Week 1: Data Warehousing and SQL for Energy Data

Topics Covered:

- Introduction to Data Warehousing for energy consumption data
- SQL for creating tables, querying, and managing energy usage, devices, and user data

Capstone Project Milestone:

- **Objective**: Design a Data Warehouse schema to store energy consumption data, device metadata, and user profiles.

Tasks:

1. **Design Schema**: Create tables to store energy consumption readings, device information (e.g., smart meters, IoT sensors), and user profiles.
2. **Querying Data**: Write SQL queries to analyze energy consumption patterns, identify peak usage periods, and calculate average energy usage.

Example Code:

```sql
-- Create schema for energy consumption data
CREATE TABLE user_dim (
    user_id INT PRIMARY KEY,
    user_name VARCHAR(255),
    location VARCHAR(255)
);

CREATE TABLE device_dim (
    device_id INT PRIMARY KEY,
    device_type VARCHAR(255),
    location VARCHAR(255)
);

CREATE TABLE energy_fact (
    reading_id INT PRIMARY KEY,
    device_id INT,
    user_id INT,
    energy_consumed DECIMAL(10, 2),  -- In kWh
    reading_time TIMESTAMP
);

-- Query to calculate average energy consumption by user
SELECT user_id, AVG(energy_consumed) AS avg_consumption
FROM energy_fact
GROUP BY user_id
ORDER BY avg_consumption DESC;
```

Outcome: By the end of Week 1, participants will have designed a Data Warehouse schema for storing energy consumption data, along with queries to monitor and analyze energy usage patterns.

Week 2: Python for Data Collection and Preprocessing

Topics Covered:

- Python for collecting and preprocessing energy data from smart meters and IoT sensors
- Feature engineering for predictive modeling (e.g., time of day, location, device type, weather data)

Capstone Project Milestone:

- **Objective**: Collect and preprocess energy consumption data using Python and generate features that can be used to predict future energy usage.

Tasks:

1. **Data Collection**: Use APIs or streaming sources to collect real-time data from smart meters and IoT sensors (e.g., energy readings, device metadata).
2. **Data Preprocessing**: Clean and preprocess the data (e.g., handle missing values, normalize readings).
3. **Feature Engineering**: Create features such as time of day, weather data, and device type to be used for prediction models.

Example Code:

```
import pandas as pd
import requests

# Collect real-time energy consumption data from an API (replace with real API)
response = requests.get("https://api.energygrid.com/consumption")
energy_data = pd.DataFrame(response.json())

# Feature engineering: extract hour of day and normalize energy readings
energy_data['reading_time'] = pd.to_datetime(energy_data['reading_time'])
energy_data['hour_of_day'] = energy_data['reading_time'].dt.hour
energy_data['normalized_energy'] = (energy_data['energy_consumed'] -
energy_data['energy_consumed'].mean()) / energy_data['energy_consumed'].std()

# Display the processed energy data
print(energy_data[['user_id', 'device_id', 'energy_consumed', 'hour_of_day',
'normalized_energy']].head())
```

Outcome: By the end of Week 2, participants will have collected and preprocessed real-time energy data and engineered features to be used in predictive models.

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

Topics Covered:

- Using Apache Spark and PySpark for processing large-scale real-time energy data
- Real-time anomaly detection for identifying unusual energy usage patterns or spikes

Capstone Project Milestone:

- **Objective**: Process real-time energy consumption data using Apache Spark and PySpark to detect anomalies such as unusual usage spikes or energy wastage.

Tasks:

1. **Set Up Spark Streaming**: Configure Apache Spark for handling real-time streaming of energy consumption data.

2. **Anomaly Detection**: Use PySpark to detect anomalies in real-time, such as usage spikes, unusual consumption patterns, or deviations from typical behavior.

Example Code:

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

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

# Read streaming data from smart meters or IoT sensors (e.g., Kafka)
energy_stream = spark.readStream.format("kafka").option("subscribe",
"energy_data").load()

# Detect anomalies: identify readings that deviate significantly from the average
anomalies = energy_stream.filter(col("energy_consumed") > col("average_consumption") *
1.5)

# 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 unusual energy consumption patterns in real-time.

Week 4: Building a Prediction Model for Energy Usage with Azure Databricks

Topics Covered:

- Azure Databricks for building and deploying machine learning models
- Training a prediction model to forecast future energy consumption based on historical data

Capstone Project Milestone:

- **Objective**: Build and deploy a machine learning model in Azure Databricks to predict future energy consumption based on historical data and other features (e.g., weather, time of day).

Tasks:

1. **Model Building**: Use Azure Databricks to train a predictive model (e.g., linear regression, decision trees) on historical energy consumption data.
2. **Deploy the Model**: Deploy the model in Databricks to forecast future energy consumption based on real-time data.

