Week 1: Data Warehousing and SQL for E-Commerce Transactions

Topics Covered:

- Introduction to Data Warehousing for e-commerce transaction data
- SQL for creating tables, querying, and managing transaction data, user profiles, and fraud indicators

Capstone Project Milestone:

- **Objective**: Design a Data Warehouse schema to store transaction data, user profiles, and fraud labels (e.g., legitimate or fraudulent transactions).

Tasks:

1. **Design Schema**: Create tables to store transaction details, user information, and fraud indicators (fraud labels based on machine learning or business rules).
2. **Querying Data**: Write SQL queries to analyze transaction patterns, detect anomalies, and calculate fraud risk scores.

Example Code:

```sql
-- Create schema for e-commerce transactions
CREATE TABLE user_dim (
    user_id INT PRIMARY KEY,
    user_name VARCHAR(255),
    location VARCHAR(255)
);

CREATE TABLE transaction_fact (
    transaction_id INT PRIMARY KEY,
    user_id INT,
    transaction_amount DECIMAL(10, 2),
    transaction_time TIMESTAMP,
    payment_method VARCHAR(50),
    location VARCHAR(255),
    fraud_label BOOLEAN
);

-- Query to detect potentially fraudulent transactions above a certain threshold
SELECT user_id, transaction_amount, transaction_time
FROM transaction_fact
WHERE transaction_amount > 1000
ORDER BY transaction_time DESC;
```

Outcome: By the end of Week 1, participants will have designed a Data Warehouse schema for storing e-commerce transactions, user profiles, and fraud labels, along with SQL queries to detect anomalies and potential fraud.

Week 2: Python for Data Collection and Preprocessing

Topics Covered:

- Python for collecting transaction data from APIs and payment gateways
- Feature engineering for fraud detection (e.g., transaction frequency, location changes, unusual purchase amounts)

Capstone Project Milestone:

- **Objective**: Collect and preprocess e-commerce transaction data using Python, and generate features that can be used to detect fraud.

Tasks:

1. **Data Collection**: Use APIs or data streams to collect real-time transaction data from payment gateways, e-commerce platforms, and user devices.
2. **Data Preprocessing**: Clean and preprocess the transaction data (e.g., handle missing values, normalize transaction amounts).
3. **Feature Engineering**: Create features such as transaction frequency, location changes, and deviation from typical user behavior for fraud detection.

Example Code:

```
import pandas as pd
import requests

# Collect real-time transaction data from a payment gateway API (replace with real
API)
response = requests.get("https://api.paymentgateway.com/transactions")
transaction_data = pd.DataFrame(response.json())

# Feature engineering: calculate transaction frequency and normalize amounts
transaction_data['transaction_time'] =
pd.to_datetime(transaction_data['transaction_time'])
transaction_data['transaction_frequency'] = transaction_data.groupby('user_id')
['transaction_id'].transform('count')
transaction_data['normalized_amount'] = (transaction_data['transaction_amount'] -
transaction_data['transaction_amount'].mean()) /
transaction_data['transaction_amount'].std()

# Display the processed transaction data
print(transaction_data[['user_id', 'transaction_amount', 'transaction_frequency',
'normalized_amount']].head())
```

Outcome: By the end of Week 2, participants will have collected and preprocessed real-time e-commerce transaction data, and generated features to be used in fraud detection 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 e-commerce transaction data
- Real-time anomaly detection for identifying potential fraud, such as unusual transaction patterns or location changes

Capstone Project Milestone:

- **Objective**: Process real-time e-commerce transaction data using Apache Spark and PySpark to detect potential fraud, such as unusual purchase patterns, location mismatches, or unusually high transaction amounts.

Tasks:

1. **Set Up Spark Streaming**: Configure Apache Spark to handle real-time streaming of transaction data from payment gateways and e-commerce platforms.
2. **Anomaly Detection**: Use PySpark to monitor and detect potential fraudulent transactions based on anomaly detection techniques (e.g., flagging large, out-of-pattern transactions).

Example Code:

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

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

# Read streaming data from payment gateways or e-commerce platforms (e.g., Kafka)
transaction_stream = spark.readStream.format("kafka").option("subscribe",
"transaction_data").load()

# Detect anomalies: flag large transactions or mismatched locations
anomalies = transaction_stream.filter((col("transaction_amount") > 1000) |
(col("location") != col("user_location")))

# 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 identify potentially fraudulent transactions in e-commerce data.

Week 4: Building a Fraud Detection Model with Azure Databricks

Topics Covered:

- Azure Databricks for building and deploying machine learning models for fraud detection
- Training a machine learning model to predict the likelihood of fraud based on transaction patterns

Capstone Project Milestone:

- **Objective**: Build and deploy a machine learning model in Azure Databricks to predict fraudulent transactions based on historical data and real-time inputs (e.g., transaction amount, user location, frequency of transactions).

