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: فاطمه مومنی**
**آدرس گیت:** https://github.com/FatemehMomeni/BigData2.git

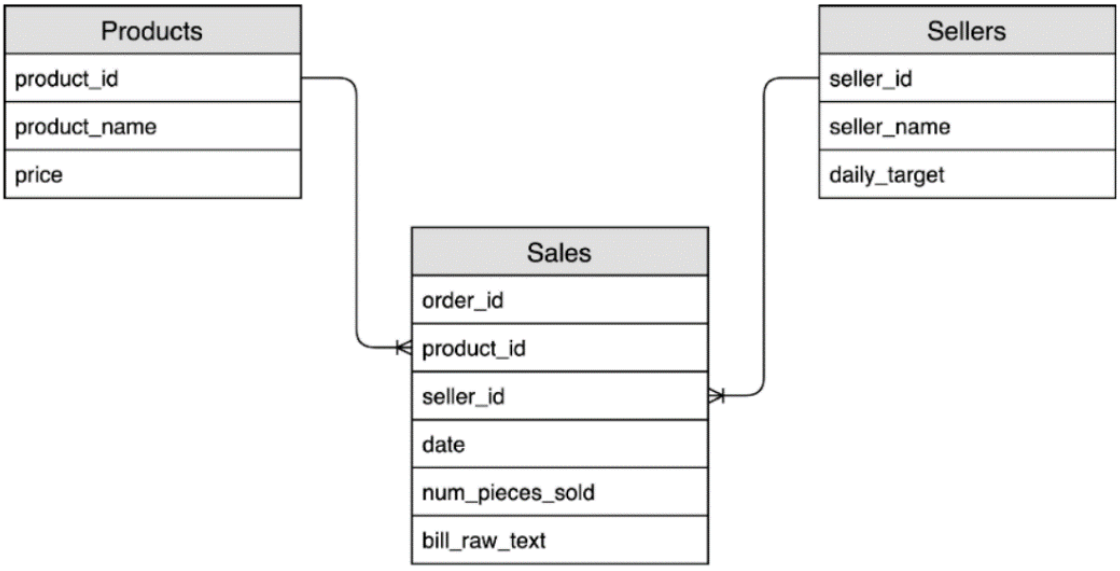
- **لطفا پاسخ تمارین حتما در سامانه‌ی کوئرا ارسال شود.**
- **لطفا پاسخ‌های خود را در خود سند سوال نوشته و در قالب یک فایل PDF ارسال کنید.**
- **نام سند ارسالی HW-{homework number}-{Name Family}-{student number}**
- **تمامی فایل‌های مورد نیاز این تمرین در این لینک قابل دسترس است.**
- **خروجی از هر مرحله‌ی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

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 که در ادامه در اختیار شما قرار گرفته است، در دسترس خواهید داشت و سپس با توجه به مجموعه داده‌های در دسترس خود با کمک زبان برنامه نویسی پایتون به سوالات مطرح شده در قسمت مربوط به همان سوال پاسخ دهید.

مجموعه داده مورد استفاده در این تمرین، از پایگاه داده یک فروشگاه، که شامل اطلاعاتی در رابطه با محصولات، فروش و فروشندگان، تشکیل شده است. نمودار رابطه موجودیت این مجموعه داده که در شکل-۱ نمایش داده می‌شود، هر کدام شامل فیلدهای زیر می‌باشند:

- ✓ محصولات (products): {کد محصول (product_id)، نام محصول (product_name)، قیمت (price)}
- ✓ فروشنده (Sellers): {کد فروشنده (seller_id)، نام فروشنده (seller_name)، مقدار فروش روزانه هر فروشنده (daily_target)}
- ✓ فروش محصولات (سفارشات): {کد سفارش (order_id)، کد محصول (product_id)، کد فروشنده (seller_id)، تاریخ (date)، تعداد محصولات فروخته شده (num_pieces_sold)، متن صورتحساب (bill_raw_text)}




فایل فشرده این مجموعه داده در لینک زیر قابل دسترس خواهد بود که با کمک دستورات برنامه نویسی در محیط گوگل کولب فراخوانی شده و در گام اول از حالت فشرده خارج می‌شود تا بتوان به هر کدام از این جداول به طور مجزا دسترسی داشت. سپس داده‌های هرکدام از جداول را بررسی کرده و از آن‌ها برای پاسخگویی به سوالات مطرح شده استفاده کنید.

---

سوالات این تمرین، به دو روش ا ستفاده از عملیات‌های spark و ا ستفاده از کتابخانه pandas و زبان SQL پا سخ داده شـده‌اند. پس از خواندن فایل‌ها از google drive به‌کمک تابع spark.read.parquet() که در شـکل ۱ آمده اسـت، با اجرای دستورات نشان داده شده در شکل ۲ data frameهای pyspark به pyspark pandas data frameهای تبدیل می شوند. data frameهای مربوط به اطلاعات محصولات، فروش و فرو شندگان به‌ترتیب در متغیرهای product، sales و sellers و products_pd، sales_pd و sellers_pd ذخیره شده است.

