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.
Business Insights: Interpret analysis findings to provide valuable business intelligence and actionable recommendations.
Documentation: Maintain comprehensive documentation of data sources, analysis methods, and results.
Security and Privacy: Implement data security measures to protect sensitive information during storage and analysis.
Scalability: Ensure that the project can handle increasingly large datasets as needed.
Feedback and Validation: Seek feedback from stakeholders to validate analysis results and refine recommendations.

### Project Deliverables:

A well-defined list of selected datasets with clear descriptions.
Configured and maintained IBM Cloud Databases with documentation.
Data exploration scripts and queries.
Analysis results, including statistical findings and machine learning models.
Visualizations for presenting insights.
Business intelligence reports with actionable recommendations.
Comprehensive project documentation, including data dictionaries and methodology.
Data security measures and privacy compliance documentation.
Scalability plan for handling larger datasets.
Stakeholder feedback reports and any refinement of analysis based on feedback.

## 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.

Business Insights:

Mentioned the ultimate goal of the project, which is to interpret analysis findings to derive valuable business intelligence and actionable recommendations. This step bridges the gap between data analysis and real-world decision-making.

## Project Innovation:

1.**Project Definition and Scope:**

Define the specific objectives of the project, including the types of insights to be extracted. Scope the project to encompass both predictive analysis and anomaly detection.

**2.Data Collection and Preparation:**

Identify and procure the relevant datasets, ensuring compatibility with IBM Cloud Databases.

Implement automated data pipelines for real-time or periodic data ingestion. Perform data cleansing, standardization, and transformation to ensure data quality and consistency.

**3.IBM Cloud Databases Setup:**

Choose the appropriate IBM Cloud Database service (e.g., Db2, Db2 Warehouse) based on the project's scale and requirements.

Provision the required resources and configure security settings.

**4.Advanced Machine Learning Algorithms:**

Identify suitable machine learning algorithms for predictive analysis (e.g., regression, time series forecasting) and anomaly detection (e.g., isolation forests, autoencoders). Implement and fine-tune these algorithms to maximize predictive accuracy and anomaly detection sensitivity.

**5.Data Ingestion and Streaming:**

Implement real-time data streaming capabilities to ingest and process data as it becomes available.

Utilize IBM Cloud services like IBM Cloud Streams for stream processing.

**6.Predictive Analysis:**

Train predictive models on historical data to forecast future trends and patterns. Implement continuous model retraining to adapt to changing data dynamics. Visualize and communicate predictive insights using dashboards.

**7.Anomaly Detection:**

Develop anomaly detection models to identify unusual patterns or events in the data. Integrate anomaly alerts and notifications for proactive response. Visualize anomalies and their context for effective decision-making.

**8.Data Integration:**

Merge results from predictive analysis and anomaly detection to gain a holistic view of the data.

Ensure seamless integration with business intelligence tools and platforms.

**9.Testing and Validation:**

Rigorously test the predictive models and anomaly detection algorithms. Validate the models' accuracy and effectiveness in identifying anomalies and making predictions.

**10.Documentation and Knowledge Sharing:**

Create comprehensive documentation covering data sources, algorithms, model training, and deployment processes.

Share knowledge within the team to facilitate understanding.

**11.Security and Compliance:**

Implement robust security measures to protect data integrity and confidentiality. Ensure compliance with relevant data privacy regulations (e.g., GDPR, HIPAA).

**12.Continuous Improvement:**

Monitor model performance and refine algorithms to adapt to changing data patterns. Solicit feedback from stakeholders and end-users to drive continuous improvement.

**13.User Training:**

Provide training to end-users on how to interpret predictive insights and anomaly alerts. Ensure that users can extract actionable information from the analysis.

**14.Business Impact Assessment:**

Evaluate the impact of the insights generated by predictive analysis and anomaly detection on business strategies and decision-making.

## DEVELOPMENT PART 1

### Step 1: Import Libraries and Load Data :

```
import pandas as pd

data = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/GlobalLandTemperaturesByCity.csv')

from google.colab import drive

drive.mount('/content/drive')
```

### 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
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  Year
0  Denmark   57.05N    10.33E  1743
1  Denmark   57.05N    10.33E  1744
2  Denmark   57.05N    10.33E  1744
3  Denmark   57.05N    10.33E  1744
4  Denmark   57.05N    10.33E  1744

After filling missing City names with 'Unknown':
           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  Year
0  Denmark   57.05N    10.33E  1743
1  Denmark   57.05N    10.33E  1744
2  Denmark   57.05N    10.33E  1744
3  Denmark   57.05N    10.33E  1744
4  Denmark   57.05N    10.33E  1744
```

```
# Remove duplicate records based on all columns
data = data.drop_duplicates()
print("\nAfter removing duplicates based on all column:
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
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  Year
0  Denmark   57.05N    10.33E  1743
1  Denmark   57.05N    10.33E  1744
2  Denmark   57.05N    10.33E  1744
3  Denmark   57.05N    10.33E  1744
4  Denmark   57.05N    10.33E  1744

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  Year
0  Denmark   57.05N    10.33E  1743
1  Denmark   57.05N    10.33E  1744
2  Denmark   57.05N    10.33E  1744
3  Denmark   57.05N    10.33E  1744
4  Denmark   57.05N    10.33E  1744
```

**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'
```

## DEVELOPMENT PART 2

**Part 1: Import Libraries and Load Data :**

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data =
pd.read_csv('/content/drive/MyDrive/ColabNotebooks/GlobalLandTemperaturesByCity.csv')
```

**Part 2: Convert 'dt' Column to DateTime :**

```
# Convert the 'dt' column to a datetime format for time series analysis
data['dt'] = pd.to_datetime(data['dt'])
print("After converting 'dt' column to datetime:")
print(data['dt'].head()) # Display the first few rows of the 'dt' column to confirm the conversion
```

**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 :**

```
# Set the '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 :**

```
# Resample the data to calculate monthly average temperatures
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 :**

```
# 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 :**