Homework #8

Name:

Define the location of the data sets and create the text RDDs

In [1]:
```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in ./opt/anaconda3/lib/python3.8/site-pac
kages (3.2.0)
Requirement already satisfied: py4j==0.10.9.2 in ./opt/anaconda3/lib/python3.8/s
ite-packages (from pyspark) (0.10.9.2)
```

In [2]:
```
from pyspark import SparkContext

sc=SparkContext()
```

In [3]:
```
path = '/Users/haileythanki/Desktop/'
storeRDD = sc.textFile(path+'wegmans_store_master.txt')
itemRDD = sc.textFile(path+'wegmans_item_master.txt')
customerRDD = sc.textFile(path+'wegmans_customer_master.txt')
postransRDD = sc.textFile(path+'partial_transaction.dat')
```

In [4]:
```
from pyspark.sql import SparkSession
import pyspark.sql.functions as sf
from pyspark.sql.functions import sum, col, desc
```

Define a pipe-delimited parsing function to split data based on the pipe character and repackage the returned results as a Row object.

In [5]:
```
from datetime import datetime
from pyspark.sql import Row
def parseStore(s):
    l = s.split('|')
    return Row(store_num = int(l[0]),
               store_name = l[1],
               store_zone = l[2],
               store_city = l[3],
               store_state = l[4],
               store_type = int(l[5]))
def parseItem(s):
    l = s.split('|')
    return Row(item_number = int(l[0]),
               dept_categ_class = l[1],
               item_des = l[2],
               item_unt_qty = float(l[3]),
               size_unit_desc = l[4],
               brand_code = l[5],
               dept_num = int(l[6]),
               dept_name = l[7],
               categ_num = int(l[8]),
               categ_name = l[9],
               class_num = int(l[10]),
               class_name = l[11])
def parseCustomer(s):
    l = s.split('|')
    return Row(hshld_acct = int(l[0]),
               birth_yr_head_hh = l[1],
               hh_income = l[2],
               hh_size = l[3],
               adult_count = l[4],
               child_count = l[5],
               birth_yr_oldest = l[6],
               birth_yr_youngest = l[7],
               bad_address = l[8],
               privacy = l[9],
               application_date = datetime.strptime(l[10],'%Y-%m-%d'),
               wine_email_sent = int(l[11]),
               wine_email_open = int(l[12]),
               wine_email_click = int(l[13]))
def parsePostrans(s):
    l = s.split('|')
    return Row(hshld_acct = int(l[0]),
               acct_num = int(l[1]),
               trans_num = int(l[2]),
               trans_date = datetime.strptime(l[3],'%Y-%m-%d'),
               store_num = int(l[4]),
               item_number = int(l[5]),
               dept_categ_class = l[6],
               unit_count = int(l[7]),
               net_sales = float(l[8]),
               gross_sales = float(l[9]),
               manuf_coupon = float(l[10]))
```

Generate the row-based RDDs

In [6]:

```
storeRowRDD = storeRDD.map(parseStore)
itemRowRDD = itemRDD.map(parseItem)
customerRowRDD = customerRDD.map(parseCustomer)
postransRowRDD = postransRDD.map(parsePostrans)
```

Generate data frames from the RDDs

In [7]:

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
storeDF = sqlContext.createDataFrame(storeRowRDD)
itemDF = sqlContext.createDataFrame(itemRowRDD)
customerDF = sqlContext.createDataFrame(customerRowRDD)
postransDF = sqlContext.createDataFrame(postransRowRDD)
```

```
/Users/haileythanki/opt/anaconda3/lib/python3.8/site-packages/pyspark/sql/contex
t.py:77: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreat
e() instead.
  warnings.warn(
```

Create temporary views for the data frames for use in a SQL context

In [8]:

```
storeDF.createOrReplaceTempView('store')
itemDF.createOrReplaceTempView('item')
customerDF.createOrReplaceTempView('customer')
postransDF.createOrReplaceTempView('postrans')
```

Question 1

- In the lecture we determined the total sales for each store. Now determine the total sales for each region. Your output should have 2 columns (store_zone and total_sales) and 5 rows, corresponding to the 5 regions.

In [9]:

```
((storeDF.join(postransDF,storeDF.store_num==postransDF.store_num,"inner")).grou
```

```
+-------------+-----------------+
|   store_zone|      total_sales|
+-------------+-----------------+
|    ROCHESTER|4797603.429994608|
|      BUFFALO|18560.36999999972|
|SOUTHERN TIER|6042.609999999964|
|     SYRACUSE|7057.149999999958|
|    JAMESTOWN|1177.310000000001|
+-------------+-----------------+
```

Question 2

- Calculate the total number of transactions (in the postransDF) for each store. Display the results in descending order by the number of transactions for the top 10 stores. Your output should have 2 columns (store_name and total_sales). The number of rows should be equal to 10.

In [10]:

```
((storeDF.join(postransDF,storeDF.store_num ==  postransDF.store_num,"inner")).g
```

```
+--------------------+-----------+
|          store_name|total_sales|
+--------------------+-----------+
|   WEGMANS PITTSFORD|     239862|
|     WEGMANS EASTWAY|     144698|
|   WEGMANS HOLT ROAD|     136332|
|    WEGMANS PERINTON|     113211|
| WEGMANS EAST AVENUE|     106501|
|    WEGMANS PENFIELD|      75607|
|WEGMANS CALKINS ROAD|      43176|
|    WEGMANS FAIRPORT|      33744|
|WEGMANS RIDGE-CULVER|      32448|
| WEGMANS IRONDEQUOIT|      31519|
+--------------------+-----------+
only showing top 10 rows
```

In [ ]: