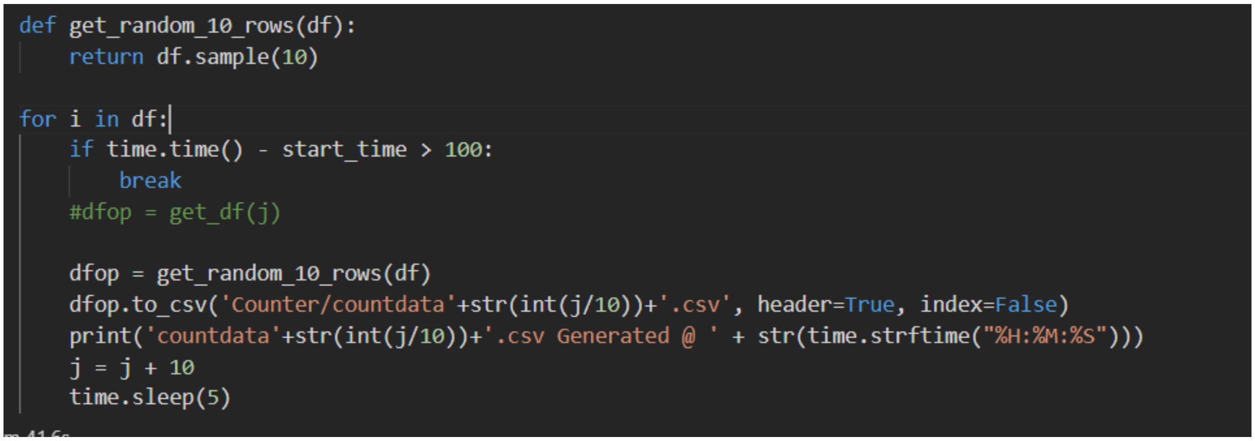
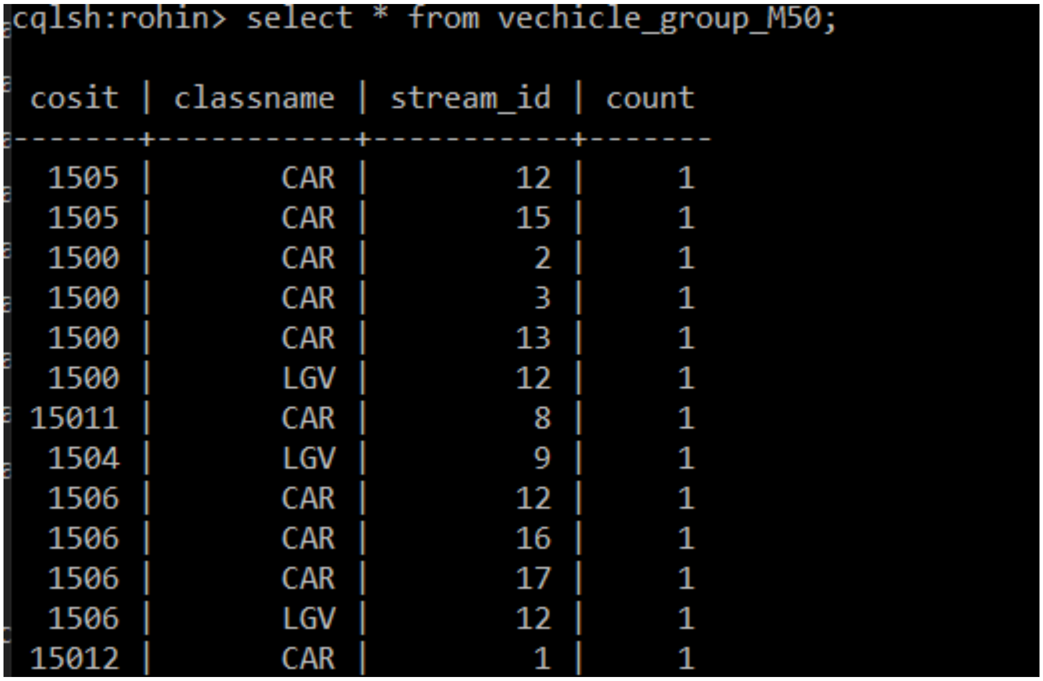
Q1 To emulate a live-stream of the traffic counter dataset, you are required to write a separate Python script that reads 10 records



```python
def get_random_10_rows(df):
    return df.sample(10)

for i in df:
    if time.time() - start_time > 100:
        break
    #dfop = get_df(j)

    dfop = get_random_10_rows(df)
    dfop.to_csv('Counter/countdata'+str(int(j/10))+'.csv', header=True, index=False)
    print('countdata'+str(int(j/10))+'.csv Generated @ ' + str(time.strftime("%H:%M:%S")))
    j = j + 10
    time.sleep(5)
```

