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: نگین شمس

آدرس گیت: https://github.com/NeginShams/spark_hw2.git

- **لطفا پاسخ تمارین حتما در سامانه‌ی کوئرا ارسال شود.**
- **لطفا پاسخ‌های خود را در خود سند سوال نوشته و در قالب یک فایل PDF ارسال کنید.**
- **نام سند ارسالی HW-{homework number}-{Name Family}-{student number}**
- **تمامی فایل‌های مورد نیاز این تمرین در این لینک قابل دسترس است.**
- **خروجی از هر مرحله‌ی تمرین را در سند خود بارگذاری کنید.**
- **کد + سند را در گیت بارگذاری کرده و لینک آن را در سند قرار دهید.**
- **لینک نوتبوک و مجموعه‌ی داده**

سوال ۱:

به‌منظور خواندن داده‌های هر جدول و ذخیره‌ی آن‌ها در قالب دیتافریم، از دستورات زیر استفاده شده است:

```
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder\
        .master("local")\
        .appName("Colab")\
        .config('spark.ui.port', '4050')\
        .getOrCreate()

#read the tables and save them as dataframes
sellers_df = spark.read.parquet("/content/sellers_parquet")
sales_df = spark.read.parquet("/content/sales_parquet")
products_df = spark.read.parquet("/content/products_parquet")
```

الف) برای به‌دست آوردن تعداد سطرهای هر دیتافریم می‌توان از دستور count استفاده کرد. نتایج به‌دست آمده به‌صورت زیر می‌باشد:

```
sales_count = sales_df.count()
sellers_count = sellers_df.count()
products_count = products_df.count()

print('the number of sales: '+ str(sales_count))
print('the number of sellers: '+ str(sellers_count))
print('the number of products: '+ str(products_count))
```

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

ب) برای به‌دست آوردن محصولاتی که حداقل یک بار فروخته شده‌اند، کافی است تعداد محصولات متمایز موجود در جدول سفارش‌ها را به‌دست آوریم.

```
from pyspark.sql.functions import countDistinct
df2=sales_df.select(countDistinct('product_id'))
df2.show()
```

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

ج) برای به‌دست آوردن میزان فروش محصولات می‌توان جدول مربوط به سفارش‌ها را براساس product_id گروه‌بندی نموده و سپس تعداد عناصر هر گروه را محاسبه نمود. در انتها می‌توان محصولات را براساس میزان فروش مرتب نمود. دستورات استفاده شده برای این سوال و بخشی از نتایج به‌صورت زیر می‌باشد:

```
df2 = sales_df.groupby('product_id').count()
df2.orderBy('count', ascending=False).show()
```

```
+----------+--------+
|product_id|   count|
+----------+--------+
|         0|19000000|
|   2839667|       3|
|  28592106|       3|
|  52606213|       3|
|  36269838|       3|
|  31136332|       3|
|  18182299|       3|
|  34681047|       3|
|  40579633|       3|
```

سوال ۲:

برای به‌دست آوردن تعداد محصولات متمایز فروخته شده در هر روز، جدول سفارشات را براساس تاریخ گروه‌بندی نموده سپس با استفاده از تابع تجمعی countDistinct تعداد محصولات متمایز در هر تاریخ محاسبه می‌شود.

```
import pyspark.sql.functions as func
sales_df.groupBy("date").agg(func.countDistinct("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|
+----------+------------------+
```

سوال ۳:

برای به‌دست آوردن درآمد حاصل از هر سفارش لازم است قیمت هر محصول را در تعداد فروخته شده از آن ضرب نماییم. برای این کار با استفاده از join می‌توان بین جدول سفارش‌ها و محصولات ارتباط برقرار کرد. سپس می‌توان یک ستون جدید با نام total price به جدول سفارشات اضافه نمود که حاوی درآمد کلی هر سفارش باشد و سپس میانگین آن ستون را محاسبه نمود.

```
from pyspark.sql.functions import col

new_df = sales_df.join(products_df, sales_df.product_id== products_df.product_id
                       ).select(sales_df["*"],products_df["price"])

df2=new_df.select(((col("price") * col("num_pieces_sold"))).alias('total_price'))

df2.agg({'total_price': 'avg'}).show()
```

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

سوال ۴:

برای به‌دست آوردن درصد سهم یک سفارش از درآمد روزانه‌ی فروشنده‌ی آن، مبلغ به‌دست آمده از آن سفارش را بر فروش روزانه‌ی فروشنده‌ی آن (daily target) تقسیم نموده و ستونی جدا در جدول سفارشات به آن اختصاص می‌دهیم. برای این کار از جدول به‌دست آمده در سوال قبل که حاصل اضافه کردن ستون total price به جدول سفارشات بود استفاده شده است. سپس این جدول با جدول مربوط به فروشندگان پیوند داده شده است. برای درصد سهم سفارش یک ستون جدید به‌نام share_percent اضافه شده است. سپس جدول سفارشات را براساس seller_id گروه‌بندی نموده و میانگین ستون share_percent برای هر گروه محاسبه شده است.

```
# add total price
df2 = new_df.withColumn("total_price", new_df.num_pieces_sold *new_df.price)

# join sales and sellers
df_joined = df2.join(sellers_df, df2.seller_id== sellers_df.seller_id).select(df2["*"],sellers_df["daily_target"])

import pyspark.sql.functions as func
df3= df_joined.withColumn("share_percent", (df_joined.total_price / df_joined.daily_target)*100)
df3.groupBy('seller_id').agg(func.avg("share_percent")).show()
```

```
+---------+--------------------+
|seller_id|  avg(share_percent)|
+---------+--------------------+
|        7|  0.1960124630631757|
|        3|  1.2318678193054804|
|        8|   0.694606099856396|
|        5|  0.3170589299015149|
|        6| 0.36093852517481845|
|        9|  0.2905299815673646|
|        1|  1.4844178645806256|
|        4|  0.2484117370480231|
|        2|  0.5064829818617168|
|        0|0.044437489776546574|
+---------+--------------------+
```

سوال ۵:

الف) اگر منظور از میزان پرفروش یا کم‌فروش بودن، تعداد محصول فروخته شده توسط هر فروشنده باشد؛ می‌توان جدول سفارشات را براساس فروشندگان گروه‌بندی کرده سپس مجموع ستون num_pieces_sold را برای هر گروه محاسبه نمود. سپس با استفاده از تابع order_by می‌توان نتایج را مرتب نمود.

```python
from pyspark.sql.types import IntegerType
from pyspark.sql.functions import element_at

# groupby seller_id and calculate sum of sold items for each seller
df1 = sales_df.groupBy("seller_id").agg(func.sum("num_pieces_sold").alias("sum_sold"))

# turn float to integer and order
df1 = df1.withColumn("sum_sold", df1["sum_sold"].cast(IntegerType())).orderBy('sum_sold', ascending=False)
df1.show()

# second least selling
second_least  = df1.collect()[-2]
print('second least selling seller: '+ str(second_least))

#second best selling
second_best  = df1.collect()[1]
print('second best selling seller: '+ str(second_best))
```

با توجه به نتایج به‌دست آمده، فروشنده‌ی با شناسه شماره ۹، دومین پرفروش‌ترین فروشنده و فروشنده‌ی با شناسه ۱، دومین کم‌فروش‌ترین فروشنده است.

```
+---------+---------+
|seller_id| sum_sold|
+---------+---------+
|        0|959445802|
|        9|  5634837|
|        3|  5629935|
|        6|  5621048|
|        4|  5617087|
|        2|  5612623|
|        7|  5610746|
|        5|  5601350|
|        1|  5598683|
|        8|  5591198|
+---------+---------+
```

همچنین با استفاده از تابع collect می‌توان به‌صورت مستقیم دومین فروشنده کم‌فروش و پرفروش را به‌دست آورد:

```
second least selling seller: Row(seller_id='1', sum_sold=5598683)
second best selling seller: Row(seller_id='9', sum_sold=5634837)
```

