**Solutions for the required SQL queries in the Weather Forecast Project**

**The database weather\_report has a table called weather\_data which contains the data imported from the “cleaned\_data\_set.csv” file.**

**The following sql query make a databse weather\_report :**

CREATE DATABASE weather\_report;

USE weather\_report;

**The following sql query make the table weather\_data for the above created database weather\_report.**

CREATE TABLE weather\_report.weather\_data (

Date\_ DATE,

Average\_temperature double,

Average\_humidity double,

Average\_dewpoint double,

Average\_barometer double,

Average\_windspeed double,

Average\_gust\_speed double,

Average\_direction double,

Rainfall\_for\_month double,

Rainfall\_for\_year double,

Maximum\_rain\_per\_minute double,

Maximum\_temperature double,

Minimum\_temperature double,

Maximum\_humidity double,

Minimum\_humidity double,

Maximum\_pressure double,

Minimum\_pressure double,

Maximum\_wind\_speed double,

Maximum\_gust\_speed double,

Maximum\_heat\_index double,

Month\_ double,

Diff\_pressure double,

PRIMARY KEY (Date\_) );

SELECT \* FROM weather\_data;

describe weather\_data;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.Give the count of the minimum number of days for the time when temperature reduced.

SELECT COUNT(\*)

FROM (

SELECT w1.Date\_, w1.Minimum\_temperature, w2.Minimum\_temperature AS Prev\_temp

FROM weather\_data w1

JOIN weather\_data w2

ON w1.Date\_ = w2.Date\_ + INTERVAL 1 DAY

WHERE w1.Minimum\_temperature < w2.Minimum\_temperature

) subq;

**Explanation:The subquery selects the date, minimum temperature, and previous day's minimum temperature from a table called "weather\_data". The subquery joins the weather\_data table with itself on the date plus one day and filters the results to only include records where the minimum temperature for a given day is less than the minimum temperature for the previous day. The outer query then uses the COUNT function to count the number of records returned by the subquery and returns the result as a single value.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2.Find the temperature as Cold / hot by using the case and avg of values of the given data set.

SELECT

AVG(Average\_temperature) AS Average\_Temperature,

CASE

WHEN AVG(Average\_temperature) < 10 THEN 'Cold'

WHEN AVG(Average\_temperature) >= 10 THEN 'Hot'

END AS Temperature\_Classification

FROM weather\_data;

**Explanation: The query calculates the average temperature using the AVG function and assigns the result to a column called "Average\_Temperature". The query also uses a CASE statement to classify the average temperature as either "Cold" or "Hot" based on the calculated value. The CASE statement checks if the average temperature is less than 10 and if so, returns "Cold". If the average temperature is not less than 10, the CASE statement returns "Hot". The final result of the query is a single row with two columns, the average temperature and the temperature classification.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3. Can you check for all 4 consecutive days when the temperature was below 30 Fahrenheit.

SELECT w1.Date\_

FROM weather\_data w1

JOIN weather\_data w2 ON w1.Date\_ = w2.Date\_ + INTERVAL 1 DAY

JOIN weather\_data w3 ON w1.Date\_ = w3.Date\_ + INTERVAL 2 DAY

JOIN weather\_data w4 ON w1.Date\_ = w4.Date\_ + INTERVAL 3 DAY

WHERE w1.Average\_temperature < 30 AND w2.Average\_temperature < 30

AND w3.Average\_temperature < 30 AND w4.Average\_temperature < 30;

**Explanation: This SQL query would return the date of consecutive days with average temperature less than 30 degrees from a table called "weather\_data". The query joins the "weather\_data" table with itself four times, each time using the "Date\_" column and an interval of 1 day, 2 days, 3 days, and 4 days, respectively. This is done to compare the average temperature of each day with the average temperature of the previous four days. The query then uses a WHERE clause to filter the results to only include records where the average temperature of the current day and the previous three days is less than 30. The final result of the query is a list of dates that meet the specified criteria.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4. Can you find the maximum number of days for which temperature dropped.

WITH cte AS (

SELECT

Date\_,

Minimum\_temperature,

ROW\_NUMBER() OVER (ORDER BY Date\_) AS rn

FROM weather\_data

)

SELECT

MAX(c1.rn) - MIN(c2.rn) + 1

FROM cte c1

JOIN cte c2

ON c1.rn + 1 = c2.rn

WHERE c1.Minimum\_temperature > c2.Minimum\_temperature;

**Explanation:** **This SQL query calculates the length of the longest streak of consecutive days with a lower minimum temperature compared to the previous day. The query starts by using a Common Table Expression (CTE) to select the date, minimum temperature, and a row number from the "weather\_data" table. The row number is calculated using the ROW\_NUMBER() function and ordered by the date.**

**The query then joins the CTE with itself on the row number and filters the results to only include records where the minimum temperature of the current day is greater than the minimum temperature of the previous day. Finally, the query calculates the length of the streak by subtracting the minimum row number from the maximum row number and adding 1, and returns the result as a single value.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5. Can you find the average of average humidity from the dataset .

