date;
```

Database_Insights_and_SQL_Solutions

- Print total number of loans, total loan amount, and average loan duration for each month. Include the month, year, total loans, total amount, and average duration?

```
select Year(fulldate) as Year, DATENAME(Month, fulldate) as Month_Name,
              COUNT(*) as Num_Loans,
              SUM(amount) as Total_Amount,
              AVG(duration) as Avg_Duration
from completedloan
Group By Year(fulldate), DATENAME(Month, fulldate), Month(Fulldate)
Order By Year DESC, Month(fulldate);
```

- create a CTE to find the top 5 clients with the highest total loan amounts. Include the client ID, name, and total loan amount in every year ?

```
with cte as (select YEAR(l.fulldate) as Year,
       l.account_id as Client_ID,
       c.Name as Client_Name,
       SUM(l.amount) as Total_Amount,
       DENSE_RANK() OVER (Partition By YEAR(l.fulldate) order By
SUM(l.amount) DESC) as Ranking
from completedloan l join completedclient c
ON l.account_id = c.account_id
Group By YEAR(l.fulldate), l.account_id, c.Name)
select * from cte where Ranking <= 5
order By Year DESC, Ranking
```

- Create a view that identifies accounts with more than 10 transactions in a day. Include the account ID, date, and the total number of transactions for that day?

```
create view MinTenTransPerDay
as
select Account_id,
       fulldate as Date,
       COUNT(*) as Num_Transactions
from CompTrans
Group By Account_id, fulldate
Having COUNT(*) > 10;

select * from MinTenTransPerDay
Order By Num_Transactions DESC
```

- Create a procedure to print Low Profit Margin,
  Input Parameters: ProfitMarginThreshold (Profit value)
  Output: Transactions with a profit margin below the specified threshold, categorized by product?

```
Create PROC LowProfitMargin (@ProfitThreshold FLOAT)
as
select Product_ID, Prod_Name,SUM(Profit) as Total_Profit
from Orders
group By Product_ID,Prod_Name
Having SUM(Profit) < @ProfitThreshold
Order By Total_Profit;

EXEC LowProfitMargin 30.0;
```

Database_Insights_and_SQL_Solutions

- create a CTE to calculate the year-over-year growth in the number of transactions for each account. Include the year, account ID, number of transactions, and growth percentage?

```
with cte as (select Account_id, Year,
COUNT(*) as Num_Trans
from CompTrans
Group By Account_id, Year)
select Account_id, Year, Num_Trans,
       Num_Trans-LAG(Num_Trans) OVER (Partition BY Account_ID Order By
Year) as YOYChangeinTransm,
       Num_Trans*100.0/SUM(Num_Trans) OVER (Partition BY Account_ID
Order By Year) as Growth_Percent
from cte where account_id is not Null
Order By Account_id, Year DESC;
```

- Create a stored procedure to print top N Customers by Region and Total Sales
  - Input Parameters: Region, Top N Value
  - It should print: Top N customers based on total sales within the specified region.

```
Create proc TopNByRegion @Region VARCHAR(80), @Ranking INT
as
with cte as (Select Region, Customer_ID,
          SUM(Sales) as Total_Sale,
          DENSE_RANK() OVER (Partition By Region Order By SUM(Sales) DESC)
as Ranking from orders
Group By Region, Customer_ID)
select * from cte where Region = @Region AND Ranking <= @Ranking;

Exec TopNByRegion 'East', 5;
```

- Create a procedure to print Category-Wise Profit and print this data for the given dates
  - Input Parameters: Category, @StartDate, EndDate
  - Profit trends for given category across provided date range?

```
create PROC ProfitByCategory (@Category VARCHAR(50), @Start_Date Datetime,
@End_Date Datetime)
as
Begin
with cte as (select Category,
    ROUND(SUM(Profit),2) as Profit
from Orders where OrderDate between @Start_Date and @End_Date AND Category =
@Category
Group By Category)
select * from cte
end;

Exec ProfitByCategory 'Furniture', '2019-11-08', '2020-10-01';
```

- Print clients who have not made any loan payments in the last 6 months (use max date as system date in the date set in CompTrans dataset). Include the client ID, name, last payment date, and loan status.