ب) برای حل این سوال جدول سفارش‌ها را براساس دو ستون seller_id و product_id گروه‌بندی نموده و سپس تعداد اقلام فروخته شده برای هر گروه را با استفاده از تابع تجمعی sum محاسبه شده است. سپس با استفاده از تابع فیلتر و انتخاب سطرهای مربوط به product_id ==0 متوجه می‌شویم که این محصول تنها یک فروشنده داشته است.

```
[64] df1 = sales_df.groupBy("product_id", "seller_id").agg(func.sum("num_pieces_sold").alias("sum_sold"))

     df1 = df1.withColumn("sum_sold", df1["sum_sold"].cast(IntegerType())).filter(df1.product_id==0)
     df1.show()
```

```
+----------+---------+---------+
|product_id|seller_id| sum_sold|
+----------+---------+---------+
|         0|        0|959445802|
+----------+---------+---------+
```

همچنین می‌توان با استفاده از window و rank رتبه‌ی هر فروشنده را براساس میزان فروش محصولات مرتب نمود.

```
from pyspark.sql import Window
from pyspark.sql.functions import rank

df = sales_df.groupBy("product_id", "seller_id").agg(func.sum("num_pieces_sold").alias("sum_sold"))


windowPartition= Window.partitionBy(col("product_id")).orderBy(col("sum_sold").desc())
df1 = df.withColumn("rank", rank().over(windowPartition))

df1.filter(df1.product_id==0).show()
```

```
+----------+---------+------------+----+
|product_id|seller_id|    sum_sold|rank|
+----------+---------+------------+----+
|         0|        0|9.59445802E8|   1|
+----------+---------+------------+----+
```

همچنین به‌صورت زیر حالت‌های مختلف را می‌توان برای دومین فروشنده پرفروش و کم‌فروش بررسی نمود:

```
df1 = sales_df.groupBy("product_id", "seller_id").agg(func.sum("num_pieces_sold").alias("sum_sold"))
df2 = df1.filter(df1.product_id==0).orderBy('sum_sold', ascending=False).withColumn(
    "sum_sold", df1["sum_sold"].cast(IntegerType()))

if df2.distinct().count() == 1:
  result = df2.collect()[0]
  print("the second best and least selling is: "+ str(result))

elif df2.distinct().count() > 2:
  second_best = df2.collect()[1]
  second_least = df2.collect()[-2]
  print("the best selling: "+ str(second_best))
  print("the least selling: "+ str(second_least))

elif df2.distinct().count() == 2:
  second_best = df2.collect()[1]
  second_least = df2.collect()[1]
  print("the best selling: "+ str(second_best))
  print("the least selling: "+ str(second_least))
```

```
the second best and least selling is: Row(product_id='0', seller_id='0', sum_sold=959445802)
```

سوال ۶:

برای حل این سوال از دو تابع udf (توابع تعریف شده توسط کاربر) استفاده شده است. یکی از این توابع به‌نام even یک رشته ورودی دریافت نموده و به تعداد کاراکتر ’A‘ موجود در آن رشته، تابع md5 را بر روی آن رشته اعمال می‌کند. تابع odd نیز sha256 را بر روی رشته ورودی اعمال نموده و آن را برمی‌گرداند. سپس در کوئری استفاده شده می‌توان بررسی نمود که اگر order_id عدد زوج باشد تابع even و اگر فرد باشد تابع odd اجرا می‌شود.

```
from pyspark.sql.functions import col,when
import hashlib
from pyspark.sql.functions import udf

@udf
def even(text):
  num = text.count('A')
  for i in range(0, num):
    text = text.encode()
    text = hashlib.md5(text).hexdigest()
  return text

@udf
def odd(text):
  return hashlib.sha256(text.encode()).hexdigest()


sales_df=sales_df.withColumn("hashed_bill",
          when(((col("order_id")) % 2 == 0), even('bill_raw_text')).otherwise(odd('bill_raw_text')))
```

سپس برای دیدن عناصر تکراری می‌توان از دستور زیر استفاده نمود. با توجه به نتیجه‌ی به‌دست آمده، هیچ عنصری در این ستون تکراری نیست.

```
[51] df_with_hash=sales_df.groupBy('hashed_bill').count().filter("count > 1").show()

     +-----------+-----+
     |hashed_bill|count|
     +-----------+-----+
     +-----------+-----+
```