Example Code:

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

# Load historical energy consumption data
energy_df = spark.read.csv('/mnt/data/energy_consumption.csv', header=True,
```

```
inferSchema=True)

# Feature engineering: prepare features for the model
assembler = VectorAssembler(inputCols=["hour_of_day", "normalized_energy",
"temperature"], outputCol="features")
energy_df = assembler.transform(energy_df)

# Train a linear regression model to predict future energy consumption
lr = LinearRegression(featuresCol="features", labelCol="energy_consumed")
model = lr.fit(energy_df)

# Deploy the model: use it to predict future energy consumption in real-time
real_time_energy = spark.readStream.format("kafka").option("subscribe",
"real_time_energy_data").load()
predictions = model.transform(real_time_energy)
predictions.select("device_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 energy consumption based on real-time data.

Week 5: Automating the Energy Monitoring and Prediction System with Azure DevOps

Topics Covered:

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

Capstone Project Milestone:

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

Tasks:

1. **Azure DevOps Pipelines**: Set up Azure DevOps pipelines to automate the deployment of the energy monitoring and prediction system.
2. **Monitoring and Alerts**: Set up monitoring and alerts to track anomalies, spikes, and predicted energy consumption patterns in real-time.

Example Code:

```
# Azure DevOps pipeline YAML for deploying the energy monitoring and prediction system
trigger:
  - main

pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: UsePythonVersion@0
```

```
    inputs:
      versionSpec: '3.x'

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

  - task: AzureMonitorMetrics@0
    inputs:
      monitorName: 'EnergyAnomalies'
      alertCriteria: 'Energy Spike Detected or Prediction Deviation'
```

Outcome: By the end of Week 5, participants will have deployed and automated the energy monitoring and prediction system using Azure DevOps, with alerts set up for real-time anomaly detection and predictions.

Summary of Outcomes:

1. **Week 1**: Design a Data Warehouse schema for energy consumption data and write SQL queries to monitor and analyze energy usage patterns.
2. **Week 2**: Collect and preprocess real-time energy data using Python, and engineer features for predictive models.
3. **Week 3**: Implement real-time anomaly detection using Apache Spark and PySpark to monitor unusual energy consumption patterns, such as spikes or deviations from typical usage.
4. **Week 4**: Build and deploy a machine learning model in Azure Databricks to predict future energy consumption based on historical data and real-time inputs, such as time of day, device type, and weather conditions.
5. **Week 5**: Automate the deployment and monitoring of the real-time energy monitoring and prediction system using Azure DevOps, including CI/CD pipelines for continuous integration, real-time monitoring, and alerting for anomalies and predictions.

Details of the System Components:

1. **Data Warehouse Schema (Week 1)**:
    - A central repository for storing energy consumption data, device metadata, and user profiles. SQL queries will be used to analyze and report on energy consumption trends, peak usage times, and overall consumption patterns.
2. **Data Collection and Preprocessing (Week 2)**:
    - Python will be used to collect real-time data from energy sources (smart meters, IoT sensors) and preprocess the data for further analysis. Feature engineering will involve creating features such as the hour of the day, normalized energy consumption, weather conditions, and other factors that can influence energy usage.

3. **Real-Time Processing with Apache Spark (Week 3)**:
    - Apache Spark Streaming will be used to process incoming real-time energy consumption data. Anomaly detection algorithms will identify outliers, such as spikes in consumption or sudden drops, and generate alerts when such anomalies are detected.
4. **Prediction Model for Energy Usage (Week 4)**:
    - Azure Databricks will be used to build and deploy a predictive model for energy consumption. The model will leverage historical data and real-time inputs to forecast future energy usage, allowing for proactive energy management and optimization.
5. **Automated Deployment and Monitoring (Week 5)**:
    - Azure DevOps will be used to automate the deployment of the energy monitoring and prediction system. CI/CD pipelines will handle updates to the system, and monitoring tools will ensure the system continues to perform as expected. Alerts will be set up for any detected anomalies or significant deviations from predicted consumption patterns.

Potential Use Cases and Benefits:

1. **Residential Energy Management**:
    - Homeowners can use the platform to monitor their energy usage in real-time, predict future energy costs, and optimize consumption during off-peak hours to save on electricity bills.
2. **Industrial Energy Optimization**:
    - Factories and large-scale industrial plants can monitor energy consumption across multiple devices and machinery in real-time, identify potential inefficiencies, and forecast future usage to plan operations more effectively.
3. **Grid Management for Utility Companies**:
    - Utility companies can use the system to monitor and manage energy distribution across the grid, predict peak usage times, and prevent outages by identifying potential overloads or energy shortages in real-time.
4. **Renewable Energy Integration**:
    - The system can help integrate renewable energy sources into the grid by predicting periods of high energy generation (e.g., during sunny or windy days) and optimizing energy storage and distribution.

Technologies Used:

- **SQL**: Data warehousing and querying for energy consumption data.
- **Python**: Data collection from APIs and IoT sensors, preprocessing, and feature engineering.
- **Apache Spark/PySpark**: Real-time data processing and anomaly detection at scale.

- **Azure Databricks**: Machine learning for predictive modeling, real-time analytics, and ETL pipelines.
- **Azure DevOps**: CI/CD automation, deployment pipelines, and monitoring tools for system deployment and maintenance.
