به‌نام خداوند بخشنده و مهربان

تمرین دوم: کار با داده‌های حجیم
درس: تحلیل سیستم داده‌های حجیم

استاد: محمدعلی نعمت‌بخش
دستیاران: فاطمه ابراهیمی، پریسا لطیفی، امیر سرتیپی

نام و نام‌خانوادگی: مهناز توحیدی‌مهر
آدرس گیت:https://github.com/mtohidimehr/BigData_HW_Two.git

سوال ۱ : الف) تعداد سفارشات، تعداد محصولات و تعداد فروشندگان ذخیره شده در دیتاست را بدست آورید.

در ابتدا یک session ایجاد کرده و از طریق دیتاست‌های باز شده و متد read سه دیتافریم Products، Sales و Sellers را می‌سازیم. سپس با استفاده از تابع count() تعداد ذخیره شده در هر دیتافریم را به‌دست می‌آورده و در خروجی چاپ می‌کنیم.

در شکل زیر در قسمت خروجی تعداد محصولات، سفارشات و فروشنده‌ها نشان داده شده است.

```
[7]  from pyspark.sql import SparkSession
     spark = SparkSession \
         .builder \
         .appName("Query on HW2_dataset") \
         .config("spark.some.config.option", "some-value") \
         .getOrCreate()
```

```
Products = spark.read.parquet('products_parquet')
Sales = spark.read.parquet('sales_parquet')
Sellers = spark.read.parquet('sellers_parquet')

Products_number = Products.count()
Sales_number = Sales.count()
Sellers_number = Sellers.count()

print('Number of Product is:',Products_number)
print('Number of Sales is:',Sales_number)
print('Number of Sellers is:',Sellers_number)
```

```
Number of Product is: 75000000
Number of Sales is: 20000040
Number of Sellers is: 10
```

ب) تعداد محصولاتی که حداقل یک بار به فروش رسیده‌اند را به‌دست آورید.

در اینجا ابتدا دیتافریم Sale را براساس product_id ها گروه‌بندی کرده و تعداد محصولاتی که در فریم جدید داریم را شمارش می‌کنیم.

```
#Group the Sales Table With Product_id
SaledProduct = Sales.groupBy(Sales.product_id).count()
#Count the number of groups
NumberOfSaledProduct = SaledProduct.count()
print(NumberOfSaledProduct)
```

```
993429
```

ج) کدام یک از محصولات به فروش رسیده، بیشترین تکرار در سفارش‌ها را دارد؟

ابتدا جدول گروه‌بندی شده در قسمت قبل را به صورت نزولی مرتب می‌کنیم و اولین سطر را که پرفروش‌ترین محصول را نشان می‌دهد برمی‌گردانیم.

```
#Sort the SaledProduct table with count attribute and pick the first row
BestSellers = SaledProduct.sort(SaledProduct[1].desc())
BestSellers.first()
```

```
Row(product_id='0', count=19000000)
```

سوال ۲) چند محصول متمایز در هر روز به فروش می‌رسد؟

ابتدا یک دیتافریم جدید با ستون‌های روز و شماره محصول می‌سازیم و سپس سطرهای تکراری را حذف نموده و با ویژگی روز گروه‌بندی می‌کنیم. حال با شمارش این گروه‌بندی تعداد محصولات متمایز به‌دست می‌آید.

```
# Make a new Table with Date and Product_id Columns
SalesDate = Sales.select(Sales.date , Sales.product_id)
# Delete duplicate rows
SalesDate = SalesDate.dropDuplicates()
# GroupBy Date and count distinct product
SalesDate.groupBy(SalesDate.date).count().collect()
```

```
[Row(date='2020-07-03', count=100017),
 Row(date='2020-07-07', count=99756),
 Row(date='2020-07-01', count=100337),
 Row(date='2020-07-08', count=99662),
 Row(date='2020-07-04', count=99791),
 Row(date='2020-07-10', count=98973),
 Row(date='2020-07-09', count=100501),
 Row(date='2020-07-06', count=100765),
 Row(date='2020-07-02', count=99807),
 Row(date='2020-07-05', count=99796)]
```

سوال ۳) میانگین درآمد سفارشات در این دیتاست را محاسبه کنید.

ابتدا دو جدول Sales وProduct را روی ویژگی product_id با هم ineer join کرده و با استفاده از agg میانگین روی ضرب تعداد قطعات فروخته شده در قیمت هر قطعه که درآمد فروش قطعه را مشخص می‌کند، میانگین درآمد سفارشات به‌دست می‌آید.

```
print(Sales.join(Products, Sales["product_id"] == Products["product_id"], "inner").
      agg(avg(Products["price"] * Sales["num_pieces_sold"])).show())
```

```
+------------------------------+
|avg((price * num_pieces_sold))|
+------------------------------+
|            1246.1338560822878|
+------------------------------+

None
```

سوال ۴)

```
Sales.join(Sellers,Sales["seller_id"] == Sellers["seller_id"], "inner"
      ).withColumn("product_percent",Sales["num_pieces_sold"]/Sellers["daily_target"]
      ).groupBy(Sales["seller_id"]).agg(avg("product_percent")).show()
```

```
+---------+--------------------+
|seller_id|avg(product_percent)|
+---------+--------------------+
|        0|2.019885898946922...|
|        7|2.595228787788170...|
|        3| 1.62888537056594E-4|
|        8|9.2130303754088 61E-5|
|        5|4.211073965904022E-5|
|        6|4.782147194369122E-5|
|        9|3.837913136180238E-5|
|        1|1.964233366461014...|
|        4|3.296428039825817E-5|
|        2|6.690408001060484E-5|
+---------+--------------------+
```

سوال ۵)

```
# the number of pieces sell  each seller each product
Sales = Sales.groupby(col("product_id"), col("seller_id")).\
    agg(sum("num_pieces_sold").alias("num_pieces_sold"))

Wdesc = Window.partitionBy(col("product_id")).orderBy(col("num_pieces_sold").desc())
Wasc = Window.partitionBy(col("product_id")).orderBy(col("num_pieces_sold").asc())

# Make Dense Rank
Sales = Sales.withColumn("rank_asc", dense_rank().over(Wasc)).\
    withColumn("rank_desc", dense_rank().over(Wdesc))

# products that  have one row ## the products that multiple sellers sell the same amount
special_sellers = Sales.where(col("rank_asc") == col("rank_desc")).select(
    col("product_id").alias("special_sellers_product_id"), col("seller_id").alias("special_sellers_seller_id"),
    lit("from special sellers").alias("seller_kind")
)

#  the second top seller
second_seller = Sales.where(col("rank_desc") == 2).select(
    col("product_id").alias("second_seller_product_id"), col("seller_id").alias("second_seller_seller_id"),
    lit("Second top seller").alias("seller_kind")
)

# Get the least sellers
least_seller = Sales.where(col("rank_asc") == 1).select(
    col("product_id"), col("seller_id"),
    lit("Least Seller").alias("seller_kind")
).join(special_sellers, (Sales["seller_id"] == special_sellers["special_sellers_seller_id"]) & (
        Sales["product_id"] == special_sellers["special_sellers_product_id"]), "left_anti"). \
    join(second_seller, (Sales["seller_id"] == second_seller["second_seller_seller_id"]) & (
       Sales["product_id"] == second_seller["second_seller_product_id"]), "left_anti")

# show all Tables
All = least_seller.select(
    col("product_id"),
    col("seller_id"),
    col("seller_kind")
).union(second_seller.select(
    col("second_seller_product_id").alias("product_id"),
    col("second_seller_seller_id").alias("seller_id"),
    col("seller_kind")
)).union(special_sellers.select(
    col("special_sellers_product_id").alias("product_id"),
    col("special_sellers_seller_id").alias("seller_id"),
    col("seller_kind")
))
All.show()

# Which are the second top seller and least seller of product 0?
All.where(col("product_id") == 0).show()
```

```
+----------+---------+------------+
|product_id|seller_id| seller_kind|
+----------+---------+------------+
|  19986717|        1|Least Seller|
|   3534470|        3|Least Seller|
|  35669461|        4|Least Seller|
|  14542470|        5|Least Seller|
|  28592106|        5|Least Seller|
|  40496308|        5|Least Seller|
|  52606213|        7|Least Seller|
|  61475460|        7|Least Seller|
|  17944574|        8|Least Seller|
|  72017876|        1|Least Seller|
|  34681047|        5|Least Seller|
|  56011040|        5|Least Seller|
|  67723231|        5|Least Seller|
|  69790381|        5|Least Seller|
|  10978356|        7|Least Seller|
|  18182299|        7|Least Seller|
|  36269838|        8|Least Seller|
|  20774718|        9|Least Seller|
|  31136332|        9|Least Seller|
|  32602520|        9|Least Seller|
+----------+---------+------------+
only showing top 20 rows

+----------+---------+--------------------+
|product_id|seller_id|         seller_kind|
+----------+---------+--------------------+
|         0|        0|from special sellers|
+----------+---------+--------------------+
```

سوال ۶)

```python
#   Define the Hash function
def Hash_func(order_id, bill_text):
    bill_encode = bill_text.encode("utf-8")
    if int(order_id) % 2 == 0:
        ACounter = bill_text.count("A")
        for _c in range(0,  ACounter):
           bill_encode = hashlib.md5(bill_encode).hexdigest().encode("utf-8")
        bill_encode = bill_encode.decode('utf-8')
    else:
        bill_encode = hashlib.sha256(bill_encode).hexdigest()
    return bill_encode


Hash_func_udf = spark.udf.register("Hash_func", Hash_func)

#   Use the `algo_udf` to apply the aglorithm and then check if there is any duplicate hash in the table
Sales.withColumn("hashed_bill", Hash_func_udf(col("order_id"), col("bill_raw_text")))\
    .groupby(col("hashed_bill")).agg(count("*").alias("counter")).where(col("counter") > 1).show()
```

```
+-----------+---+
|hashed_bill|cnt|
+-----------+---+
+-----------+---+
```