```
select l.account_id as Client_ID,
       c.Name as Client_Name, l.status as Loan_Status,
       MAX(l.fulldate) as Last_Payment_Date
from completedloan l Join completedclient c
ON l.account_id = c.account_id
Group By l.account_id, c.Name, l.status
Having DATEDIFF(Month, MAX(l.fulldate), GETDATE()) > 6;
```

Database_Insights_and_SQL_Solutions

- Create a procedure to print High-Discount Transactions
  Input Parameters: DiscountThreshold
  Output: List of transactions where the discount exceeds the threshold (Discount value), including Order ID, Product Name, and Profit?

```
Create PROC High_Discount_Orders (@DiscountThreshold FLOAT)
as
begin
with cte as (select Order_ID,
       Product_ID,
       Prod_Name,
       FORMAT(Cast(Discount as FLOAT),'N2') as Discount,
       Profit,
       Sales
from orders where FORMAT(CAST(Discount as FLOAT),'N2') >
@DiscountThreshold)
select * from cte
end;

Exec High_Discount_Orders 0.30;
```

- Create a procedure that accepts a minimum and maximum price range, locality, and type of property as inputs, and returns all matching properties along with their broker information?

```
create PROC Housing_Search
       (@Minimum_Price FLOAT, @Maximum_Price FLOAT, @Locality
VARCHAR(80), @Property_Type Varchar(40))
as
begin
select * from NY_Housing where
       (Price between @Minimum_Price and @Maximum_Price) and (LOCALITY =
@Locality) and (TYPE = @Property_Type)
end;

Exec Housing_Search 10000, 310000, 'New York','House For sale'
```

- Create a temporary table to store filtered results for further calculations like average price, total properties, and distribution by type?

```
with cte as (select [Type],AVG(CAST(PRICE as FLOAT))as Average_price,
       COUNT(*) as Num_Properties
       from NY_Housing
       Group By [Type])
select [Type], Average_price, Num_Properties,
       CONCAT(FORMAT(ROUND(Num_Properties*100.0/SUM(Num_Properties)
OVER(),2),'N2'),'%') as Dist_Percentage
into #t3 from cte;

select * from #t3
Order By Num_Properties Desc;
```

Database_Insights_and_SQL_Solutions

- Create a procedure to calculate the average price of properties grouped by property type and locality, using parameters for filtering based on beds, baths, or price range?

```
Create PROC Average_Cost (@Num_beds INT, @Num_Baths INT, @Min_Price
FLOAT, @Max_Price FLOAT)
as
begin
with cte as
(select Type, LOCALITY ,AVG(PRICE) as Average_Price from NY_Housing
       where BEDS = @Num_beds AND BATH = @Num_Baths AND PRICE between
@Min_Price AND @Max_Price
       Group By Type, LOCALITY)
select * from cte
Order By Average_Price DESC
end;

Exec Average_Cost 3,2,10000, 200000;
```

- Create a temp table to store intermediary calculations for debugging and performance improvement.

```
with cte as (select TYPE, LOCALITY, AVG(CAST(PRICE as FLOAT)) as
Average_Price,
       COUNT(*) as Num_Properties
       from NY_Housing
       Group By TYPE, LOCALITY)
select * into #Intermediary_calcs from cte;

select * from #Intermediary_calcs
Order By Num_Properties DESC;
```

- Create a procedure to identify the top 5 most expensive properties within a specific locality?

```
create PROC Top5Locality (@Locality VARCHAR(80))
as
begin
with cte as (select LOCALITY, SUBLOCALITY as County, CAST(PRICE as
FLOAT) as Price,
       DENSE_RANK() OVER (Partition By LOCALITY ORDER BY CAST(PRICE as
FLOAT) DESC) as RNK
       from NY_Housing)
select * from cte where RNK <= 5 and LOCALITY = @Locality
end;

Top5Locality 'New York'
```

- Create temp table to store intermediate results of all properties sorted by price before applying the top 5 filter?

```
with cte as (select *,
       DENSE_RANK() OVER (Order By CAST(Price as FLOAT) DESC) as RNK
       from NY_Housing)
select * into #Properties_High_Low from cte

select * from #Properties_High_Low
Order By RNK ASC;
```

Database_Insights_and_SQL_Solutions

- Create a stored procedure to generate summary statistics (count, min, max, average, and median price) for each locality and property type?