Tasks:

1. **Model Building**: Use Azure Databricks to train a machine learning model (e.g., logistic regression, decision trees) on historical transaction data labeled as fraudulent or legitimate.
2. **Deploy the Model**: Deploy the fraud detection model in Databricks to flag suspicious transactions in real-time.

Example Code:

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import VectorAssembler

# Load historical transaction data
transaction_df = spark.read.csv('/mnt/data/transaction_data.csv', header=True,
inferSchema=True)

# Feature engineering: prepare features for the model
assembler = VectorAssembler(inputCols=["transaction_amount", "transaction_frequency",
"location_change"], outputCol="features")
transaction_df = assembler.transform(transaction_df)

# Train a logistic regression model to predict fraudulent transactions
lr = LogisticRegression(featuresCol="features", labelCol="fraud_label")
model = lr.fit(transaction_df)

# Deploy the model: use it to score real-time transactions for fraud detection
real_time_transactions = spark.readStream.format("kafka").option("subscribe",
"real_time_transactions").load()
predictions = model.transform(real_time_transactions)
predictions.select("transaction_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 potential fraud in real-time e-commerce transactions.

Week 5: Automating the Fraud Detection System with Azure DevOps

Topics Covered:

- Automating deployment of the fraud detection system with Azure DevOps
- Implementing CI/CD pipelines for deploying and monitoring fraud detection pipelines and models

Capstone Project Milestone:

- **Objective**: Deploy and automate the fraud detection system using Azure DevOps for continuous integration and monitoring of e-commerce data pipelines and fraud detection models.

Tasks:

1. **Azure DevOps Pipelines**: Set up Azure DevOps pipelines to automate the deployment of the fraud detection system.
2. **Monitoring and Alerts**: Set up monitoring and alerts to track suspicious transactions and trigger real-time alerts for potential fraud.

Example Code:

```
# Azure DevOps pipeline YAML for deploying the fraud detection system
trigger:
  - main

pool:
```

```
  vmImage: 'ubuntu-latest'

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

  - script: |
      pip install -r requirements.txt
      python deploy_fraud_detection.py
    displayName: 'Deploy Fraud Detection System'

  - task: AzureMonitorMetrics@0
    inputs:
      monitorName: 'FraudAlerts'
      alertCriteria: 'Suspicious Transaction Detected'
```

Outcome: By the end of Week 5, participants will have deployed and automated the fraud detection system using Azure DevOps, with real-time monitoring and alerts set up for detecting and responding to suspicious e-commerce transactions.

Summary of Outcomes:

1. **Week 1**: Design a Data Warehouse schema for storing e-commerce transactions, user profiles, and fraud labels. Write SQL queries to detect potentially fraudulent transactions and monitor transaction patterns.
2. **Week 2**: Collect and preprocess real-time e-commerce transaction data using Python, and generate features for fraud detection models, such as transaction frequency, location changes, and unusual transaction amounts.
3. **Week 3**: Implement real-time anomaly detection using Apache Spark and PySpark to monitor e-commerce transactions in real-time, helping to identify unusual patterns or behavior indicative of potential fraud.
4. **Week 4**: Build and deploy a machine learning model in Azure Databricks to predict the likelihood of fraud based on historical transaction data and real-time inputs. The model will analyze transaction patterns and flag suspicious transactions for further investigation.
5. **Week 5**: Automate the deployment and monitoring of the fraud detection system using Azure DevOps. Implement CI/CD pipelines for continuous integration, real-time monitoring, and alerting of suspicious activities, allowing the system to respond quickly to potential fraud in e-commerce transactions.

Potential Use Cases and Benefits:

1. **E-Commerce Fraud Prevention**:
    - The system can help e-commerce companies detect and prevent fraudulent transactions in real-time, reducing financial losses and protecting both the company and its customers from fraud.
2. **Payment Gateway Security**:

    - Payment processors can use the system to monitor transactions across various merchants, identifying fraudulent patterns and blocking suspicious transactions before they are processed.

3. **Customer Protection**:
    - The fraud detection system helps protect customers by flagging unusual transactions (e.g., purchases from unusual locations or abnormal transaction amounts), reducing the risk of identity theft and unauthorized purchases.

4. **Real-Time Alerts**:
    - With real-time monitoring and alerting, e-commerce platforms can take immediate action when potential fraud is detected, such as freezing accounts or requesting additional verification before completing transactions.

5. **Regulatory Compliance**:
    - The system can assist companies in meeting regulatory requirements for fraud detection and reporting by providing a robust, automated solution for tracking and preventing fraudulent activities.

Technologies Used:

- **SQL**: Data warehousing and querying for transaction data and fraud indicators.
- **Python**: Data collection from payment gateways and e-commerce platforms, preprocessing, and feature engineering.
- **Apache Spark/PySpark**: Real-time data processing and anomaly detection for transaction data.
- **Azure Databricks**: Machine learning for fraud detection, real-time analytics, and ETL pipelines.
- **Azure DevOps**: CI/CD automation, deployment pipelines, and monitoring tools for system deployment and maintenance.
