# SP25: DATA - 228 Sec -12
# Big Data Technologies and Applications
# Homework -3

**Name -** Manav Anandani
**SJSU ID :** 018194917

**Mandatory question:**

**Did you find or come across solutions to similar problems by using Generative AI or other sources?**

**Ans -** No, didn't use any LLMs to generate the answers for the assignments.

**Problem Definition :**

For this single exercise homework, you're tasked with developing a real-time credit card fraud detection system using Apache Kafka, Kafka Streams (KSQLDB), MongoDB, and Flink. This system will ingest simulated (or real – your choice) credit card transaction data, analyze it for suspicious patterns, and generate alerts in real time.

**Step 1:**

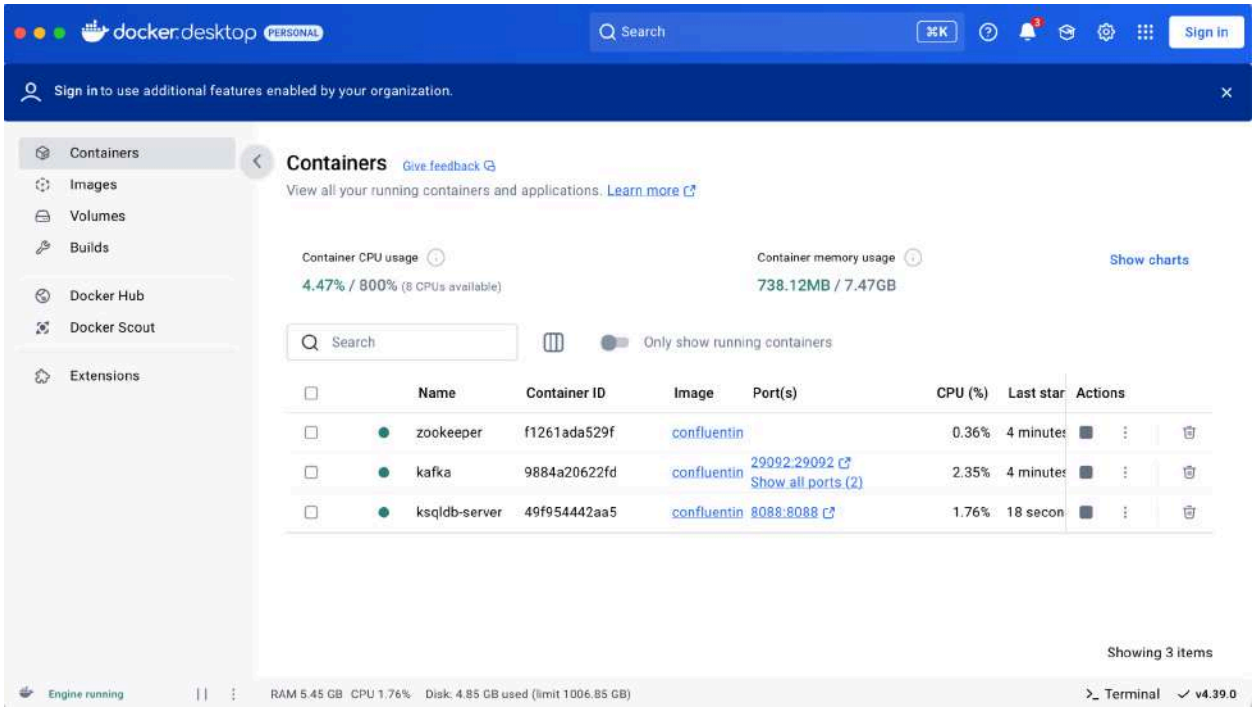
**Set up the Environment:**

Here i manually stelled up the entire environment all the libraries and functions which were required by this assignment here i installed all the dependencies :

Used : *pip install kafka-python confluent-kafka ksql kafka-streams faker pyflink pymongo*

```
spartan — -zsh — 98×30
(base) spartan@MLK-SCS-FVFHW4ULQ05N ~ % pip install kafka-python confluent-kafka ksql faker pyflin
k pymongo

Collecting kafka-python
  Using cached kafka_python-2.0.6-py2.py3-none-any.whl.metadata (9.0 kB)
Collecting confluent-kafka
  Using cached confluent_kafka-2.8.2-cp312-cp312-macosx_11_0_arm64.whl.metadata (22 kB)
Collecting ksql
  Using cached ksql-0.10.2.tar.gz (15 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: faker in /opt/anaconda3/lib/python3.12/site-packages (19.13.0)
Collecting pyflink
  Downloading pyflink-1.0-py3-none-any.whl.metadata (239 bytes)
Collecting pymongo
  Downloading pymongo-4.11.2-cp312-cp312-macosx_11_0_arm64.whl.metadata (22 kB)
Requirement already satisfied: requests in /opt/anaconda3/lib/python3.12/site-packages (from ksql)
 (2.32.3)
Requirement already satisfied: six in /opt/anaconda3/lib/python3.12/site-packages (from ksql) (1.1
6.0)
Requirement already satisfied: urllib3 in /opt/anaconda3/lib/python3.12/site-packages (from ksql)
(2.2.3)
Collecting hyper (from ksql)
  Downloading hyper-0.7.0-py2.py3-none-any.whl.metadata (15 kB)
Requirement already satisfied: python-dateutil>=2.4 in /opt/anaconda3/lib/python3.12/site-packages
 (from faker) (2.9.0.post0)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.7.0-py3-none-any.whl.metadata (5.8 kB)
Collecting h2<3.0,>=2.4 (from hyper->ksql)
  Downloading h2-2.6.2-py2.py3-none-any.whl.metadata (27 kB)
Collecting hyperframe<4.0,>=3.2 (from hyper->ksql)
```

I personally used docker for setting up the entire tools which are required by the project here i created containers for kafka, zookeeper, ksqldb and kafka connect



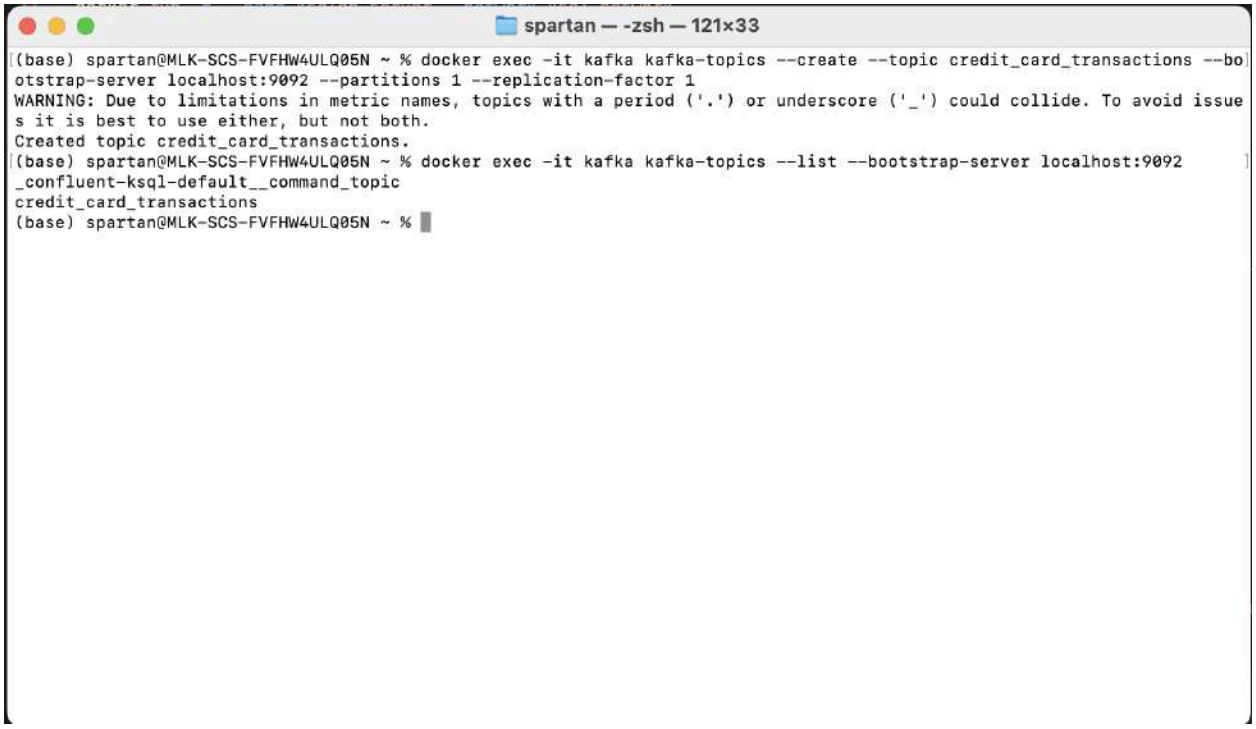
Here we see that all the containers are up and running and we have installed all the required dependencies here which even include java and docker.

**Creation of Kafka topics**

Here using the docker container for kafka we were able to create the kafka topic named **credit_card_transactions**

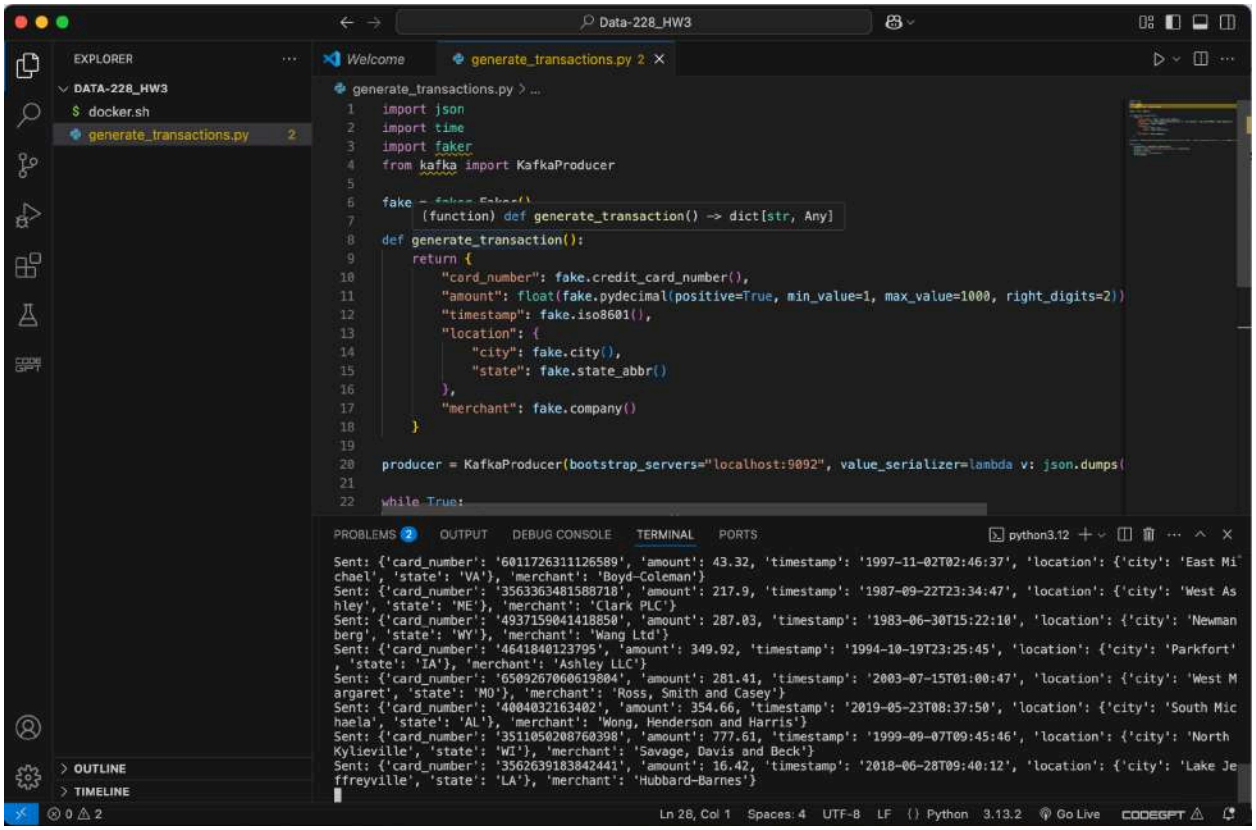

Verification of the topics created

Now we will verify the created kafka topic which is **credit_card_transactions**:




**Step 2:**

**Generate Data Simulated using the Faker python package**




Step 3:

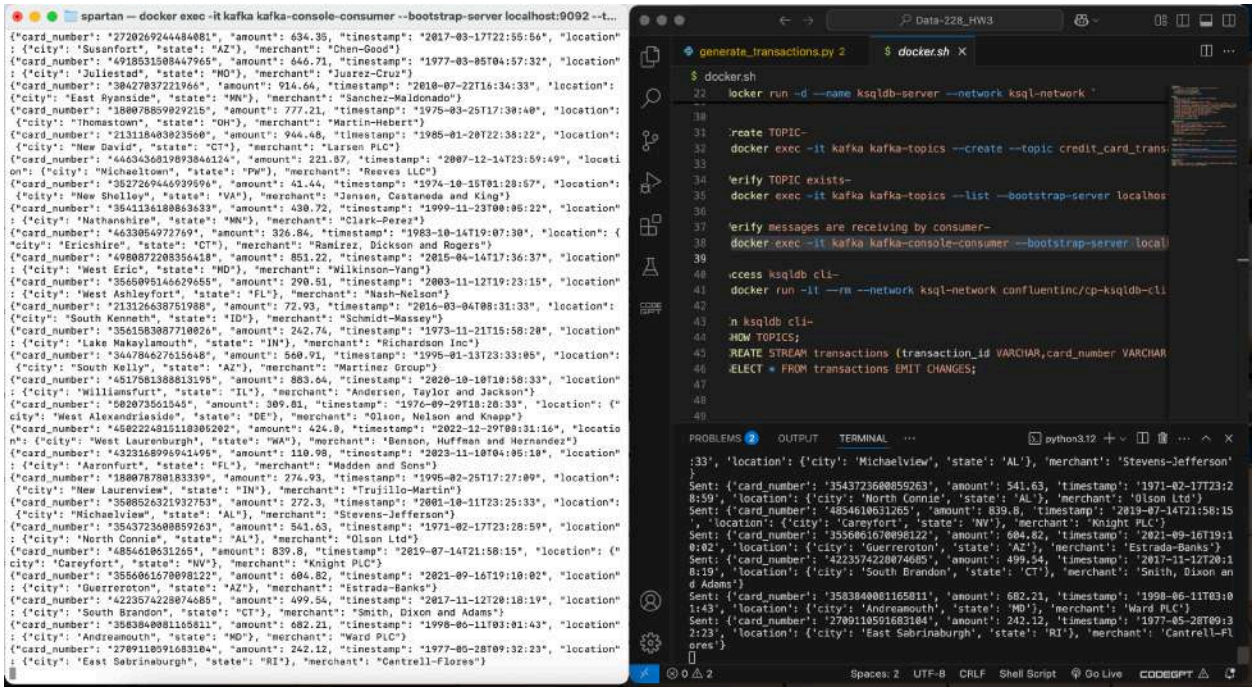
**Simulate transactions and send them to Kafka**

Verification that the message are received by consumer in kafka

spartan — docker exec -it kafka kafka-console-consumer --bootstrap-server localhost:9092 --topic credit_card_trans...

```
(base) spartan@MLK-SCS-FVFHW4ULQ05N ~ % docker exec -it kafka kafka-console-consumer --bootstrap-server localhost:9092 --
topic credit_card_transactions --from-beginning

{"card_number": "6011894551123607", "amount": 342.57, "timestamp": "1975-03-02T19:01:43", "location": {"city": "West Chri
stopher", "state": "AZ"}, "merchant": "Jones and Sons"}
{"card_number": "4514301638447", "amount": 397.53, "timestamp": "2001-11-15T08:21:22", "location": {"city": "New Judyburg
h", "state": "NM"}, "merchant": "Harvey Group"}
{"card_number": "38768369668239", "amount": 786.21, "timestamp": "2002-11-13T22:53:42", "location": {"city": "Nicoleville
", "state": "MO"}, "merchant": "Shannon Group"}
{"card_number": "565760482044", "amount": 926.88, "timestamp": "1990-04-14T00:32:25", "location": {"city": "New Darrensta
d", "state": "RI"}, "merchant": "Smith, Brown and Haynes"}
{"card_number": "3508190301146924", "amount": 976.71, "timestamp": "2021-08-10T15:43:04", "location": {"city": "Palmerche
ster", "state": "AZ"}, "merchant": "Olson and Sons"}
{"card_number": "3580116946922974", "amount": 325.77, "timestamp": "2010-05-13T17:34:44", "location": {"city": "Lake Kris
tin", "state": "CA"}, "merchant": "Jackson-Lawrence"}
{"card_number": "5398972959522284", "amount": 471.41, "timestamp": "1974-03-08T02:07:11", "location": {"city": "Jessicato
wn", "state": "VA"}, "merchant": "Young-Cannon"}
{"card_number": "4692160257275", "amount": 322.56, "timestamp": "1989-09-29T14:16:23", "location": {"city": "Port Elizabe
th", "state": "CO"}, "merchant": "Dean-Sanchez"}
{"card_number": "3589253873880841", "amount": 930.36, "timestamp": "2013-07-03T16:44:53", "location": {"city": "Lake Case
yfurt", "state": "KY"}, "merchant": "Williams, Valdez and Torres"}
{"card_number": "180028263494786", "amount": 199.97, "timestamp": "1988-12-23T03:50:24", "location": {"city": "New Charle
ston", "state": "WI"}, "merchant": "Khan and Sons"}
{"card_number": "6011688461666097", "amount": 445.12, "timestamp": "2017-12-17T17:55:41", "location": {"city": "Elizabeth
furt", "state": "CO"}, "merchant": "Kennedy PLC"}
{"card_number": "180084926118775", "amount": 244.42, "timestamp": "1973-10-14T06:03:14", "location": {"city": "Smithberg"
, "state": "PR"}, "merchant": "Cain, Ferguson and Griffin"}
{"card_number": "3584330188640600", "amount": 186.99, "timestamp": "1999-07-08T22:47:42", "location": {"city": "Kevinview
", "state": "NE"}, "merchant": "Burke and Sons"}
{"card_number": "4564083468634684", "amount": 984.25, "timestamp": "2022-09-09T07:07:25", "location": {"city": "East Tony
", "state": "NY"}, "merchant": "Rice Inc"}
{"card_number": "4367117201219", "amount": 570.45, "timestamp": "1984-05-29T03:42:06", "location": {"city": "New Kevinbur
y", "state": "NJ"}, "merchant": "Thomas-Lee"}
```

