KGiSL INSTITUTE OF TECHNOLOGY

Coimbatore – 641035

Institution code: 7117

Big Data Analysis Using IBM Cloud Databases

MENTOR:
MRS.INDU POORNIMA.R
TEAM MEMBERS:
YOGANATHAN.R
SABAREESAN.R
VENKATESH.R
SIBIRAJ.M

Big Data Analysis Using IBM Cloud Databases

Problem Definition: The project involves delving into big data analysis using IBM Cloud Databases. The objective is to extract valuable insights from extensive datasets, ranging from climate trends to social patterns. The project includes designing the analysis process, setting up IBM Cloud Databases, performing data analysis, and visualizing the results for business intelligence.

Project Objective:

Data Selection: Identify and collect relevant datasets for analysis, including climate data and social media trends.
Database Setup: Configure and maintain IBM Cloud Databases to store and manage large datasets securely.
Data Exploration: Develop and implement scripts and queries to explore and understand the datasets.
Analysis Techniques: Apply statistical analysis and machine learning techniques to extract insights and patterns from the data.
Visualization: Create visualizations to effectively communicate analysis results to stakeholders.

Design Thinking:

Data Selection: Clearly defined the first step, which is identifying the datasets to be analyzed. It's essential to have a clear understanding of what data you'll be working with before proceeding with analysis.
Database Setup: Mentioned the importance of setting up IBM Cloud Databases for storing and managing the datasets. This step is crucial for data organization and accessibility.
Data Exploration: Highlighted the need to develop queries and scripts to explore the datasets. Data exploration helps in understanding the data's structure, quality, and initial insights.
Analysis Techniques: Emphasized the application of appropriate analysis techniques, such as statistical analysis or machine learning. This step is where the core analysis takes place to uncover meaningful insights.
Visualization: Recognized the importance of designing visualizations to present analysis results. Visualizations make complex data more understandable and can communicate insights effectively.

Development part 1:

Dataset: GlobalLandTemperaturesByCity.csv
Description: The GlobalLandTemperaturesByCity dataset is a comprehensive and widely used dataset that contains historical temperature records from various cities around the world. This dataset is an invaluable resource for climate scientists, researchers, and data analysts interested in climate change, weather patterns, and temperature trends over time.

Step 1: Import Libraries and Load Data :

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/GlobalLandTemperaturesByCity.csv')
```

OUTPUT :

```
Mounted at /content/drive
```

Step 2: Data Cleaning :

```
# Drop rows with missing temperature data
data = data.dropna(subset=['AverageTemperature'])
print("\nAfter dropping rows with missing AverageTemperature:")
print(data.head(5))
# Display the first 5 rows of the DataFrame
# Fill missing city names with 'Unknown'
data['City'] = data['City'].fillna('Unknown')
print("\nAfter filling missing City names with 'Unknown':")
print(data.head(5))
```

OUTPUT :

```
After dropping rows with missing AverageTemperature:
           dt  AverageTemperature  AverageTemperatureUncertainty   City  \
0  1743-11-01               6.068                          1.737  Århus
5  1744-04-01               5.788                          3.624  Århus
6  1744-05-01              10.644                          1.283  Århus
7  1744-06-01              14.051                          1.347  Århus
8  1744-07-01              16.082                          1.396  Århus

   Country Latitude Longitude
0  Denmark   57.05N    10.33E
5  Denmark   57.05N    10.33E
6  Denmark   57.05N    10.33E
7  Denmark   57.05N    10.33E
8  Denmark   57.05N    10.33E

After filling missing City names with 'Unknown':
           dt  AverageTemperature  AverageTemperatureUncertainty   City  \
0  1743-11-01               6.068                          1.737  Århus
5  1744-04-01               5.788                          3.624  Århus
6  1744-05-01              10.644                          1.283  Århus
7  1744-06-01              14.051                          1.347  Århus
8  1744-07-01              16.082                          1.396  Århus

   Country Latitude Longitude
0  Denmark   57.05N    10.33E
5  Denmark   57.05N    10.33E
6  Denmark   57.05N    10.33E
7  Denmark   57.05N    10.33E
8  Denmark   57.05N    10.33E
```

```
# Remove duplicate records based on all columns
data = data.drop_duplicates()
print("\nAfter removing duplicates based on all columns:")
print(data.head(5))
# Display the first 5 rows of the DataFrame # Optionally, reset the index
data = data.reset_index(drop=True)
print("\nAfter resetting the index:")
print(data.head(5))
```

OUTPUT :

```
After removing duplicates based on all columns:
           dt  AverageTemperature  AverageTemperatureUncertainty   City  \
0  1743-11-01               6.068                          1.737  Århus
5  1744-04-01               5.788                          3.624  Århus
6  1744-05-01              10.644                          1.283  Århus
7  1744-06-01              14.051                          1.347  Århus
8  1744-07-01              16.082                          1.396  Århus

   Country Latitude Longitude
0  Denmark   57.05N    10.33E
5  Denmark   57.05N    10.33E
6  Denmark   57.05N    10.33E
7  Denmark   57.05N    10.33E
8  Denmark   57.05N    10.33E

