Credit Card Fraud Detection System

Task List

Background Activity:

Infer structure of the two datasets "**member_details**" and "**member_score**" from AWS-RDS by using the following credentials.

Hostname: database-2.cl4c0rtglkdz.ap-south-1.rds.amazonaws.com

Username: admin

Password: Bankingprj1

Database: BankingPrj

Tables: member_details and member_score

==========

Task 1:

Copy "**card_transactions.csv"** file from local system to HDFS.

Table creation tasks :

Task 2:

Create the "**card_transactions"** table in MySQL based on the card_transactions.csv file structure.

Task 3:

Do a sqoop export to the database for card_transactions.csv and delete the file from HDFS.

Task 4:

On **"member_score"** and **"member_details"** create a normal hive external table.

Task 5:

Create a special **"card_transactions"** Hbase table managed by Hive.

Task 6:

Create a Hbase **"lookup"** table with columns - member_id, card_id, UCL, timestamp, zipcode, credit_score.

Batch Processing tasks:

Task 7:

Sqoop import member_score from AWS-RDS to Hive. (Full load import, has to be refreshed every week)

Task 8:

Sqoop import member_details from AWS-RDS to Hive. (Incremental load import in append mode based on member_id for every 8hrs)

Task 9:

Sqoop import card_transactions to HDFS from MySQL. (This is a one-time full load activity. The card_transactions table will be updated with new transactions while in streaming mode.)

Scheduling tasks:

Task 10:

Schedule a sqoop import job using Airflow to import member_score from AWS-RDS to Hive on a full-load.

Task 11:

Schedule a sqoop import job using Airflow to import member_details from AWS-RDS to Hive on an incremental append mode for every 8hrs.

Integration tasks:

Task 12:

Spark-HBase Integration

1. For populating the card_transactions table.
2. For populating the look_up table.

Task 13:

Spark-Hive Integration for spark stream processing.

Task 14:

Access the hive tables using apache spark and calculate the UCL.

Streaming tasks:

Task 15:

Producer to create the transactions in JSON format, to be added and queued in Kafka topics.

Task 16:

Spark structured streaming program as a Consumer that will consume the data from the kafka topics.

Task 17:

Retrieve the timestamp and zipcode of the last transaction of each card.

Task 18:

Processing in Spark Streaming -

Task 19.1 : Validating RULE 1 -> **"credit_score > 200"**

Task 19.2 : Validating RULE 2 -> **"transaction amount <= UCL"**

Task 19.3 : Validating RULE 3 -> **"zipcode distance within threshold"**

Task 19:

Based on the above rules, the entire transaction along with status should be updated in the card_transactions table.

Task 20:

Schedule a job for validating rules by comparing the incoming data from the POS terminals in JSON format with the values in the lookup table.

Task 21:

If the transaction was marked genuine, then we need to update the lookup table with the new timestamp and the zipcode.

Task 22:

Schedule a job for populating the lookup table.