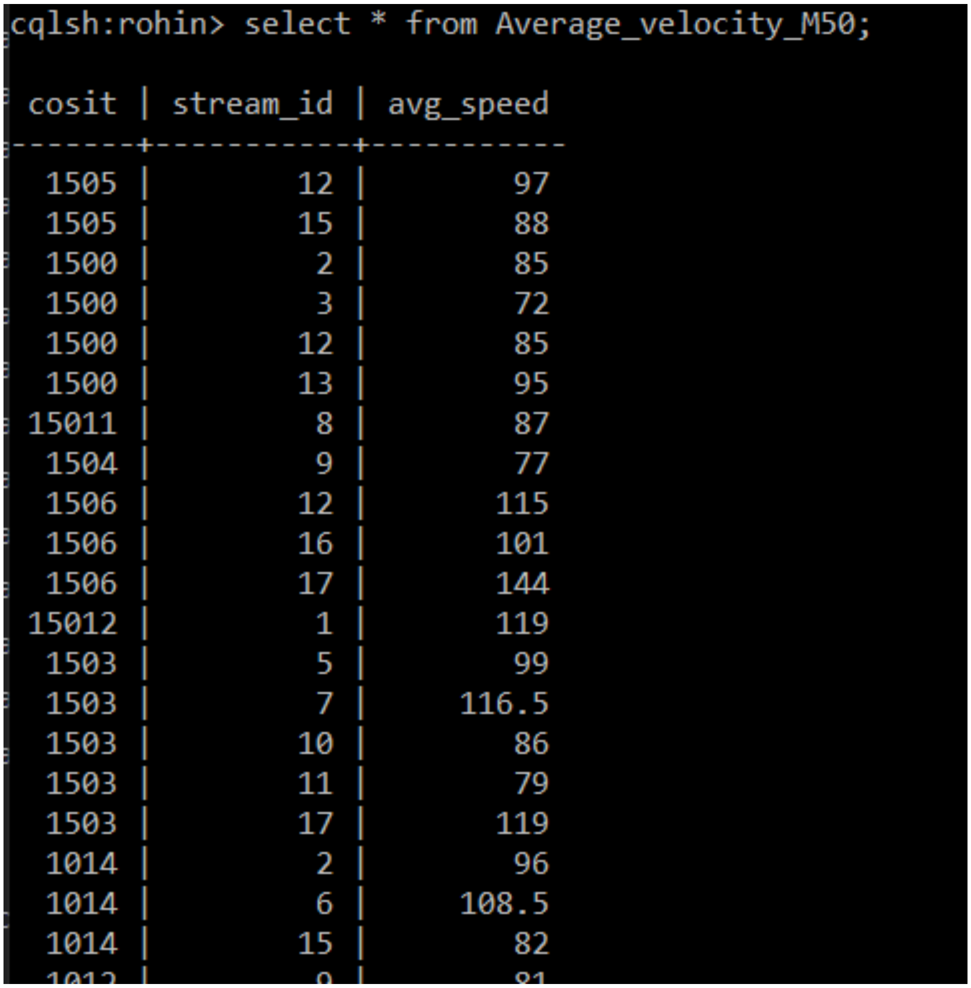
Q2

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



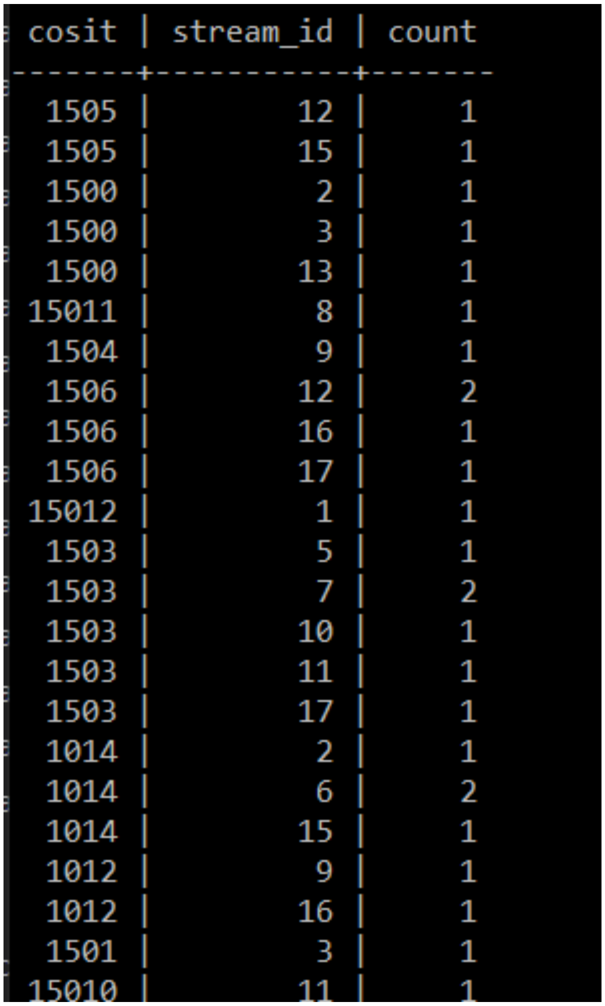
```
cqlsh:rohin> select * from vechicle_group_M50;

 cosit | classname | stream_id | count
-------+-----------+-----------+-------
  1505 |       CAR |        12 |     1
  1505 |       CAR |        15 |     1
  1500 |       CAR |         2 |     1
  1500 |       CAR |         3 |     1
  1500 |       CAR |        13 |     1
  1500 |       LGV |        12 |     1
 15011 |       CAR |         8 |     1
  1504 |       LGV |         9 |     1
  1506 |       CAR |        12 |     1
  1506 |       CAR |        16 |     1
  1506 |       CAR |        17 |     1
  1506 |       LGV |        12 |     1
 15012 |       CAR |         1 |     1
```

2. Compute the average speed (on each site on M50) by vehicle class.
