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

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

in MySQL Table

3.) Setup Confluent Kafka

4.) Create Topic

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

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

7.) Producer will publish data in Kafka Topic

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

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

**Solutions**

**Producer code:-**

from pyspark.sql import SparkSession

from pyspark.sql.functions import \*

from pyspark.sql.types import \*

import os

from pyspark.sql import DataFrame

import time

import logging

from datetime import datetime

#Variable declaration

#cassandra database detail

KEYSPACE = "ineuron1"

TABLE="employee"

#kafka server detail

KAFKA\_BOOTSTRAP\_SERVER="kafka:9092"

KAFKA\_TOPIC = "employee"

#Cassandra database connectivity credentails

CASSANDRA\_HOST="cassandra"

CASSANDRA\_USER="cassandra"

CASSANDRA\_PASSWORD="cassandra"

#Maining log

#log file name

LOG\_FILE\_NAME = f"{datetime.now().strftime('%m%d%Y\_\_%H%M%S')}.log"

#log directory

LOG\_FILE\_DIR = os.path.join(os.getcwd(),"logs")

#create folder if not available

os.makedirs(LOG\_FILE\_DIR,exist\_ok=True)

logging.basicConfig(

filename=os.path.join(LOG\_FILE\_DIR,LOG\_FILE\_NAME),

format="[ %(asctime)s ] %(lineno)d %(name)s - %(levelname)s - %(message)s",

level=logging.INFO,

)

#create spark session with cassandar configuration

sparkSesison = (SparkSession.builder

.config("spark.cassandra.connection.host","cassandra")

.config("spark.cassandra.auth.username","cassandra")

.config("spark.cassandra.auth.password","cassandra")

.appName("demo").getOrCreate()

)

#Reading table from cassandra db and returning spark dataframe

def dataFrameFromCassandaDbTable(sparkSession:SparkSession,keyspace:str,table:str)->DataFrame:

df = (sparkSession.read

.format("org.apache.spark.sql.cassandra")

.options(table=table, keyspace=keyspace)

.load())

return df

def sendDataToKafkaTopic(kafkaBootstrapServer,topicName,dataFrame:DataFrame):

logging.info(f"Started writing data to kafka topic {topicName} and server: {kafkaBootstrapServer}")

dataFrame = dataFrame.select(col("emp\_id").cast(StringType()).alias("key"),to\_json(struct("emp\_id","emp\_name","city","state")).alias("value"))

dataFrame.show(2,truncate=False)

(dataFrame

.write

.format("kafka")

.option("kafka.bootstrap.servers",kafkaBootstrapServer)

.option("failOnDataLoss", "false")

.option("topic",topicName )

.save())

logging.info(f"Data has been written to kafka topic: {topicName}")

if \_\_name\_\_=="\_\_main\_\_":

# Read data from cassandra database

df = dataFrameFromCassandaDbTable(sparkSession=sparkSesison,table=TABLE,keyspace=KEYSPACE)

#Print the schema

df.printSchema()

#showing dataframe

df.show(truncate=False)

nRows = df.count()

columns = df.columns

logging.info(f"{TABLE} has columns: [{columns}]")

logging.info(f"{nRows} rows found in table: {KEYSPACE}.{TABLE}")

if nRows==0:

logging.info(f"No data found hence data will not be written to kafka topic")

else:

sendDataToKafkaTopic(kafkaBootstrapServer=KAFKA\_BOOTSTRAP\_SERVER,

topicName=KAFKA\_TOPIC,

dataFrame=df

)

**Consumer code:-**

from pyspark.sql import SparkSession

from pyspark.sql.functions import \*

from pyspark.sql.types import \*

import os

from pyspark.sql import DataFrame

import time

import logging

from datetime import datetime

#Variable declaration

#cassandra database detail

KEYSPACE = "ineuron1"

TABLE="employee"

#kafka server detail

KAFKA\_BOOTSTRAP\_SERVER="kafka:9092"

KAFKA\_TOPIC = "employee"

#Cassandra database connectivity credentails

CASSANDRA\_HOST="cassandra"

CASSANDRA\_USER="cassandra"

CASSANDRA\_PASSWORD="cassandra"

PROCESSING\_INTERVAL = f"5 seconds"

#Maining log

#log file name

LOG\_FILE\_NAME = f"{datetime.now().strftime('%m%d%Y\_\_%H%M%S')}.log"

#log directory

LOG\_FILE\_DIR = os.path.join(os.getcwd(),"logs")

#create folder if not available

os.makedirs(LOG\_FILE\_DIR,exist\_ok=True)

logging.basicConfig(

filename=os.path.join(LOG\_FILE\_DIR,LOG\_FILE\_NAME),

format="[ %(asctime)s ] %(lineno)d %(name)s - %(levelname)s - %(message)s",

level=logging.INFO,

)

