

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*

```

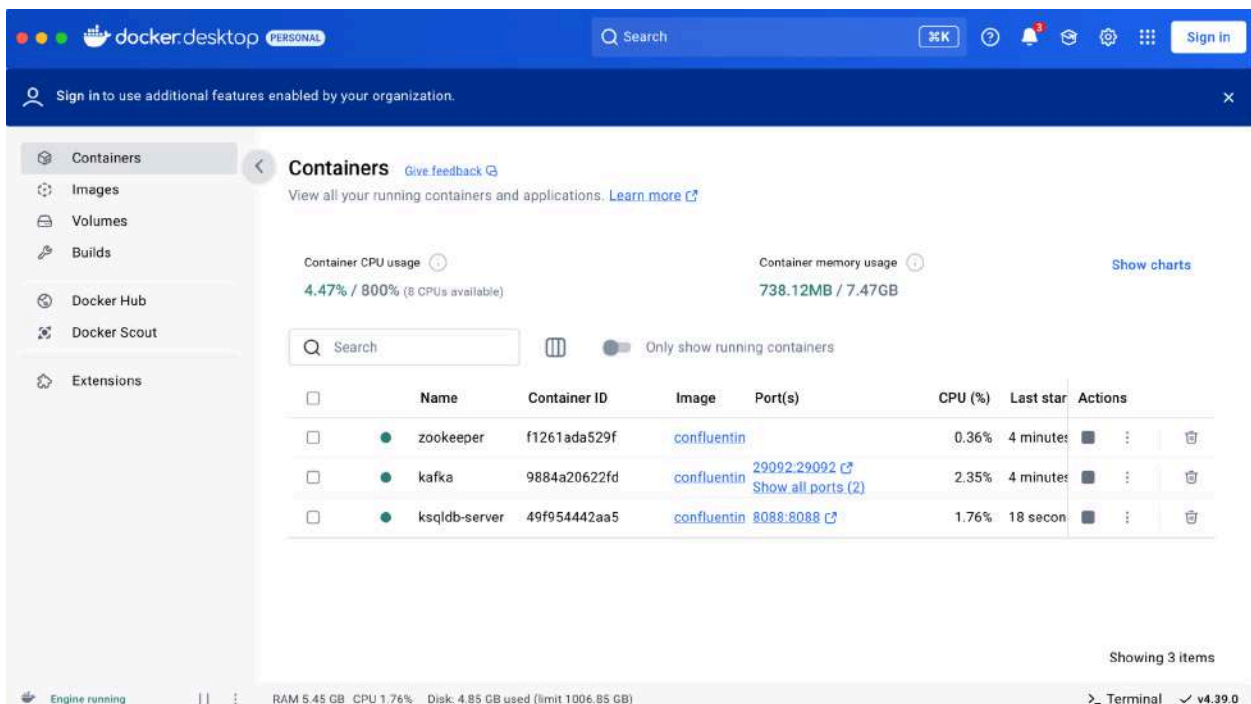
spartan — zsh — 98x30

(base) spartan@MLK-SCS-FVFHW4ULQ05N ~ % pip install kafka-python confluent-kafka ksql faker pyflink
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

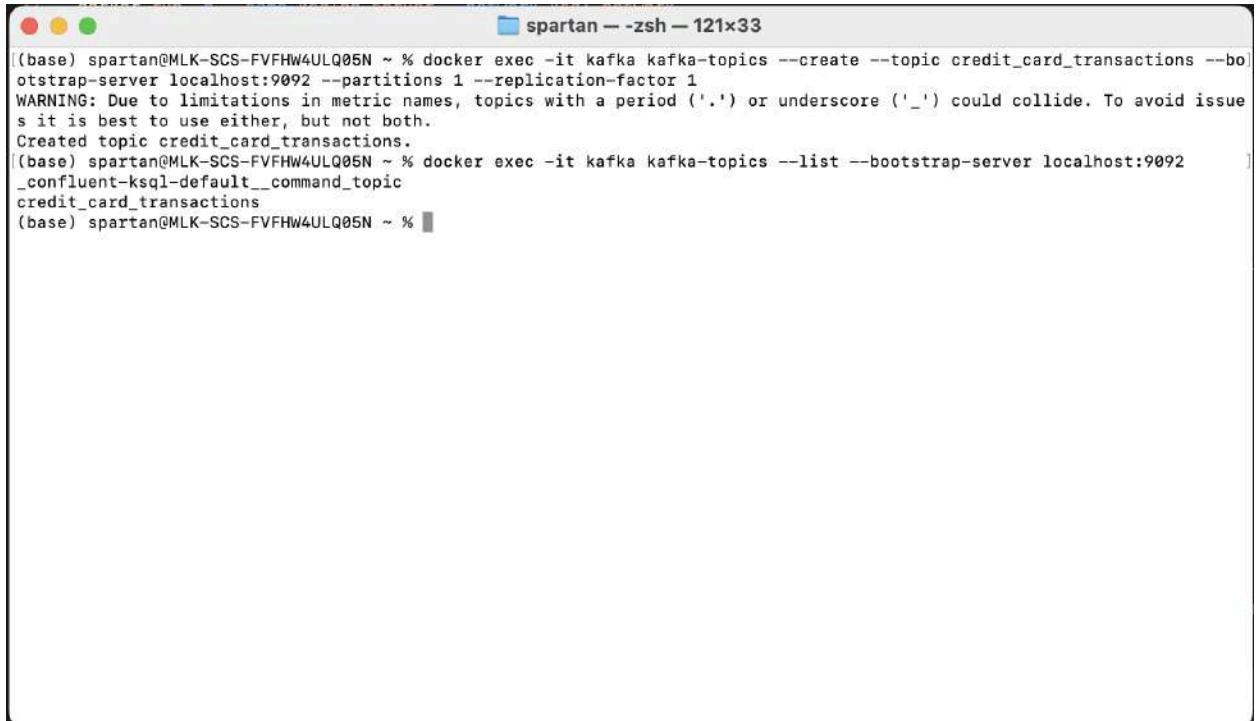
Here using the docker container for kafka we were able to create the kafka topic named **credit_card_transactions**

A terminal window titled 'spartan -- zsh -- 121x33' showing a Docker command to create a Kafka topic. The command is: 'docker exec -it kafka kafka-topics --create --topic credit_card_transactions --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1'. The output shows a warning about metric names and confirms the topic was created.