(10 marks)



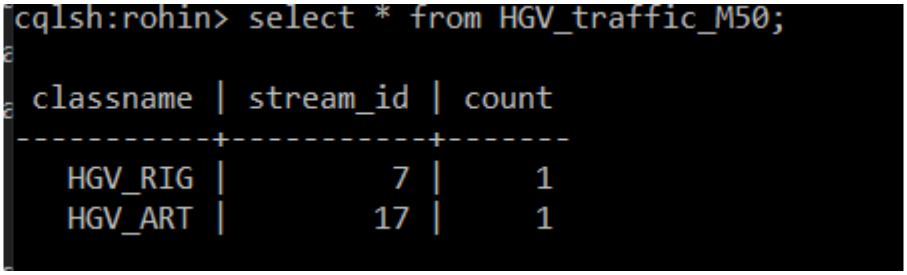
```
cqlsh:rohin> select * from Average_velocity_M50;

 cosit | stream_id | avg_speed
-------+-----------+-----------
  1505 |        12 |        97
  1505 |        15 |        88
  1500 |         2 |        85
  1500 |         3 |        72
  1500 |        12 |        85
  1500 |        13 |        95
 15011 |         8 |        87
  1504 |         9 |        77
  1506 |        12 |       115
  1506 |        16 |       101
  1506 |        17 |       144
 15012 |         1 |       119
  1503 |         5 |        99
  1503 |         7 |     116.5
  1503 |        10 |        86
  1503 |        11 |        79
  1503 |        17 |       119
  1014 |         2 |        96
  1014 |         6 |     108.5
  1014 |        15 |        82
  1012 |         9 |        81
```

