Week 1: Data Warehousing and SQL for Environmental Data

Topics Covered:

- Introduction to Data Warehousing for environmental data (e.g., air quality, weather, pollution levels)
- SQL for creating tables, querying, and managing environmental data, such as air quality readings, weather conditions, and sensor metadata

Capstone Project Milestone:

- **Objective**: Design a Data Warehouse schema to store environmental data, such as air quality levels (PM2.5, PM10), weather conditions (temperature, humidity), and sensor information.

Tasks:

1. **Design Schema**: Create tables to store sensor data, air quality readings, weather conditions, and pollution levels.
2. **Querying Data**: Write SQL queries to monitor air quality, track pollution levels over time, and calculate average temperature and humidity.

Example Code:

```sql
-- Create schema for environmental data
CREATE TABLE sensor_dim (
    sensor_id INT PRIMARY KEY,
    sensor_location VARCHAR(255),
    sensor_type VARCHAR(50)
);

CREATE TABLE air_quality_fact (
    record_id INT PRIMARY KEY,
    sensor_id INT,
    pm25 DECIMAL(5, 2),  -- PM2.5 in µg/m³
    pm10 DECIMAL(5, 2),  -- PM10 in µg/m³
    co2 DECIMAL(6, 2),   -- CO2 in ppm
    record_time TIMESTAMP
);

CREATE TABLE weather_fact (
    record_id INT PRIMARY KEY,
    sensor_id INT,
    temperature DECIMAL(4, 2),  -- Temperature in Celsius
    humidity DECIMAL(4, 2),     -- Humidity as a percentage
    wind_speed DECIMAL(4, 2),   -- Wind speed in m/s
    record_time TIMESTAMP
);

-- Query to calculate average PM2.5 levels by location
SELECT sensor_location, AVG(pm25) AS avg_pm25
FROM air_quality_fact aqf
JOIN sensor_dim sd ON aqf.sensor_id = sd.sensor_id
GROUP BY sensor_location
ORDER BY avg_pm25 DESC;
```

Outcome: By the end of Week 1, participants will have set up a Data Warehouse schema for storing environmental data, including air quality and weather information, and written SQL queries to analyze trends in pollution levels and weather conditions.

Week 2: Python for Data Collection and Preprocessing

Topics Covered:

- Python for collecting environmental data from sensors and APIs (e.g., air quality monitors, weather stations)
- Feature engineering for predictive modeling (e.g., pollution levels, temperature, humidity, wind speed)

Capstone Project Milestone:

- **Objective**: Collect and preprocess environmental data using Python, and generate features that can be used to predict air quality trends or weather conditions.

Tasks:

1. **Data Collection**: Use APIs or streaming data sources to collect real-time environmental data (e.g., air quality levels, weather updates).
2. **Data Preprocessing**: Clean and preprocess the data (e.g., handle missing values, normalize pollution and weather readings).
3. **Feature Engineering**: Create features such as pollution levels (PM2.5, PM10, CO2), temperature, humidity, and wind speed for predictive modeling.

Example Code:

```
import pandas as pd
import requests

# Collect real-time environmental data from an API (replace with real API)
response = requests.get("https://api.environment.com/data")
environmental_data = pd.DataFrame(response.json())

# Feature engineering: normalize PM2.5 and temperature readings
environmental_data['normalized_pm25'] = (environmental_data['pm25'] -
environmental_data['pm25'].mean()) / environmental_data['pm25'].std()
environmental_data['normalized_temp'] = (environmental_data['temperature'] -
environmental_data['temperature'].mean()) / environmental_data['temperature'].std()

# Display the processed environmental data
print(environmental_data[['sensor_id', 'pm25', 'normalized_pm25', 'temperature',
'normalized_temp']].head())
```

Outcome: By the end of Week 2, participants will have collected and preprocessed real-time environmental data, and generated features to be used in predictive models for air quality and weather conditions.

Week 3: Real-Time Data Processing with Apache Spark and PySpark

Topics Covered:

- Using Apache Spark and PySpark for processing large-scale real-time environmental data

- Real-time anomaly detection for pollution levels and weather anomalies (e.g., pollution spikes, temperature extremes)

Capstone Project Milestone:

- **Objective**: Process real-time environmental data using Apache Spark and PySpark to detect anomalies, such as sudden pollution spikes or extreme weather conditions.

Tasks:

1. **Set Up Spark Streaming**: Configure Apache Spark for handling real-time streaming of environmental data from sensors and weather stations.
2. **Anomaly Detection**: Use PySpark to detect anomalies in real-time, such as high pollution levels, abnormal temperatures, or sudden changes in humidity or wind speed.

Example Code:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# Create Spark session for streaming environmental data
spark = SparkSession.builder.appName("EnvironmentalMonitoring").getOrCreate()

# Read streaming data from sensors or weather stations (e.g., Kafka)
environment_stream = spark.readStream.format("kafka").option("subscribe",
"environment_data").load()

# Detect anomalies: identify pollution spikes or extreme weather conditions
anomalies = environment_stream.filter((col("pm25") > 150) | (col("temperature") > 40)
| (col("temperature") < 0))

