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: سارا معینی
آدرس گیت: https://github.com/saramoeini20/bigdata-H2.git

در این تمرین هدف ما آشنایی با دیتافریم‌ها و کار با داده‌های حجیم در موتور تحلیل spark است.

برای این منظور در ابتدا فایل دیتاست را به کمک قطعه کدی که در فایل نوت بوکی قرار گرفته است، دریافت کرده و در گام اول از حالت فشرده خارج میکنیم تا بتوان به هر کدام از جداول موجود به طور مجزا دسترسی داشت.

در ادامه یک session ساخته و سپس هر کدام از جداول را خوانده و در دیتافریم مربوطه قرار می‌دهیم (شکل ۱).

```
from pyspark.sql import SparkSession
from pyspark import SparkContext
sc = SparkContext('local')
spark = SparkSession(sc)
parDF_Product=spark.read.parquet("./products_parquet/")
parDF_Sales=spark.read.parquet("./sales_parquet/")
parDF_Sellers=spark.read.parquet("./sellers_parquet/")
```

شکل ۱

سوال ۱)

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

کافی است با تابع count() این کار ار انجام دهیم. نتیجه در شکل ۲ موجود است.

```
#1
print(parDF_Product.count())
print(parDF_Sales.count())
print(parDF_Sellers.count())
```

```
75000000
20000040
10
```

شکل ۲

ب) تعداد محصولاتی که حداقل یکبار به فروش رسیده‌اند را بدست آورید.

ابتدا ستون product-id در جدول سفارشات را انتخاب کرده و سپس با تابع distinct() و count() تعداد محصولات متمایزی که حداقل یکبار به فروش رسیده‌اند مشخص میشود.

```
#2
parDF_Sales.select('product_id').distinct().count()
```

```
993429
```

شکل ۳

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

ابتدا در جدول سفارش، بر اساس product-id گروهبندی کرده و سپس تعداد در هر گروه محاسبه شده و به صورت نزولی مرتب میشود. و سپس اولین عنصر برگردانده میشود که همان عنصر با بیشترین تکرار است یعنی محصول با id=0.

```
#3
parDF_Sales.groupBy('product_id').count().orderBy('count',ascending=False).take(1)
```

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

شکل ۴

سوال ۲)

چند محصول متمایز در هر روز به فروش می‌رسد؟

در جدول سفارشات ابتدا بر اساس تاریخ، گروه‌بندی انجام داده و سپس با تابع agg() ، تعدادproduct-idهای متمایز را حساب میکنیم (شکل ۵). برای استفاده از تابع count و distinct در agg ، تابع func را از کتابخانه ی functions ایمپورت میکنیم.

```
import pyspark.sql.functions as func
parDF_Sales.groupby('date').agg(func.expr('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|
+----------+-----------------+
```

شکل ۵

سوال ۳)

میانگین درآمد سفارشات در این دیتاست را محاسبه کنید.

ابتدا مانند شکل ۶ جدولی که حاصل join جداول سفارشات و محصولات است را ایجاد میکنیم که خروجی آن مانند شکل ۷ میشود.

```
new_df = parDF_Sales.join(parDF_Product,parDF_Sales["product_id"] == parDF_Product["product_id"])
```

شکل ۶

order_id	product_id	seller_id	date	num_pieces_sold	bill_raw_text	product_id	product_name	price
12478308	10005243	6	2020-07-04	98	qfvpgiscflyjxphcq...	10005243	product_10005243	44
12481548	1000879	5	2020-07-09	20	wdslrrocazrovktgm...	1000879	product_1000879	70
15490686	10010167	4	2020-07-05	3	veyxxxgodgNpntiXj...	10010167	product_10010167	23
12986886	10015577	3	2020-07-04	74	fzbfbqephcwfqelxu...	10015577	product_10015577	126
15996052	10017874	7	2020-07-07	80	xnadslnmyotjouDtn...	10017874	product_10017874	142
8996776	10023464	9	2020-07-03	59	jjbyqkzcimBfoehbv...	10023464	product_10023464	19
479116	10027897	5	2020-07-03	1	xyknodccptzxixqeo...	10027897	product_10027897	136

شکل ۷

سپس جدول جدیدی ساخته که شامل ستون جدید است که در آن تعداد فروش در هر سفارش، ضرب در قیمت آن محصول میشود (شکل ۸). ستون p*num.

```
from pyspark.sql.functions import col
df1=new_df.withColumn("p*num", col("num_pieces_sold")* col("price"))
df1.show()
```

order_id	product_id	seller_id	date	num_pieces_sold	bill_raw_text	product_id	product_name	price	p*num
12478308	10005243	6	2020-07-04	98	qfvpgiscflyjxphcq...	10005243	product_10005243	44	4312.0
12481548	1000879	5	2020-07-09	20	wdslrrocazrovktgm...	1000879	product_1000879	70	1400.0
15490686	10010167	4	2020-07-05	3	veyxxxgodgNpntiXj...	10010167	product_10010167	23	69.0
12986886	10015577	3	2020-07-04	74	fzbfbqephcwfqelxu...	10015577	product_10015577	126	9324.0
15996052	10017874	7	2020-07-07	80	xnadslnmyotjouDtn...	10017874	product_10017874	142	11360.0

شکل ۸

سپس مانند شکل ۹ جمع مقادیر ستون جدید یعنی p*num را حساب کرده و بر تعدا سفارشات تقسیم میکنیم تا میانگین به دست آید.

```
g=df1.agg({'p*num': 'sum'}).collect()

print(g[0][0]/parDF_Sales.count())
```

```
1246.1338560822878
```

شکل ۹

سوال ٤)

به ازای هر فروشنده، میانگین درصد سهم یک سفارش در سهمیه روزانه فروشندگان چقدر است؟

(به عنوان مثال می‌توانیم بین جدول فروشنده و همچنین جدول فروش که نمایانگر ارتباط بین سفارشات، محصولات و فروشندگان می‌باشد، ارتباط برقرار کرده و سپس مقدار درصد سهمیه را برای هر سفارش خاص محاسبه کرده و پس از محاسبه میانگین سهمیه سفارش محصولات، مقدار بدست آمده در خروجی را براساس شماره فروشنده (seller_id) گروه‌بندی کنید.)

ابتدا جداول سفارش و فروشنده را بر طبق SELLER-ID متصل میکنیم. (شکل ۱۰).

```
Sales_Seller_df = parDF_Sales.join(parDF_Sellers,["seller_id"] )
```

شکل ۱۰

سپس درصد سهمیه برای هر سفارش مانند شکل زیر محاسبه میشود. یعنی مقدار آیتم فروخته شده تقسیم بر مقدار فروش روزانه ی فروشنده که در آخر در ۱۰۰ ضرب میشود در ستون percentage قرار میگیرد.

```
Sales_Seller_df=Sales_Seller_df.withColumn("percentage", (col("num_pieces_sold")/col("daily_target"))*100)
Sales_Seller_df.show()
```

```
+---------+--------+----------+----------+---------------+--------------------+-----------+------------+--------------------+
|seller_id|order_id|product_id|      date|num_pieces_sold|       bill_raw_text|seller_name|daily_target|          percentage|
+---------+--------+----------+----------+---------------+--------------------+-----------+------------+--------------------+
|        0|       1|         0|2020-07-10|             26|kyeibuumwlyhuwksx...|   seller_0|     2500000|0.001040000000000...|
|        0|       2|         0|2020-07-08|             13|jfyuoyfkeyqkckwbu...|   seller_0|     2500000|5.2000000000000001E-4|
|        0|       3|         0|2020-07-05|             38|uyjihlzhzcswxcccx...|   seller_0|     2500000|             0.00152|
|        0|       4|         0|2020-07-05|             56|umnxvoqbdzpbwjqmz...|   seller_0|     2500000|             0.00224|
|        0|       5|         0|2020-07-05|             11|zmqexmaawmvdpqhih...|   seller_0|     2500000|              4.4E-4|
|        0|       6|         0|2020-07-01|             82|lmuhhkpyuoyslwmvX...|   seller_0|     2500000|             0.00328|
|        0|       7|         0|2020-07-04|             15|zoqweontumefxbgvu...|   seller_0|     2500000|6.0000000000000001E-4|
|        0|       8|         0|2020-07-08|             79|sgldfgtcxufasnvsc...|   seller_0|     2500000|             0.00316|
|        0|       9|         0|2020-07-10|             25|jnykelwjjebgkwgmu...|   seller_0|     2500000|               0.001|
|        0|      10|         0|2020-07-08|              8|yywjfihneygcvfnyl...|   seller_0|     2500000|3.199999999999999...|
|        0|      11|         0|2020-07-01|             10|nxwejyoeznltdhcam...|   seller_0|     2500000|3.999999999999999...|
|        0|      12|         0|2020-07-06|             45|efmymeftivwsfljzt...|   seller_0|     2500000|              0.0018|
```

