Week 1: Data Warehousing and SQL for Stock Market Data

Topics Covered:

- Introduction to Data Warehousing for stock market data (e.g., stock prices, trading volumes, market indicators)
- SQL for creating tables, querying, and managing stock data, company information, and historical market trends

Capstone Project Milestone:

- **Objective**: Design a Data Warehouse schema to store stock market data, such as stock prices, volumes, and market indicators (e.g., RSI, moving averages).

Tasks:

1. **Design Schema**: Create tables to store stock data, company profiles, and market indicators (e.g., RSI, moving averages, Bollinger Bands).
2. **Querying Data**: Write SQL queries to analyze stock market trends, calculate average prices, and identify sudden price changes or anomalies.

Example Code:

```
-- Create schema for stock market data
CREATE TABLE company_dim (
    company_id INT PRIMARY KEY,
    company_name VARCHAR(255),
    ticker_symbol VARCHAR(10)
);

CREATE TABLE stock_fact (
    stock_id INT PRIMARY KEY,
    company_id INT,
    stock_price DECIMAL(10, 2),
    trading_volume BIGINT,
    moving_avg DECIMAL(10, 2),
    rsi DECIMAL(5, 2),
    market_cap DECIMAL(15, 2),
    record_time TIMESTAMP
);

-- Query to calculate average stock price for a company
SELECT company_name, AVG(stock_price) AS avg_stock_price
FROM stock_fact sf
JOIN company_dim cd ON sf.company_id = cd.company_id
GROUP BY company_name
ORDER BY avg_stock_price DESC;
```

Outcome: By the end of Week 1, participants will have designed a Data Warehouse schema for storing stock market data and written SQL queries to analyze stock prices and market indicators.

Week 2: Python for Data Collection and Preprocessing

Topics Covered:

- Python for collecting stock market data from APIs and financial data providers (e.g., Yahoo Finance, Alpha Vantage)
- Feature engineering for stock market analysis (e.g., price changes, trading volumes, moving averages, volatility)

Capstone Project Milestone:

- **Objective**: Collect and preprocess stock market data using Python, and generate features that can be used to predict stock prices and identify market trends.

Tasks:

1. **Data Collection**: Use APIs to collect real-time stock data from financial data providers, including stock prices, trading volumes, and market indicators.
2. **Data Preprocessing**: Clean and preprocess the data (e.g., handle missing values, normalize stock prices, and calculate indicators such as moving averages).
3. **Feature Engineering**: Create features such as price changes, RSI, volatility, moving averages, and Bollinger Bands for stock market prediction.

Example Code:

```
import pandas as pd
import requests

# Collect real-time stock data from a financial data API (replace with real API)
response = requests.get("https://api.stockmarket.com/data?ticker=AAPL")
stock_data = pd.DataFrame(response.json())

# Feature engineering: calculate price changes and moving averages
stock_data['price_change'] = stock_data['close'] - stock_data['open']
stock_data['moving_avg'] = stock_data['close'].rolling(window=10).mean()

# Display the processed stock data
print(stock_data[['ticker', 'price_change', 'moving_avg']].head())
```

Outcome: By the end of Week 2, participants will have collected and preprocessed real-time stock market data, and generated features to be used in predictive models for stock price prediction.

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

Topics Covered:

- Using Apache Spark and PySpark for processing large-scale real-time stock market data
- Real-time anomaly detection for identifying sudden price movements, unusual trading volumes, or market volatility

Capstone Project Milestone:

- **Objective**: Process real-time stock market data using Apache Spark and PySpark to detect anomalies, such as sudden price movements or unusually high trading volumes.

Tasks:

1. **Set Up Spark Streaming**: Configure Apache Spark for handling real-time streaming of stock market data from financial APIs and data providers.

2. **Anomaly Detection**: Use PySpark to detect anomalies in real-time, such as sudden price drops, trading volume spikes, or significant deviations from moving averages.

Example Code:

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

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

# Read streaming data from stock market APIs (e.g., Kafka)
stock_stream = spark.readStream.format("kafka").option("subscribe",
"stock_data").load()

# Detect anomalies: identify price drops or spikes in trading volume
anomalies = stock_stream.filter((col("price_change") < -10) | (col("trading_volume") >
1_000_000))

# 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 stock market movements in real-time, helping identify abnormal price or volume changes.

Week 4: Building a Stock Price Prediction Model with Azure Databricks

Topics Covered:

- Azure Databricks for building and deploying machine learning models for stock price prediction
- Training a machine learning model to predict future stock prices based on historical data and real-time inputs

Capstone Project Milestone:

- **Objective**: Build and deploy a machine learning model in Azure Databricks to predict future stock prices based on historical stock data and real-time inputs (e.g., price changes, moving averages, volatility).

Tasks:

1. **Model Building**: Use Azure Databricks to train a machine learning model (e.g., time series forecasting, regression) on historical stock price data to predict future prices.
2. **Deploy the Model**: Deploy the stock price prediction model in Databricks to forecast future stock prices in real-time.

Example Code:

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

```
# Load historical stock price data
stock_df = spark.read.csv('/mnt/data/stock_prices.csv', header=True, inferSchema=True)

# Feature engineering: prepare features for the model
assembler = VectorAssembler(inputCols=["price_change", "trading_volume", "moving_avg",
"volatility"], outputCol="features")
stock_df = assembler.transform(stock_df)

# Train a linear regression model to predict stock prices
lr = LinearRegression(featuresCol="features", labelCol="close")
model = lr.fit(stock_df)

# Deploy the model: use it to predict stock prices in real-time
real_time_stock = spark.readStream.format("kafka").option("subscribe",
"real_time_stock").load()
predictions = model.transform(real_time_stock)
predictions.select("ticker",
"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 stock prices based on real-time data and market indicators.

Week 5: Automating the Stock Market Monitoring System with Azure DevOps

Topics Covered:

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

Capstone Project Milestone:

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

Tasks:

1. **Azure DevOps Pipelines**: Set up Azure DevOps pipelines to automate the deployment of the stock market monitoring and prediction system.
2. **Monitoring and Alerts**: Set up monitoring and alerts to track stock market anomalies, price predictions, and trading volume spikes in real-time.

Example Code:

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

pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: UsePythonVersion@0
```

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

  - script: |
      pip install -r requirements.txt
      python deploy_stock_market_monitoring.py
    displayName: 'Deploy Stock Market Monitoring System'

  - task: AzureMonitorMetrics@0
    inputs:
      monitorName: 'StockMarketAnomalies'
      alertCriteria: 'Price Drop or Volume Spike Detected'
```

Outcome: By the end of Week 5, participants will have deployed and automated the stock market monitoring and prediction system using Azure DevOps, with real-time monitoring and alerts set up for price drops, volume spikes,