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

A)

import mysql.connector as myconn

mydb = myconn.connect(

host="localhost",

user="root",

password="Vasavi@123",

database="one"

)

mycursor=mydb.cursor()

mycursor.execute("CREATE TABLE IF NOT EXISTS employee(id INT PRIMARY KEY AUTO\_INCREMENT,name VARCHAR(50),age INT,created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP);")

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

in MySQL Table

A)

import mysql.connector as myconn

import random

import time

from datetime import datetime, timedelta

mydb = myconn.connect(

host="localhost",

user="root",

password="Vasavi@123",

database="one"

)

# defining a function to generate dummy data

def generate\_data():

names = ['Ravi', 'Ram', 'Sam', 'Teja', 'Varun', 'Raja', 'sham', 'Drudra', 'Remo', 'sandeep', 'Kishan', 'Alice']

name = random.choice(names)

age = random.randint(24, 65)

return (name, age)

# insert dummy data into the table employee in an infinite loop

while True:

data = generate\_data()

sql = "INSERT INTO employee (name, age) VALUES (%s, %s)"

val = (data[0], data[1])

mycursor = mydb.cursor()

mycursor.execute(sql, val)

mydb.commit()

print("Inserted record: ", data)

time.sleep(1)

3.) Setup Confluent Kafka

A) Setuped Confluent Kafka with cluster\_name as "kafka\_cluster"

4.) Create Topic

A) created a Topic with Topic\_name as "employees"

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

A)

{

"$id": "http://example.com/myURI.schema.json",

"$schema": "http://json-schema.org/draft-07/schema#",

"additionalProperties": false,

"description": "Sample schema to help you get started.",

"properties": {

"age": {

"type": "integer"

},

"created\_at": {

"format": "date-time",

"type": "string"

},

"id": {

"type": "integer"

},

"name": {

"type": "string"

}

},

"title": "SampleRecord",

"type": "object"

}

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

A)

from confluent\_kafka import Producer

import mysql.connector as myconn

from datetime import datetime, timedelta

import json

# MySQL connection details

mydb = myconn.connect(

host="localhost",

user="root",

password="Vasavi@123",

database="one"

)

mycursor = mydb.cursor()

# Kafka connection details

API\_KEY = 'TG2UFQD5JK55IFWH'

API\_SECRET\_KEY = '4vWJBxZkXMi44tu93nbYCG4mjQVzYAI7stM4UIV8LszmMxmKQyUEV9u3RfNZ23qZ'

BOOTSTRAP\_SERVER = 'pkc-6ojv2.us-west4.gcp.confluent.cloud:9092'

SECURITY\_PROTOCOL = 'SASL\_SSL'

SSL\_MACHENISM = 'PLAIN'

ENDPOINT\_SCHEMA\_URL = 'https://psrc-epkz2.ap-southeast-2.aws.confluent.cloud'

SCHEMA\_REGISTRY\_API\_KEY = 'KIPLIUBH5A3PIIGG'

SCHEMA\_REGISTRY\_API\_SECRET = 'hN4RXRXBcobHDTZnwBmlRpxggaNSFyHOvl7be5tTbei1Xx+mr0z84ettA5HlEcOV'

conf = {

'bootstrap.servers': BOOTSTRAP\_SERVER,

'security.protocol': SECURITY\_PROTOCOL,

'sasl.mechanism': SSL\_MACHENISM,

'sasl.username': API\_KEY,

'sasl.password': API\_SECRET\_KEY

}

producer = Producer(conf)

# Start timestamp for incremental reads

start\_time = datetime.now() - timedelta(minutes=1)

# Function to fetch data from MySQL table incrementally

def fetch\_data():

global start\_time

sql = f"SELECT \* FROM employee WHERE created\_at >= '{start\_time}'"

mycursor.execute(sql)

result = mycursor.fetchall()

start\_time = datetime.now()

return result

# Function to publish data to Kafka topic

def publish\_to\_topic(data):

for row in data:

record = {

'id': row[0],

'name': row[1],

'age': row[2],

'created\_at': str(row[3])

}

print(record.values())

producer.produce(topic='employees', value=json.dumps(record).encode('utf-8'))

producer.flush()

# Fetch data from MySQL table and publish to Kafka topic in an infinite loop

while True:

data = fetch\_data()

if data:

publish\_to\_topic(data)

7.) Producer will publish data in Kafka Topic

A) Already answered in question number "6".

