Questions

Q1. To emulate a live-stream of the trafc counter dataset, you are required to write a separate Python script that reads 10 records (of 2 each counter site - ignore the test site) every 5 seconds from traffc counter data fle and stores them as separate fles (countdata1, countdata2, countdata3, etc.) in the streaming directory on which your application is listening. (20 marks)

Ans: Python Code

```python
import pandas as pd
import time

input_file = "per-vehicle-records-2021-01-15.csv"
output_file_prefix = "counterdata"

def read_vehicle_data():
    print("Reading data")
    reader = pd.read_csv(input_file, sep=',', chunksize=10, iterator=True)
    for i, j in enumerate(reader):
        df = next(reader)
        output_file = output_file_prefix + str(i) + ".csv"
        df.to_csv(output_file, index=False)
        time.sleep(5)
        print("Finished chunk")


if __name__ == "__main__":
    read_vehicle_data()
```

Explanation : This python script will read the file as pandas dataframe but in chunks size of 10 that will read 10 lines of file at time period of 5 seconds (used time.sleep(5)). Once the dataframe is generated it is converted to a CSV file. The script basically simulates the streaming data from a batch data.

Q2. Prepare the streaming application to read the data streams from the streaming directory using a batch length of 5 seconds. (10 marks)

Ans: Spark Code + Output

```
1  from pyspark.streaming import StreamingContext
2  ssc = StreamingContext(sc, 5)
```

```
# define the schema
TrafSchema = StructType([ StructField("cosit", StringType(), True), StructField("year", StringType(), True), StructField("month", StringType(), True), StructField("day", StringType(), True), StructField("hour", StringType(), True), StructField("minute", StringType(), True), StructField("second", StringType(), True), StructField("millisecond", StringType(), True), StructField("minuteofday", StringType(), True), StructField("lane", StringType(), True), StructField("lanename", StringType(), True), StructField("straddlelane", StringType(), True), StructField("straddlelanename", StringType(), True), StructField("class", StringType(), True), StructField("classname", StringType(), True), StructField("length", StringType(), True), StructField("headway", StringType(), True), StructField("gap", StringType(), True), StructField("speed", StringType(), True), StructField("weight", StringType(), True), StructField("temperature", StringType(), True), StructField("duration", StringType(), True), StructField("validitycode", StringType(), True),StructField("numberofaxles", StringType(), True),StructField("axleweights", StringType(), True),StructField("axlespacings", StringType(), True)])

```

```
streamingInputDF = (
  spark
    .readStream
    .schema(TrafSchema)
    .option("maxFilesPerTrigger", 1)  # Treat a sequence of files as a stream by picking one file at a time
    .format("csv")
    .load("dbfs:/FileStore/shared_uploads/email@gmail.com/streaming-files/")
)
```

Explanation : The code here is setting a streaming context in spark using a batch length of 5 seconds that means the streams will be read as a batch for 5 seconds for doing aggregations. It is also needed to set a schema for reading the input streams and setting the maxFilesPerTrigger as 1 to pick one file at a time as a stream.

Q3. Show total number of counts (on each site of M50) by vehicle class.

Ans: Spark code + Cassandra output

```
# Total number of counts by vehicle class
streamingVehicleClassCountsDF = (
  streamingInputDF
    .groupBy(
      streamingInputDF.classname)
    .count()
)
```

```
# Writing the Dataframe to Cassandra
%scala
//Writing the dataframe directly to cassandra
import org.apache.spark.sql.cassandra._


streamingVehicleClassCountsDF.write
  .format("org.apache.spark.sql.cassandra")
  .mode("overwrite")
  .options(Map( "table" -> "streamingVehicleClassCountsDF", "keyspace" -> "test_keyspace"))
  .save()
```

Explanation : Here we are grouping by vehicle class name to count the number of vehicles.

Stream 1 Output

Table

	classname	count
1	CAR	5
2	HGV_RIG	4
3	LGV	1
4	classname	1

Stream 2 Output

Table

	classname	count
1	CAR	8
2	HGV_ART	2
3	HGV_RIG	9
4	LGV	1

Stream 3 Output

Table

	classname	count
1	CAR	11
2	HGV_ART	3
3	HGV_RIG	15
4	LGV	1

Stream 4 Output

Table +

	classname	count
1	CAR	15
2	HGV_ART	4
3	HGV_RIG	17
4	LGV	4

Q4. Compute the average speed (on each site on M50) by vehicle class.

Ans: Spark + Cassandra code + output

```
import pyspark.sql.functions as F
streamingVehicleClassAvgSpeedCountsDF = (
  streamingInputDF
    .groupBy(
      streamingInputDF.classname)
    .agg(F.round(F.avg(streamingInputDF.speed), 3))
)
```

```
# Writing the Dataframe to Cassandra
%scala
//Writing the dataframe directly to cassandra
import org.apache.spark.sql.cassandra._


streamingVehicleClassAvgSpeedCountsDF.write
  .format("org.apache.spark.sql.cassandra")
  .mode("overwrite")
  .options(Map( "table" -> "streamingVehicleClassAvgSpeedCountsDF", "keyspace" -> "test_keyspace"))
  .save()
```

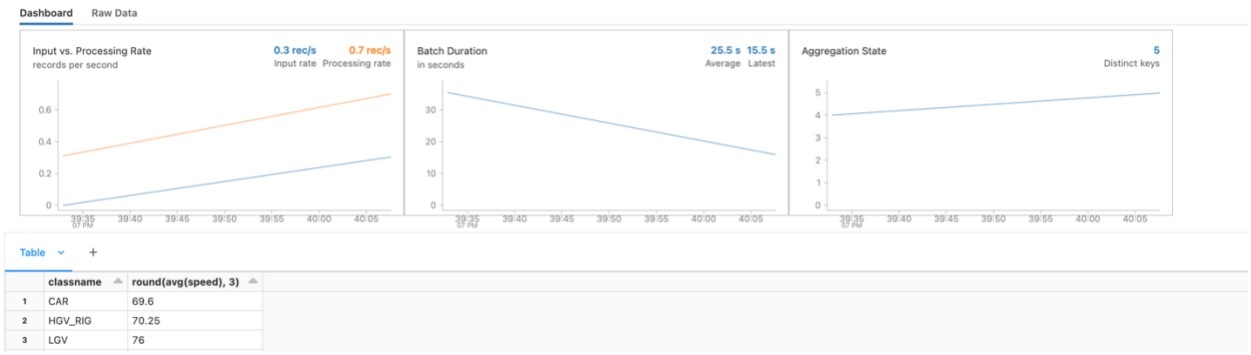
Explanation : Here we are grouping by vehicle class name and calculating the average speed

Stream 1 Output

Table +

	classname	round(avg(speed), 3)
1	CAR	70.625
2	HGV_ART	70
3	HGV_RIG	70.333
4	LGV	76

Stream 2 Output



Q5: Find the top 3 busiest counter sites on M50.

Ans: Spark + Cassandra code + Output

```
streamingCountCositDF = (
  streamingInputDF
    .groupBy(
      streamingInputDF.cosit)
    .count()
    .withColumnRenamed('count', 'total_count')
    .orderBy()
)
```

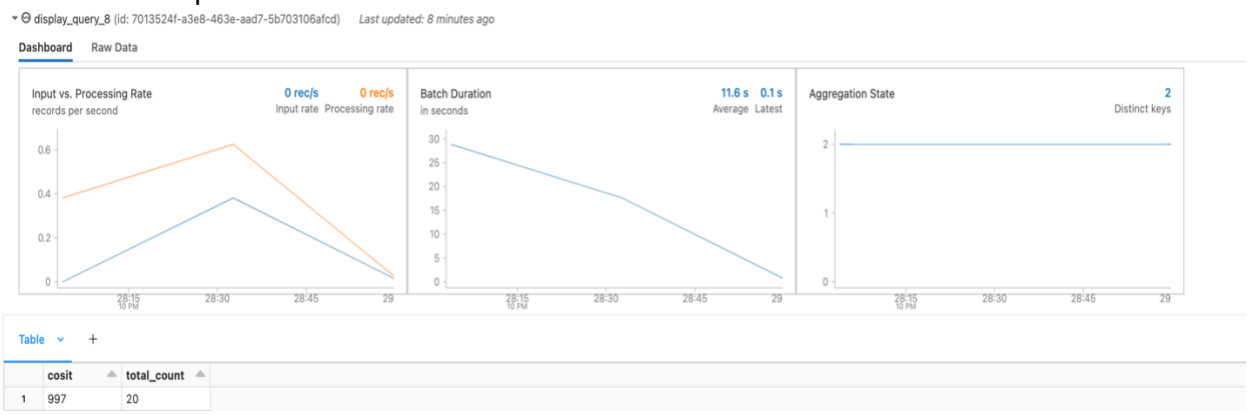
```
streamingCountCositDF.createTempView('count_cosit')
```

```
streamingTop3CositDF = spark.sql('select cosit, total_count from count_cosit order by total_count desc LIMIT 3')

```

```
# Writing the Dataframe to Cassandra
%scala
//Writing the dataframe directly to cassandra
import org.apache.spark.sql.cassandra._


streamingTop3CositDF.write
  .format("org.apache.spark.sql.cassandra")
  .mode("overwrite")
  .options(Map( "table" -> "streamingTop3CositDF", "keyspace" -> "test_keyspace"))
  .save()
```

Explanation : As streaming doesn't support ranking function yet, so using the Spark SQL to find the top 3 busiest counter sites, firstly the total count of cosit is calculated and then using that in descending order to calculate top 3 by using LIMIT 3 in the query.

Stream 1 Output



Stream 2 Output

	cosit	total_count
1	997	10

Stream 3 Output

	cosit	total_count
1	997	30

Q.6 Find total number of counts for HGVs on M50.

Ans:

```
streamingHGVCountsDF = (
  streamingInputDF
    .where((streamingInputDF.classname == 'HGV_RIG') | (streamingInputDF.classname == 'HGV_ART'))
    .groupBy(
      streamingInputDF.classname)
    .count()
)
```

```
# Writing the Dataframe to Cassandra
%scala
//Writing the dataframe directly to cassandra
import org.apache.spark.sql.cassandra._


streamingHGVCountsDF.write
  .format("org.apache.spark.sql.cassandra")
  .mode("overwrite")
  .options(Map( "table" -> "streamingHGVCountsDF", "keyspace" -> "test_keyspace"))
  .save()
```

Explanation: Here we are first filtering the vehicles by HGV and then grouping by vehicle classname to find the counts of HGV

Stream 1 Output

	classname	count
1	HGV_RIG	4

Stream 2 Output

	classname	count
1	HGV_ART	2

Stream 3 Output

	classname	count
1	HGV_ART	3