syedprogramming@gmail.com

QUESTION -1 (General Coding)

Description: From the sample data given below, remove duplicates on the combination of Name and Age and print results.

- Please do not use high level API/Framework like pandas /spark-sql etc.
- Solve this problem by using simple data structures given in a programming language.
- Please try to optimize the solution for efficiency in terms of space and time.

Solution:

```
data = [["Amit", 21, "London"],
        ["Cynthia", 28, "Belfast"],
        ["Wendy", 26, "Manchester"],
        ["Gareth", 21, "Cardiff"],
        ["Charles", 29, "Edinburgh"],
        ["Amit", 21, "London"]]

data_all = []
keys = set()

for i in data:
    key = (i[0], i[1])
    if key not in keys:
        data_all.append(i)
        keys.add(key)

for row in data_all:
    print(row)
```

syedprogramming@gmail.com

QUESTION - 2

Description: Given a time series data which is a clickstream of user activity is stored in any flat flies, ask is to enrich the data with session id.

Session Definition:

- Session expires after inactivity of 30 mins, because of inactivity no clickstream record will be generated.
- Session remains active for a total duration of 2 hours

Steps:

- Load Data in any flat file format.
- Read the data and use spark batch (pyspark/scala) to do the computation.
- Save the results in parquet with enriched data. Note: Please do not use direct spark-sql.

Solution:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import lag, unix_timestamp, col, when
from pyspark.sql.window import Window

# Initialize SparkSession
spark = SparkSession.builder.appName("Clickstream Session
Enrichment").getOrCreate()

# Read input data
input_data = spark.read.option("header",
"true").csv("/home/jasyed/data.csv")

# Convert timestamp to Unix timestamp
input_data = input_data.withColumn("timestamp", unix_timestamp("timestamp",
"yyyy-MM-dd'T'HH:mm:ss'Z'"))

# Sort data by user and timestamp
sorted_data = input_data.sort(["userid", "timestamp"])

# Define window to partition data by user and order by timestamp
w = Window.partitionBy("userid").orderBy("timestamp")

# Calculate time since last click for each user
sorted_data = sorted_data.withColumn("time_since_last_click",
input_data.timestamp - lag(input_data.timestamp, 1).over(w))

# Define a new session for each user if time since last click is greater
than 30 minutes
sorted_data = sorted_data.withColumn("new_session",
when(sorted_data.time_since_last_click.isNull() |
(sorted_data.time_since_last_click >= 1800), 1).otherwise(0))

# Define a cumulative session ID for each user
sorted_data = sorted_data.withColumn("session_id",
col("new_session").cumsum())
```

```
# Define a new session if the session has been inactive for more than 2
hours
sorted_data = sorted_data.withColumn("time_since_start",
sorted_data.timestamp - sorted_data.first("timestamp").over(w))
sorted_data = sorted_data.withColumn("new_session",
when(sorted_data.time_since_start >= 7200, 1).otherwise(0))

# Define a cumulative session ID for each user
sorted_data = sorted_data.withColumn("session_id",
col("new_session").cumsum())

# Write output to Parquet file
sorted_data.write.mode("overwrite").parquet("/home/jasyed/sessions_stream_o
utput.parquet")

# Stop SparkSession
spark.stop()
```

syedprogramming@gmail.com

QUESTION 3

Description: In addition to the problem statement given in question 2 assume below scenario as well

and design schema based on it:

- Get Number of sessions generated in a day.
- Total time spent by a user in a day
- Total time spent by a user over a month.

Here are the guidelines and instructions for the solution of above queries:

- Design the table in any flat file format
- Write the script to create the file
- Load data into file
- Write all the queries in spark-sql
- Think in the direction of using partitioning, bucketing, etc.

Solution:

1. Table schema design:
   - userid: String
   - timestamp: Timestamp
   - time_since_last_click: Long
   - new_session: Integer
   - session_id: Integer

2. Get the number of sessions generated in a day.

```
SELECT COUNT(DISTINCT session_id) AS num_sessions
FROM table_name
WHERE DATE(timestamp) = '2023-02-18';
```

3. Get the total time spent by a user in a day.

```
SELECT userid, SUM(time_since_last_click) AS time_spent
FROM table_name
WHERE DATE(timestamp) = '2023-02-18'
GROUP BY userid;
```

4. Get the total time spent by a user over a month.

```
SELECT userid, SUM(time_since_last_click) AS time_spent
FROM table_name
WHERE YEAR(timestamp) = 2023 AND MONTH(timestamp) = 2
GROUP BY userid;
```