```
create PROC SummaryStats @Locality VARCHAR(256), @Type VARCHAR(256)
as
with cte as (select LOCALITY,Type,
             COUNT(*) as Num_Properties,
             MIN(CAST(Price as FLOAT)) as Min_Price,
             MAX(CAST(Price as FLOAT)) as Max_Price,
             AVG(CAST(Price as FLOAT)) as Avg_Price
from NY_Housing
Group By LOCALITY,Type),
cte2 as (select LOCALITY, TYPE,
       PERCENTILE_CONT(0.5) within group (Order By CAST(Price as FLOAT))
OVER (Partition By LOCALITY,Type) as Median
from NY_Housing)
select DISTINCT a.LOCALITY, a.TYPE, a.Num_Properties, a.Avg_Price,
a.Min_Price, a.Max_Price, b.Median
from cte a Join cte2 b on a.LOCALITY = b.LOCALITY AND a.TYPE = b.TYPE
where a.LOCALITY = @Locality AND a.TYPE = @Type;

Exec SummaryStats 'Kings County', 'House for Sale';
```

- Create a procedure to compare the average and median property prices across counties, grouped by property type?

```
Create PROC AvgMedianPriceCounty (@County VARCHAR(256), @Type
VARCHAR(256))
as
with cte as (select SUBLOCALITY as County,Type, AVG(Cast(Price as
FLOAT)) as Avg_Price
from NY_Housing
Group By SUBLOCALITY,TYPE),
cte2 as (select SUBLOCALITY as County,TYPE,
       PERCENTILE_CONT(0.5) within group (order By Cast(Price as FLOAT))
OVER (Partition By Type) as Median
from NY_Housing)
select DISTINCT a.County,a.TYPE, a.Avg_Price, b.Median from cte a join
cte2 b
on a.TYPE = b.Type and a.County = b.County where a.County = @County AND
a.TYPE = @Type;

Exec AvgMedianPriceCounty 'New York' ,'House for sale';
```

- Create a procedure that accepts a price range and locality as inputs and finds properties offering the highest number of beds and baths?

```
Create PROC Max_Beds_Baths (@Min_Price FLOAT, @Max_Price FLOAT,
@Locality VARCHAR(256))
as
with cte as (select LOCALITY,CAST(Price as FLOAT) as Price,
             MAX(Beds) as Max_Beds,
             MAX(BATH) as Max_Baths
from NY_Housing where Price between @Min_Price and @Max_Price and
LOCALITY = @Locality
Group By LOCALITY, CAST(Price as FLOAT))
select * from cte;

Exec Max_Beds_Baths 10000, 1000000, 'New York';
```

Database_Insights_and_SQL_Solutions

- ❖ Calculate Average Heart Rate, BMI, and Blood Pressure by Condition and Gender and Create a temporary table to store aggregated data for average heart rate, BMI, and systolic/diastolic blood pressure grouped by condition and gender. Use this table to identify whether specific conditions or genders are associated with higher health risks?

```
select Gender, Condition, AVG(Heart_Rate) as Avg_HeartRate,
       AVG(BMI) as Avg_BMI, AVG(CAST(SUBSTRING(Blood_Pressure, 1,
CHARINDEX('/',Blood_Pressure)-1) as INT)) as Avg_systolic,
       AVG(CAST(SUBSTRING(Blood_Pressure,
CHARINDEX('/',Blood_Pressure)+1, LEN(Blood_Pressure)) as INT)) as
Avg_diastolic
into #HighRiskPatients
from HealthcareData
Group By Gender, Condition;


select * from #HighRiskpatients where Avg_systolic > 90 and
Avg_diastolic > 70;
```

- ❖ Create a temporary table to store patients with critical vitals, defined as:
  - Heart rate > 100 bpm.
  - Systolic blood pressure > 140 or diastolic > 90.
  - BMI > 30 (obese).
  - Include columns for `Patient_ID`, `Condition`, `Heart_Rate`, `Blood_Pressure`, and `BMI`.
  - This temp table will help generate alerts or notifications for critical patients.