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

A)

from confluent\_kafka import Consumer, KafkaError

# Kafka connection details

API\_KEY = 'TG2UFQD5JK55IFWH'

API\_SECRET\_KEY = '4vWJBxZkXMi44tu93nbYCG4mjQVzYAI7stM4UIV8LszmMxmKQyUEV9u3RfNZ23qZ'

BOOTSTRAP\_SERVER = 'pkc-6ojv2.us-west4.gcp.confluent.cloud:9092'

SECURITY\_PROTOCOL = 'SASL\_SSL'

SSL\_MACHENISM = 'PLAIN'

conf = {

'bootstrap.servers': BOOTSTRAP\_SERVER,

'security.protocol': SECURITY\_PROTOCOL,

'sasl.mechanism': SSL\_MACHENISM,

'sasl.username': API\_KEY,

'sasl.password': API\_SECRET\_KEY,

'group.id': 'my-group'

}

consumer = Consumer(conf)

consumer.subscribe(['employees'])

while True:

msg = consumer.poll(1.0)

if msg is None:

continue

if msg.error():

if msg.error().code() == KafkaError.\_PARTITION\_EOF:

print(f'{msg.topic()} reached end at offset {msg.offset()}')

else:

print(f'Error while consuming message: {msg.error()}')

else:

print(f"Received message: {msg.value().decode('utf-8')}")

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

A)

import json

import cassandra

from cassandra.cluster import Cluster

from cassandra.auth import PlainTextAuthProvider

from confluent\_kafka import Consumer, KafkaError

cloud\_config= {

'secure\_connect\_bundle': '/home/teja/practice/kafka/confluent\_kafka/confluent\_kafka\_Assignment/secure-connect-kafka-cassandra.zip'

}

auth\_provider = PlainTextAuthProvider('WNzbmMvdFyNuZUhLYEnyZFQU', 'Ri5\_KPROGUE+wBqK.8ZOpbv.yTF+GJ+eU7idrNR9hhDsZm5dt,Wym1Zas0wkqqMhA\_SMZ2T-WibWnh.nE-O\_M4F9frKfqvoHG5stYhFfkbbLAE9lBZofC8wuU2AmtK2K')

cluster = Cluster(cloud=cloud\_config, auth\_provider=auth\_provider)

session = cluster.connect()

query = "use new\_kc"

session.execute(query)

try:

session.execute("CREATE TABLE IF NOT EXISTS emp (id BIGINT PRIMARY KEY, name TEXT, age INT)")

print("created")

except Exception as err:

print(err)

# Kafka connection details

API\_KEY = 'TG2UFQD5JK55IFWH'

API\_SECRET\_KEY = '4vWJBxZkXMi44tu93nbYCG4mjQVzYAI7stM4UIV8LszmMxmKQyUEV9u3RfNZ23qZ'

BOOTSTRAP\_SERVER = 'pkc-6ojv2.us-west4.gcp.confluent.cloud:9092'

SECURITY\_PROTOCOL = 'SASL\_SSL'

SSL\_MACHENISM = 'PLAIN'

conf = {

'bootstrap.servers': BOOTSTRAP\_SERVER,

'security.protocol': SECURITY\_PROTOCOL,

'sasl.mechanism': SSL\_MACHENISM,

'sasl.username': API\_KEY,

'sasl.password': API\_SECRET\_KEY,

'group.id': 'my-group'

}

consumer = Consumer(conf)

consumer.subscribe(['employees'])

while True:

msg = consumer.poll(1.0)

if msg is None:

continue

if msg.error():

if msg.error().code() == KafkaError.\_PARTITION\_EOF:

print(f'{msg.topic()} reached end at offset {msg.offset()}')

else:

print(f'Error while consuming message: {msg.error()}')

else:

# parse the message to extract the data

try:

data = json.loads(msg.value().decode('utf-8'))

id = int(data['id'])

name = data['name']

age = int(data['age'])

# performing some tranformation on data i.e we are inserting records only if the age is between 25 and 35

if age>=25 and age<=35:

# insert the data into Cassandra

session.execute(f"INSERT INTO emp (id, name, age) VALUES ({id}, '{name}', {age})")

except Exception as e:

print(f'Error while processing message: {e}')