1.) MySQL Table (Table should have some column like created\_at or updated\_at so that can be used for incremental read)

Sol. CREATE TABLE employee(

id INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(255),

Salary INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP);

2.) Write a python script which is running in infinite loop and inserting 4-5 dummy/dynamically prepared records

in MySQL Table

Sol.

# MySQL connection configuration

employee= {

'id': 'id',

'name': 'employee \_name',

'salary': 'employee\_salarye',

'age': 'employee\_age'

}

def generate\_random\_string(length):

letters = string.ascii\_lowercase

return ''.join(random.choice(letters) for i in range(length))

# Insert dummy records into MySQL table

def insert\_dummy\_records():

try:

# Connect to MySQL

conn = mysql.connector.connect(\*\*employee)

# Create a cursor object

cursor = conn.cursor()

# Prepare and execute the INSERT statement

for \_ in range(4):

employee\_name = generate\_random\_string(5)

employee\_age = random.randint(1, 10)

insert\_query = "INSERT INTO employee(name,age) VALUES (%s, %s)"

record\_values = (name,age)

cursor.execute(insert\_query, record\_values)

# Commit the changes

conn.commit()

# Close the cursor and connection

cursor.close()

conn.close()

print("Records inserted successfully!")

except Error as e:

print("Error inserting records into MySQL table:", e)

# Run the script in an infinite loop

while True:

insert\_dummy\_records()

3.) Setup Confluent Kafka

Sol. setup

4.) Create Topic

Sol. bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --topic my\_topic --partitions 3 --replication-factor 2

curl -X POST -H "Content-Type: application/json" -d '{"numPartitions": 3, "replicationFactor": 2}' http://localhost:8080/admin/v2/persistent/public/default/my\_topic

5.) Create json schema on schema registry (depends on what kind of data you are publishing in mysql table)

Sol. {

"type": "record",

"name": "employee",

"fields": [ {"name": "id", "type": "int"},

{"name": "name", "type": "string"},

{"name": "age", "type": "int"}

]

}

6.) Write a producer code which will read the data from MySQL table incrementally (hint : use and maintain create\_at column)

Sol.

# MySQL connection configuration

mysql\_config = {

'host': 'your\_host',

'database': 'your\_database',

'user': 'your\_username',

'password': 'your\_password'

}

# Kafka producer configuration

kafka\_config = {

'bootstrap\_servers': 'your\_bootstrap\_servers'

}

def get\_incremental\_data(last\_updated):

try:

# Connect to MySQL

conn = mysql.connector.connect(\*\*mysql\_config)

# Create a cursor object

cursor = conn.cursor()

select\_query = "SELECT \* FROM employee WHERE created\_at > %s"

cursor.execute(select\_query, (last\_updated,))

# Fetch all rows

rows = cursor.fetchall()

def publish\_data(data):

producer = KafkaProducer(\*\*kafka\_config)

for row in data:

# Format the row data as per your requirement

7.) Producer will publish data in Kafka Topic

Sol.bin/kafka-console-producer.sh --broker-list localhost:5467 --topic Hello-Kafka

8.) Write consumer group to consume data from Kafka topic

Sol. from confluent\_kafka import Consumer, KafkaException

# Kafka consumer configuration

consumer\_config = {

'bootstrap.servers': 'your\_bootstrap\_servers',

'group.id': 'your\_consumer\_group\_id',

'auto.offset.reset': 'earliest',

'enable.auto.commit': False

}

# Kafka topic to consume from

topic = 'data'

# Create Kafka consumer instance

consumer = Consumer(consumer\_config)

consumer.subscribe([topic])

try:

while True:

message = consumer.poll(1.0)

if message is None:

continue

if message.error():

if message.error().code() == KafkaException.\_PARTITION\_EOF:

# Reached end of partition, continue to the next message

continue

else:

print(f"Error occurred: {message.error().str()}")

break

# Process the message

value = message.value().decode('utf-8')

print(f"Received message: {value}")

# Commit the consumed message offset

consumer.commit(message)

except KeyboardInterrupt:

pass

finally:

# Close the consumer

consumer.close()

9.) In Kafka consumer code do some changes or transformation for each record and write it in Cassandra table

Sol. from confluent\_kafka import Consumer, KafkaException

from cassandra.cluster import Cluster

# Kafka consumer configuration

consumer\_config = {

'bootstrap.servers': 'your\_bootstrap\_servers',

'group.id': 'your\_consumer\_group\_id',

'auto.offset.reset': 'earliest',

'enable.auto.commit': False

}

# Kafka topic to consume from

topic = 'your\_topic\_name'

# Cassandra connection configuration

cassandra\_config = {

'contact\_points': ['your\_contact\_point'],

'port': 9042,

'keyspace': 'your\_keyspace',

'table': 'your\_table'

}

# Prepare Cassandra INSERT statement

insert\_statement = session.prepare(

f"INSERT INTO {cassandra\_config['table']} (id, transformed\_data) VALUES (?, ?)"

)

# Subscribe to the topic

consumer.subscribe([topic])

# Consume messages from Kafka topic

try:

while True:

message = consumer.poll(1.0)

if message is None:

continue

if message.error():

if message.error().code() == KafkaException.\_PARTITION\_EOF:

# Process the message

value = message.value().decode('utf-8')

transformed\_data = transform\_data(value) # Perform your data transformation here

# Write the transformed data to Cassandra

session.execute(insert\_statement, (message.key(), transformed\_data))

# Commit the consumed message offset

consumer.commit(message)

except KeyboardInterrupt:

pass

finally:

# Close the consumer and Cassandra session

consumer.close()

session.shutdown()

cluster.shutdown()