Week5_FinalAssignment_MySolutions

Instructions

Now that you are equipped with the knowledge and skills to extract, transform and load data you will use these skills to perform ETL, create a pipeline and upload the data into a database. You will use both Airflow and Kafka in the Hands-on labs.

Scenario

You are a data engineer at a data analytics consulting company. You have been assigned to a project that aims to de-congest the national highways by analyzing the road traffic data from different toll plazas. Each highway is operated by a different toll operator with different IT setup that use different file formats. **In the first Hands-on lab your job is to collect data available in different formats and, consolidate it into a single file.**

As a vehicle passes a toll plaza, the vehicle's data like vehicle_id,vehicle_type,toll_plaza_id and timestamp are streamed to Kafka. **In the second Hands-on lab your job is to create a data pipe line that collects the streaming data and loads it into a database.**

Grading Criteria

There are a total of 30 points possible for this final project.

Your final assignment will be graded by your peers who are also completing this assignment within the same session. Your grade will be based on the following tasks:

- Task 1.1: Define DAG arguments (2pts)
- Task 1.2: Define the DAG (2pts)
- Task 1.3: Create a task to download data (2pts)
- Task 1.4: Create a task to extract data from csv file (2pts)
- Task 1.5: Create a task to extract data from tsv file (2pts)
- Task 1.6: Create a task to extract data from fixed width file (2pts)
- Task 1.7: Create a task to consolidate data extracted from previous tasks (2pts)
- Task 1.8: Transform the data (2 pts)
- Task 1.9: Define the task pipeline (1pt)
- Task 1.10: Submit the DAG (1pt)
- Task 1.11: Unpause the DAG (1pt)
- Task 1.12: Monitor the DAG (1pt)
- Task 2.1: Start Zookeeper (1pt)

- Task 2.2: Start Kafka server (1pt)
- Task 2.3: Create a topic named toll (1pt)
- Task 2.4: Download the Toll Traffic Simulator (1pt)
- Task 2.5: Configure the Toll Traffic Simulator (1pt)
- Task 2.6: Run the Toll Traffic Simulator (1pt)
- Task 2.7: Configure streaming_data_reader.py (2pts)
- Task 2.8: Run streaming_data_reader.py (1pt)
- Task 2.9: Health check of the streaming data pipeline (1pt)

My Solutions

Part I

Task 1.1 - Define DAG arguments

```
default_args = {
    'owner': 'jalvi',
    'start_date':  dt.datetime(2022, 9, 9),
    'email': ['joaopaulonobregaalvim@gmail.com'],
    'email_on_failure': True,
    'email_on_retry': True,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}
```

Task 1.2: Define the DAG

```
dag = DAG(
    'ETL_toll_data',
    default_args=default_args,
    description='Apache Airflow Final Assignment',
    schedule_interval='@daily',
)
```

Task 1.3 - Create a task to unzip data

```

unzip_data = BashOperator(
    task_id='unzip',
    bash_command='tar -xvzf tolldata.tgz',
    dag=dag,
)
```

Task 1.4 - Create a task to extract data from csv file

```
extract_data_from_csv = BashOperator(
    task_id="extract_csv",
    bash_command="sudo chmod 777 /home/project/airflow/dags/finalassignment
| cut -d',' -f1,2,3,4  /home/project/airflow/dags/finalassignment/vehicle-
data.csv >  /home/project/airflow/dags/finalassignment/csv_data.csv",
    dag=dag

)
```

Task 1.5 - Create a task to extract data from tsv file

```
extract_data_from_tsv = BashOperator(

    task_id = "extract_tsv",
    bash_command="cut -d$'\t'  -f5,6,7
/home/project/airflow/dags/finalassignment/tollplaza-data.tsv  --output-
delimiter=',' > /home/project/airflow/dags/finalassignment/tsv_data.csv",
    dag=dag
)
```

Task 1.6 - Create a task to extract data from fixed width file

```
extract_data_from_fixed_width = BashOperator(

    task_id = "extract_fixed_width",
    bash_command="cut -b 49-57,59-61
/home/project/airflow/dags/finalassignment/payment-data.txt --output-
delimiter=','  >
/home/project/airflow/dags/finalassignment/fixed_width_data.csv",
    dag=dag
)
```

Task 1.7 - Create a task to consolidate data extracted from previous tasks

```
consolidate_data = BashOperator(
task_id = "consolidate",
bash_command = "paste -d ','
/home/project/airflow/dags/finalassignment/csv_data.csv
/home/project/airflow/dags/finalassignment/fixed_width_data.csv
/home/project/airflow/dags/finalassignment/tsv_data.csv  >
```

```
/home/project/airflow/dags/finalassignment/extracted_data.csv",
dag = dag
)
```

```
transform_data = BashOperator(
    task_id="transform",
    bash_command = "awk '$5 = toupper($5)'
/home/project/airflow/dags/finalassignment/extracted_data.csv >
/home/project/airflow/dags/finalassignment/staging/transformed_data.csv",
    dag=dag
)
```

Task 1.9 - Define the task pipeline

`unzip_data >> extract_data_from_csv >> extract_data_from_tsv >> extract_data_from_fixed_width >> consolidate_data >> transform_data`

Task 1.10 - Submit the DAG

`sudo cp ETL_toll_data.py $AIRFLOW_HOME/dags`

Task 1.11 - Unpause the DAG

`airflow dags unpause ETL_toll_data`

Part II

You are a data engineer at a data analytics consulting company. You have been assigned to a project that aims to de-congest the national highways by analyzing the road traffic data from different toll plazas. As a vehicle passes a toll plaza, the vehicle's data like vehicle_id,vehicle_type,toll_plaza_id and timestamp are streamed to Kafka. Your job is to create a data pipe line that collects the streaming data and loads it into a database.

- Step 1: Download Kafka.
  `wget https://archive.apache.org/dist/kafka/2.8.0/kafka_2.12-2.8.0.tgz`
- Step 2: Extract Kafka.
  `tar -xzf kafka_2.12-2.8.0.tgz`
- Step 3: Start MySQL server.
  `start_mysql`
- Step 4: Connect to the mysql server, using the command below. Make sure you use the password given to you when the MySQL server starts. Please make a note or record of the password because you will need it later.

```
mysql --host=127.0.0.1 --port=3306 --user=root --password=Mjk2MDItam9hb3Bh
```

- Step 5: Create a database named tolldata.
  `create database tolldata;`

- Step 6: Create a table named livetolldata with the schema to store the data generated by the traffic simulator.

```
use tolldata;

create table livetolldata(timestamp datetime,vehicle_id int,vehicle_type
char(15),toll_plaza_id smallint);
```

This is the table where you would store all the streamed data that comes from kafka. Each row is a record of when a vehicle has passed through a certain toll plaza along with its type and anonymized id.

- Step 7: Disconnect from MySQL server.
  `exit`
- Step 8: Install the python module kafka-python using the pip3 command.
  `pip3 install kafka-python`
- Step 9: Install the python module mysql-connector-python using the pip3 command.
  `pip3 install mysql-connector-python`

- Task 2.1 - Start Zookeeper
  Start zookeeper server.

```
cd kafka_2.12-2.8.0
bin/zookeeper-server-start.sh config/zookeeper.properties
```

-Task 2.2 - Start Kafka server
`cd kafka_2.12-2.8.0 bin/kafka-server-start.sh config/server.properties`

- Task 2.3 - Create a topic named toll

```
cd kafka_2.12-2.8.0
bin/kafka-topics.sh --create --topic toll --bootstrap-server localhost:9092
```

- Task 2.4 Download the toll_traffic_generator.py from the url given below using 'wget'.

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-
DB0250EN-SkillsNetwork/labs/Final%20Assignment/toll_traffic_generator.py
```

- Task 2.5 - Configure the Toll Traffic Simulator
  Open the toll_traffic_generator.py and set the topic to toll.

```
"""
Top Traffic Simulator
"""
from time import sleep, time, ctime
from random import random, randint, choice
from kafka import KafkaProducer
```

```
producer = KafkaProducer(bootstrap_servers='localhost:9092')

TOPIC = 'toll'

VEHICLE_TYPES = ("car", "car", "car", "car", "car", "car", "car", "car",
                 "car", "car", "car", "truck", "truck", "truck",
                 "truck", "van", "van")
for _ in range(100000):
    vehicle_id = randint(10000, 10000000)
    vehicle_type = choice(VEHICLE_TYPES)
    now = ctime(time())
    plaza_id = randint(4000, 4010)
    message = f"{now},{vehicle_id},{vehicle_type},{plaza_id}"
    message = bytearray(message.encode("utf-8"))
    print(f"A {vehicle_type} has passed by the toll plaza {plaza_id} at
{now}.")
    producer.send(TOPIC, message)
    sleep(random() * 2)
```

- Task 2.6 - Run the Toll Traffic Simulator
  Run the toll_traffic_generator.py.

`python3 toll_traffic_generator`

- Task 2.7 - Configure streaming_data_reader.py
  Download the streaming_data_reader.py from the url below using 'wget'.

```
"""
Streaming data consumer
"""
from datetime import datetime
from kafka import KafkaConsumer
import mysql.connector

TOPIC='toll'
DATABASE = 'tolldata'
USERNAME = 'root'
PASSWORD = 'Mjk2MDItam9hb3Bh'

print("Connecting to the database")
try:
    connection = mysql.connector.connect(host='localhost',
database=DATABASE, user=USERNAME, password=PASSWORD)
except Exception:
```

```
    print("Could not connect to database. Please check credentials")
else:
    print("Connected to database")
cursor = connection.cursor()

print("Connecting to Kafka")
consumer = KafkaConsumer(TOPIC)
print("Connected to Kafka")
print(f"Reading messages from the topic {TOPIC}")
for msg in consumer:

    # Extract information from kafka

    message = msg.value.decode("utf-8")

    # Transform the date format to suit the database schema
    (timestamp, vehcile_id, vehicle_type, plaza_id) = message.split(",")

    dateobj = datetime.strptime(timestamp, '%a %b %d %H:%M:%S %Y')
    timestamp = dateobj.strftime("%Y-%m-%d %H:%M:%S")

    # Loading data into the database table

    sql = "insert into livetolldata values(%s,%s,%s,%s)"
    result = cursor.execute(sql, (timestamp, vehcile_id, vehicle_type,
plaza_id))
    print(f"A {vehicle_type} was inserted into the database")
    connection.commit()
connection.close()
```

- Task 2.8 - Run streaming_data_reader.py
  `python3 streaming_data_reader.py`
- Task 2.9 - Health check of the streaming data pipeline.

`mysql> select * from livetolldata limit 10;`