After resetting the index:
           dt  AverageTemperature  AverageTemperatureUncertainty   City  \
0  1743-11-01               6.068                          1.737  Århus
1  1744-04-01               5.788                          3.624  Århus
2  1744-05-01              10.644                          1.283  Århus
3  1744-06-01              14.051                          1.347  Århus
4  1744-07-01              16.082                          1.396  Århus

   Country Latitude Longitude
0  Denmark   57.05N    10.33E
1  Denmark   57.05N    10.33E
2  Denmark   57.05N    10.33E
3  Denmark   57.05N    10.33E
4  Denmark   57.05N    10.33E
```

Step 3: Data Transformation :

```
data['Year'] = data['dt'].str[:4].astype(int)
# Calculate the average temperature for each city and year
agg_data = data.groupby(['City', 'Year'])['AverageTemperature'].mean().reset_index()
print(agg_data)
print("\nAggregated data by City and Year with average temperature:")
```

OUTPUT :

```
            City  Year  AverageTemperature
0       A Coruña  1743           10.779000
1       A Coruña  1744           13.678125
2       A Coruña  1745            9.170500
3       A Coruña  1750           13.489273
4       A Coruña  1751           13.698500
...          ...   ...                 ...
681564    Ürümqi  2009            7.287417
681565    Ürümqi  2010            6.650083
681566    Ürümqi  2011            6.806083
681567    Ürümqi  2012            6.600167
681568    Ürümqi  2013            9.472000

[681569 rows x 3 columns]

Aggregated data by City and Year with average temperature:
```

Step 4: Save the Cleaned and Transformed Data :

```
# Save the cleaned and transformed data to a new CSV file
agg_data.to_csv('cleaned_and_transformed_data.csv', index=False)
print(agg_data)
print("\nCleaned and transformed data saved to 'cleaned_and_transformed_data.csv'")
```

OUTPUT :

```
            City  Year  AverageTemperature
0       A Coruña  1743           10.779000
1       A Coruña  1744           13.678125
2       A Coruña  1745            9.170500
3       A Coruña  1750           13.489273
4       A Coruña  1751           13.698500
...          ...   ...                 ...
681564    Ürümqi  2009            7.287417
681565    Ürümqi  2010            6.650083
681566    Ürümqi  2011            6.806083
681567    Ürümqi  2012            6.600167
681568    Ürümqi  2013            9.472000

[681569 rows x 3 columns]

Cleaned and transformed data saved to 'cleaned_and_transformed_data.csv'
```

Analysis Techniques:

Time Series Analysis: The code leverages time series analysis to examine how a particular variable (in this case, average temperature) changes over time. It accomplishes this by converting the 'dt' column to a datetime format, setting it as the index, and resampling the data to calculate monthly average temperatures.

Visualization Methods:

Matplotlib: The code employs Matplotlib, a widely-used Python library, for creating visualizations. Specifically, it uses Matplotlib to create a line plot of the monthly average temperatures over time.

Seaborn: Seaborn, a data visualization library based on Matplotlib, is used for improving the aesthetics and style of the plot. It sets the plot's style using sns.set_style('whitegrid').

Development Part 2:

Part 1: Import Libraries and Load Data

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/GlobalLandTemperaturesByCity.csv')
```

OUTPUT:

```
Mounted at /content/drive
```

Part 2: Convert 'dt' Column to DateTime

```
data['dt'] = pd.to_datetime(data['dt'])
print("After converting 'dt' column to datetime:")
print(data['dt'].head())
```

OUTPUT:

```
After converting 'dt' column to datetime:
0   1743-11-01
1   1743-12-01
2   1744-01-01
3   1744-02-01
4   1744-03-01
Name: dt, dtype: datetime64[ns]
```

Part 3: Set 'dt' Column as the Index for Time Series Analysis

```
data.set_index('dt', inplace=True)
print("After setting 'dt' column as the index:")
print(data.head())
```

OUTPUT:

```
After setting 'dt' column as the index:
            AverageTemperature  AverageTemperatureUncertainty   City  Country  \
dt
1743-11-01               6.068                          1.737  Århus  Denmark
1743-12-01                 NaN                            NaN  Århus  Denmark
1744-01-01                 NaN                            NaN  Århus  Denmark
1744-02-01                 NaN                            NaN  Århus  Denmark
1744-03-01                 NaN                            NaN  Århus  Denmark

           Latitude Longitude
dt
1743-11-01   57.05N    10.33E
1743-12-01   57.05N    10.33E
1744-01-01   57.05N    10.33E
1744-02-01   57.05N    10.33E
1744-03-01   57.05N    10.33E
```

Part 4: Resample the Data to Monthly Averages

```
monthly_data = data['AverageTemperature'].resample('M').mean()
print("Monthly Average Temperatures:")
print(monthly_data.head())
```

OUTPUT:

```
Monthly Average Temperatures:
dt
1743-11-30    4.882424
1743-12-31         NaN
1744-01-31         NaN
1744-02-29         NaN
1744-03-31         NaN
Freq: M, Name: AverageTemperature, dtype: float64
```

Part 5: Visualize the Time Series Data

```
plt.figure(figsize=(12, 6))
sns.set_style('whitegrid')
plt.plot(monthly_data.index, monthly_data, label='Monthly Average Temperature',
color='blue')
plt.title('Monthly Average Temperature Over Time')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.legend()
plt.tight_layout()
plt.show()
```

OUTPUT: