به‌نام خداوند بخشنده و مهربان

تمرین اول: مقدمه‌ای بر اسپارک
استاد: محمدعلی نعمت‌بخش
درس: پایگاه‌داده پیشرفته
دستیاران: فاطمه ابراهیمی، پریسا لطیفی، امیر سرتیپی

نام و نام‌خانوادگی: سید عمید اسدالهی مجد
آدرس گیت: https://github.com/amidmajd/HW-2-spark-dataframes

در ابتدا با انجام دستورات زیر یک Session جدید اسپارک می‌سازیم و داده‌های دانلودشده را خوانده و به DataFrame تبدیل می‌کنیم. سپس برای راحتی کار نوع ستون‌های قیمت، تعداد فروخته‌شده و مقدار فروش روزانه را به نوع داده‌ای Integer تبدیل می‌کنیم.

```
[3]  from pyspark.sql import SparkSession, Window
     import pyspark.sql.functions as sf

     # Creating a Spark Session
     spark = SparkSession.builder.appName("homework").getOrCreate()
```

```
[4]  # Reading parquet files into corresponding dataframes
     products_df = spark.read.parquet("/content/products_parquet")
     sales_df = spark.read.parquet("/content/sales_parquet")
     sellers_df = spark.read.parquet("/content/sellers_parquet")

     # Converting column types to the correct ones
     products_df = products_df.withColumn('price', products_df['price'].cast('int'))
     sales_df = sales_df.withColumn('num_pieces_sold', sales_df['num_pieces_sold'].cast('int'))
     sellers_df = sellers_df.withColumn('daily_target', sellers_df['daily_target'].cast('int'))
```

۱. الف)

```
[6]  # number of products
     print('number of products:', products_df.count())

     # number of sales
     print('number of sales:', sales_df.count())

     # number of sellers
     print('number of sellers:', sellers_df.count())
```

```
number of products: 75000000
number of sales: 20000040
number of sellers: 10
```

۱. ب)

```
[7]  sales_df.filter(sales_df['num_pieces_sold']>=1).select('product_id').agg(sf.count_distinct('product_id')).show()

     +-----------------+
     |count(product_id)|
     +-----------------+
     |           993429|
     +-----------------+
```

۱. ج) در این بخش ابتدا بر اساس product_id گروه‌بندی می‌کنیم و یک دیتافریم موقت می‌سازیم و تعداد هر product_id را با count() حساب می‌کنیم. سپس با استفاده از aggregate مقدار بیشینه تعداد محصولات موجود در داده‌های فروش را حساب می‌کنیم و در متغییر highest_product_count ذخیره می‌کنیم. سپس با دستور filter شماره product_id مربوط به آن را از دیتافریم موقتی می‌یابیم و در انتها نام محصول را پیدا شد.

```
[8]  temp_df = sales_df.groupBy('product_id').count()                    # temp dataframe of each product_id & its count in sales
     highest_product_count = temp_df.agg({'count': 'max'}).collect()[0][0]   # getting maximum count  (19000000)

     temp_df.filter(temp_df['count'] == highest_product_count).show()        # getting product_id of maximum count
     products_df.filter(products_df['product_id'] == 0).show()               # getting product info

     +----------+--------+
     |product_id|   count|
     +----------+--------+
     |         0|19000000|
     +----------+--------+

     +----------+------------+-----+
     |product_id|product_name|price|
     +----------+------------+-----+
     |         0|   product_0|   22|
     +----------+------------+-----+
```

۲.

```
[9]  sales_df.groupBy('date').agg(sf.count_distinct('product_id')).show()

     +----------+-----------------+
     |      date|count(product_id)|
     +----------+-----------------+
     |2020-07-03|           100017|
     |2020-07-07|            99756|
     |2020-07-01|           100337|
     |2020-07-08|            99662|
     |2020-07-04|            99791|
     |2020-07-10|            98973|
     |2020-07-09|           100501|
     |2020-07-06|           100765|
     |2020-07-02|            99807|
     |2020-07-05|            99796|
     +----------+-----------------+
```

۳. در این بخش ابتدا دو دیتافریم فروش و محصول join زده شدند و سپس ستون جدید income با ضرب تعداد فروش هر محصول و قیمت آن به دیتافریم جدید اضافه شد. در انتها میانگین درآمد سفارشات محاسبه شد.

```
[10] sales_w_income_df = sales_df.join(products_df, sales_df['product_id'] == products_df['product_id'])
     sales_w_income_df = sales_w_income_df.withColumn('income', sales_w_income_df['num_pieces_sold'] * sales_w_income_df['price'])

[11] sales_w_income_df.agg(sf.avg('income')).show()

     +------------------+
     |       avg(income)|
     +------------------+
     |1246.1338560822878|
     +------------------+
```

۴.

```
[12] sales_w_seller_df = sales_df.join(sellers_df, "seller_id")
     sales_w_seller_df = sales_w_seller_df.withColumn('percent', sales_w_seller_df['num_pieces_sold'] / sales_w_seller_df['daily_target'])

[13] sales_w_seller_df.groupBy('seller_id').agg(sf.avg('percent')).show()

     +---------+--------------------+
     |seller_id|        avg(percent)|
     +---------+--------------------+
     |        0|2.019885898946922...|
     |        7|2.595228787788170...|
     |        3| 1.62888537056594E-4|
     |        8|9.213030375408861E-5|
     |        5|4.211073965904022E-5|
     |        6|4.782147194369122E-5|
     |        9|3.837913136180238E-5|
     |        1|1.964233366461014...|
     |        4|3.296428039825817E-5|
     |        2|6.690408001060484E-5|
     +---------+--------------------+
```

۵. الف) در این بخش ابتدا داده‌ها بر اساس seller_id گروه‌بندی شدند و جمع تعداد فروش آن‌ها حساب شد. سپس بر اساس مجموع تعداد فروش مرتب شدند تا بتوان دومین پرفروش‌ترین و دومین کم‌فروش‌ترین فروشنده‌ها بدست آورده شود.

```
[14] temp_df = sales_df.groupBy('seller_id').sum('num_pieces_sold').sort('sum(num_pieces_sold)')

[15] temp = temp_df.collect()
     print('second highest seller (id):', temp[-2][0], 'total pieces sold:', temp[-2][1])
     print('second lowest  seller (id):', temp[1][0], 'total pieces sold:', temp[1][1])

     second highest seller (id): 9 total pieces sold: 5634837
     second lowest  seller (id): 1 total pieces sold: 5598683
```

۵. ب)

```
[16] sales_df.filter(sales_df['product_id'] == 0).select('seller_id').distinct().show()

     +---------+
     |seller_id|
     +---------+
     |        0|
     +---------+
```

۶. تابع انجام عملیات خواسته شده:

```
[17] from hashlib import md5, sha256


     def apply_enc(row):
       new_row = list(row)
       new_row.append('')  # new hashed_bill column

       if int(row.order_id) % 2 == 0:
         a_count = row.bill_raw_text.count('A')
         if a_count >= 1:
           for _ in range(a_count):
             new_row[-1] = md5(row.bill_raw_text.encode('utf-8')).hexdigest()
       else:
         new_row[-1] = sha256(row.bill_raw_text.encode('utf-8')).hexdigest()

       return new_row
```

تبدیل به rdd و اعمال تابع با استفاده از map:

```
[18] new_sales_rdd = sales_df.rdd.map(lambda row: apply_enc(row))
```

تبدیل مجدد خروجی به دیتافریم با نام ستون‌های صحیح:

```
new_sales_df = new_sales_rdd.toDF(sales_df.columns + ['hashed_bill'])
new_sales_df.show()
```