و بعد بر اساس seller-id گروهبندی کرده و در آخر بااستفاده از تابع avg، میانگین percentage را برای هر گروه حساب میکنیم.

```
from pyspark.sql.functions import avg as _avg
Sales_Seller_df.groupBy('seller_id').agg(_avg('percentage')).show()
```

```
+---------+--------------------+
|seller_id|     avg(percentage)|
+---------+--------------------+
|        0|0.002019885898947...|
|        7|0.002595228787788...|
|        3| 0.01628885370565917|
|        8|0.009213030375408902|
|        5|0.004211073965904032|
|        6|0.004782147194369067|
|        9|0.003837913136180...|
|        1|  0.0196423336646103|
|        4|0.003296428039825792|
|        2|0.006690408001060533|
+---------+--------------------+
```

سوال ۵)

الف) دومین پرفروش‌ترین فروشنده و همچنین دومین کم فروش‌ترین را در بین فروشندگان بیابید.

ب) کدام فروشندگان محصول "product_id = 0" را بیابید.

الف) طبق شکل ۱۱ ابتدا جدول سفارش را طبق seller_id با تابع groupBy گروهبندی میکنیم و سپس num_pieces_sold ها را با هم جمع میکنیم تا به ازای هر فروشنده تمام ایتم هایی که در همه ی سفارش های موجود فروخته شده به دست اید. سپس آن ها را مرتب کرده و سپس طبق سوال عناصر مربوطه را برمیگردانیم. سطر اول مربوط به دومین کم فروشنده (فروشنده۱) و سطر دوم مربوط به دومین پرفروشنده (فروشنده ۹) است.

```
print(parDF_Sales.groupBy(parDF_Sales.seller_id).agg({'num_pieces_sold': 'sum'}).orderBy('sum(num_pieces_sold)',ascending=True).collect()[1])
print(parDF_Sales.groupBy(parDF_Sales.seller_id).agg({'num_pieces_sold': 'sum'}).orderBy('sum(num_pieces_sold)',ascending=False).collect()[1])
```

```
Row(seller_id='1', sum(num_pieces_sold)=5598683.0)
Row(seller_id='9', sum(num_pieces_sold)=5634837.0)
```

شکل ۱۱

ب) برای این قسمت به صورت شکل ۱۲ عمل کرده و ابتدا ستون seller-id در جدول سفارش را انتخاب کرده و سپس شرط product_id برابر با صفر را اعمال میکنیم. تا این مرحله فروشندگانی را داریم که محصول صفر را به فروش رسانده اند. چون امکان وجود عناصر تکراری وجود دارد در آخر از distinct() استفاده شده است. مقط فروشنده صفر محصول صفر را فروخته است.

```
parDF_Sales.select('seller_id').where(parDF_Sales['product_id']==0).distinct().show()
```

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

شکل ۱۲

سوال ٦)

در این قسمت ستونی به نام "hashed_bill" ایجاد کنید که به صورت زیر تعریف می‌شود:

- ✓ اگر شماره سفارش زوج (order_Id) باشد: تابع رمزنگار (Hash Function)، MD5 را به صورت متوالی روی قسمت "bill_raw_text" یک بار برای هر مقدار "A" موجود در متن اعمال کنید. (به عنوان مثال اگر متن صورتحساب به صورت "nbAAnllA" باشد، تابع hashing سه بار تکرار می‌شود.)
- ✓ اگر شماره سفارش فرد (order_id) باشد: تابع رمزنگار (Hash Function)، SHA256 را بر روی داده‌های درج شده در ستون "bill_raw_text" اعمال کنید.

در پایان وجود و یا عدم وجود موارد تکراری در ستون جدید را بررسی کنید.

ابتدا دو تابع تعریف کرده که در یکی از MD5 استفاده شده و در یکی از sha256. (شکل ۱۳).

```
import hashlib
def func_MD5(bill_raw_text):
  return hashlib.md5(bill_raw_text.encode()).digest()

def func_sha256(bill_raw_text):
  number=bill_raw_text.count('a')
  S=bill_raw_text
  hex_dig=0
  for x in range(number):
    S = hashlib.sha256(S.encode())
    hex_dig = S.hexdigest()
    S=hex_dig
  return S


func_udf_MD5 = udf(func_MD5, StringType())
func_udf_sha256 = udf(func_sha256, StringType())
```

