Microsoft

SQL QUESTIONS

1.Calculate a Cumulative Percentage of Total Sales by Region

Compute the cumulative percentage of total sales (amount) for each region, ordered by order_date. Display the cumulative percentage alongside other details.

Region	Order Date	Order_ID	Amount
North	2024-01-15	1	500
North	2024-03-05	5	600
South	2024-01-20	2	700
East	2024-02-18	4	200

2. Identify the First and Last Orders for Each Customer

Find the first and last orders placed by each customer based on order_date. Include their order IDs, amounts, and the difference in days between the first and last orders.

Order_ID	Customer_ID	Order_Date	Amount
1	101	2024-01-15	500
2	102	2024-01-20	700
3	101	2024-02-10	300
4	103	2024-02-18	200
5	101	2024-03-05	600

3. Create a Rolling 3-Month Sales Average

Calculate the 3-month rolling average of sales for each region. If there aren't enough months for a 3-month window, the rolling average should still be computed based on the available data.

Region	Month	Year	Sales
North	January	2024	15000
North	February	2024	18000
North	March	2024	20000
South	January	2024	14000
South	February	2024	16000

4. Rank Customers by Spending but Exclude Orders Above $500

Rank customers by their total spending, but only consider orders with an amount less than or equal to $500.

Order_ID	Customer_ID	Order_Date	Order_Amount
1	C001	2024-01-10	300
2	C002	2024-01-12	450
3	C003	2024-04-15	600


```
-- DDL for the table
CREATE OR REPLACE TABLE sales_region_data (
   region VARCHAR(50),
   order_date DATE,
   order_id INT,
   amount DECIMAL(10, 2)
);

-- Insert sample data
INSERT INTO sales_region_data (region, order_date, order_id, amount)
VALUES
('North', '2024-01-15', 1, 500.00),
('North', '2024-03-05', 5, 600.00),
('South', '2024-01-20', 2, 700.00),
('East', '2024-02-18', 4, 200.00);

CREATE OR REPLACE TABLE customer_orders (
   order_id INT,
   customer_id INT,
   order_date DATE,
   amount DECIMAL(10, 2)
);

INSERT INTO customer_orders (order_id, customer_id, order_date, amount)
VALUES
(1, 101, '2024-01-15', 500.00),
```

```
(2, 102, '2024-01-20', 700.00),
(3, 101, '2024-02-10', 300.00),
(4, 103, '2024-02-18', 200.00),
(5, 101, '2024-03-05', 600.00);

CREATE OR REPLACE TABLE rolling_sales (
    region VARCHAR(50),
    month VARCHAR(20),
    year INT,
    sales DECIMAL(10, 2)
);

INSERT INTO rolling_sales (region, month, year, sales)
VALUES
('North', 'January', 2024, 15000.00),
('North', 'February', 2024, 18000.00),
('North', 'March', 2024, 20000.00),
('South', 'January', 2024, 14000.00),
('South', 'February', 2024, 16000.00);


CREATE OR REPLACE TABLE customer_spending (
    order_id INT,
    customer_id VARCHAR(50),
    order_date DATE,
    order_amount DECIMAL(10, 2)
```

```
);

INSERT INTO customer_spending (order_id, customer_id, order_date, order_amount)
VALUES
(1, 'C001', '2024-01-10', 300.00),
(2, 'C002', '2024-01-12', 450.00),
(3, 'C003', '2024-04-15', 600.00);

-- 1.Calculate a Cumulative Percentage of Total Sales by Region
WITH sales_with_total AS (
  SELECT
    region, order_date, order_id,
    amount,
    SUM(amount) OVER (PARTITION BY region) AS region_total,
    SUM(amount) OVER () AS grand_total
  FROM sales_region_data
),
cumulative_sales AS (
  SELECT
    region, order_date, order_id, amount,
    region_total,
    grand_total,
    SUM(amount) OVER (PARTITION BY region ORDER BY order_date) AS
cumulative_amount
  FROM sales_with_total
)
```

```
SELECT
    region, order_date, order_id,
    amount,
    region_total,
    cumulative_amount,
    ROUND((cumulative_amount / region_total) * 100, 2) AS cumulative_percentage
FROM cumulative_sales
ORDER BY region, order_date;
```

	REGION	ORDER_DATE	ORDER_ID	AMOUNT	CUMULATIVE_SALES	TOTAL_SALES	CUMULATIVE_PERCENTAGE
1	East	2024-02-18	4	200.00	200.00	200.00	100.00
2	North	2024-01-15	1	500.00	500.00	1100.00	45.45
3	North	2024-03-05	5	600.00	1100.00	1100.00	100.00
4	South	2024-01-20	2	700.00	700.00	700.00	100.00

--2. Identify the First and Last Orders for Each Customer

```
WITH CustomerOrders AS (
    SELECT
        Customer_ID, Order_ID, Order_Date,
        Amount,
        ROW_NUMBER() OVER (PARTITION BY Customer_ID ORDER BY Order_Date ASC) AS Row_First,
        ROW_NUMBER() OVER (PARTITION BY Customer_ID ORDER BY Order_Date DESC) AS Row_Last
    FROM customer_orders
),
```

```
FirstOrders AS (
    SELECT
        Customer_ID,
        Order_ID AS First_Order_ID,
        Order_Date AS First_Order_Date,
        Amount AS First_Order_Amount
    FROM CustomerOrders
    WHERE Row_First = 1
),
LastOrders AS (
    SELECT
        Customer_ID,
        Order_ID AS Last_Order_ID,
        Order_Date AS Last_Order_Date,
        Amount AS Last_Order_Amount
    FROM CustomerOrders
    WHERE Row_Last = 1
)
SELECT
    f.Customer_ID,
    f.First_Order_ID,
    f.First_Order_Date,
    f.First_Order_Amount,
    l.Last_Order_ID,
    l.Last_Order_Date,
    l.Last_Order_Amount,
```

DATEDIFF(day,f.First_Order_Date, l.Last_Order_Date) AS Days_Between_Orders

FROM **FirstOrders** f

JOIN **LastOrders** l

ON f.Customer_ID = l.Customer_ID

ORDER BY f.Customer_ID;

	CUSTOMER_ID	FIRST_ORDER_ID	FIRST_ORDER_DATE	FIRST_ORDER_AMOUNT	LAST_ORDER_ID	LAST_ORDER_DATE	LAST_ORDER_AMOUNT	DAYS_BETWEEN
1	101	1	2024-01-15	500.00	5	2024-03-05	600.00	50
2	102	2	2024-01-20	700.00	2	2024-01-20	700.00	0
3	103	4	2024-02-18	200.00	4	2024-02-18	200.00	0

--3. Create a Rolling 3-Month Sales Average

WITH **SalesData** AS (

SELECT

Region,

CONCAT(Month, ' ', Year) AS Month_Year,

Year,

CASE

WHEN Month = 'January' THEN 1

WHEN Month = 'February' THEN 2

WHEN Month = 'March' THEN 3

WHEN Month = 'April' THEN 4

WHEN Month = 'May' THEN 5

WHEN Month = 'June' THEN 6

WHEN Month = 'July' THEN 7

WHEN Month = 'August' THEN 8

WHEN Month = 'September' THEN 9

WHEN Month = 'October' THEN 10

```
        WHEN Month = 'November' THEN 11
        WHEN Month = 'December' THEN 12
    END AS Month_Num,
    Sales
  FROM rolling_sales
),
RankedSales AS (
  SELECT
    Region,
    Month_Year,
    Year,
    Month_Num,
    Sales,
    ROW_NUMBER() OVER (PARTITION BY Region ORDER BY Year, Month_Num) AS Month_Rank
  FROM SalesData
)
SELECT
  Region,
  Month_Year,
  Sales,
  ROUND(AVG(Sales) OVER (PARTITION BY Region ORDER BY Month_Rank ROWS BETWEEN 2 PRECEDING AND CURRENT ROW), 2) AS Rolling_3_Month_Avg
FROM RankedSales
ORDER BY Region, Month_Rank;
```

	REGION	MONTH_YEAR	SALES	ROLLING_3_MONTH_AVG
1	North	January 2024	15000.00	15000.00
2	North	February 2024	18000.00	16500.00
3	North	March 2024	20000.00	17666.67
4	South	January 2024	14000.00	14000.00
5	South	February 2024	16000.00	15000.00

-- 4. Rank Customers by Spending but Exclude Orders Above $500

```
WITH filtered_orders AS (
    SELECT
        customer_id,
        SUM(order_amount) AS total_spending
    FROM customer_spending
    WHERE order_amount <= 500
    GROUP BY customer_id
),
ranked_customers AS (
    SELECT
        customer_id,
        total_spending,
        RANK() OVER (ORDER BY total_spending DESC) AS rank
    FROM filtered_orders
)
SELECT
    customer_id,
    total_spending,
    rank
FROM ranked_customers;
```