3. Find the top 3 busiest counter sites on M50. (10 marks)



```
 cosit | stream_id | count
-------+-----------+-------
  1505 |        12 |     1
  1505 |        15 |     1
  1500 |         2 |     1
  1500 |         3 |     1
  1500 |        13 |     1
 15011 |         8 |     1
  1504 |         9 |     1
  1506 |        12 |     2
  1506 |        16 |     1
  1506 |        17 |     1
 15012 |         1 |     1
  1503 |         5 |     1
  1503 |         7 |     2
  1503 |        10 |     1
  1503 |        11 |     1
  1503 |        17 |     1
  1014 |         2 |     1
  1014 |         6 |     2
  1014 |        15 |     1
  1012 |         9 |     1
  1012 |        16 |     1
  1501 |         3 |     1
 15010 |        11 |     1
```

4. Find total number of counts for HGVs on M50. (10 marks)



```
cqlsh:rohin> select * from HGV_traffic_M50;

 classname | stream_id | count
-----------+-----------+-------
   HGV_RIG |         7 |     1
   HGV_ART |        17 |     1
```

Q3

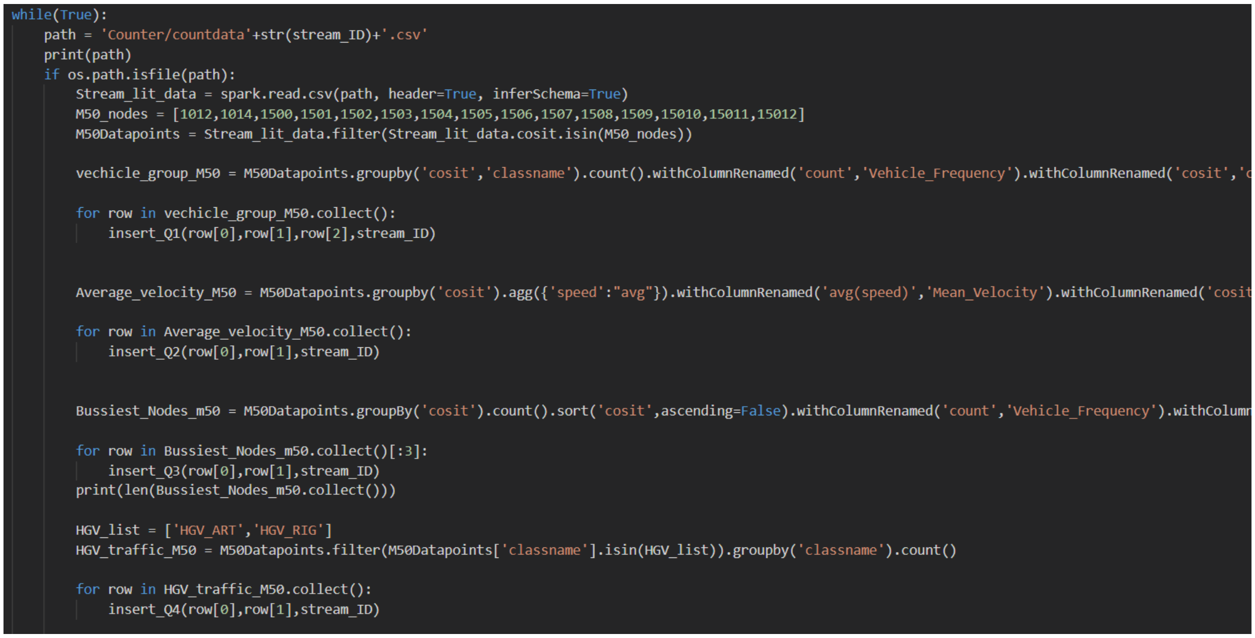
1. Prepare Cassandra data structures to store the results. (10 marks)

```
statement = session.prepare("create table rohin.vechicle_group_M50 (cosit
int, classname text, count int, stream_id int, primary key (cosit,
classname, stream_id))")
```

```
session.execute(statement)

statement = session.prepare("create table rohin.Average_velocity_M50
(cosit int, avg_speed float, stream_id int, primary key (cosit,
stream_id))")
session.execute(statement)

statement = session.prepare("create table rohin.Bussiest_Nodes_m50 (cosit
int, count int, stream_id int, primary key (cosit,stream_id))")
session.execute(statement)

statement = session.prepare("create table rohin.HGV_traffic_M50 (classname
text, count int, stream_id int, primary key (classname,stream_id))"  )
session.execute(statement)
```