#create spark session with cassandar configuration

sparkSesison = (SparkSession.builder

.config("spark.cassandra.connection.host","cassandra")

.config("spark.cassandra.auth.username","cassandra")

.config("spark.cassandra.auth.password","cassandra")

.appName("demo").getOrCreate()

)

schema = StructType(fields=[

StructField(name="emp\_id",dataType=IntegerType()),

StructField(name="emp\_name",dataType=StringType()),

StructField(name="city",dataType=StringType()),

StructField(name="state",dataType=StringType()),

])

dataSink = os.path.join("employee\_data")

def processEachInterval(df:DataFrame,epoch\_id):

# print(epoch\_id)

# df.show(truncate=False)

df = (df.withColumn("value",

from\_json(decode("value",charset="UTF-8"),schema=schema)

.alias("value"))

.select("value.\*")

)

if df.count()>0:

df.show(truncate=False)

df.write.mode("append").parquet(dataSink)

if \_\_name\_\_=="\_\_main\_\_":

df = (sparkSesison

.readStream

.format("kafka")

.option("kafka.bootstrap.servers",KAFKA\_BOOTSTRAP\_SERVER)

.option("subscribe",KAFKA\_TOPIC)

.option("startingOffsets","earliest")

.load()

)

df.printSchema()

query = (df.writeStream

.trigger(processingTime=PROCESSING\_INTERVAL)

.foreachBatch(processEachInterval)

.start()

)

query.awaitTermination()

**Create a keyspace with name "ineuron1"**

CREATE KEYSPACE ineuron1

WITH REPLICATION = {

'class': 'org.apache.cassandra.locator.SimpleStrategy',

'replication\_factor': '3'

}

AND DURABLE\_WRITES = true;

**Create a table Employee**

CREATE TABLE EMPLOYEE(

EMP\_ID INT,

EMP\_NAME text,

CITY text,

STATE text,

primary key (EMP\_ID)

);

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES

(2,'Avnish','Bengalore','Karnataka');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (4,'Stephen','Mumbai','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (5,'Berry','Chennai','Tamilnadu');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (6,'Barton','Pune','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (7,'Natasha','Hyderabad','Andra Pradesh');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (8,'Sundar','Noida','Uttar Pradesh');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (9,'Shashank','Delhi','Delhi');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (10,'Sudhanshu','New Mumbai','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (11,'Krish','Nagpur','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (12,'Aman','Hyderabad','Andra Pradesh');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (13,'Rahul','Noida','Uttar Pradesh');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (14,'Sunny','Delhi','Delhi');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (15,'Vishal','New Mumbai','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (16,'Vikash','Nagpur','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (17,'Aravind','Chennai','Tamilnadu');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (18,'Adam','Pune','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (19,'Scarlett','Hyderabad','Andra Pradesh');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (20,'Robert','Noida','Uttar Pradesh');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (21,'Shivam','Delhi','Delhi');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (22,'Deepak','New Mumbai','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (23,'Amit','Nagpur','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (24,'Susmita','Hyderabad','Andra Pradesh');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (25,'Iris','Mumbai','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (26,'Pia','Chennai','Tamilnadu');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (27,'Ankit','Pune','Maharashtra');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (28,'Akash','Hyderabad','Andra Pradesh');

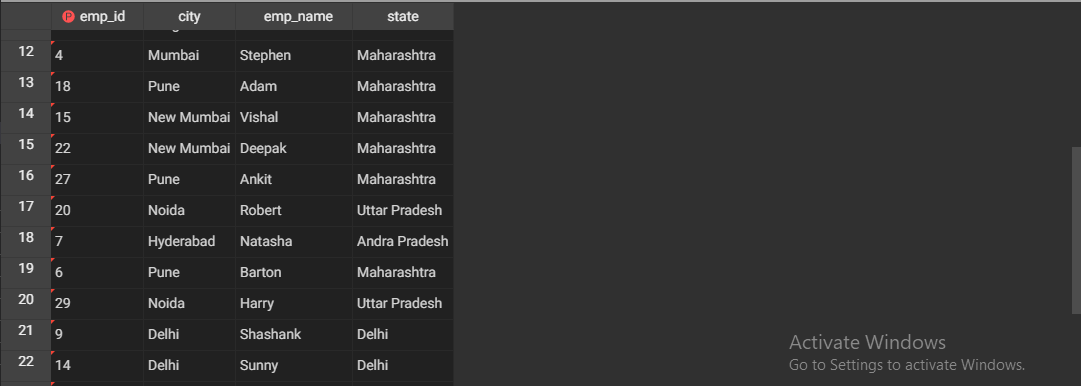
INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (29,'Harry','Noida','Uttar Pradesh');

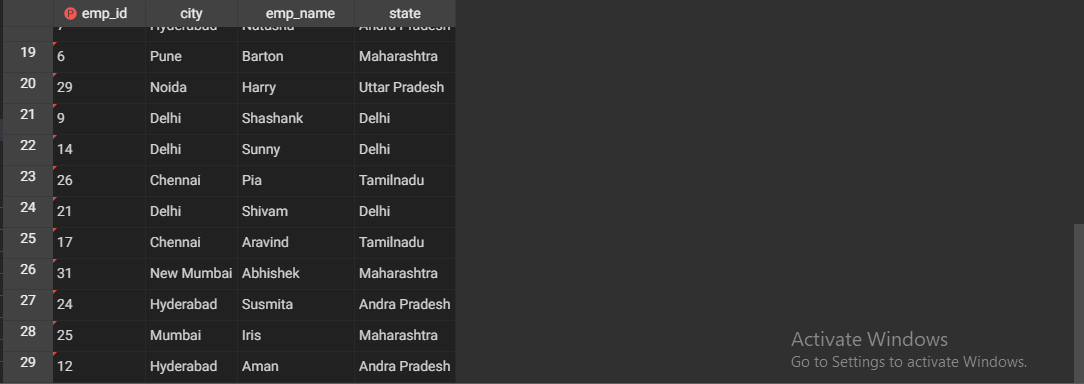
INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (30,'Emma','Delhi','Delhi');

INSERT INTO ineuron1.employee (emp\_id,emp\_name,city,state) VALUES (31,'Abhishek','New Mumbai','Maharashtra');

select \* from *ineuron1*.employee;







**To launch producer script**

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.0.1,com.datastax.spark:spark-cassandra-connector\_2.12:3.0.0 producer.py

**To launch consumer script**

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.0.1 consumer.py

truncate table *ineuron1*.employee;

INSERT INTO *ineuron1*.employee (emp\_id,emp\_name,city,state) VALUES (30,'Sunny','Delhi','Delhi');

INSERT INTO *ineuron1*.employee (emp\_id,emp\_name,city,state) VALUES (31,'Vishal','New Mumbai','Maharashtra');

**Again launch producer script**

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.0.1,com.datastax.spark:spark-cassandra-connector\_2.12:3.0.0 producer.py

**Again launch consumer script**

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.0.1 consumer.py

ALTER TABLE *ineuron1*.employee ADD created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP;

ALTER TABLE *ineuron1*.employee

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

INSERT INTO *ineuron1*.employee (emp\_id,emp\_name,city,state) VALUES (36,'Sunny','Delhi','Delhi');

INSERT INTO *ineuron1*.employee (emp\_id,emp\_name,city,state) VALUES (37,'Vishal','New Mumbai','Maharashtra');

UPDATE *ineuron1*.employee

SET

    city='Pune',

    created\_at='2023-05-01 21:59:36',

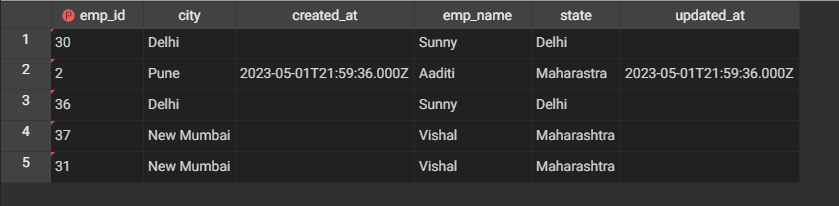
    emp\_name='Aaditi',

    state='Maharastra',

    updated\_at='2023-05-01 21:59:36'

WHERE emp\_id=2;

select \* from *ineuron1*.employee;



UPDATE *ineuron1*.employee

SET

    city='Delhi',

    created\_at='2023-05-01 22:09:42',

    emp\_name='Sunny',

    state='Delhi',

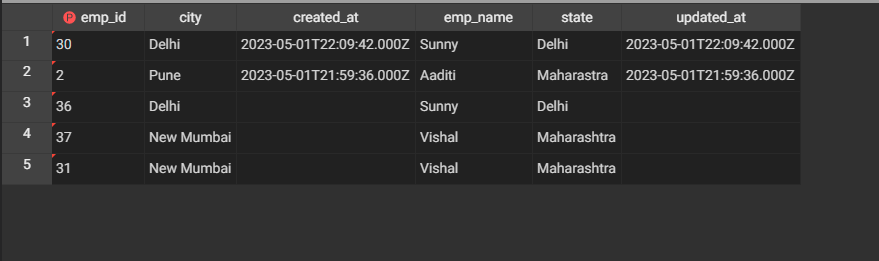
    updated\_at='2023-05-01 22:09:42'

WHERE emp\_id=30;

SELECT emp\_id, city, created\_at, emp\_name, state, updated\_at

FROM *ineuron1*.employee

LIMIT 500;



UPDATE *ineuron1*.employee

SET

    city='Delhi',

    created\_at='2023-05-01 22:18:12',

    emp\_name='Sunny',

    state='Delhi',

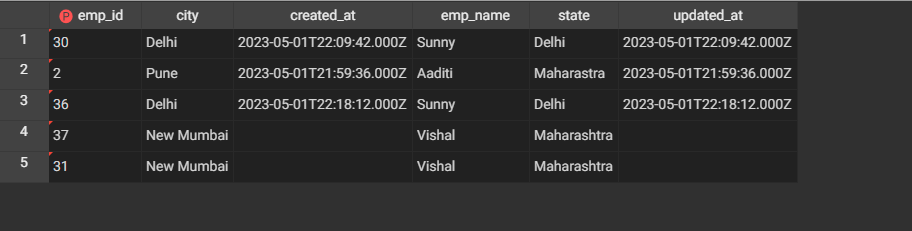
    updated\_at='2023-05-01 22:18:12'

WHERE emp\_id=36;

SELECT emp\_id, city, created\_at, emp\_name, state, updated\_at

FROM *ineuron1*.employee

LIMIT 500;



UPDATE *ineuron1*.employee

SET

    city='New Mumbai',

    created\_at='2023-05-01 22:25:54',

    emp\_name='Vishal',

    state='Maharashtra',

    updated\_at='2023-05-01 22:25:54'

WHERE emp\_id=37;

UPDATE *ineuron1*.employee

SET

    city='New Mumbai',

    created\_at='2023-05-01 22:25:54',

    emp\_name='Vishal',

    state='Maharashtra',

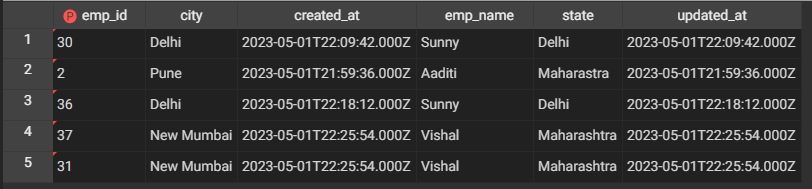
    updated\_at='2023-05-01 22:25:54'

WHERE emp\_id=37;

SELECT emp\_id, city, created\_at, emp\_name, state, updated\_at

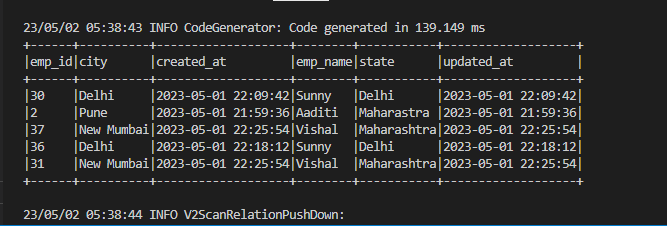
FROM *ineuron1*.employee

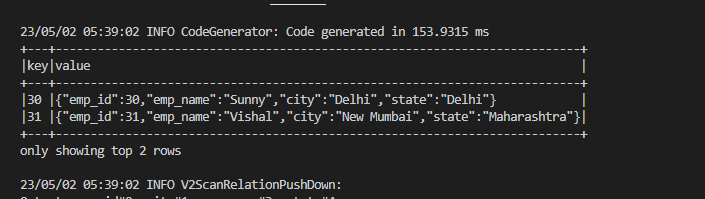
LIMIT 500;



**Again launch producer script**

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.0.1,com.datastax.spark:spark-cassandra-connector\_2.12:3.0.0 producer.py





**Again launch consumer script**

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.0.1 consumer.py

emp\_id|emp\_name|city |state |

+------+--------+----------+-------------+

|24 |Susmita |Hyderabad |Andra Pradesh|

|15 |Vishal |New Mumbai|Maharashtra |

|9 |Shashank|Delhi |Delhi |

|4 |Stephen |Mumbai |Maharashtra |

|18 |Adam |Pune |Maharashtra |

|11 |Krish |Nagpur |Maharashtra |

|27 |Ankit |Pune |Maharashtra |

|6 |Barton |Pune |Maharashtra |

|26 |Pia |Chennai |Tamilnadu |

|20 |Robert |Noida |Uttar Pradesh|

|21 |Shivam |Delhi |Delhi |

|19 |Scarlett|Hyderabad |Andra Pradesh|

|31 |Abhishek|New Mumbai|Maharashtra |

|5 |Berry |Chennai |Tamilnadu |

|28 |Akash |Hyderabad |Andra Pradesh|

|23 |Amit |Nagpur |Maharashtra |

|13 |Rahul |Noida |Uttar Pradesh|

|22 |Deepak |New Mumbai|Maharashtra |

|25 |Iris |Mumbai |Maharashtra |

|16 |Vikash |Nagpur |Maharashtra |

+------+--------+----------+-------------+

only showing top 20 rows