q4_pyspark-gcp

March 23, 2021

1 HW3 Q4 [10 pts]

1.1 Important Notices

WARNING: Do NOT add any cells to this Jupyter Notebook, because that will cras

All instructions, code comments, etc. in this notebook **are part of the assignment instructions**.
That is, if there is instructions about completing a task in this notebook, that task is not optional.

You must implement the following functions in this notebook to receive credit

user()

load_data()

exclude_no_pickuplocations()

exclude_no_tripdistance()

include_fare_range()

get_highest_tip()

get_total_toll()

Each method will be auto-graded using different sets of parameters or data, to ensure that values are not hard-coded. You may assume we will only use your code to work with data from NYC Taxi Trips during auto-grading. You do not need to write code for unreasonable scenarios.

Since the overall correctness of your code will require multiple function to work together correctly (i.e., all methods are interdepedent), implementing only a subset of the functions likely will lead to a low score.

1.1.1 Helper functions

You are permitted to write additional helper functions, or use additional instance variables so long as the previously described functions work as required.

Pyspark Imports *Please don't modify the below cell*

[1]:
```
import pyspark
from pyspark.sql import SQLContext
from pyspark.sql import *
```

Define Spark Context *Please don't modify the below cell*

[2]:
```
sc
sqlContext = SQLContext(sc)
```

1.1.2 Student Section - Please compete all the functions below

Function to return GT Username

[3]:
```
def user():
        """
        :return: string
        your GTUsername, NOT your 9-Digit GTId
        """
        return 'wwu395'
```

Function to load data

[13]:
```
def load_data(gcp_storage_path):
    """
        :param gcp_storage_path: string (full gs path including file name e.g␣
↪gs://bucket_name/data.csv)
        :return: spark dataframe
    """

    ######################################################################
    # code to load yellow_tripdata_2019-01.csv data from your GCP storage␣
↪bucket#
    df = spark.read.csv(gcp_storage_path, header=True)
    df = df.withColumn("trip_distance",df["trip_distance"].
↪cast("Decimal(20,8)"))
    df = df.withColumn("fare_amount",df["fare_amount"].cast("Decimal(20,8)"))
    df = df.withColumn("tip_amount",df["tip_amount"].cast("Decimal(20,8)"))
    df = df.withColumn("tolls_amount",df["tolls_amount"].cast("Decimal(20,8)"))
    #                                                                    #
    ######################################################################

    return df
```

```
root
 |-- VendorID: string (nullable = true)
 |-- tpep_pickup_datetime: string (nullable = true)
 |-- tpep_dropoff_datetime: string (nullable = true)
 |-- passenger_count: string (nullable = true)
 |-- trip_distance: decimal(20,8) (nullable = true)
 |-- RatecodeID: string (nullable = true)
 |-- store_and_fwd_flag: string (nullable = true)
 |-- PULocationID: string (nullable = true)
 |-- DOLocationID: string (nullable = true)
```

```
 |-- payment_type: string (nullable = true)
 |-- fare_amount: decimal(20,8) (nullable = true)
 |-- extra: string (nullable = true)
 |-- mta_tax: string (nullable = true)
 |-- tip_amount: decimal(20,8) (nullable = true)
 |-- tolls_amount: decimal(20,8) (nullable = true)
 |-- improvement_surcharge: string (nullable = true)
 |-- total_amount: string (nullable = true)
 |-- congestion_surcharge: string (nullable = true)
```

[13]: 7667792

Function to exclude trips that don't have a pickup location

[14]:
```
def exclude_no_pickuplocations(df):
    """
        :param nyc tax trips dataframe: spark dataframe
        :return: spark dataframe
    """

    ######################################################################
    # code to exclude trips with no pickup locations                     #
    # Note: Exclude  nulls, blanks and zeros                             #
    df = df.na.drop(subset=["PULocationID"])
    ######################################################################

    return df
```

[14]: 7667792

Function to exclude trips with no distance

[16]:
```
def exclude_no_tripdistance(df):
    """
        :param nyc tax trips dataframe: spark dataframe
        :return: spark dataframe
    """

    ######################################################################
    # code to exclude trips with no trip distances                       #
    # Note: Exclude  nulls, blanks and zero                              #
    df = df.na.drop(subset=["trip_distance"])
    ######################################################################

    return df
```