SELECT AVG(Average\_humidity)

FROM weather\_data

GROUP BY Date\_

ORDER BY Date\_;

**Explanation:This SQL query would return the average humidity for each day in a table called "weather\_data". The query uses the AVG function to calculate the average humidity for each day and groups the results by the date using the GROUP BY clause. The query then orders the results by the date using the ORDER BY clause and returns the average humidity for each day as a list of values.**

窗体顶端

窗体底端

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6. Use the GROUP BY clause on the Date column and make a query to fetch details for average windspeed .

SELECT

Date\_,

AVG(Average\_windspeed) AS avg\_windspeed

FROM weather\_data

GROUP BY Date\_

ORDER BY Date\_;

**Explanation: This SQL query would return the average wind speed for each day in a table called "weather\_data". The query uses the AVG function to calculate the average wind speed for each day and groups the results by the date using the GROUP BY clause. The query then orders the results by the date using the ORDER BY clause and returns the average wind speed for each day along with the date as a list of records. The average wind speed for each day is assigned to a column named "avg\_windspeed".**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7. Please add the data in the dataset for 2034 and 2035 as well as forecast predictions for these years .

WITH forecast AS (

SELECT

Date\_,

Average\_temperature,

AVG(Average\_temperature) OVER (ORDER BY Date\_ ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS avg\_temp\_2

FROM weather\_data

),

forecast\_2034\_2035 AS (

SELECT

DATE\_ADD(Date\_, INTERVAL 1 YEAR) AS Date\_,

avg\_temp\_2 AS Average\_temperature

FROM forecast

) SELECT \*FROM forecast\_2034\_2035;

**Explanation: This SQL query generates a forecast of the average temperature for the year 2034 to 2035. The query starts by creating a Common Table Expression (CTE) named "forecast" that calculates the average temperature for each day and the average temperature for the past two days. The query then creates another CTE named "forecast\_2034\_2035" that adds one year to the date and selects the average temperature of the past two days as the average temperature for the new date. Finally, the query selects all records from the "forecast\_2034\_2035" CTE and returns the result as a list of records with two columns: "Date\_" and "Average\_temperature".**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

8.If the maximum gust speed increases from 55mph, fetch the details for the next 4 days.

SELECT \*

FROM weather\_data

WHERE Date\_ BETWEEN (SELECT MIN(Date\_) FROM weather\_data WHERE Maximum\_gust\_speed > 55) AND DATE\_ADD((SELECT MIN(Date\_) FROM weather\_data WHERE Maximum\_gust\_speed > 55), INTERVAL 4 DAY);

**Explanation**:This SQL query returns all the records from the "weather\_data" table for a five-day period starting from the date on which the maximum gust speed exceeded 55. The query uses two subqueries to first find the minimum date from the "weather\_data" table where the maximum gust speed is greater than 55. The first subquery returns the minimum date, and the second subquery returns the minimum date plus four days. The final query selects all records from the "weather\_data" table where the date is between the minimum date and the minimum date plus four days.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

9. Find the number of days when the temperature went below 0 degrees Celsius .

SELECT COUNT(\*)

FROM weather\_data

WHERE Average\_temperature < 0;

**Explanation:**This SQL query returns the count of records in the "weather\_data" table where the average temperature is less than 0. The query uses a WHERE clause to filter the records based on the average temperature and returns the number of records that meet the condition using the COUNT function. The result of the query is a single value representing the count of records where the average temperature is less than 0.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

10. Create another table with a “Foreign key” relation with the existing given data set.

CREATE TABLE weather\_data\_2 (

id INT AUTO\_INCREMENT PRIMARY KEY,

weather\_data\_id DATE,

some\_other\_column VARCHAR(255),

FOREIGN KEY (weather\_data\_id) REFERENCES weather\_data(Date\_));

**Explanation:**This SQL create a new table named "weather\_data\_2". The table has four columns:

"id" - an auto-incrementing primary key.

"weather\_data\_id" - a date column that references the "Date\_" column in the "weather\_data" table.

"some\_other\_column" - a column to store a string value with a maximum length of 255 characters.

The statement creates a foreign key constraint between the "weather\_data\_id" column in the "weather\_data\_2" table and the "Date\_" column in the "weather\_data" table, enforcing referential integrity between the two tables.