```
with cte as
(select Patient_ID, Condition, Heart_Rate, BMI, Blood_Pressure,
       CAST(SUBSTRING(Blood_Pressure, 1, CHARINDEX('/',Blood_Pressure)-
1) as INT) as Systolic,
       CAST(SUBSTRING(Blood_Pressure, CHARINDEX('/', Blood_Pressure)+1,
LEN(Blood_Pressure)) as INT) as Diastolic
from HealthcareData)
select Patient_ID, Condition, Heart_Rate, BMI, Blood_Pressure
into #HighIrsk from cte
where Heart_Rate > 100 or BMI > 30 or Systolic > 140 or Diastolic > 90;

select * from #HighIrsk;
```

- ❖ Create a temporary table to analyze medication usage trends across different conditions.

  - Include the count of distinct medications used for each condition, grouped by blood type.
  - Add an additional column to calculate the percentage of patients on each medication for a given condition.

```
with cte as (select Condition, Blood_Type,
       COUNT(DISTINCT Medication) as Num_Medications from HealthcareData
Group By Medication, Condition, Blood_Type),
cte2 as (select Condition, COUNT(DISTINCT Medication) as Med_Count,
       FORMAT(COUNT(DISTINCT Medication) * 100.0/SUM(COUNT(DISTINCT
Medication)) OVER(),'N2') as Perc_Dist from HealthcareData
Group By Condition)
select a.Condition, a.Blood_Type, a.Num_Medications,b.Perc_Dist
into #MedicationTrends from cte a join cte2 b on a.Condition =
b.Condition;

select * from #MedicationTrends;
```

Database_Insights_and_SQL_Solutions

- Create a temporary table to analyze the distribution of insurance providers across different conditions.
    - Include the count of patients per insurance provider, average BMI, and heart rate for each provider.
    - This table will assist in understanding insurance-related trends in patient demographics.

```
with cte as (select Insurance_Provider, Condition,
     COUNT(*) as Num_Patients,
     AVG(BMI) as Avg_BMI,
     AVG(Heart_Rate) as AVG_HeartRate
from HealthcareData
Group By Insurance_Provider, Condition)
select Insurance_Provider, Condition, Num_Patients, Avg_BMI, AVG_HeartRate,
     FORMAT(Num_Patients*100.0/SUM(Num_Patients)OVER(Partition By
Condition),'N2') as Percentage_Dist
into #Insurance_Provider_Trends
from cte;


select * from #Insurance_Provider_Trends
Order By Num_Patients DESC;
```

- Create a temporary table to store upcoming appointments for high-risk patients.
    - High-risk patients are defined as those with any of the following:
    - Age > 70.
    - BMI > 35.
    - Heart rate > 90.
    - The table should include `Patient_ID`, `Age`, `Condition`, `Heart_Rate`, `BMI`, and `Appointment_Date` to prioritize scheduling.

```
with cte as (select Patient_ID, Age, Condition, Heart_Rate, BMI,
Appointment_Date from HealthcareData
where Age > 70 or BMI > 35 or Heart_Rate > 90)
select * into #Priority_Appintments from cte;

select * from #Priority_Appintments;
```

- create a procedure to Identify and retrieve a list of high-risk patients based on age, BMI, and heart rate thresholds.

  Input Parameters

  AgeThreshold
  BMIThreshold
  HeartRateThreshold

  It should print Patient details, including `Patient_ID`, `Condition`, `Vitals`, and `Appointment_Date`?

```
create PROC Critical_Trends (@Age INT, @BMI FLOAT, @Heartrate FLOAT)
as
Begin
with cte as (select Patient_ID, Condition, Notes as Vitals,
Appointment_Date  from HealthcareData
where (Age >  @Age) AND (BMI > @BMI) AND (Heart_Rate > @Heartrate))
select *  from cte
End;

Exec Critical_Trends 70, 35, 90;
```

Database_Insights_and_SQL_Solutions

- Create a procedure to print Low Profit Margin
  - Input Parameters: ProfitMarginThreshold (Profit value)
  - Output: Transactions with a profit margin below the specified threshold, categorized by product

```
create PROC Profit_Margin (@Profit_Margin FLOAT)
as
select Product_ID, Prod_Name, SUM(Profit) as Profit_Margin
from Orders
Group By Product_ID, Prod_Name
Having SUM(Profit) <= @Profit_Margin;

Exec Profit_Margin 0.30
```

- Create a stored procedure to Analyze the impact of specific medications on patients' vitals for a given condition. It requires following Input Parameters
  - Condition
  - Medication
  - And, it should print the following result set Aggregated metrics (average heart rate, BMI, and blood pressure) grouped by Gender, Age_Range, and Blood_Type?

