Task-2: Catching Fraud

Q1. Regarding the question whether the query will be working or not, the answer is that it depends on the sql version you are using to run the query. If you are running the query on sqlite3 then it would not run because power and select left do not exist and will give syntactical errors. However, if you try to run it on MS SQL server(pre 2008 don't have some of the functions, it would be prudent to use a recent version like 2019), it will run without giving any syntactical errors (assuming that you have kept the schema of the tables as per the specification provided by the data team).

Assuming the query is run in MS SQL server, the output of the query would be an empty table. The aim of the query is to find the sum of the total amount spent by unique groups of customers in merchant countries with specific conditions on the transaction. The query progresses sequentially to check for common customer and merchant operation countries, then it finds the appropriate exchange rate with the base currency being set to EUR. It moves on to find the sum of the transactions for those whose source is 'GAIA' (In between, it uses a join over currency details and the intermediate result to find the appropriate exponent value). During the inner join process, it outlines appropriate join conditions so that only the right records make it to the final output. Finally, the resultant table is ordered in decreasing order of the total sales value for each customer, merchant country pair.

Q2.

```
SELECT t.*
FROM (SELECT t.USER_ID,
          (t.amount / fx.rate / power(10, cd.exponent)) AS amount,
          t.CURRENCY,
          t.CREATED_DATE,
          ROW_NUMBER() OVER (PARTITION BY t.USER_ID ORDER BY t.CREATED_DATE) as sn
     FROM transactions t JOIN
        fx_rates fx
        ON fx.ccy = t.currency AND fx.base_ccy = 'USD'
                    JOIN
        currency_details cd
        ON cd.currency = t.currency
     WHERE t.STATE = 'COMPLETED' AND t.TYPE = 'CARD_PAYMENT'
    ) t
WHERE sn = 1 AND amount > 10;
```

{Please run this on MS SQL Server to ensure it doesn't give syntax errors.}

Q3.

(Please note that for this classification, I am assuming that such cases can be looked into further by appropriate personnel, they may not be immediately dropped from the system, but their account may be suspended for the time being.)

To find likely fraudsters, we will have to use ML and the task will not be possible with SQL alone. For this task, I used the Random Forest Classifier which is available in the sklearn library of Python. Technically speaking, Python was used for the ETL process and the underlying sql environment was PostgreSQL.

Intuitively,in the beginning, we figured out that the current account status(Active or Locked) and the KYC pass result are useful indicators. In case someone has a locked status or they have a failure in the KYC pass result (or both), we indicated them to be likely fraudsters. However, there were a lot of data points where both of these conditions were not satisfied.

Next on, we also took into consideration the nationality risk. In case more than 50 % of the total applicants from a particular country are fraudsters, it would be prudent to classify a potential user from that country to be a fraudster so that they can be investigated further.

For many data points, these two conditions were passed, so for these, we proceeded on to focus on other features (which were 'birth_year', 'country', 'has_email', 'failed_sign_in_attempts') and constructed a random forest classifier to arrive at a conclusion. The data was split into training and testing in the ratio of 80:20. For the classifier, we also calculated the accuracy and selected an accuracy threshold of 95% by virtue of which we will concretely comment about a user if and only if the classifier model for that user has an accuracy greater than or equal to 95%. In case the model doesn't pass the threshold, we will report this by saying that the model doesn't have sufficient accuracy to determine the nature of the user(fraudster or not). This reporting is done in the Python runtime. However for our purpose, this effectively means that we need personnel review, so we took them as potential fraudsters as well.This was the method used to identify potential fraudsters. To make better models and have more concrete predictions, we would require more data, because the fraudsters.csv file contains only around 300 data points.

Following are 5 user_ids(outside the fraudsters.csv dataset) that are potential fraudsters. I have also scanned the entire csv to find all possible fraudsters that the model is confident about. The same is attached as a txt file(named as Frauds_2.txt) for your reference.

User_id: ada797f9-2e13-4cad-934f-00e861b5ebd8
User_id: d966bbc8-4f4a-477b-b00e-034aec8004f1
User_id: a2e41203-a8f5-4744-97ca-dfb04479cde9
User_id: 0ab0a2d3-f7dd-4ad3-acbf-badebeb0c69d
User_id: 4d612d5b-b627-4084-a734-777f64f8a876

From the entire csv file of 10,300 entries, the algorithm classified 1318 users as potential fraudsters. Fraud classification rate is around 12.796%. (Will change with a new dataset)