MICROSOFT SQL & PYTHON INTEVIEW QUESTIONS FOR DATA ANALYSTS

	CUSTOMER_ID	TOTAL_SPENDING	RANK
1	C002	450.00	1
2	C001	300.00	2

Microsoft

PYTHON QUESTIONS

1.Filtering Data Frame Based on Conditions:

Create a Data Frame with data like this and Filter the Data Frame to show only employees in the `IT` department who earn more than $60,000.

```
data = {
    'employee': ['John', 'Jane', 'Jim', 'Jill', 'Jack'],
    'department': ['HR', 'IT', 'Finance', 'IT', 'HR'],
    'salary': [50000, 60000, 70000, 65000, 55000]
}
```

2.Renaming Columns:

Create a Data Frame with the following data and rename the columns to ID, Name, and Price.

```
data = {
    'Product_ID': [101, 102, 103],
    'Product_Name': ['Laptop', 'Phone', 'Tablet'],
    'Product_Price': [1000, 800, 600]
}
```

3.Handling Missing Values:

Create a Data Frame with the following data and fill the missing values in the price column with the average price of the products and drop the rows where the `price` column is still missing after filling

```
data = {
    'product': ['Laptop', 'Tablet', 'Phone', 'Monitor', 'Mouse'],
    'price': [1000, 500, None, 150, None],
    'stock': [50, 150, 100, 200, 300]
}
```

4.Creatig Extra Column:

Read an Excel file into a Pandas Data Frame. Add a new column called State based on the values in the city column.

Input:

City	Population
New York	20000
Los Angeles	30000
Houston	40000
Chicago	60000

Expected Output:

City	Population	State
New York	20000	New York
Los Angeles	30000	California
Houston	40000	Illinois
Chicago	60000	Texas

MICROSOFT SQL & PYTHON INTEVIEW QUESTIONS FOR DATA ANALYSTS

-- 1.Filtering Data Frame Based on Conditions:

```
import pandas as pd
data = {
   'employee': ['John', 'Jane', 'Jim', 'Jill', 'Jack'],
   'department': ['HR', 'IT', 'Finance', 'IT', 'HR'],
   'salary': [50000, 60000, 70000, 65000, 55000]
}
df = pd.DataFrame(data)
filtered_df = df[(df['department'] == 'IT') & (df['salary'] > 60000)]
print(filtered_df)
```

```
   employee department  salary
3      Jill         IT   65000
```

--2.Renaming Columns

```
import pandas as pd
data = {
   'Product_ID': [101, 102, 103],
   'Product_Name': ['Laptop', 'Phone', 'Tablet'],
   'Product_Price': [1000, 800, 600]
}
df = pd.DataFrame(data)
df.rename(columns={
   'Product_ID': 'ID',
   'Product_Name': 'Name',
   'Product_Price': 'Price'
```

}, inplace=True)

print(df)

```
     ID    Name  Price
0   101  Laptop   1000
1   102   Phone    800
2   103  Tablet    600
```

--3.Handling Missing Values

import pandas as pd

data = {

'product': ['Laptop', 'Tablet', 'Phone', 'Monitor', 'Mouse'],

'price': [1000, 500, None, 150, None],

'stock': [50, 150, 100, 200, 300]

}

df = pd.DataFrame(data)

average_price = df['price'].mean(skipna=True)

df['price'].fillna(average_price, inplace=True)

df.dropna(subset=['price'], inplace=True)

print(df)

```
    product   price  stock
0    Laptop  1000.0     50
1    Tablet   500.0    150
2     Phone   550.0    100
3   Monitor   150.0    200
4     Mouse   550.0    300
```

--4.Creatig Extra Column

import pandas as pd

file_path = **'C:/Users/Anand Jha/Downloads/city_pop.xlsx**' # Replace with the actual file path by creating an excel file with the datasets(already given)

df = pd.read_excel(file_path)

city_to_state = {

'New York': 'New York',

'Los Angeles': 'California',

'Houston': 'Texas',

'Chicago': 'Illinois'

}

df['State'] = df['City'].map(city_to_state)

print(df)

output_path = 'C:/Users/Anand Jha/Downloads/city_pop_updated.xlsx' # Replace the path in your local system where you want to save the file

df.to_excel(output_path, index=False)

	City	Population	State
0	New York	20000	New York
1	Los Angeles	30000	California
2	Houston	40000	Texas
3	Chicago	6000	Illinois