Now here we can see that messages are generated in real time from faker and are received by kafka consumer in real time

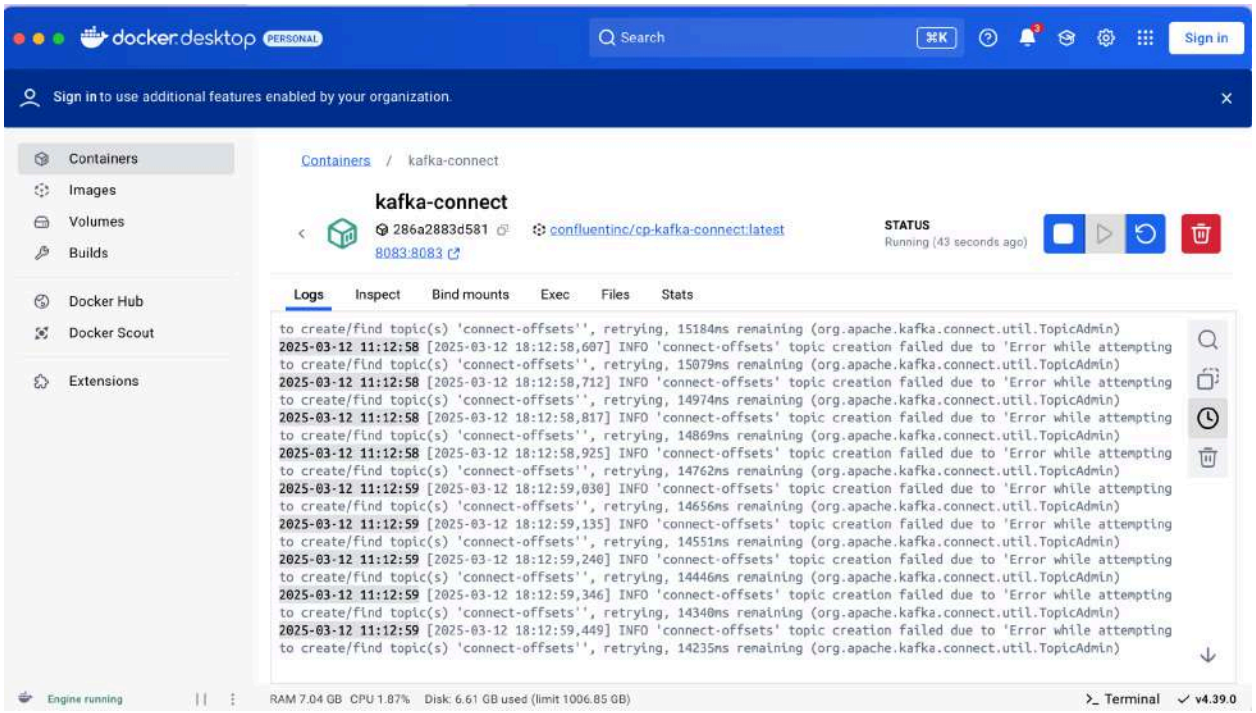



**Step 4:**

**Real-time Fraud Detection with KSQLDB:**

Starting and working with Kafka connect:

1. Configuring kafka connect using docker:

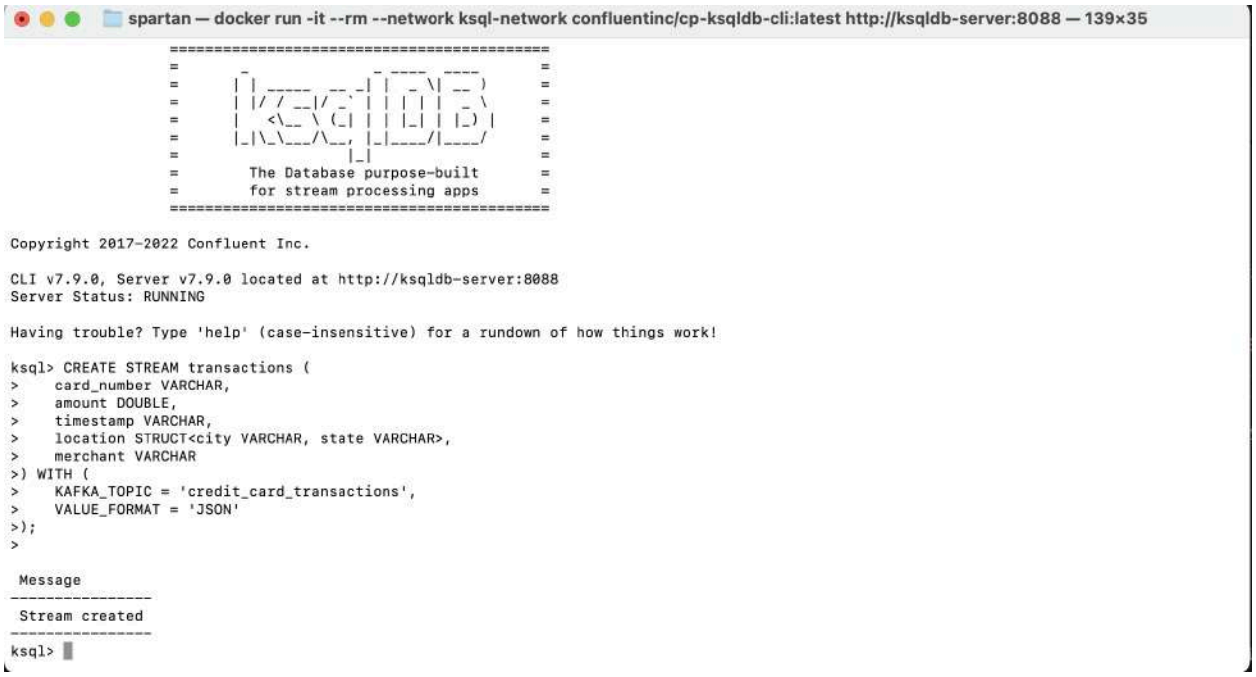



Here we used kafka connect using docker and here we see this is up and running and now and here we will use this to transfer the stream data from Kafka to ksqlDB

Setting up and installing of ksqlDB:

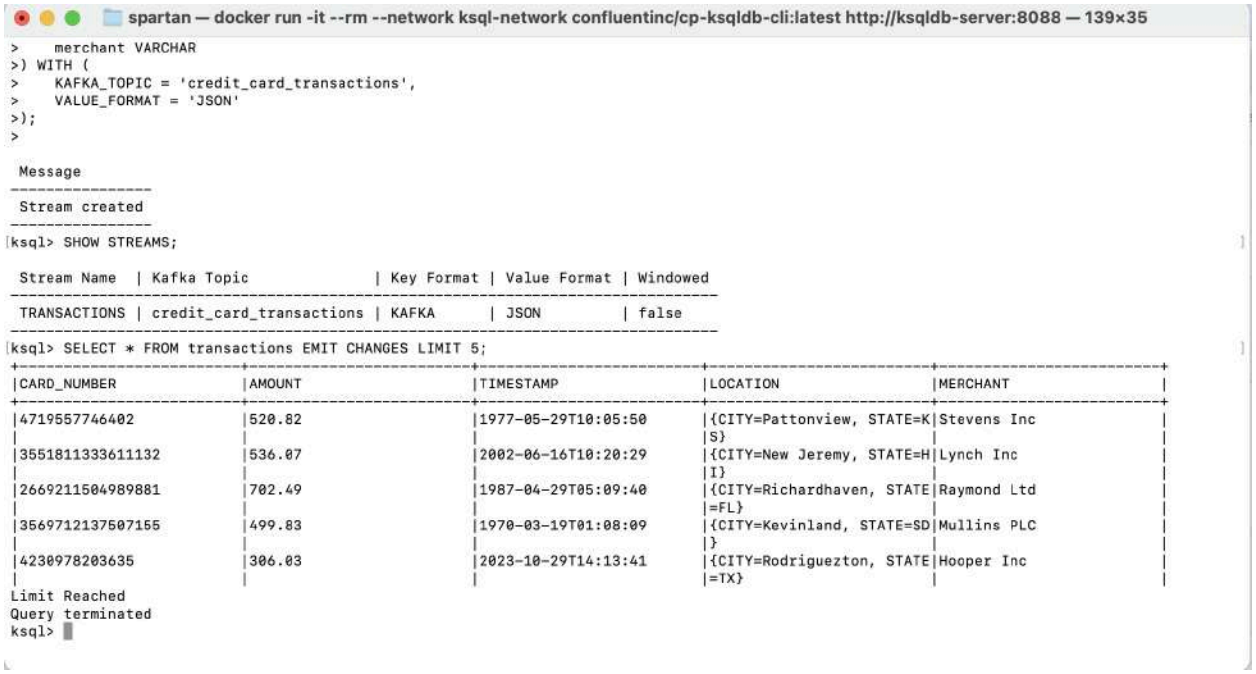

Now here we are inside the ksqlDB and here on we are going to perform various queries on this:

- Create a ksqlDB stream to consume the transactions received from kafka




Here we created a ksql stream which will allow us to query the consumed data from kafka

- Now we will do the verification of the stream created and view the incoming data from kafka.