# Write detected anomalies to console or database
query = anomalies.writeStream.outputMode("append").format("console").start()
query.awaitTermination()
```

Outcome: By the end of Week 3, participants will have implemented real-time anomaly detection using Apache Spark and PySpark to monitor environmental conditions in real-time, helping to detect and alert on pollution spikes or extreme weather conditions.

Week 4: Building Environmental Prediction Models with Azure Databricks

Topics Covered:

- Azure Databricks for building and deploying machine learning models for environmental prediction
- Training a prediction model to forecast air quality trends and weather conditions based on historical data

Capstone Project Milestone:

- **Objective**: Build and deploy a machine learning model in Azure Databricks to predict future air quality and weather conditions based on historical data and real-time inputs (e.g., pollution levels, weather conditions).

Tasks:

1. **Model Building**: Use Azure Databricks to train a machine learning model (e.g., time series forecasting, regression) on historical environmental data to predict air quality and weather trends.
2. **Deploy the Model**: Deploy the model in Databricks to forecast future air quality and weather conditions based on real-time inputs.

Example Code:

```
from pyspark.ml.regression import DecisionTreeRegressor
from pyspark.ml.feature import VectorAssembler

# Load historical environmental data
env_df = spark.read.csv('/mnt/data/environmental_data.csv', header=True,
inferSchema=True)

# Feature engineering: prepare features for the model
assembler = VectorAssembler(inputCols=["temperature", "humidity", "wind_speed",
"pm25", "pm10", "co2"], outputCol="features")
env_df = assembler.transform(env_df)

# Train a decision tree model to predict air quality levels
dt = DecisionTreeRegressor(featuresCol="features", labelCol="pm25")
model = dt.fit(env_df)

# Deploy the model: use it to predict future air quality levels in real-time
real_time_env = spark.readStream.format("kafka").option("subscribe",
"real_time_environment").load()
predictions = model.transform(real_time_env)
predictions.select("sensor_id",
"prediction").writeStream.outputMode("append").format("console").start()
```

Outcome: By the end of Week 4, participants will have built and deployed a machine learning model in Azure Databricks that predicts future air quality levels and weather conditions based on real-time data.

Week 5: Automating the Environmental Monitoring System with Azure DevOps

Topics Covered:

- Automating deployment of the environmental monitoring and prediction system with Azure DevOps
- Implementing CI/CD pipelines for deploying and monitoring environmental data pipelines and prediction models

Capstone Project Milestone:

- **Objective**: Deploy and automate the environmental monitoring and prediction system using Azure DevOps for continuous integration and monitoring of environmental data pipelines and prediction models.

Tasks:

1. **Azure DevOps Pipelines**: Set up Azure DevOps pipelines to automate the deployment of the environmental monitoring and prediction system.
2. **Monitoring and Alerts**: Set up monitoring and alerts to track environmental anomalies, pollution spikes, and extreme weather conditions in real-time.

Example Code

```
# Azure DevOps pipeline YAML for deploying the environmental monitoring system
trigger:
  - main

pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: UsePythonVersion@0
    inputs:
      versionSpec: '3.x'

  - script: |
      pip install -r requirements.txt
      python deploy_environment_monitoring.py
    displayName: 'Deploy Environmental Monitoring System'

  - task: AzureMonitorMetrics@0
    inputs:
      monitorName: 'EnvironmentalAnomalies'
      alertCriteria: 'Pollution Spike or Extreme Weather Detected'
```

Outcome: By the end of Week 5, participants will have deployed and automated the environmental monitoring and prediction system using Azure DevOps. Alerts will be set up to track real-time environmental anomalies such as pollution spikes and extreme weather conditions.

Summary of Outcomes:

1. **Week 1**: Design a Data Warehouse schema for storing environmental data such as air quality levels, weather conditions, and sensor metadata. Write SQL queries to analyze trends in air quality and weather.
2. **Week 2**: Collect and preprocess real-time environmental data using Python, including air quality and weather data from sensors and APIs. Generate features such as pollution levels, temperature, humidity, and wind speed for predictive models.
3. **Week 3**: Implement real-time anomaly detection using Apache Spark and PySpark to monitor environmental data in real-time, helping to detect pollution spikes, temperature extremes, and other environmental anomalies.
4. **Week 4**: Build and deploy a machine learning model in Azure Databricks to predict future air quality and weather conditions based on historical data and real-time inputs. The model will provide insights into future trends and help mitigate health risks associated with poor air quality or extreme weather.
5. **Week 5**: Automate the deployment and monitoring of the environmental monitoring and prediction system using Azure DevOps. Implement CI/CD pipelines for continuous integration, real-time monitoring, and alerting of environmental anomalies.

Potential Use Cases and Benefits:

1. **Public Health Monitoring**:
    - The system can monitor air quality in real-time and issue alerts when pollution levels become harmful, helping individuals avoid health risks such as respiratory issues.
2. **Environmental Protection Agencies**:
    - Environmental agencies can use the platform to track pollution levels across different regions, forecast future trends, and plan interventions to reduce pollution.
3. **Smart City Initiatives**:
    - The system can be integrated into smart city infrastructures to monitor environmental conditions in urban areas, improve air quality, and optimize responses to extreme weather events.
4. **Weather Forecasting**:
    - Predictive models can provide more accurate localized weather forecasts, helping residents and authorities prepare for sudden weather changes, such as storms or heatwaves.
5. **Industry Monitoring**:
    - Factories and industries can monitor the pollution they produce in real-time and take action to reduce emissions based on predictive insights and alerts.

Technologies Used:

- **SQL**: Data warehousing and querying for environmental data, such as air quality and weather conditions.
- **Python**: Data collection from environmental sensors and APIs, preprocessing, and feature engineering.
- **Apache Spark/PySpark**: Real-time data processing and anomaly detection for pollution and weather conditions.
- **Azure Databricks**: Machine learning for predicting air quality and weather conditions, real-time analytics, and ETL pipelines.
- **Azure DevOps**: CI/CD automation, deployment pipelines, and monitoring tools for system deployment and maintenance.