```
((base) spartan@MLK-SCS-FVFHW4ULQ05N ~ % docker exec -it kafka kafka-topics --create --topic credit_card_transactions --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Created topic credit_card_transactions.
(base) spartan@MLK-SCS-FVFHW4ULQ05N ~ %
```

Verification of the topics created

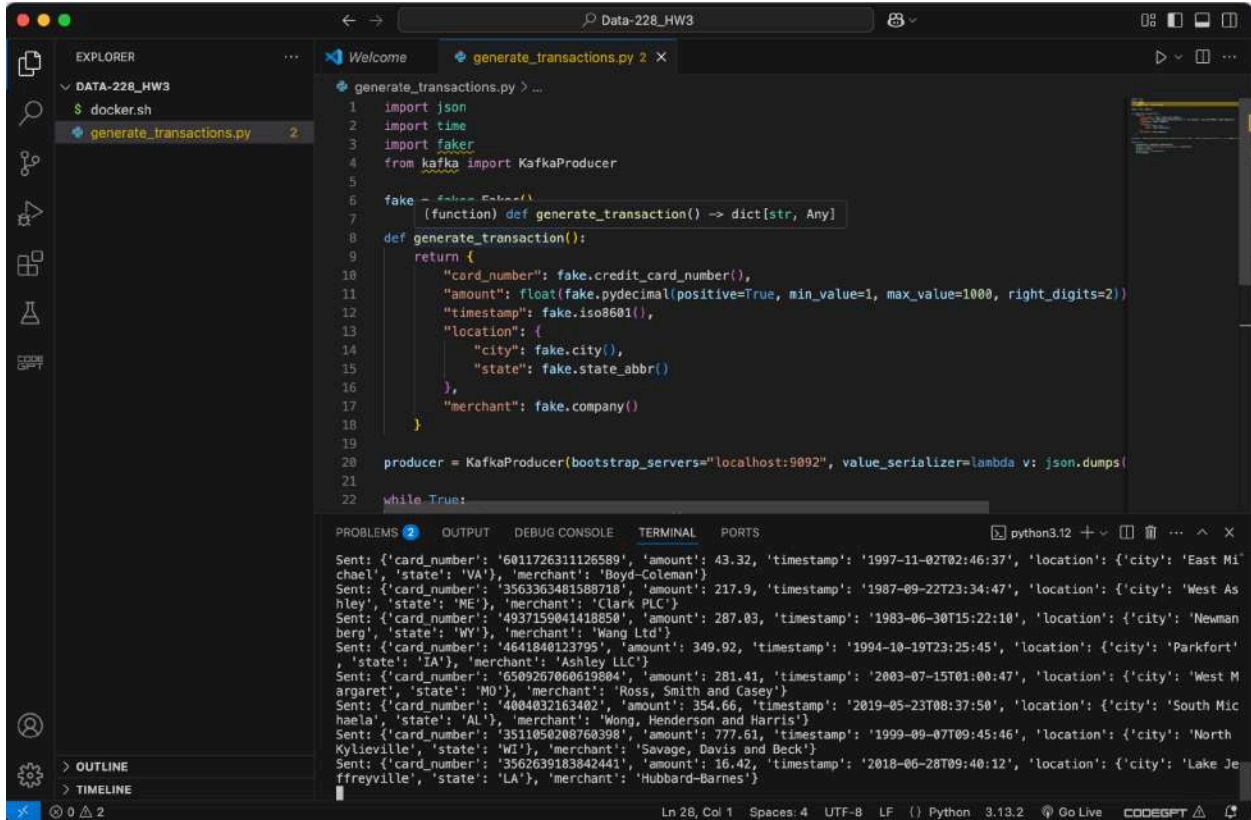
Now we will verify the created kafka topic which is **credit_card_transactions**:



```
spartan -- zsh -- 121x33
(base) spartan@MLK-SCS-FVFHW4ULQ05N ~ % docker exec -it kafka kafka-topics --create --topic credit_card_transactions --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Created topic credit_card_transactions.
(base) spartan@MLK-SCS-FVFHW4ULQ05N ~ % docker exec -it kafka kafka-topics --list --bootstrap-server localhost:9092
_confluent-ksql-default__command_topic
credit_card_transactions
(base) spartan@MLK-SCS-FVFHW4ULQ05N ~ %
```

Step 2:

Generate Data Simulated using the Faker python package



```
1 import json
2 import time
3 import faker
4 from kafka import KafkaProducer
5
6 fake = faker.Faker()
7
8 def generate_transaction():
9     return {
10         "card_number": fake.credit_card_number(),
11         "amount": float(fake.pydecimal(positive=True, min_value=1, max_value=1000, right_digits=2)),
12         "timestamp": fake.iso8601(),
13         "location": {
14             "city": fake.city(),
15             "state": fake.state_abbr()
16         },
17         "merchant": fake.company()
18     }
19
20 producer = KafkaProducer(bootstrap_servers="localhost:9092", value_serializer=lambda v: json.dumps(v).encode('utf-8'))
21
22 while True:
```

Problems: 2 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

python3.12

```
Sent: {'card_number': '6011726311126589', 'amount': 43.32, 'timestamp': '1997-11-02T02:46:37', 'location': {'city': 'East M', 'state': 'VA'}, 'merchant': 'Boyd-Coleman'}
Sent: {'card_number': '3563363481588710', 'amount': 217.9, 'timestamp': '1987-09-22T23:34:47', 'location': {'city': 'West As', 'state': 'ME'}, 'merchant': 'Clark PLC'}
Sent: {'card_number': '4937159041418850', 'amount': 287.03, 'timestamp': '1983-06-30T15:22:10', 'location': {'city': 'Newman', 'state': 'WY'}, 'merchant': 'Wang Ltd'}
Sent: {'card_number': '4641840123795', 'amount': 349.92, 'timestamp': '1994-10-19T23:25:45', 'location': {'city': 'Parkfort', 'state': 'IA'}, 'merchant': 'Ashley LLC'}
Sent: {'card_number': '6509267060619804', 'amount': 281.41, 'timestamp': '2003-07-15T01:00:47', 'location': {'city': 'West M', 'state': 'MO'}, 'merchant': 'Ross, Smith and Casey'}
Sent: {'card_number': '4004032163402', 'amount': 354.66, 'timestamp': '2019-05-23T08:37:50', 'location': {'city': 'South Mic', 'state': 'AL'}, 'merchant': 'Wong, Henderson and Harris'}
Sent: {'card_number': '3511050208760308', 'amount': 777.61, 'timestamp': '1999-09-07T09:45:46', 'location': {'city': 'North', 'state': 'WV'}, 'merchant': 'Savage, Davis and Beck'}
Sent: {'card_number': '3502639183842441', 'amount': 16.42, 'timestamp': '2018-06-28T09:40:12', 'location': {'city': 'Lake Je', 'state': 'LA'}, 'merchant': 'Hubbard-Barnes'}
```

Ln 28, Col 1 | Spaces: 4 | UTF-8 | LF | Python 3.13.2 | Go Live | CODEGPT

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-FVFW4ULQ05N ~ % 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": "3876836968239", "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

```

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

{"card_number": "272026924484081", "amount": 434.35, "timestamp": "2017-03-17T22:55:56", "location": {"city": "Susanfort", "state": "AZ"}, "merchant": "Chen-Good"}
{"card_number": "491853108047965", "amount": 646.71, "timestamp": "1977-03-05T04:57:32", "location": {"city": "Juliestad", "state": "MO"}, "merchant": "Juarez-Cruz"}
{"card_number": "58427837221366", "amount": 914.66, "timestamp": "2018-07-22T16:34:33", "location": {"city": "East Ryanside", "state": "NM"}, "merchant": "Sanchez-Maldonado"}
{"card_number": "38807889829215", "amount": 777.21, "timestamp": "1975-03-25T17:38:40", "location": {"city": "Thomasoan", "state": "OH"}, "merchant": "Martin-Hebert"}
{"card_number": "213138483023560", "amount": 946.48, "timestamp": "1986-01-20T22:38:22", "location": {"city": "New David", "state": "CT"}, "merchant": "Larson PLC"}
{"card_number": "446343681993846124", "amount": 221.87, "timestamp": "2007-12-14T23:59:49", "location": {"city": "Michaeltoan", "state": "PW"}, "merchant": "Reeves LLC"}
{"card_number": "3052769446939596", "amount": 41.44, "timestamp": "1974-10-15T01:29:57", "location": {"city": "New Shelley", "state": "VA"}, "merchant": "Jansen, Custaneda and Kling"}
{"card_number": "3541136180863639", "amount": 430.72, "timestamp": "1999-11-23T00:05:22", "location": {"city": "Nathanshire", "state": "NM"}, "merchant": "Clark-Perez"}
{"card_number": "46630864972769", "amount": 326.86, "timestamp": "1983-10-14T19:07:30", "location": {"city": "Erichshire", "state": "CT"}, "merchant": "Bamirez, Dickson and Rogers"}
{"card_number": "498087220356418", "amount": 851.22, "timestamp": "2015-04-14T17:36:37", "location": {"city": "West Eric", "state": "MD"}, "merchant": "Wilkinson-Yang"}
{"card_number": "3565099146629658", "amount": 290.51, "timestamp": "2003-11-12T19:23:15", "location": {"city": "Lake Makaylanouth", "state": "FL"}, "merchant": "Richardson Inc"}
{"card_number": "56478627615648", "amount": 540.91, "timestamp": "1996-01-31T23:33:05", "location": {"city": "South Kelly", "state": "AZ"}, "merchant": "Martinez Group"}
{"card_number": "451788138881195", "amount": 883.64, "timestamp": "2020-10-10T18:58:33", "location": {"city": "Williamsfurt", "state": "IL"}, "merchant": "Andersen, Taylor and Jackson"}
{"card_number": "502073561545", "amount": 309.81, "timestamp": "1976-09-29T18:20:33", "location": {"city": "West Alexandriasside", "state": "DE"}, "merchant": "Olson, Nelson and Knapp"}
{"card_number": "450222485110305202", "amount": 424.0, "timestamp": "2022-12-29T08:31:16", "location": {"city": "West Laurenburgh", "state": "WA"}, "merchant": "Benson, Huffman and Hernandez"}
{"card_number": "432316999614108", "amount": 110.98, "timestamp": "2023-11-10T04:05:19", "location": {"city": "Aaronfurt", "state": "FL"}, "merchant": "Madden and Sons"}
{"card_number": "38807878018339", "amount": 274.93, "timestamp": "1996-02-25T17:27:89", "location": {"city": "New Laurenview", "state": "IN"}, "merchant": "Trujillo-Martin"}
{"card_number": "3580526321932753", "amount": 272.3, "timestamp": "2002-10-11T23:25:33", "location": {"city": "Michaelview", "state": "AL"}, "merchant": "Stevens-Jefferson"}
{"card_number": "3543723608059263", "amount": 541.63, "timestamp": "1971-02-17T23:28:59", "location": {"city": "North Connie", "state": "AL"}, "merchant": "Olson Ltd"}
{"card_number": "4854610531265", "amount": 839.8, "timestamp": "2019-07-14T21:58:15", "location": {"city": "Guerreron", "state": "NV"}, "merchant": "Knight PLC"}
{"card_number": "3550861040950122", "amount": 604.82, "timestamp": "2021-09-16T19:18:02", "location": {"city": "Guerreron", "state": "AZ"}, "merchant": "Estrada-Banks"}
{"card_number": "4223574220874605", "amount": 499.54, "timestamp": "2011-11-12T20:18:19", "location": {"city": "South Brandon", "state": "CT"}, "merchant": "Smith, Dixon and Adams"}
{"card_number": "3583840061165011", "amount": 602.21, "timestamp": "1998-06-11T03:01:43", "location": {"city": "Andreanouth", "state": "MD"}, "merchant": "Ward PLC"}
{"card_number": "2700110501683104", "amount": 242.12, "timestamp": "1977-05-28T09:31:23", "location": {"city": "East Sabrinaburgh", "state": "RI"}, "merchant": "Cantrell-Flores"}
{"card_number": "242.12", "timestamp": "1977-05-28T09:31:23", "location": {"city": "East Sabrinaburgh", "state": "RI"}, "merchant": "Cantrell-Flores"}

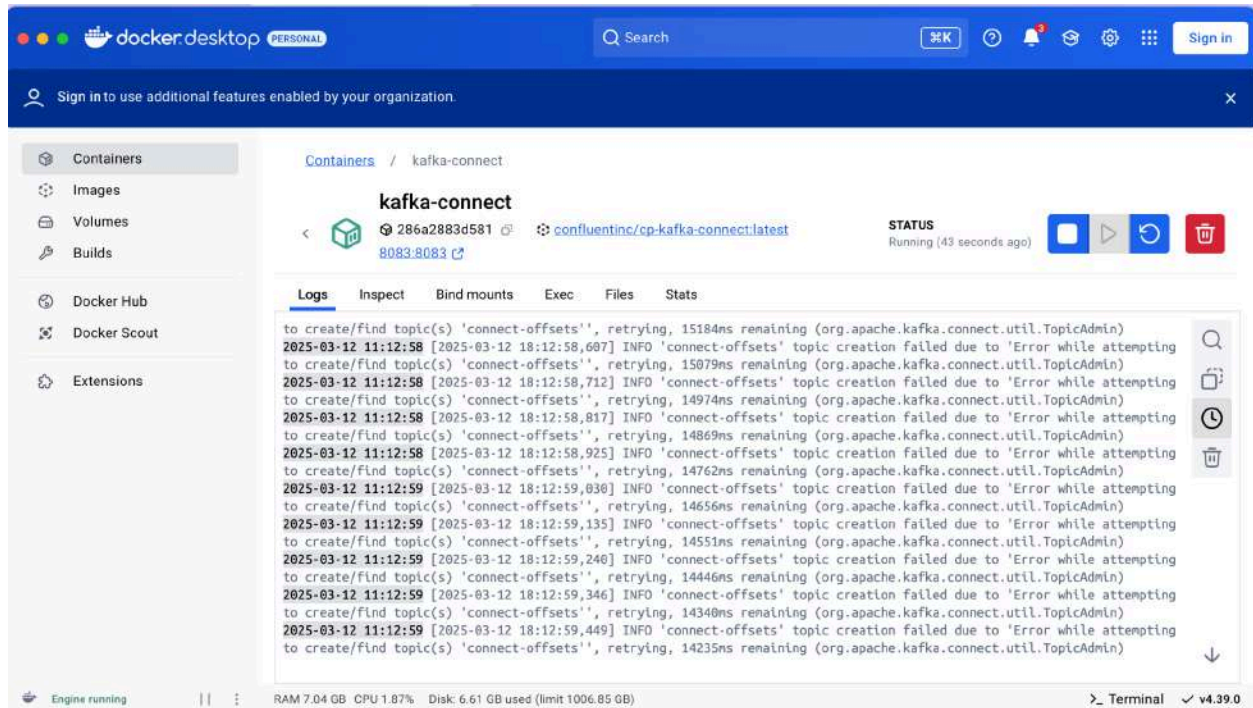
```

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:


```

spartan — docker run -it --rm --network ksql-network confluentinc/cp-ksqldb-cli:latest http://ksqldb-server:8088 — 139x35
> merchant VARCHAR
>) WITH (
>   KAFKA_TOPIC = 'credit_card_transactions',
>   VALUE_FORMAT = 'JSON'
>);
>

Message
-----
Stream created

ksql> SHOW STREAMS;

Stream Name | Kafka Topic           | Key Format | Value Format | Windowed
-----
TRANSACTIONS | credit_card_transactions | KAFKA      | JSON        | false

ksql> SELECT * FROM transactions EMIT CHANGES LIMIT 5;

+-----+-----+-----+-----+-----+
| CARD_NUMBER | AMOUNT | TIMESTAMP | LOCATION | MERCHANT |
+-----+-----+-----+-----+-----+
| 4719557746402 | 520.82 | 1977-05-29T10:05:50 | {CITY=Pattonview, STATE=KS} | Stevens Inc |
| 3551811333611132 | 536.07 | 2002-06-16T10:20:29 | {CITY=New Jeremy, STATE=HI} | Lynch Inc |
| 2669211504989881 | 702.49 | 1987-04-29T05:09:40 | {CITY=Richardhaven, STATE=FL} | Raymond Ltd |
| 3569712137507155 | 499.83 | 1970-03-19T01:08:09 | {CITY=Kevinland, STATE=SD} | Mullins PLC |
| 4230978203635 | 306.03 | 2023-10-29T14:13:41 | {CITY=Rodriguezton, STATE=TX} | Hooper Inc |
+-----+-----+-----+-----+-----+
Limit Reached
Query terminated
ksql>

```

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:

```

ksql> CREATE STREAM flagged_transactions AS
>SELECT card_number, amount, timestamp, location
>FROM transactions
>WHERE amount > 500;

Message
-----
Created query with ID CSAS_FLAGGED_TRANSACTIONS_1

ksql>

```

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 , STATE=OR}	Campbell LLC

```
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
180859935313161	851.73
4424791882021299	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	WI	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:

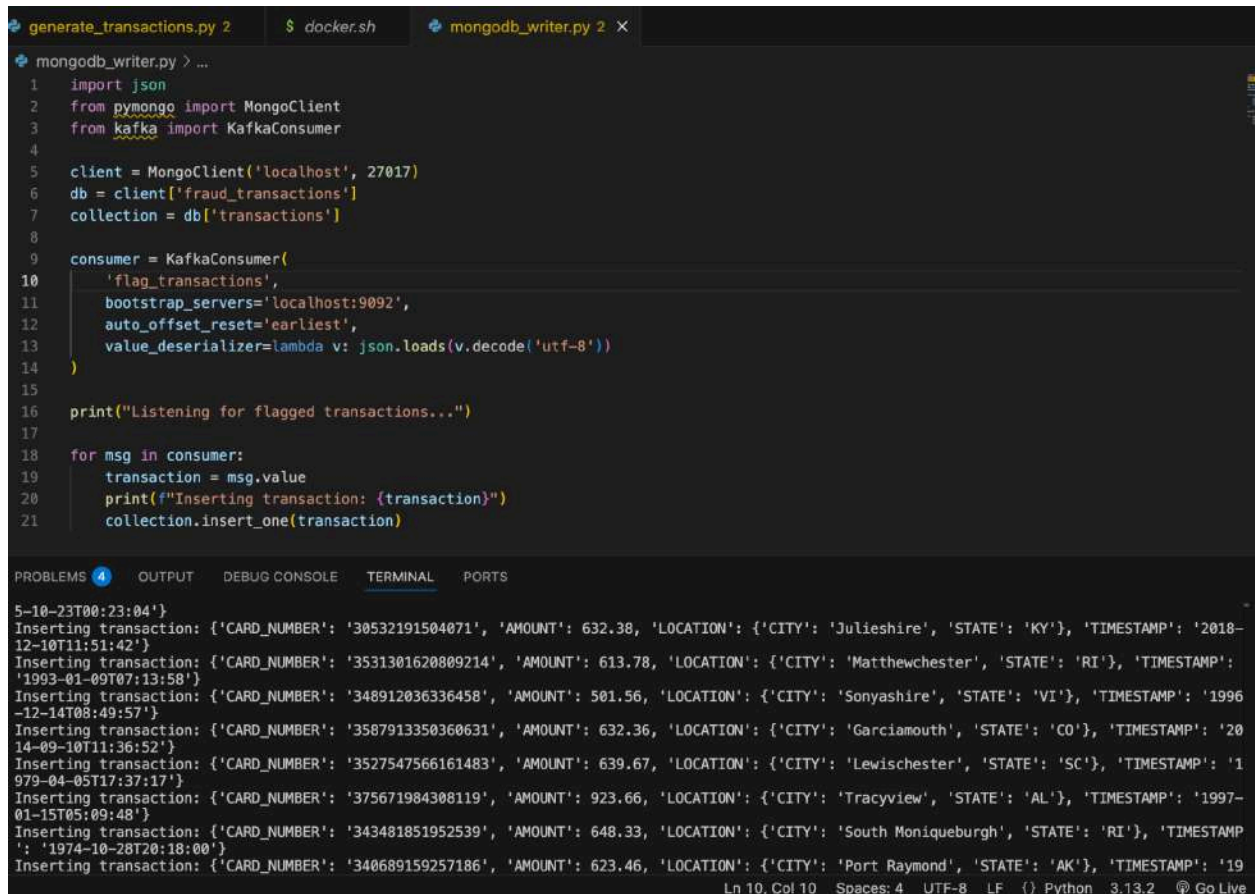
Store Data in MongoDB:

For storing the data in mongo first of all:

1. We started the mongo db server and check mongodb is correctly running on our system.
2. We created a file called **mongodb_writer.py** which contains the logic from 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...
^C
```

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

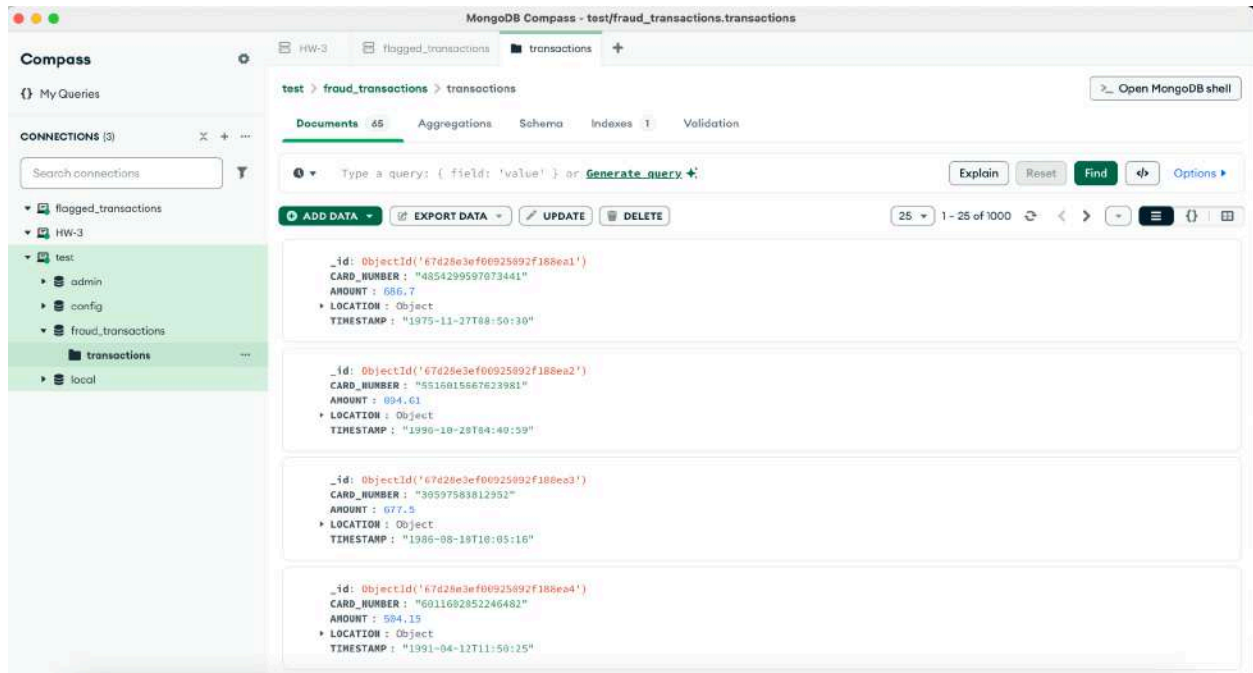


```
generate_transactions.py 2    $ docker.sh    mongodb_writer.py 2 X
mongodb_writer.py > ...
1  import json
2  from pymongo import MongoClient
3  from kafka import KafkaConsumer
4
5  client = MongoClient('localhost', 27017)
6  db = client['fraud_transactions']
7  collection = db['transactions']
8
9  consumer = KafkaConsumer(
10     'flag_transactions',
11     bootstrap_servers='localhost:9092',
12     auto_offset_reset='earliest',
13     value_deserializer=lambda v: json.loads(v.decode('utf-8'))
14 )
15
16 print("Listening for flagged transactions...")
17
18 for msg in consumer:
19     transaction = msg.value
20     print(f"Inserting transaction: {transaction}")
21     collection.insert_one(transaction)

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS
5-10-23T00:23:04'}
Inserting transaction: {'CARD_NUMBER': '30532191504071', 'AMOUNT': 632.38, 'LOCATION': {'CITY': 'Julieshire', 'STATE': 'KY'}, 'TIMESTAMP': '2018-
12-10T11:51:42'}
Inserting transaction: {'CARD_NUMBER': '3531301620009214', 'AMOUNT': 613.78, 'LOCATION': {'CITY': 'Matthewchester', 'STATE': 'RI'}, 'TIMESTAMP':
'1993-01-09T07:13:58'}
Inserting transaction: {'CARD_NUMBER': '348912036336458', 'AMOUNT': 501.56, 'LOCATION': {'CITY': 'Sonyashire', 'STATE': 'VI'}, 'TIMESTAMP': '1996
-12-14T08:49:57'}
Inserting transaction: {'CARD_NUMBER': '3587913350360631', 'AMOUNT': 632.36, 'LOCATION': {'CITY': 'Garciamouth', 'STATE': 'CO'}, 'TIMESTAMP': '20
14-09-10T11:36:52'}
Inserting transaction: {'CARD_NUMBER': '3527547566161483', 'AMOUNT': 639.67, 'LOCATION': {'CITY': 'Lewischester', 'STATE': 'SC'}, 'TIMESTAMP': '1
979-04-05T17:37:17'}
Inserting transaction: {'CARD_NUMBER': '375671984308119', 'AMOUNT': 923.66, 'LOCATION': {'CITY': 'Tracyview', 'STATE': 'AL'}, 'TIMESTAMP': '1997-
01-15T05:09:48'}
Inserting transaction: {'CARD_NUMBER': '343481851952539', 'AMOUNT': 648.33, 'LOCATION': {'CITY': 'South Moniqueburgh', 'STATE': 'RI'}, 'TIMESTAMP
': '1974-10-28T20:18:00'}
Inserting transaction: {'CARD_NUMBER': '340689159257186', 'AMOUNT': 623.46, 'LOCATION': {'CITY': 'Port Raymond', 'STATE': 'AK'}, 'TIMESTAMP': '19
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

card_number	amount	timestamp	location	merchant
6811894551123687	342.57	1975-03-02 19:01:43.000	(West Christopher, AZ)	Jones and Sons
4514381638447	397.53	2001-11-15 08:21:22.000	(New Judyburgh, NM)	Harvey Group
3876836968239	786.21	2002-11-13 22:53:42.000	(Nicoleville, MD)	Shannon Group
565768482844	926.88	1996-04-14 00:32:25.000	(New Darrenstad, RI)	Smith, Brown and Haynes
3588198381146924	976.71	2021-08-10 15:43:04.000	(Palmerchester, AZ)	Olson and Sons
3588116946922974	325.77	2010-05-13 17:34:44.000	(Lake Kristin, CA)	Jackson-Lawrence
5398972959522284	471.41	1974-03-08 02:07:11.000	(Jessicatown, VA)	Young-Cannon
4692168257275	322.56	1989-09-29 14:16:23.000	(Port Elizabeth, CO)	Dean-Sanchez
3589253873880841	930.36	2013-07-03 16:44:53.000	(Lake Caseyfurt, KY)	Williams, Valdez and Torres
188828263494786	199.97	1988-12-23 03:58:24.000	(New Charleston, WI)	Khan and Sons

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

Table creations :

```

[INFO] Execute statement succeeded.

Flink SQL> CREATE TABLE transactions (
>   card_number STRING,
>   amount DECIMAL(10, 2),
>   'timestamp' TIMESTAMP(3),
>   location ROW<city STRING, state STRING>,
>   merchant STRING
> ) WITH (
>   'connector' = 'kafka',
>   'topic' = 'credit_card_transactions',
>   'properties.bootstrap.servers' = 'localhost:9092',
>   'scan.startup.mode' = 'earliest',
>   'format' = 'json'
> );
[INFO] Execute statement succeeded.

Flink SQL> CREATE TABLE flagged_transactions (
>   card_number STRING,
>   amount DECIMAL(10, 2),
>   'timestamp' TIMESTAMP(3),
>   location ROW<city STRING, state STRING>
> ) WITH (
>   'connector' = 'kafka',
>   'topic' = 'flagged_transactions',
>   'properties.bootstrap.servers' = 'localhost:9092',
>   'properties.group.id' = 'flink_consumer_group',
>   'format' = 'json'
> );
[INFO] Execute statement succeeded.

Flink SQL> show tables;
+-----+
| table name |
+-----+
| flagged_transactions |
| transactions |
+-----+
2 rows in set

Flink SQL>

```

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)
Page: Last of 224 Updated: 02:13:23.468

Refresh: 1 s

window_start	window_end	transaction_count	total amount	min amount	max amount
2025-02-06 07:15:50.000	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	528.33	528.33	528.33
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)
Page: Last of 1 Updated: 02:22:23.777

Refresh: 1 s

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>

```


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

...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)
Page: Last of 184
Refresh: 1 s Updated: 02:23:23.391

card number	amount	location	timestamp
344162619333977	877.16	(Diazbury, DC)	1979-04-23 11:42:29.000
4992257387680539	515.24	(Port Markmouth, KS)	2082-07-85 11:49:23.000
502017559746	935.47	(Martinview, MD)	1993-18-04 01:27:08.000
4754967799329	936.76	(Jeffreymouth, TN)	2020-01-26 17:57:42.000
2228350892601510	513.30	(West Shane, WY)	1992-04-04 23:07:32.000
188015386101791	876.07	(South Mackenzie, LA)	2003-09-04 02:52:49.000
4642326263094528336	830.71	(South Ashley, FL)	1990-00-10 13:14:23.000
6011605687293730	639.55	(Leslieville, LA)	1992-01-15 22:50:56.000

Quit Refresh Inc Refresh Dec Refresh Goto Page Last Page Next Page Prev Page Open Row

- 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]
```

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)
Page: Last of 112 Updated: 16:55:11.427

CARD_NUMBER	AMOUNT	TIMESTAMP	LOCATION
4342869279846	595.99	2014-12-22T05:27:00	(South Carolineshire, SC)
4024369172531747945	968.84	2000-03-12T21:48:20	(Dunnland, NY)
4767518851787	658.04	2024-05-15T20:37:21	(North Rachel, IN)
38953321435866	541.20	1987-10-16T12:54:57	(Batesbury, KS)
3529052979191142	788.03	1973-03-05T01:36:16	(Watsonview, NY)
601199840463399	812.82	2023-02-24T34:28:52	(Port Kelseyland, IA)
2262837143798365	652.47	1990-12-07T17:33:11	(Karenstad, OK)
3559329711164286	600.94	1976-08-02T23:50:06	(West Lataashaside, MP)
676295894395	872.09	2010-04-21T10:28:07	(East Desireemouth, RI)
4824228521469974	905.81	1978-11-16T02:35:59	(Robinsonmouth, PA)
2717246626796383	906.05	1986-06-06T10:37:33	(South Brianna, PW)
213118531461176	928.39	1976-05-22T18:14:34	(Coletown, AR)
3570034747402525	929.59	2014-09-20T00:59:56	(New Stephanie, UT)
342617370080960	665.44	2007-04-22T11:50:40	(West Timothyberg, NE)
379139932335028	788.29	1993-11-05T09:01:03	(South Robertbury, MI)

Quit Refresh Inc Refresh Dec Refresh Goto Page Last Page Next Page Prev Page Open Row

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> █

...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)
Page: Last of 192 Updated: 16:34:24.648

CARD_NUMBER	AMOUNT_CATEGORY	TRANSACTION_COUNT
6504982719800141	High	1
344991779708143	High	1
2592744523835221	High	1
3511329386785329	High	1
3536385873253597	High	1
4919685922574	High	1
675928037802	High	1
213163266806651	High	1

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>
```

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
3520004167719624	712.51
4018782894469917058	667.28
4594893562130932	697.18
3530943945611634	514.43
4723501461883068232	882.23
3591907286762286	501.02
4100992772256377	606.03
30147377530434	766.58
4961805980954047275	727.90
3581243306825677	770.14
6011406989267656	608.08
566125472571	942.06
586186809990	754.19
676372562741	603.17
3539842770042226	542.27
371503301097774	758.07
30415752520423	526.44
30086145919745	649.23
3514474357874387	659.80

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>
```

CARD_NUMBER	total_amount	average_amount
3512797300183975	804.47	804.470000
30006231094245	794.98	794.980000
2523123593737202	548.27	548.270000
213163872209918	734.08	734.080000
3507948248393550	645.45	645.450000
571750824900	613.27	613.270000
30411420503618	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>
```

CITY	TOTAL_FLAGGED_AMOUNT
Michelleshire	728.84
Shannonland	728.14
West Tammybury	894.43
Ortizhaven	828.08
North Lisamouth	582.57

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 .