**KSQL UDF :**

Here we defined the user defined function to classify the fraud detection with my own rules here:

**Rules for fraud detection :**

1. Any transactions exceeding over 500 will be detected as fraud
2. Transactions happening in an unusual locations are also flagged as fraud

    - Create a new kafka topic named as flagged transactions which would only contain the flagged transactions data in it:


This shows that here we have successfully created a flagged_transactions topic which will help us to identify the flagged transactions here.

- Verification of our UDF and the fraud detection system that we created it in a correct manner or not:

```
ksql> SHOW TOPICS;
>

 Kafka Topic              | Partitions | Partition Replicas
-------------------------------------------------------------
 FLAGGED_TRANSACTIONS     | 1          | 1
 credit_card_transactions | 1          | 1
-------------------------------------------------------------
ksql> SELECT * FROM flagged_transactions EMIT CHANGES LIMIT 10;
>
+--------------------------------+--------------------------------+--------------------------------+--------------------------------+
|CARD_NUMBER                     |AMOUNT                          |TIMESTAMP                       |LOCATION                        |
+--------------------------------+--------------------------------+--------------------------------+--------------------------------+
|4212830592736                   |892.28                          |1991-10-10T08:27:27             |{CITY=Evansmouth, STATE=AR}     |
|4864538114286                   |559.95                          |2023-06-23T14:01:32             |{CITY=Kevinside, STATE=GU}      |
|4496316271529292080             |601.49                          |1998-11-15T14:14:31             |{CITY=Jasonmouth, STATE=NC}     |
|6530601653837204                |833.68                          |2017-07-01T17:57:07             |{CITY=Port Justin, STATE=VT}    |
|3521417887589901                |840.61                          |2022-07-29T21:37:47             |{CITY=South Markmouth, STATE=MA}|
|2720895056282203                |574.36                          |1997-09-25T03:59:01             |{CITY=South Karen, STATE=NY}    |
|3522536404818110                |728.83                          |1993-08-01T12:19:57             |{CITY=South Terryside, STATE=UT}|
|3572763632103441                |679.12                          |1988-12-26T03:16:25             |{CITY=North Kennethside, STATE=M|
|                                |                                |                                |A}                              |
|4031728309948595                |804.96                          |1970-12-14T09:43:24             |{CITY=West Christopher, STATE=MP|
|                                |                                |                                |}                               |
|213122602102373                 |692.33                          |2011-10-21T16:10:59             |{CITY=Georgemouth, STATE=FM}    |
Limit Reached
Query terminated
```

Now here we can see that the real time fraud data is being generated from our system in real time this table shows the verification of our fraud detection system in ksqlDB and we can perform various queries and analysis on the very same data.

- Create a new ksqldb stream to detect the fraud transactions

  Creation and verification of streams :

```
ksql> Show streams;

 Stream Name         | Kafka Topic              | Key Format | Value Format | Windowed
-------------------------------------------------------------------------------------
 FLAG_TRANSACTIONS   | flag_transactions        | KAFKA      | JSON         | false
 HIGH_RISK_MERCHANTS | HIGH_RISK_MERCHANTS      | KAFKA      | JSON         | false
 TRANSACTIONS        | credit_card_transactions | KAFKA      | JSON         | false
-------------------------------------------------------------------------------------
```

Now i have created all the streams and all the topics which are required now we will perform our analysis on the real credit card transactions data using transactions stream and on flagged data using the flag_transactions stream.

```
ksql> SHOW TOPICS;
>

 Kafka Topic              | Partitions | Partition Replicas
-------------------------------------------------------------
 FLAGGED_TRANSACTIONS     | 1          | 1
 credit_card_transactions | 1          | 1
-------------------------------------------------------------
ksql> SELECT * FROM flagged_transactions EMIT CHANGES LIMIT 10;
>
+--------------------------------+--------------------------------+--------------------------------+--------------------------------+
|CARD_NUMBER                     |AMOUNT                          |TIMESTAMP                       |LOCATION                        |
+--------------------------------+--------------------------------+--------------------------------+--------------------------------+
|4212830592736                   |892.28                          |1991-10-10T08:27:27             |{CITY=Evansmouth, STATE=AR}     |
|4864538114286                   |559.95                          |2023-06-23T14:01:32             |{CITY=Kevinside, STATE=GU}      |
|4496316271529292080             |601.49                          |1998-11-15T14:14:31             |{CITY=Jasonmouth, STATE=NC}     |
|6530601653837204                |833.68                          |2017-07-01T17:57:07             |{CITY=Port Justin, STATE=VT}    |
|3521417887589901                |840.61                          |2022-07-29T21:37:47             |{CITY=South Markmouth, STATE=MA}|
|2720895056282203                |574.36                          |1997-09-25T03:59:01             |{CITY=South Karen, STATE=NY}    |
|3522536404818110                |728.83                          |1993-08-01T12:19:57             |{CITY=South Terryside, STATE=UT}|
|3572763632103441                |679.12                          |1988-12-26T03:16:25             |{CITY=North Kennethside, STATE=M|
|                                |                                |                                |A}                              |
|4031728309948595                |804.96                          |1970-12-14T09:43:24             |{CITY=West Christopher, STATE=MP|
|                                |                                |                                |}                               |
|213122602102373                 |692.33                          |2011-10-21T16:10:59             |{CITY=Georgemouth, STATE=FM}    |
Limit Reached
Query terminated
```

## Query and analysis on both streams and topics:

1. **Flagging transactions with high risk merchants if transactions occur at fraud prone merchants**

```
ksql> CREATE STREAM high_risk_merchants AS
>SELECT * FROM transactions
>WHERE merchant IN ('ScamCorp', 'FraudTech', 'ShadyBiz');

 Message
----------------------------------------------------
 Created query with ID CSAS_HIGH_RISK_MERCHANTS_3
----------------------------------------------------
ksql>
```

2. **Query using tumbling window**

```
ksql> SELECT card_number, COUNT(*) AS txn_count
>FROM transactions
>WINDOW TUMBLING (SIZE 10 MINUTES)
>GROUP BY card_number
>EMIT CHANGES;
+-----------------------------------------------------------------------+-----------------------------------------------------------------------+
|CARD_NUMBER                                                            |TXN_COUNT                                                              |
+-----------------------------------------------------------------------+-----------------------------------------------------------------------+
|180044116617390                                                        |1                                                                      |
|4432759808142823                                                       |1                                                                      |
|373557418074322                                                        |1                                                                      |
|3500857371720774                                                       |1                                                                      |
|503802807415                                                           |1                                                                      |
|30152148983623                                                         |1                                                                      |
|4214403765160568667                                                    |1                                                                      |
|676306583987                                                           |1                                                                      |
|4013074380364572                                                       |1                                                                      |
|4198263463950                                                          |1                                                                      |
|3580756536538902                                                       |1                                                                      |
|2709052543224092                                                       |1                                                                      |
|3557339998772712                                                       |1                                                                      |
|6011061451253385                                                       |1                                                                      |
|30404529548156                                                         |1                                                                      |
|36000024018507                                                         |1                                                                      |
|3565266993554854                                                       |1                                                                      |
|30329078031207                                                         |1                                                                      |
|4950157915010                                                          |1                                                                      |
|4532006619964782                                                       |1                                                                      |
|6011537616233388                                                       |1                                                                      |
^CQuery terminated
ksql>
```

3. Transactions with particular card number:

```
ksql>  SELECT * FROM transactions
>WHERE card_number = '30160354313765';
+-------------------------+-------------------------+-------------------------+-------------------------+-------------------------+
|CARD_NUMBER              |AMOUNT                   |TIMESTAMP                |LOCATION                 |MERCHANT                 |
+-------------------------+-------------------------+-------------------------+-------------------------+-------------------------+
|30160354313765           |355.75                   |1985-07-25T17:37:23      |{CITY=North Cassieborough|Campbell LLC             |
|                         |                         |                         |, STATE=OR}              |                         |
Query Completed
Query terminated
ksql>
```

4. Average transactions amount by account number in flag transactions

```
ksql> SELECT
>    CARD_NUMBER,
>    AVG(AMOUNT) AS AVG_TRANSACTION_AMOUNT
>FROM FLAG_TRANSACTIONS
>GROUP BY CARD_NUMBER
>EMIT CHANGES;
+-----------------------------------------------------------------------------------+-----------------------------------------------------------------------------------+
|CARD_NUMBER                                                                        |AVG_TRANSACTION_AMOUNT                                                             |
+-----------------------------------------------------------------------------------+-----------------------------------------------------------------------------------+
|2272462528559031                                                                   |901.85                                                                             |
|180059935313161                                                                    |851.73                                                                             |
|4424791802021299                                                                   |901.34                                                                             |
|213126215043213                                                                    |955.38                                                                             |
^CQuery terminated
ksql>
```

5. Counting transactions per location

```
ksql> SELECT
>    LOCATION->CITY AS CITY,
>    LOCATION->STATE AS STATE,
>    COUNT(*) AS TRANSACTION_COUNT
>FROM FLAG_TRANSACTIONS
>GROUP BY LOCATION->CITY, LOCATION->STATE
>EMIT CHANGES;
>
+-----------------------------------------------------------------------------------+-----------------------------------------------------------------------------------+-----------------------------------------------------------------------------------+
|CITY                                                                               |STATE                                                                              |TRANSACTION_COUNT                                                                  |
+-----------------------------------------------------------------------------------+-----------------------------------------------------------------------------------+-----------------------------------------------------------------------------------+
|Cynthiaview                                                                        |UT                                                                                 |1                                                                                  |
|Whitneyville                                                                       |AZ                                                                                 |1                                                                                  |
|South Susanhaven                                                                   |ID                                                                                 |1                                                                                  |
|Michelleport                                                                       |VA                                                                                 |1                                                                                  |
|East Joseph                                                                        |MI                                                                                 |1                                                                                  |
|Russellmouth                                                                       |CO                                                                                 |1                                                                                  |
|West Stevehaven                                                                    |FL                                                                                 |1                                                                                  |
|North Meganview                                                                    |DE                                                                                 |1                                                                                  |
|North Zacharyland                                                                  |FM                                                                                 |1                                                                                  |
|South Dave                                                                         |LA                                                                                 |1                                                                                  |
|Lake Sara                                                                          |GU                                                                                 |1                                                                                  |
^CQuery terminated
ksql>
```

**Step 5:**

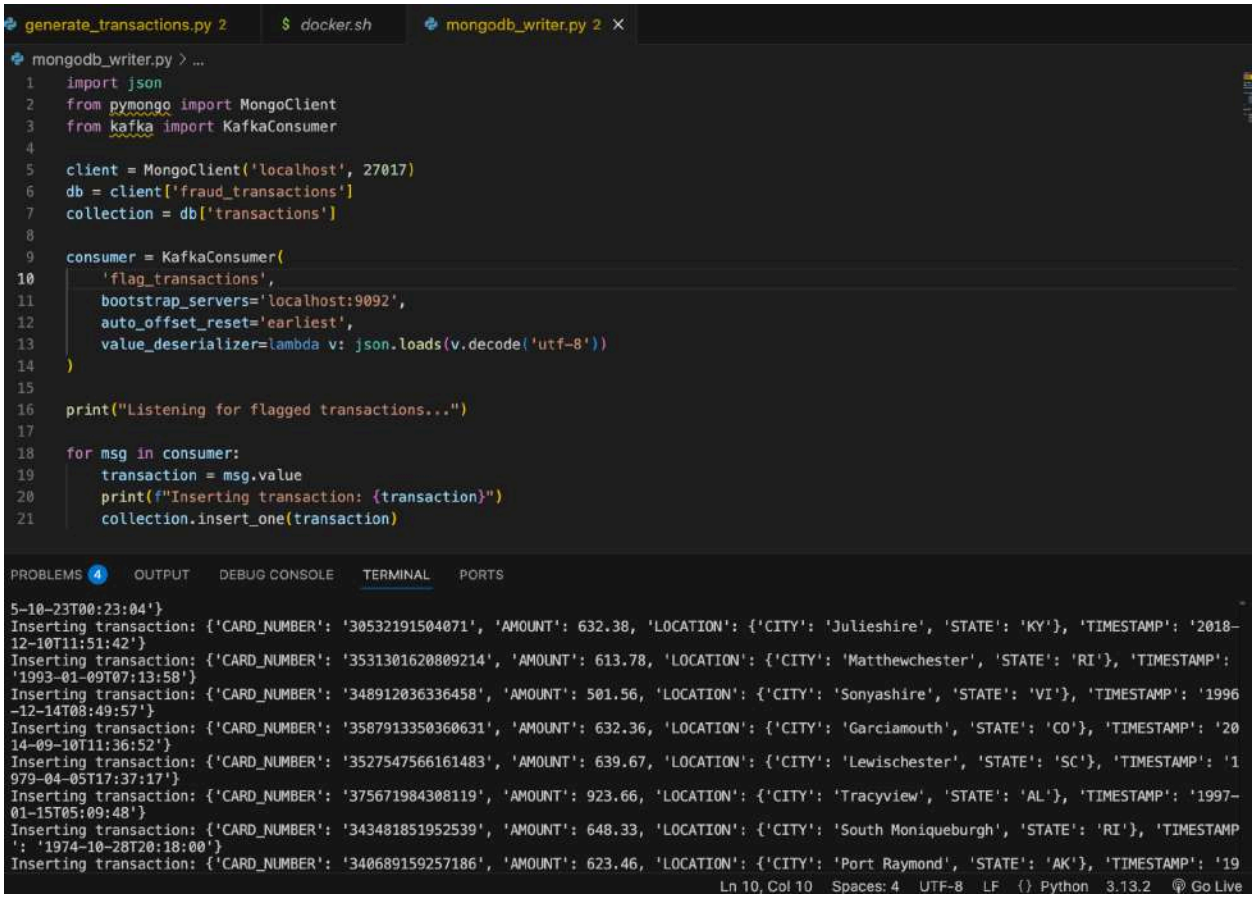
**Store Data in MongoDB:**

For storing the data in mongo first of all:

1. We started the ming db server and check mongodb is correctly tunning on our system.
2. We created a file called **mongodb_writer.py** which contains the logic fromm which the flagged data will be stored in mongodb by using connection from kafka and ksqldb

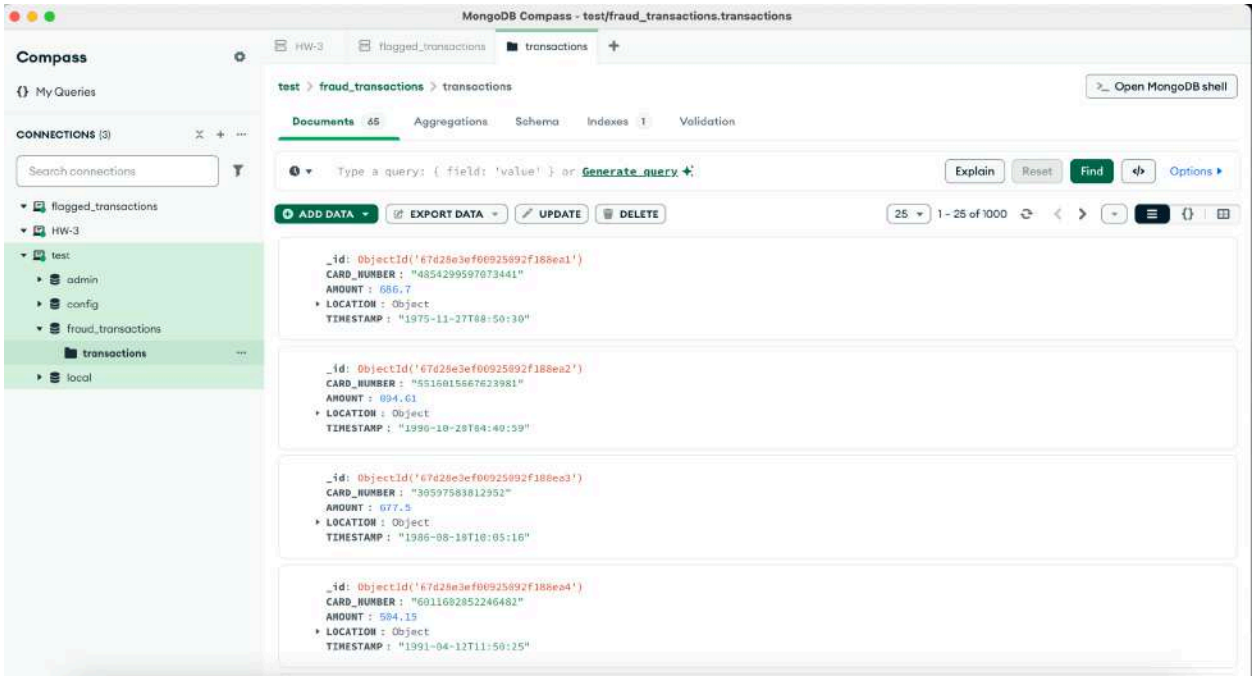
```
KeyboardInterrupt

(base) spartan@MLK-SCS-FVFHW4ULQ05N Data-228_HW3 % python mongodb_writer.py
Listening for flagged transactions...
```

Ln 10, Col 10 Spaces: 4 UTF-8 LF {} Python 3.13.2 Go Live




Now here we see that mongo db is listening and presenting the flagged transitions all in one:

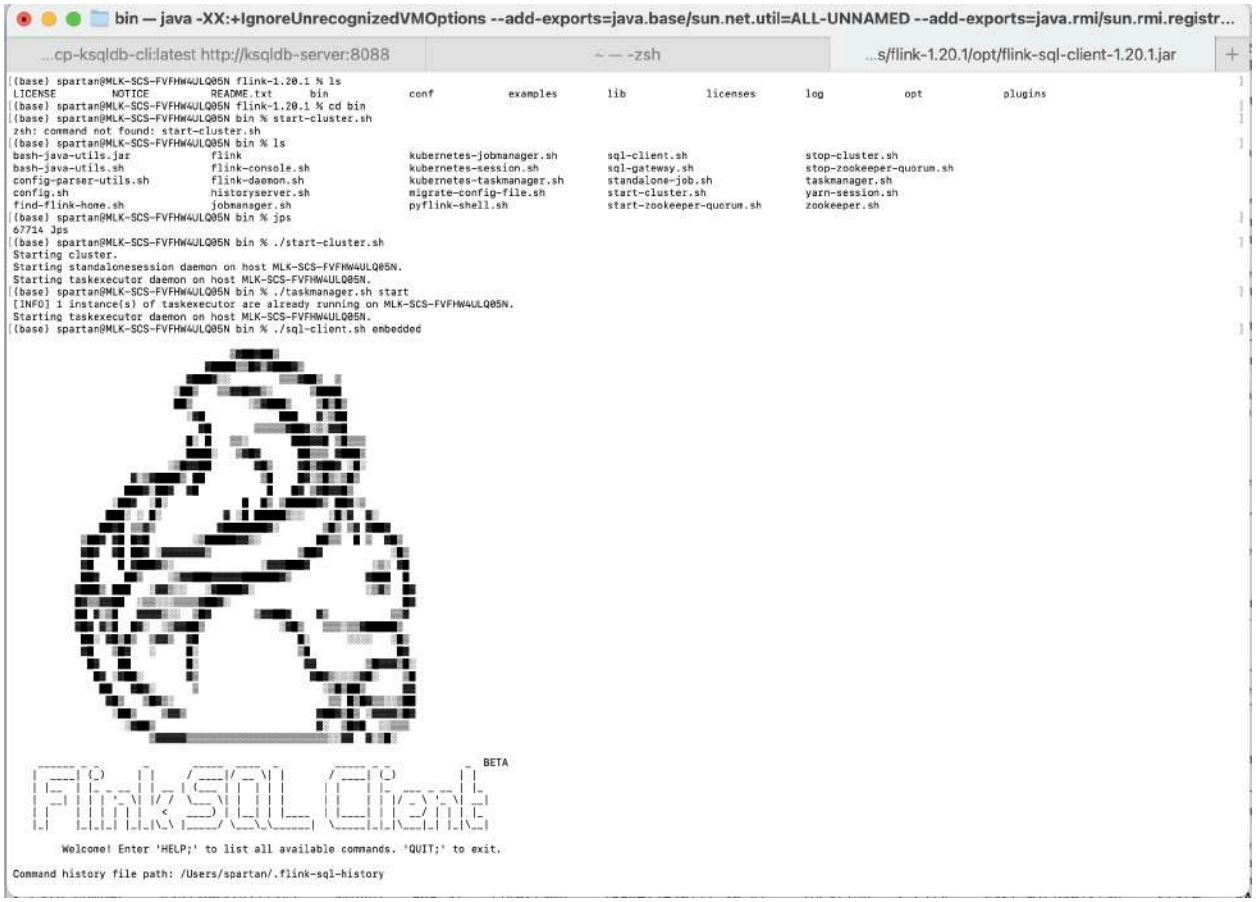
Now storing the data in mongodb



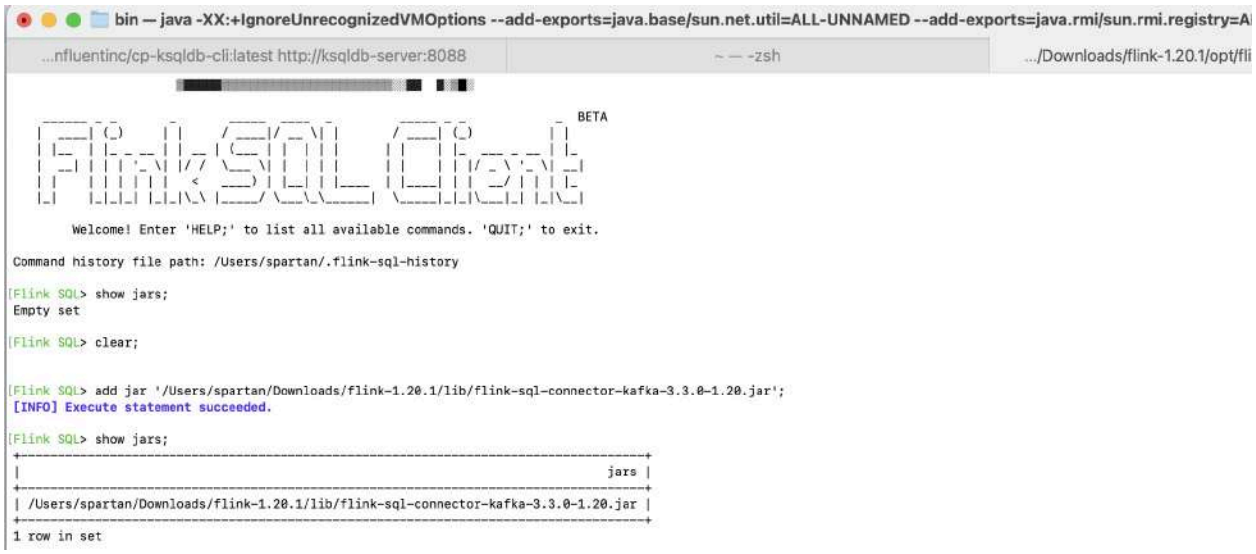

**Step 6:**

**Real-time Analytics with Apache Flink:**

Installing apache Flink on the system

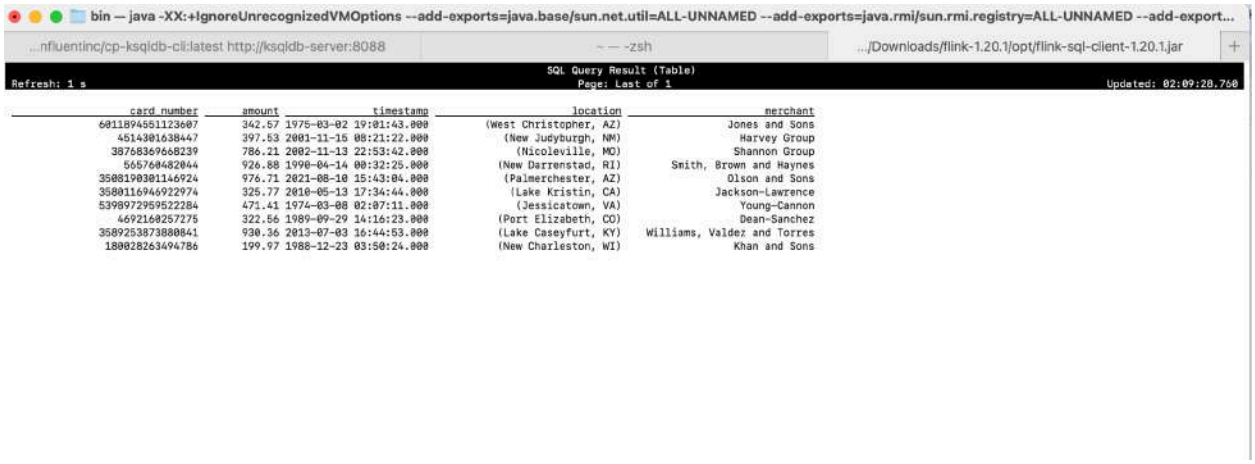



Installing apache flink and creation of jars:



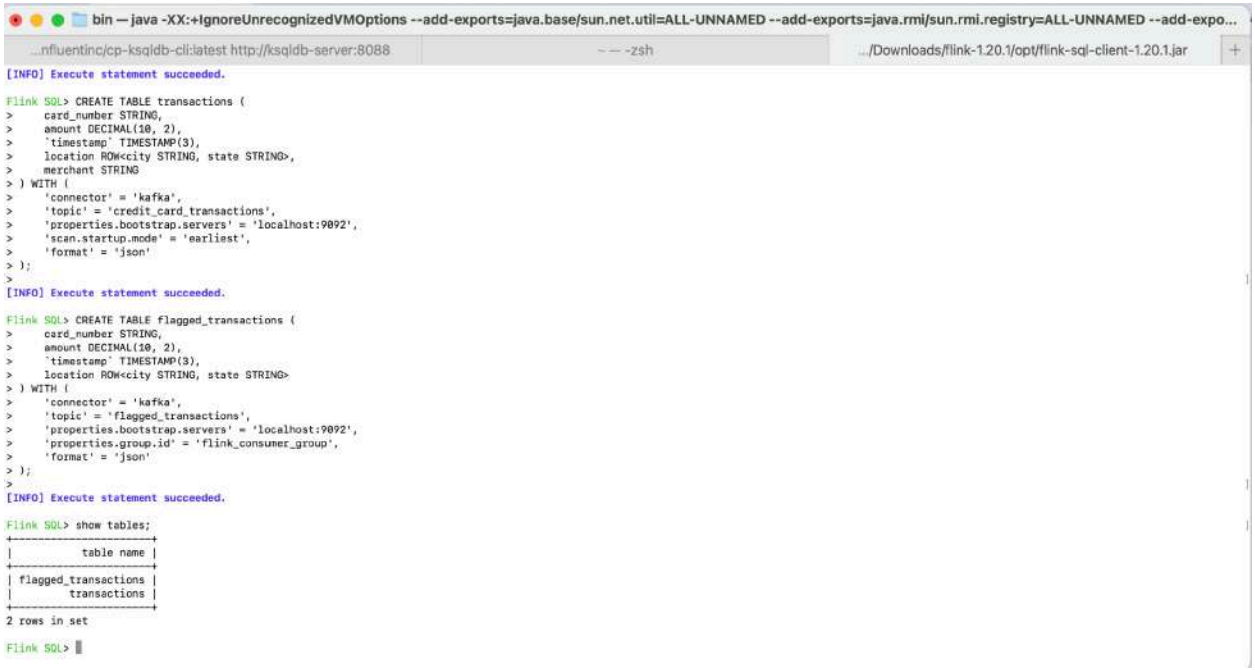

Here the apache flink is running here and we have created a jar to run our system here:

**Data generation on apache flink:**




Here we see that fraud data is correctly generated in apache flink here in real time from ksqldb

Table creations :




Performing and analysis on apache flink:

Querying on **transactions data** in flink:

```
Flink SQL> SELECT
>     TUMBLE_START(`timestamp`, INTERVAL '10' SECOND) AS window_start,
>     TUMBLE_END(`timestamp`, INTERVAL '10' SECOND) AS window_end,
>     COUNT(*) AS transaction_count,
>     SUM(amount) AS total_amount,
>     MIN(amount) AS min_amount,
>     MAX(amount) AS max_amount
> FROM transactions
> GROUP BY TUMBLE(`timestamp`, INTERVAL '10' SECOND);
[INFO] Result retrieval cancelled.

Flink SQL>
```

bin — java -XX:+IgnoreUnrecognizedVMOptions --add-exports=java.base/sun.net.util=ALL-UNNAMED --add-exports=java.rmi/sun.rmi.registry=ALL-UNNAMED --add-export...

...nfluentinc/cp-ksqldb-cli:latest http://ksqldb-server:8088 | ~ — -zsh | .../Downloads/flink-1.20.1/opt/flink-sql-client-1.20.1.jar

SQL Query Result (Table)
Refresh: 1 s | Page: Last of 224 | Updated: 02:13:23.468

| window_start | window_end | transaction_count | total_amount | min_amount | max_amount |
|---|---|---|---|---|---|
| 2025-02-06 07:16:50.000 | 2025-02-06 07:17:00.000 | 1 | 965.60 | 965.60 | 965.60 |
| 2025-02-13 12:28:20.000 | 2025-02-13 12:28:30.000 | 1 | 143.80 | 143.80 | 143.80 |
| 2025-02-13 15:09:40.000 | 2025-02-13 15:09:50.000 | 1 | 885.42 | 885.42 | 885.42 |
| 2025-02-14 21:15:40.000 | 2025-02-14 21:15:50.000 | 1 | 383.36 | 383.36 | 383.36 |
| 2025-02-18 04:54:30.000 | 2025-02-18 04:54:40.000 | 1 | 520.33 | 520.33 | 520.33 |
| 2025-02-18 14:09:00.000 | 2025-02-18 14:09:10.000 | 1 | 657.39 | 657.39 | 657.39 |
| 2025-02-22 03:45:00.000 | 2025-02-22 03:45:10.000 | 1 | 826.71 | 826.71 | 826.71 |
| 2025-02-25 09:54:40.000 | 2025-02-25 09:54:50.000 | 1 | 227.21 | 227.21 | 227.21 |
| 2025-02-28 19:18:20.000 | 2025-02-28 19:18:30.000 | 1 | 277.02 | 277.02 | 277.02 |
| 2025-03-03 17:47:20.000 | 2025-03-03 17:47:30.000 | 1 | 201.78 | 201.78 | 201.78 |
| 2025-03-04 22:01:20.000 | 2025-03-04 22:01:30.000 | 1 | 38.63 | 38.63 | 38.63 |
| 2025-03-09 07:00:30.000 | 2025-03-09 07:00:40.000 | 1 | 917.73 | 917.73 | 917.73 |
| 2025-03-11 10:12:50.000 | 2025-03-11 10:13:00.000 | 1 | 611.39 | 611.39 | 611.39 |

## 2. Count of total transactions

```
>
[INFO] Result retrieval cancelled.

Flink SQL> SELECT COUNT(*) AS total_transactions FROM transactions;
[INFO] Result retrieval cancelled.

Flink SQL>
```

bin — java -XX:+IgnoreUnrecognizedVMOptions --add-exports=java.base/sun.net.util=ALL-UNNAMED --add-exports=java.rmi/sun.rmi.registry=ALL-UNNAMED --add-export...

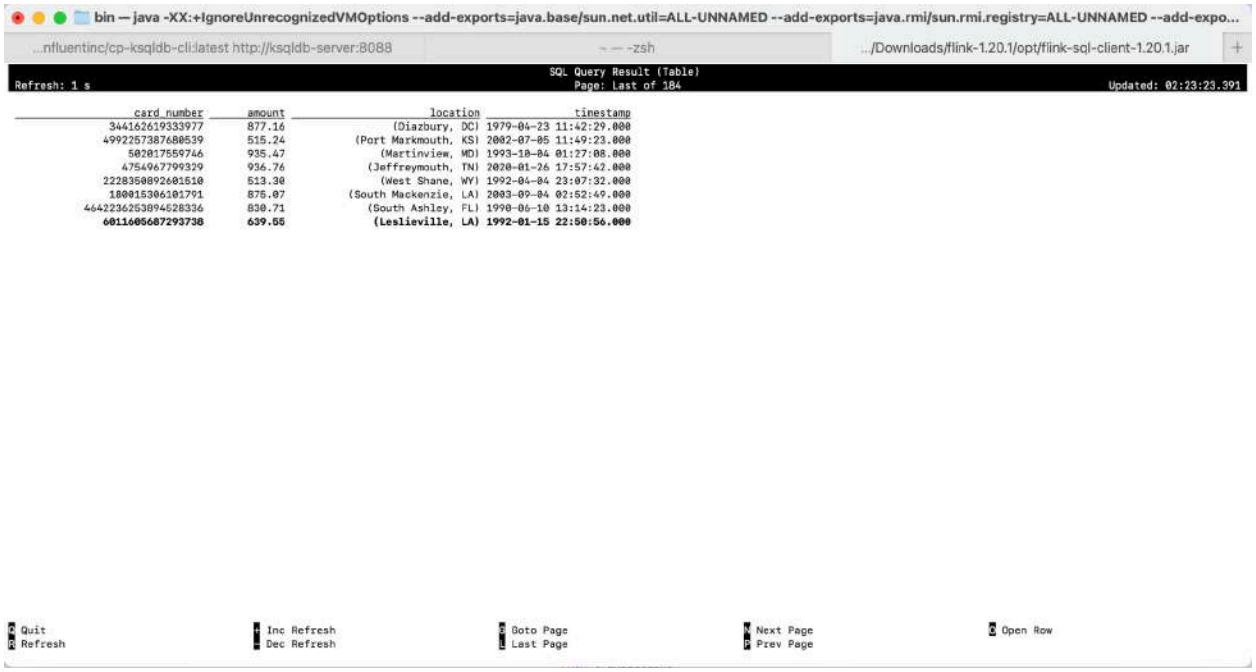
...nfluentinc/cp-ksqldb-cli:latest http://ksqldb-server:8088 | ~ — -zsh | .../Downloads/flink-1.20.1/opt/flink-sql-client-1.20.1.jar

SQL Query Result (Table)
Refresh: 1 s | Page: Last of 1 | Updated: 02:22:23.777

| total_transactions |
|---|
| 13620 |

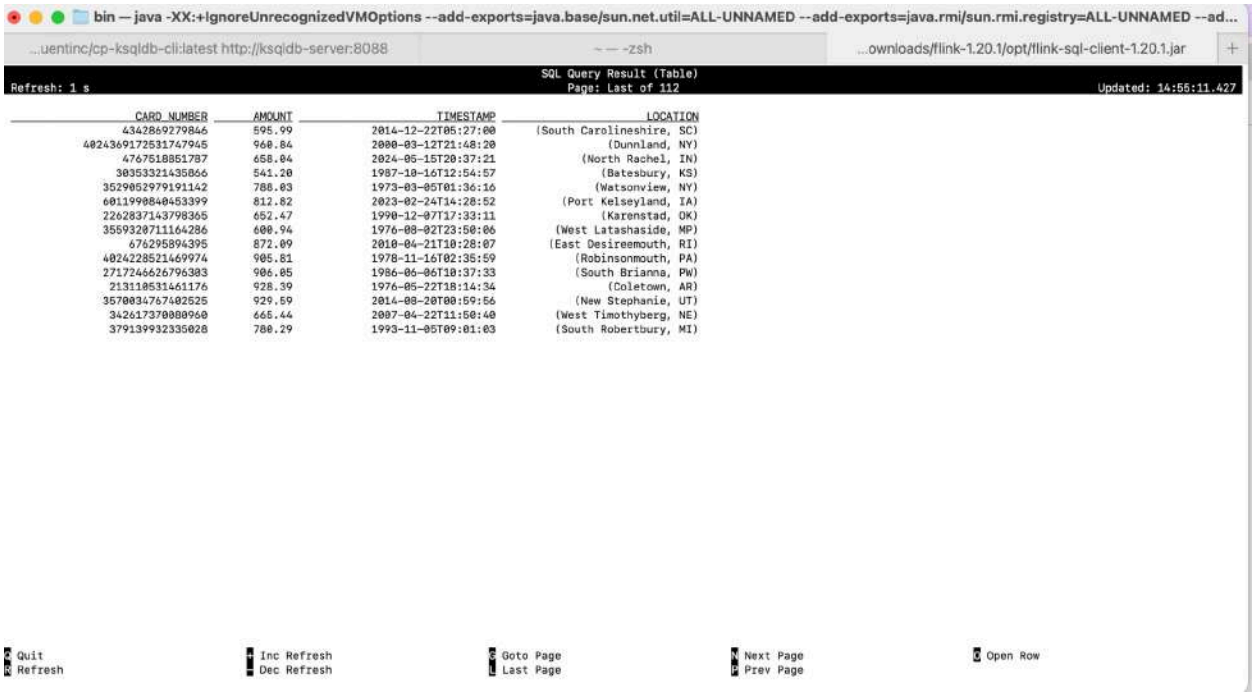
3. Query where transactions are greater than 500

```
Flink SQL> SELECT card_number, amount, location, `timestamp`
> FROM transactions
> WHERE amount > 500;
>
[INFO] Result retrieval cancelled.

Flink SQL>
```




- **Querying on the flagged data in flink**

```
[Flink SQL> drop table FLAG_TRANSACTIONS;
[INFO] Execute statement succeeded.

Flink SQL> CREATE TABLE FLAG_TRANSACTIONS (
>     CARD_NUMBER STRING,
>     AMOUNT DECIMAL(10,2),
>     `TIMESTAMP` STRING,  -- Change to STRING to avoid parsing issues
>     LOCATION ROW<CITY STRING, STATE STRING>
> ) WITH (
>     'connector' = 'kafka',
>     'topic' = 'flag_transactions',
>     'properties.bootstrap.servers' = 'localhost:9092',
>     'properties.group.id' = 'flink_consumer_group',
>     'scan.startup.mode' = 'earliest-offset',
>     'format' = 'json'
> );
[INFO] Execute statement succeeded.

[Flink SQL> SELECT * FROM FLAG_TRANSACTIONS LIMIT 10;
[INFO] Result retrieval cancelled.

Flink SQL>
```

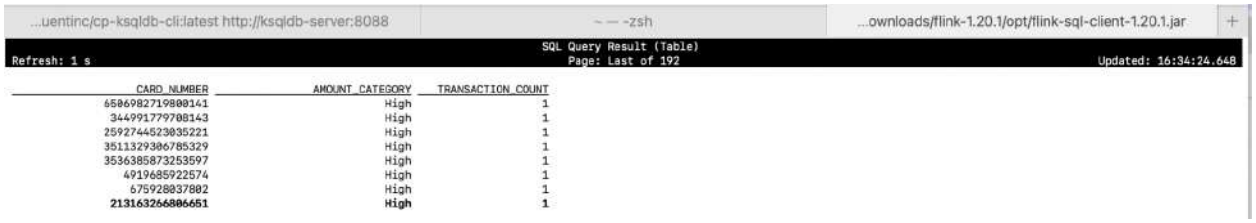



Here the flagged data from flag_ransctions is visible in flink from ksqlDB

- Querying on the data

1. Group transactions based on amount ranges

```
Flink SQL> SELECT
>     CARD_NUMBER,
>     CASE
>         WHEN AMOUNT < 100 THEN 'Low'
>         WHEN AMOUNT BETWEEN 100 AND 500 THEN 'Medium'
>         WHEN AMOUNT BETWEEN 500 AND 1000 THEN 'High'
>         ELSE 'Very High'
>     END AS AMOUNT_CATEGORY,
>     COUNT(*) AS TRANSACTION_COUNT
> FROM FLAG_TRANSACTIONS
> GROUP BY CARD_NUMBER,
>          CASE
>             WHEN AMOUNT < 100 THEN 'Low'
>             WHEN AMOUNT BETWEEN 100 AND 500 THEN 'Medium'
>             WHEN AMOUNT BETWEEN 500 AND 1000 THEN 'High'
>             ELSE 'Very High'
>          END;
>
[INFO] Result retrieval cancelled.

Flink SQL>
```




2. Query to find amount by group by card number

```
Flink SQL> SELECT CARD_NUMBER, SUM(AMOUNT) AS total_amount
> FROM FLAG_TRANSACTIONS
> GROUP BY CARD_NUMBER;
[INFO] Result retrieval cancelled.

Flink SQL>
```

bin — java -XX:+IgnoreUnrecognizedVMOptions --add-exports=java.base/sun.net.util=ALL-UNNAMED --add-exports=java.rmi/sun.rmi.registry=ALL-UNNAMED --ad...

...uentinc/cp-ksqldb-cli:latest http://ksqldb-server:8088 | ~ — -zsh | ...ownloads/flink-1.20.1/opt/flink-sql-client-1.20.1.jar

SQL Query Result (Table)
Refresh: 1 s | Page: Last of 114 | Updated: 14:57:39.697

| CARD_NUMBER | total_amount |
|---|---|
| 3559320711164286 | 600.94 |
| 676295894395 | 872.09 |
| 4024228521469974 | 905.81 |
| 2717246626796303 | 906.05 |
| 213110531461176 | 928.39 |
| 3570034767402525 | 929.59 |
| 342617370080960 | 665.44 |
| 379139932335028 | 780.29 |
| 4937623455694868 | 891.54 |
| 3520804167719624 | 712.51 |
| 401878289446991098 | 667.28 |
| 4594893562130932 | 697.18 |
| 3530943945611634 | 514.43 |
| 4723501461883068232 | 882.23 |
| 3591987286762286 | 581.82 |
| 4108992772256377 | 606.03 |
| 30147377530434 | 768.58 |
| 4961885988954047275 | 727.90 |
| 3581243306825677 | 770.14 |
| 6011406989267656 | 608.08 |
| 566125472571 | 942.06 |
| 586186809990 | 754.19 |
| 676372562741 | 603.17 |
| 3539842778042225 | 542.27 |
| 371503301097774 | 758.07 |
| 30415752520423 | 526.44 |
| 30086145919745 | 649.23 |
| **3514474357874387** | **659.80** |

Q Quit | R Refresh | I Inc Refresh | D Dec Refresh | G Goto Page | L Last Page | N Next Page | P Prev Page | O Open Row

3. Query to find the average amount of flagged transactions from a card

```
Flink SQL> SELECT
>     CARD_NUMBER,
>     SUM(AMOUNT) AS total_amount,
>     AVG(AMOUNT) AS average_amount
> FROM FLAG_TRANSACTIONS
> GROUP BY CARD_NUMBER;
>
[INFO] Result retrieval cancelled.

Flink SQL>
```

bin — java -XX:+IgnoreUnrecognizedVMOptions --add-exports=java.base/sun.net.util=ALL-UNNAMED --add-exports=java.rmi/sun.rmi.registry=ALL-UNNAMED --ad...

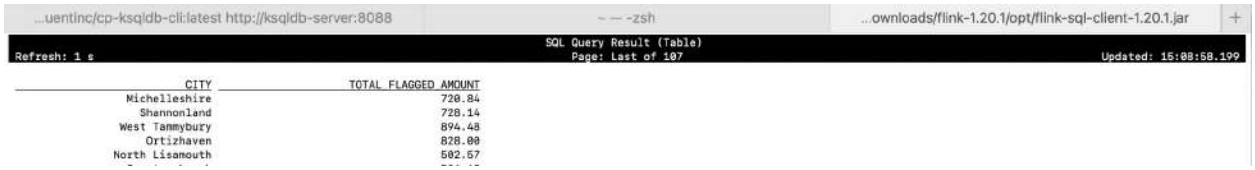
...uentinc/cp-ksqldb-cli:latest http://ksqldb-server:8088 | ~ — -zsh | ...ownloads/flink-1.20.1/opt/flink-sql-client-1.20.1.jar

SQL Query Result (Table)
Refresh: 1 s | Page: Last of 118 | Updated: 15:01:50.737

| CARD_NUMBER | total_amount | average_amount |
|---|---|---|
| 3512797308183975 | 804.47 | 804.470000 |
| 30006231094245 | 794.98 | 794.980000 |
| 2523123593737202 | 548.27 | 548.270000 |
| 213163072209918 | 734.08 | 734.080000 |
| 3502968248393550 | 645.45 | 645.450000 |
| 571759824980 | 613.27 | 613.270000 |
| 30411420503610 | 514.59 | 514.590000 |
| **370466464511216** | **639.75** | **639.750000** |

4. Query to group by flagged transactions by location

```
Flink SQL> SELECT LOCATION.CITY AS CITY, SUM(AMOUNT) AS TOTAL_FLAGGED_AMOUNT
> FROM FLAG_TRANSACTIONS
> GROUP BY LOCATION.CITY;
>
[INFO] Result retrieval cancelled.

Flink SQL>
```




**7. Reflect and share:**

**Summary:**

Students gained practical knowledge of using Kafka alongside MongoDB and Flink for creating a live fraud detection solution through this assignment.

**Challenges:**

**Kafka Integration:** Ensuring smooth communication between producer and consumer. The system efficiently processes massive streams of current data.
**Data Synchronization:** Ensuring data integrity between Kafka and MongoDB.

**Insights:**

Real-time analytics stands essential to prevent fraud from occurring.
The combination of Kafka and MongoDB and Flink provides powerful solutions although their integration process becomes complex.
Performance improvement in systems demands both data validation and asynchronous processing steps.

**Technical Difficulties:**
The application two systems achieved synchronization between Kafka offset management and MongoDB high-throughput performance.

**Best Practices:**

Efficient error handling and data validation.
The system inserts data asynchronously into MongoDB to achieve better performance.

Blog link for more insights and detailed information :

https://github.com/manavanandani/Real_time_Fraud_detection_system

For more detailed information i have shared the link for my blog for reflect and share .