شکل ۱۳

سپس طبق شرایط صورت سوال هر تابع را صدا میزنیم مثلا وقتی شماره سفارش زوج بود ، تابع func_udf_MD5 صدا زده میشود و کار هشینگ را انجام میدهد. (شکل ۱۴) نتیجه در شکل ۱۵ موجود است.

```
from pyspark.sql.functions import when
Sales_new = parDF_Sales.withColumn("hashed_bill", when(parDF_Sales['order_id'] %2==0,func_udf_MD5( parDF_Sales['bill_raw_text']))
                                .when(parDF_Sales['order_id'] %2!=0,func_udf_sha256( parDF_Sales['bill_raw_text'])))
```

شکل ۱۴

order_id	product_id	seller_id	date	num_pieces_sold	bill_raw_text	hashed_bill
1	0	0	2020-07-10	26	kyeibuumwlyhuwksx...	3bab1ec836f4351cb...
2	0	0	2020-07-08	13	jfyuoyfkeyqkckwbu...	[B@47595958
3	0	0	2020-07-05	38	uyjihlzhzcswxcccx...	f733a1a2278e48fa5...
4	0	0	2020-07-05	56	umnxvoqbdzpbwjqmz...	[B@7ce2e0bf
5	0	0	2020-07-05	11	zmqexmaawmvdpqhih...	27bca7d6b9249e398...
6	0	0	2020-07-01	82	lmuhhkpyuoyslwmvX...	[B@5c5d77e2
7	0	0	2020-07-04	15	zoqweontumefxbgvu...	745f83ca1f54ff242...
8	0	0	2020-07-08	79	sgldfgtcxufasnvsc...	[B@7cf98947
9	0	0	2020-07-10	25	jnykelwjjebgkwgmu...	bdb979d55f6b8aa55...
10	0	0	2020-07-08	8	yywjfihneygcvfnyl...	[B@11d51a0
11	0	0	2020-07-01	10	nxwejyoeznltdhcam...	765d7feeb8d759c05...
12	0	0	2020-07-06	45	efmymeftivwsfljzt...	[B@df71541
13	0	0	2020-07-10	63	nxhvtospPhfnkavdy...	7f63579f43f746ab5...
14	0	0	2020-07-03	22	ypyusdsjzfpfbucnn...	[B@162337eb
15	0	0	2020-07-09	75	ymjvbhaxffyjcwzyn...	e32898dc0565f91b9...
16	0	0	2020-07-10	83	phbcykkhvqsbkipwa...	[B@3ea2485b
17	0	0	2020-07-04	54	qgnGqqnjmbqZytoug...	16dd91a1e2221fc22...
18	0	0	2020-07-04	58	ozmllbabrnhebWcex...	[B@b9e485a
19	0	0	2020-07-07	33	kbrvXuzgiuinodtkg...	e37fa73adb7a0329a...
20	0	0	2020-07-09	73	jnqjzaigjtqlfwpug...	[B@7c884b6d

شکل ۱۵

طبق بررسی های انجام شده در شکل ۱۶ مشخص میشود که مقادیر تکراری وجود دارند چون کل سطرها برابر ۲۰۰۰۰۰۴۰ و سطرهای دارای hashed-bill متفاوت برابر ۱۹۹۷۶۶۹۴ شد یعنی ۲۳۳۴۶ سطر عناصر تکراری وجود دارد.

```
Sales_new.count()
```

```
20000040
```

```
Sales_new.select('hashed_bill').distinct().count()
```

```
19976694
```

شکل ۱۶

در شکل ۱۷ تعداد تکرار برای هر عبارت دیده می‌شود.

```
Sales_new.groupBy('hashed_bill').count().orderBy('count',ascending=False).show()
```

```
+-----------+-----+
|hashed_bill|count|
+-----------+-----+
|[B@45f24105|    3|
|[B@6786207f|    3|
|[B@1ba8e25f|    3|
|[B@3b5d416f|    3|
| [B@a620fbc|    3|
|[B@56fce6cc|    3|
|[B@570c1916|    3|
|[B@1079e128|    3|
|[B@39a51f24|    3|
|[B@1fde6e8c|    3|
|[B@3e4bea80|    3|
|[B@38ca20ff|    3|
|[B@1df6d6c1|    3|
|[B@1a600474|    3|
|[B@7deb107b|    3|
|[B@549b88e8|    3|
|[B@34559cdb|    3|
|[B@6e1e1a2e|    3|
|[B@7e8d2ed7|    3|
|[B@564354fe|    3|
+-----------+-----+
```

شکل ۱۷