```
create PROC Patient_Vitals (@Condition VARCHAR(120), @Medication VARCHAR(256))
as
Begin
with cte as (
select Gender, Blood_Type,
        CASE when Age <= 18 then '<=18'
        WHEN Age between 19 and 30 then '19-30'
        when Age between 30 and 50 then '30-50'
        when Age between 50 AND 70 then '50-70'
        else '>70' end as Age_Group,
        AVG(BMI) as Avg_BMI,
        AVG(Heart_Rate) as Avg_HeartRate,
        AVG(CAST(SUBSTRING(Blood_Pressure, 1, CHARINDEX('/',Blood_Pressure)-1)
as INT)) as Avg_Systolic,
        AVG(CAST(SUBSTRING(Blood_Pressure,
CHARINDEX('/',Blood_Pressure)+1,LEN(Blood_Pressure)) as INT)) as Diastolic
from HealthcareData where Condition = @Condition AND Medication = @Medication
Group By Gender, Blood_Type,
        CASE when Age <= 18 then '<=18'
        WHEN Age between 19 and 30 then '19-30'
        when Age between 30 and 50 then '30-50'
        when Age between 50 AND 70 then '50-70'
        else '>70' end)
select * from cte
end;

Exec Patient_Vitals 'Obesity', 'Too';
```

- create a temp table to store OrderID, CustomerName, ShipMode, ShippingDelay, and find shipments delayed beyond the expected delivery date where order and ship date difference > 3 days?

```
with cte as (select Order_ID, Customer_Name, Ship_Mode,
        Case when DATEDIFF(day, OrderDate, ShipDate) > 3 then 'Delayed'
        else 'On Time' end as Shipping_Delayed
from Orders)
select DISTINCT * into #Shipping_Status from cte;

select * from #Shipping_Status;
```

Database_Insights_and_SQL_Solutions

- Create a procedure to find Frequent Buyers,
  - Input Parameters: StartDate, EndDate`
  - Output: List of customers with more than X orders (Number of orders, take the count of Order ID's) within the date range?

```
create PROC Cust_Order_Count (@StartDate DATETIME,  @EndDate DATETIME,
@Order_Threshold INT)
as
select Customer_ID, COUNT(Order_ID)as Num_Orders from Orders
where OrderDate  between @StartDate AND @EndDate
Group By Customer_ID
Having COUNT(Order_ID) > @Order_Threshold
Order By Num_Orders DESC;


Exec Cust_Order_Count '2024-01-01', '2024-02-28', 20
```

- Create a procedure to print Yearly Sales Growth by Category
  - Input Parameters: Category
  - Output: Year-over-year sales growth for the given category?

```
Create Proc CategorySaleYoy @Category VARCHAR(256)
as
with cte as (select Category,YEAR(OrderDate) as Year,
    SUM(Sales) as Total_Sale,
    SUM(Sales)*100.0/SUM(SUM(Sales)) OVER(Partition By Category) as
Perc_Dist
from Orders
Group By Category,YEAR(OrderDate))
select Category,Year, Total_Sale,
    ISNULL(Total_Sale-LAG(Total_Sale)OVER(Partition By Category order By
Year),0) as ChangeInSales,
    Perc_Dist,
    ISNULL(Perc_Dist-LAG(Perc_Dist) OVER (Partition By Category Order By
Year),0) as GrowthPercent
from cte where Category = @Category
Order By YEAR DESC;

Exec CategorySaleYoY 'Furniture';
```

- Create a procedure to print Underperforming Products
  - Input Parameters: SalesThreshold (Total sales), ProfitThreshold (Total Profit)
  - Output: Products with sales and profits below the respective thresholds?