[16]: 7667792

Function to include fare amount between the range of 20 to 60 Dollars

[40]:
```
def include_fare_range(df):
    """
        :param nyc tax trips dataframe: spark dataframe
        :return: spark dataframe
    """

    ##########################################################################
    # code to include trips with only within the fare range of     #
    # 20 to 60 dollars (including 20 and 60 dollars)               #
    df = df.filter((df["fare_amount"]<=60)&(df["fare_amount"]>=20))
    ##########################################################################

    return df
```

[40]:
```
969417
```

Function to get the highest tip amount

[33]:
```
def get_highest_tip(df):
    """
        :param nyc tax trips dataframe: spark dataframe
        :return: decimal (rounded to 2 digits)  (NOTE: DON'T USE FLOAT)
    """

    ##########################################################################
    # code to get the highest tip                                  #
    #                                                              #
    from pyspark.sql.functions import round
    max_df = df.agg({"tip_amount":"max"}).
 ↪withColumn("max(tip_amount)",round("max(tip_amount)",2))
    max_tip = max_df.first()[0]
    ##########################################################################

    return max_tip
```

```
+---------------+
|max(tip_amount)|
+---------------+
|         787.25|
+---------------+
```

[33]:
```
Decimal('787.25')
```

Function to get total toll amount

[34]:
```
def get_total_toll(df):
    """
        :param nyc tax trips dataframe: spark dataframe
        :return: decimal (rounded to 2 digits)  (NOTE: DON'T USE FLOAT)
    """

    ###################################################################
    # code to get total toll amount                                   #
    #                                                                 #
    from pyspark.sql.functions import round
    toll_df = df.agg({"tolls_amount":"sum"}).
↪withColumn("sum(tolls_amount)",round("sum(tolls_amount)",2))
    total_toll = toll_df.first()[0]
    ###################################################################

    return total_toll
```

[34]:
```
Decimal('2430066.70')
```

1.1.3 Run above functions and print

Uncomment the cells below and test your implemented functions

Load data from yellow_tripdata_2019-01.csv

[35]:
```
gcp_storage_path = "gs://wwu395/yellow_tripdata_2019-01.csv"
df = load_data(gcp_storage_path)
df.printSchema()
```

```
root
 |-- VendorID: string (nullable = true)
 |-- tpep_pickup_datetime: string (nullable = true)
 |-- tpep_dropoff_datetime: string (nullable = true)
 |-- passenger_count: string (nullable = true)
 |-- trip_distance: decimal(20,8) (nullable = true)
 |-- RatecodeID: string (nullable = true)
 |-- store_and_fwd_flag: string (nullable = true)
 |-- PULocationID: string (nullable = true)
 |-- DOLocationID: string (nullable = true)
 |-- payment_type: string (nullable = true)
 |-- fare_amount: decimal(20,8) (nullable = true)
 |-- extra: string (nullable = true)
 |-- mta_tax: string (nullable = true)
 |-- tip_amount: decimal(20,8) (nullable = true)
 |-- tolls_amount: decimal(20,8) (nullable = true)
 |-- improvement_surcharge: string (nullable = true)
 |-- total_amount: string (nullable = true)
 |-- congestion_surcharge: string (nullable = true)
```

Print total numbers of rows in the dataframe

[36]:
```
df.count()
```

[36]: 7667792

Print total number of rows in the dataframe after excluding trips with no pickup location

[37]:
```
df_no_pickup_locations = exclude_no_pickuplocations(df)
df_no_pickup_locations.count()
```

[37]: 7667792

Print total number of rows in the dataframe after exclude trips with no distance

[38]:
```
df_no_trip_distance = exclude_no_tripdistance(df)
df_no_trip_distance.count()
```

[38]: 7667792

Print total number of rows in the dataframe after including trips with fair amount between the range of 20 to 60 Dollars

[39]:
```
df_include_fare_range = include_fare_range(df)
df_include_fare_range.count()
```

[39]: 969417

Print the highest tip amount

[41]:
```
max_tip = get_highest_tip(df)
print(max_tip)
```

```
+---------------+
|max(tip_amount)|
+---------------+
|         787.25|
+---------------+

787.25
```

Print the total toll amount

[42]:
```
total_toll = get_total_toll(df)
print(total_toll)
```

```
2430066.70
```