REPORT ON
WEATHER DATA SET

Import Necessary Libraries:

Start by importing the necessary Python libraries such as Pandas, Numpy and other libraries.

```
import pandas as pd
import numpy as np
and etc….,
```

Load Given Data Set

Load the weather dataset into a DataFrame and inspect it to understand its structure, available columns, and data types.

```
df = pd.read_csv(“File Location of dataset”)
print(df.head()
```

NOTE: This Analysis is done in the JupiterNoteBook ,there may be change in code when you use different platfroms.

Iterating on the Data Frame as per the Problem Statement

Q. 1) Find all the unique 'Wind Speed' values in the data.

Python Code:

```
# Q. 1) Find all the unique 'Wind Speed' values in the data.
unique_WindSpeed_values = df['Wind Speed_km/h'].unique()
print(unique_WindSpeed_values)
```

```
[ 9 24 26 15  4  0 19 17 11 22 35 13 20  6  7 30 32 41 39 28 44 33 37 52
 46  2 50 48 57 63 43 83 70 54]
```

Code Explanation:

This code will extract the 'Wind Speed' column from weather DataFrame and then use the unique() method to find all the unique values in that column.

Q. 2) Find the number of times when the 'Weather is exactly Clear'.

Python Code:

```
# Q. 2) Find the number of times when the 'Weather is exactly Clear'.
clear_weather_count = df['Weather'].value_counts().get('Clear', 0)
print(clear_weather_count)
```

```
1326
```

Code Explanation:

In this code, we use the value_counts() method on the 'Weather' column to get a count of unique values in that column. Then, we use the get() method to retrieve the count for the 'Clear' value. If 'Clear' is not present in the 'Weather' column, it returns 0. The result is printed to the console.

Or

Python Code Using For Loop:

```
clear_weather_count = 0
for i in df.Weather:
    if i =='Clear':
        clear_weather_count+=1
print(clear_weather_count)
```

```
1326
```

Code Explanation:

In this code, we initialize a counter variable clear_weather_count to 0. Then, we iterate through each row in the DataFrame using a for loop. For each row, we check if the value in the 'Weather' column is equal to 'Clear'. If it is, we increment the clear_weather_count by 1. Finally, we print the count after the loop is done.

Q. 3) Find the number of times when the 'Wind Speed was exactly 4 km/h'.

Python Code Using For Loop:

```
wind_speed_count  = 0
for i in df['Wind Speed_km/h']:
    if i == 4:
        wind_speed_count+=1
print(wind_speed_count)
```

```
474
```

Code Explanation:

In this code, we initialize a counter variable wind_speed_count to 0. Then, we iterate through each row in the DataFrame using a for loop. For each row, we check if the value in the 'Wind Speed' column is equal to '4 km/h'. If it is, we increment the wind_speed_count by 1. Finally, we print the count after the loop is done.

Or

Python Code:

```
wind_speed_count = df['Wind Speed_km/h'].value_counts().get(4, 0)
print(wind_speed_count)
```

```
474
```

Code Explanation:

In this code, we first use the value_counts() method to count the occurrences of each unique value in the 'Wind Speed' column. Then, we use the get() method to retrieve the count for '4 km/h' specifically. If '4 km/h' is not found in the series (i.e., it doesn't exist in the dataframe), we default to 0. Finally, we print the count

Q. 4) Find out all the Null Values in the data.

Python Code:

```
# Q. 4) Find out all the Null Values in the data.
Weather_isnull = df.isnull().sum()
print(Weather_isnull)
```

```
Date/Time           0
Temp_C              0
Dew Point Temp_C    0
Rel Hum_%           0
Wind Speed_km/h     0
Visibility_km       0
Press_kPa           0
Weather             0
dtype: int64
```

Code Explanation:

In this code, weather.isnull() creates a DataFrame of the same shape as your original 'weather' DataFrame, where each cell is True if it's a null value and False otherwise. Then, sum() is used to count the number of True values (which are nulls) along each column. The result is a Series where the column names are the columns in your DataFrame, and the values represent the count of null values in each column.

Q. 5) Rename the column name 'Weather' of the dataframe to 'Weather Condition'

Python Code :

```
# Q. 5) Rename the column name 'Weather' of the dataframe to 'Weather Condition'.
df.rename(columns = {'Weather':'Weather Condition'},inplace = True)
```

Code Explanation:

df.rename(columns={'Weather': 'Weather Condition'}, inplace=True): This renames the 'Weather' column to 'Weather Condition' in the DataFrame df.