```
Create PROC Product_Sales (@SalesThreshold FLOAT, @ProfitThreshold FLOAT)
as
with cte as (select Product_ID, SUM(Sales) as Total_Sale, SUM(Profit) as
Total_Profit
    from Orders
    Group By Product_ID
    Having SUM(Sales) < @SalesThreshold AND SUM(Profit) < @ProfitThreshold)
select * from cte;

Exec Product_Sales 100000, 0.30;
```

Database_Insights_and_SQL_Solutions

❖ Create a procedure to print Region-Specific Profitability
   - Input Parameters: Region
   - Output: Trends of profitability by state within the region?

```
Create PROC Region_Profits (@Region VARCHAR(40))
As
select Region, State, SUM(Profit) as Total_Profit
from Orders where Region = @Region
Group By Region, State
Order By Total_Profit DESC;

Exec Region_Profits 'West'
```

❖ Create a procedure to print Shipping Performance
   - Input Parameters: ShipMode
   - Output: Average shipping time and delay (Find the difference between order data and ship date, where the difference is more 3 days, consider it as delay) counts for the selected shipping mode?

```
create PROC ShippingDelay @Ship_Mode VARCHAR(256)
as
select Ship_Mode, AVG(DATEDIFF(Day, OrderDate, ShipDate)) as Avg_Delay,
       SUM(case when DATEDIFF(Day, OrderDate, ShipDate) > 3 then 1 else
0 end) as Num_Delays
from Orders where Ship_Mode = @Ship_Mode
Group By Ship_Mode
Order By Num_Delays DESC;

Exec ShippingDelay 'Standard Class';
```

❖ Create a procedure to print Top Selling Products by Segment
   - Input Parameters: Segment, TopN
   - Output: List of top-selling products in the specified segment?

```
Create proc SegmentSales @Segment VARCHAR(256), @Ranking INT
as
with cte as (
select Segment, Product_ID, SUM(Sales) as Total_Sale,
       DENSE_RANK() OVER (Partition By Segment Order By SUM(Sales) DESC)
as Ranking from Orders
Group By Segment, Product_ID)
select * from cte where Segment = @Segment AND Ranking <= @Ranking;

Exec SegmentSales 'Consumer', 5;
```

❖ Create a temp table to store Regional Sales it should contain Region, State, Categoru TotalSales, TotalProfit, Category Ranks based on total profit?

```
with cte as (select Region, State, Category, SUM(Sales) as Total_Sale,
       SUM(Profit) as Total_Profit,
       DENSE_RANK() OVER (Partition By Region,State Order By SUM(Profit)
DESC) as Ranks
from Orders
Group By Region, State, Category)
Select * into #Regional_Sales from cte;

Select * from #Regional_Sales
Order By Region,State, Ranks;
```

Database_Insights_and_SQL_Solutions

- Create a temp table to store CustomerID, TotalOrders, TotalSales, AverageProfit?

```
select Customer_ID,
              COUNT(Order_ID) as Num_Order,
              SUM(Sales) as Total_Sale,
              AVG(Profit) as Avg_Profit into #Customer_SalesTrend
from Orders
Group By Customer_ID;

Select * from #Customer_SalesTrend
Order By Total_Sale DESC;
```

- create a temp table to store ProductID, AverageDiscount, AverageProfit and compare how discounts impact profit margins per product?

```
with cte as (select Product_ID,
              AVG(Discount) as Avg_Discount,
              AVG(Profit) as Avg_Profit
from Orders
Group By Product_ID)
SELECT
    Product_ID, Avg_Discount, Avg_Profit,
    Case When Avg_Discount > 0.25 AND Avg_Profit < 0 then 'High
Discount, Low Profit'
        When Avg_Discount > 0.25 Then 'High Discount'
        When Avg_Profit < 0 Then 'Low Profit'
        Else 'Balanced' end as Impact_Analysis into
#DiscountVSProfit_Analysis
FROM cte;

Select * from #DiscountVSProfit_Analysis
Order By Avg_Discount DESC;
```

- create a temp table to store Category, Order Year, TotalSales, TotalProfit?

```
with cte as (select Category,
              Year(OrderDate) as Year,
              SUM(Sales) as Total_Sale,
              SUM(Profit) as Total_Profit from Orders
       Group By Category,Year(OrderDate))
select * into #Yearly_Trend_Category from cte;

select * from #Yearly_Trend_Category
Order By Year DESC, Total_Sale DESC;
```