2. Prepare code for writing the results into the Cassandra tables. (20 marks)

```
def insert_Q1(cosit,classname,count,stream_id):
    '''prepare statment and execute in cassandra'''
    statement = session.prepare("insert into rohin.vechicle_group_M50
(cosit, classname, count, stream_id) values (?,?,?,?)")
    session.execute(statement, (cosit,classname,count,stream_id))
def insert_Q2(cosit,avg_speed,stream_id):
    statement = session.prepare("insert into rohin.Average_velocity_M50
(cosit, avg_speed, stream_id) values (?,?,?)")
    session.execute(statement, (cosit,avg_speed,stream_id))
def insert_Q3(cosit,count,stream_id):
    statement = session.prepare("insert into rohin.Bussiest_Nodes_m50
(cosit, count, stream_id) values (?,?,?)")
    session.execute(statement, (cosit,count,stream_id))
def insert_Q4(classname,count,stream_id):
    statement = session.prepare("insert into rohin.HGV_traffic_M50
(classname, count, stream_id) values (?,?,?)")
    session.execute(statement, (classname,count,stream_id))
```



```python
while(True):
    path = 'Counter/countdata'+str(stream_ID)+'.csv'
    print(path)
    if os.path.isfile(path):
        Stream_lit_data = spark.read.csv(path, header=True, inferSchema=True)
        M50_nodes = [1012,1014,1500,1501,1502,1503,1504,1505,1506,1507,1508,1509,15010,15011,15012]
        M50Datapoints = Stream_lit_data.filter(Stream_lit_data.cosit.isin(M50_nodes))

        vechicle_group_M50 = M50Datapoints.groupby('cosit','classname').count().withColumnRenamed('count','Vehicle_Frequency').withColumnRenamed('cosit','c

        for row in vechicle_group_M50.collect():
            insert_Q1(row[0],row[1],row[2],stream_ID)


        Average_velocity_M50 = M50Datapoints.groupby('cosit').agg({'speed':"avg"}).withColumnRenamed('avg(speed)','Mean_Velocity').withColumnRenamed('cosit

        for row in Average_velocity_M50.collect():
            insert_Q2(row[0],row[1],stream_ID)


        Bussiest_Nodes_m50 = M50Datapoints.groupBy('cosit').count().sort('cosit',ascending=False).withColumnRenamed('count','Vehicle_Frequency').withColumn

        for row in Bussiest_Nodes_m50.collect()[:3]:
            insert_Q3(row[0],row[1],stream_ID)
        print(len(Bussiest_Nodes_m50.collect()))

        HGV_list = ['HGV_ART','HGV_RIG']
        HGV_traffic_M50 = M50Datapoints.filter(M50Datapoints['classname'].isin(HGV_list)).groupby('classname').count()

        for row in HGV_traffic_M50.collect():
            insert_Q4(row[0],row[1],stream_ID)
```