The inplace=True argument allows you to modify the DataFrame in place without the need to assign it back to a variable.

Q. 6) What is the mean 'Visibility' ?

Python Code :

```
# Q. 6) What is the mean 'Visibility' ?
mean_Visibility = df.Visibility_km.mean()
mean_Visibility
```

```
27.664446721311162
```

Code Explanation:

mean_visibility will now contain the mean visibility value for weather data.

Q. 7) What is the Standard Deviation of 'Pressure' in this data?

Python Code :

```
: # Q. 7) What is the Standard Deviation of 'Pressure' in this data?
  Pressure_Std= df['Press_kPa'].std()
  print(Pressure_Std)
```

```
0.8440047459486459
```

Code Explanation:

This code will compute the standard deviation of the 'Pressure' column in your weather dataframe and print the result.

Q. 8) What is the Variance of 'Relative Humidity' in this data ?

Python Code :

```
# Q. 8) What is the Variance of 'Relative Humidity' in this data ?
Rel_Hum = df['Rel Hum_%'].var()
print(Rel_Hum)
```

```
286.2485501985015
```

Code Explanation:

This code will compute and print the variance of the 'Relative Humidity' column in your weather dataframe.

Q. 9) Find all instances when 'Snow' was recorded.

Python Code :

```
# Q. 9) Find all instances when 'Snow' was recorded.
Snow_Weather= df[df['Weather Condition']=='Snow']
print(Snow_Weather)
```

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather Condition
11	01-01-2012 11:00	-6.2	-9.6	37	35	4.8	101.56	Snow
70	03-01-2012 22:00	-4.0	-6.6	62	22	16.1	100.48	Snow
73	04-01-2012 01:00	2.3	-3.4	64	35	25.0	103.43	Snow
105	05-01-2012 09:00	-1.8	-4.2	73	15	6.4	101.28	Snow
112	05-01-2012 16:00	1.7	-0.3	75	6	9.7	101.47	Snow
...	...	...	...	...	...	...	...	...
8573	9/22/2012 13:00	-6.0	-10.2	82	19	16.1	101.71	Snow
8650	9/25/2012 18:00	-4.6	-6.6	52	4	12.9	100.48	Snow
8671	9/26/2012 15:00	-5.9	-10.5	60	13	16.1	101.01	Snow
8713	9/28/2012 1:00	-5.2	-7.8	72	33	4.0	101.33	Snow

Code Explanation:

This code will create a new dataframe called snow_Weather that contains only the rows where 'Snow' is recorded in the 'Weather Condition' column. You can then use this dataframe for further analysis or display the specific instances when 'Snow' occurred in your weather data.

Q. 10) Find all instances when 'Wind Speed is above 24' and 'Visibility is 25'.

Python Code :

```
# Q. 10) Find all instances when 'Wind Speed is above 24' and 'Visibility is 25'.
filtered_data = df[(df['Wind Speed_km/h']>24)&(df['Visibility_km']==25)]
print(filtered_data)
```

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather Condition
2	01-01-2012 02:00	15.7	13.4	21	26	25.0	99.84	Cloudy
73	04-01-2012 01:00	2.3	-3.4	64	35	25.0	103.43	Snow
126	06-01-2012 06:00	10.0	5.4	77	39	25.0	101.30	Cloudy
158	07-01-2012 14:00	1.9	-2.1	87	26	25.0	100.87	Rain,Snow Grains
184	08-01-2012 16:00	14.2	9.2	35	44	25.0	99.49	Mostly Cloudy
...	...	...	...	...	...	...	...	...
8707	9/27/2012 5:00	-1.0	-6.0	70	33	25.0	98.56	Mostly Cloudy
8714	9/28/2012 10:00	2.6	0.3	72	26	25.0	101.60	Rain
8738	9/29/2012 10:00	22.8	12.3	80	28	25.0	101.60	Mostly Cloudy
8745	9/29/2012 17:00	-10.3	-12.9	82	28	25.0	102.16	Cloudy
8776	9/30/2012 23:00	19.2	13.2	93	43	25.0	101.60	Mainly Clear

308 rows × 8 columns

Code Explanation:

This code will create a new dataframe called filtered_data that contains only the rows where 'Wind Speed' is above 24 and 'Visibility' is 25.

Q. 11) What is the Mean value of each column against each 'Weather Condition ?

Python Code :

```
# Q. 11) What is the Mean value of each column against each 'Weather Condition ?
Mean_eachcolumn = df.groupby('Weather Condition').mean()
Mean_eachcolumn
```

Weather Condition	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa
Clear	6.825716	0.089367	67.127451	10.557315	30.153243	101.084495
Cloudy	7.970544	2.375810	67.349537	16.127315	26.625752	101.056852
Drizzle	7.353659	5.504878	69.048780	16.097561	17.931707	101.099268
Drizzle,Fog	8.067500	7.033750	70.062500	11.862500	5.257500	100.820750
Drizzle,Ice Pellets,Fog	0.400000	-0.700000	52.000000	20.000000	4.000000	99.440000
Drizzle,Snow	1.050000	0.150000	44.000000	14.000000	10.500000	100.490000
Drizzle,Snow,Fog	0.693333	0.120000	69.800000	15.533333	5.513333	100.971333
Fog	4.303333	3.159333	66.466667	7.946667	6.248000	101.149400
Freezing Drizzle	-5.657143	-8.000000	68.857143	16.571429	9.200000	101.070000
Freezing Drizzle,Fog	-2.533333	-4.183333	64.000000	17.000000	5.266667	100.851667
Freezing Drizzle,Haze	-5.433333	-8.000000	63.333333	10.333333	2.666667	101.136667
Freezing Drizzle,Snow	-5.109091	-7.072727	62.454545	16.272727	5.872727	100.380909
Freezing Fog	-7.575000	-9.250000	68.000000	4.750000	0.650000	101.222500

Code Explanation:

This code will group your DataFrame by the 'Weather Condition' column and then calculate the mean for each numerical column in the grouped data for each unique 'Weather Condition.' The resulting DataFrame, means_by_weather_condition, will contain the mean values for each column against each 'Weather Condition.

Q. 12) What is the Minimum & Maximum value of each column against each 'Weather Condition ?

Python Code :

```
# Q. 12) What is the Minimum & Maximum value of each column against each 'Weather Condition ?
result = df.groupby('Weather Condition').agg([min,max])
result
```

	Date/Time		Temp_C		Dew Point Temp_C		Rel Hum_%		Wind Speed_km/h		Visibility_km		Press_kPa	
	min	max	min	max	min	max	min	max	min	max	min	max	min	max
Weather Condition														
Clear	01-01-2012 00:00	9/30/2012 7:00	-23.3	32.8	-28.5	20.4	18	100	0	33	11.3	48.3	97.75	103.63
Cloudy	01-01-2012 02:00	9/30/2012 8:00	-21.4	30.5	-26.8	22.6	20	100	0	54	11.3	48.3	97.52	103.52
Drizzle	01-06-2012 08:00	9/15/2012 22:00	1.1	18.8	-0.2	17.7	37	97	0	30	6.4	25.0	98.29	103.58
Drizzle,Fog	01-04-2012 01:00	9/19/2012 15:00	0.0	19.9	-1.6	19.1	38	98	0	28	1.0	9.7	98.32	103.56
Drizzle,Ice Pellets,Fog	7/24/2012 5:00	7/24/2012 5:00	0.4	0.4	-0.7	-0.7	52	52	20	20	4.0	4.0	99.44	99.44
Drizzle,Snow	05-02-2012 09:00	3/17/2012 1:00	0.9	1.2	0.1	0.2	39	49	9	19	9.7	11.3	100.27	100.71

Code Explanation:

This code will group the DataFrame by 'Weather Condition' and then calculate the minimum and maximum values for each group.The resulting DataFrame result will display the minimum and maximum values for each column for each 'Weather Condition'.

Q. 13) Show all the Records where Weather Condition is Fog.

Python Code :

```
# Q. 13) Show all the Records where Weather Condition is Fog.
Fog_Records = df[df['Weather Condition']=='Fog']
Fog_Records
```

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather Condition
13	01-01-2012 13:00	9.5	7.8	40	13	6.4	100.90	Fog
53	03-01-2012 05:00	-3.6	-4.3	57	7	9.7	101.32	Fog
136	06-01-2012 16:00	14.8	13.5	80	19	9.7	100.86	Fog
197	09-01-2012 05:00	2.1	0.7	43	11	8.0	101.44	Fog
278	12-01-2012 14:00	1.2	0.6	70	13	6.4	103.22	Fog
...	...	...	...	...	...	...	...	...
8475	9/18/2012 11:00	6.2	5.4	56	7	4.8	102.03	Fog
8511	9/19/2012 22:00	15.7	15.4	66	7	8.0	101.93	Fog
8518	9/19/2012 8:00	-2.9	-4.5	68	6	6.4	100.41	Fog
8537	9/20/2012 3:00	-0.5	-2.1	74	7	4.0	100.81	Fog
8771	9/30/2012 19:00	12.8	12.2	91	19	4.8	100.60	Fog

150 rows × 8 columns

Code Explanation: This code will display all records where the weather condition is "Fog."

Q. 14) Find all instances when 'Weather is Clear' or 'Visibility is above 40'.

Python Code :

```
# Q. 14) Find all instances when 'Weather is Clear' or 'Visibility is above 40'.
result = df[(df['Weather Condition']=='Clear') | (df['Visibility_km']>40)]
result
```

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather Condition
0	01-01-2012 00:00	-1.3	-3.5	18	9	25.0	98.67	Clear
9	01-01-2012 09:00	20.0	3.8	35	17	48.3	100.11	Clear
16	01-01-2012 16:00	23.8	17.6	42	9	25.0	100.52	Clear
17	01-01-2012 17:00	-6.8	-9.8	42	20	48.3	100.76	Mainly Clear
18	01-01-2012 18:00	2.3	-2.4	42	6	48.3	101.05	Cloudy
...	...	...	...	...	...	...	...	...
8774	9/30/2012 21:00	23.0	14.7	92	13	48.3	101.93	Mostly Cloudy
8777	9/30/2012 3:00	9.3	5.8	95	9	48.3	101.25	Mainly Clear
8779	9/30/2012 5:00	1.4	-3.7	97	22	48.3	100.16	Cloudy
8780	9/30/2012 6:00	-4.6	-9.5	98	11	48.3	101.46	Mostly Cloudy
8781	9/30/2012 7:00	1.5	-6.3	99	30	24.1	101.48	Clear

3027 rows × 8 columns

Code Explanation:

This code will create a new dataframe result data frame that contains only the rows where 'Weather' is 'Clear' or 'Visibility' is above 40.

Q. 15) Find all instances when :

A. 'Weather is Clear' and 'Relative Humidity is greater than 50'

or

B. 'Visibility is above 40'.

Python Code :

```
'''Q. 15) Find all instances when :
A. 'Weather is Clear' and 'Relative Humidity is greater than 50'
or
B. 'Visibility is above 40'''
result =df[((df['Weather Condition']=='Clear') & (df['Rel Hum_%']>50))|(df['Visibility_km']>40)]
result
```

	Date/Time	Temp_C	Dew Point Temp_C	Rel Hum_%	Wind Speed_km/h	Visibility_km	Press_kPa	Weather Condition
9	01-01-2012 09:00	20.0	3.8	35	17	48.3	100.11	Clear
17	01-01-2012 17:00	-6.8	-9.8	42	20	48.3	100.76	Mainly Clear
18	01-01-2012 18:00	2.3	-2.4	42	6	48.3	101.05	Cloudy
19	01-01-2012 19:00	-12.7	-17.2	43	17	48.3	101.16	Clear
23	01-01-2012 23:00	29.5	16.8	45	4	48.3	101.07	Mainly Clear
...	...	...	...	...	...	...	...	...
8774	9/30/2012 21:00	23.0	14.7	92	13	48.3	101.93	Mostly Cloudy
8777	9/30/2012 3:00	9.3	5.8	95	9	48.3	101.25	Mainly Clear
8779	9/30/2012 5:00	1.4	-3.7	97	22	48.3	100.16	Cloudy
8780	9/30/2012 6:00	-4.6	-9.5	98	11	48.3	101.46	Mostly Cloudy
8781	9/30/2012 7:00	1.5	-6.3	99	30	24.1	101.48	Clear

2864 rows × 8 columns

Code Explanation:

The **result** will contain the rows that satisfy either condition A or condition B or both.