*Reading Files From Google Drive*

```
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("Spark1").config('spark.ui.port', '4040').getOrCreate()

products = spark.read.parquet("products_parquet")
sales = spark.read.parquet("sales_parquet")
sellers = spark.read.parquet("sellers_parquet")
```

**شکل ۱: خواندن فایل‌ها از google drive**

*Converting Pyspark DataFrame to Pyspark Pandas Dataframe*

```
import pyspark.pandas as ps

products_pd = products.to_pandas_on_spark()
sales_pd = sales.to_pandas_on_spark()
sellers_pd = sellers.to_pandas_on_spark()
```

**شکل ۲: تبدیل فایل‌ها**

## سوال ۱

الف) شکل ۳، پرسش تعداد سفارشات، تعداد محصولات و تعداد فروشندگان ذخیره شده در دیتاست و خروجی آن را نشان می‌دهد. برای به‌دست آوردن هر یک از موارد خواسته شده، باید به‌ترتیب تعداد ردیف‌های data frameهای مربوط به فروش، محصولات و فروشندگان را با تابع count() شمارش کنیم.

```
orders_num = ps.sql("select count(order_id) from {sales_pd}")
products_num = ps.sql("select count(product_id) from {products_pd}")
sellers_num = ps.sql("select count(seller_id) from {sellers_pd}")
print("Number of orders:\n"+str(orders_num)+"\n\nNumber of products:\n"+str(products_num)+"\n\nNumber of sellers:\n"+str(sellers_num)
```

```
Number of orders:
   count(order_id)
0         20000040

Number of products:
   count(product_id)
0          75000000

Number of sellers:
   count(seller_id)
0                10
```

**شکل ۳: تعداد سفارشات، محصولات و فروشندگان**

شکل ۴، پرسش قبل را با به‌کارگیری عملیات‌های spark نشان می‌دهد.

```
orders_num = sales.count()
products_num = products.count()
sellers_num = sellers.count()
print("Number of orders:\n"+str(orders_num)+"\n\nNumber of products:\n"+str(products_num)+"\n\nNumber of sellers:\n"+str(sellers_num))
```

```
Number of orders:
20000040

Number of products:
75000000

Number of sellers:
10
```

**شکل ۴: تعداد سفارشات، محصولات و فروشندگان**

ب) شکل ۵، پرسش تعداد محصولاتی که حداقل یک‌بار به فروش رسیده‌اند و خروجی آن را نشان می‌دهد. تعداد محصولات به فروش رسیده را می‌توان از شمارش ردیف‌های data frame سفارشات با حذف ردیف‌های تکراری به‌کمک عبارت distinct به‌دست آورد.

```
ps.sql("select count(distinct product_id) from {sales_pd}")
```

| | count(DISTINCT product_id) |
|---|---|
| 0 | 993429 |

**شکل ۵: تعداد محصولاتی که حداقل یک‌بار به فروش رسیده‌اند**

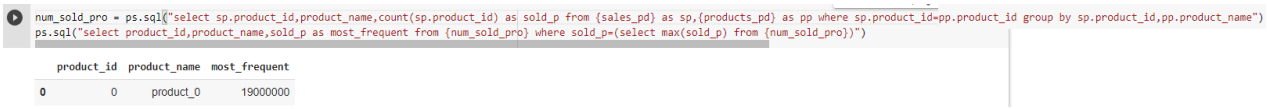
شکل ۶، پرسش قبل را با به‌کارگیری عملیات‌های spark نشان می‌دهد. در این روش، ردیف‌های data frame سفارشات با حذف ردیف‌های تکراری به‌کمک تابع countDistinct() شمارش شده است.

```
from pyspark.sql.functions import countDistinct
sold_products = sales.select(countDistinct("product_id"))
sold_products.show()
```

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

**شکل ۶: تعداد محصولاتی که حداقل یک‌بار به فروش رسیده‌اند**

ج) شکل ۷، پرسش محصول با بیشترین تکرار و خروجی آن را نشان می‌دهد. برای پاسخ به این پرسش، ابتدا با گروه‌بندی data frame سفارشات براساس product_id، تعداد تکرار هر محصول را به‌دست می‌آوریم. خروجی این پرسش که شامل کد، نام و تعداد تکرار محصول است در متغیر num_sold_pro ذخیره شده است. سپس بیشینه تعداد تکرار را از خروجی حاصل از مرحله قبل، محاسبه کرده و همراه با نام و کد محصول نمایش می‌دهیم.




**شکل ۷: محصول با بیشترین تکرار در سفارشات**

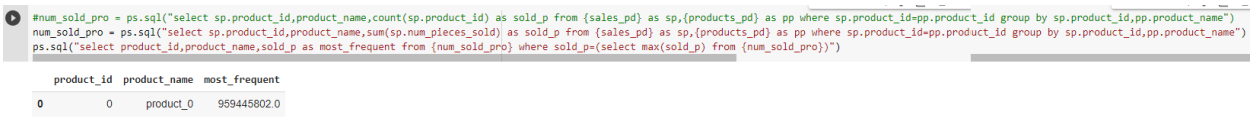
شکل ۸، پرسش قبل را با به‌کارگیری عملیات‌های spark نشان می‌دهد.

```
num_sold = sales.groupBy('product_id').count().withColumnRenamed("count","frequency")
num_sold.filter(num_sold.frequency == num_sold.agg({'frequency':'max'}).collect()[0][0]).show()
```

```
+----------+---------+
|product_id|frequency|
+----------+---------+
|         0| 19000000|
+----------+---------+
```

**شکل ۸: محصول با بیشترین تکرار در سفارشات**

روش دیگر برای به‌دست آوردن محصول با بیشترین تکرار، محاسبه مجموع num_pieces_sold پس از گروه‌بندی data frame سفارشات براساس product_id و یافتن بیشترین مقدار است که در شکل ۹ مشاهده می‌شود.




**شکل ۹: محصول با بیشترین تکرار**

شکل ۱۰،روش دوم را با به‌کارگیری عملیات‌های spark نشان می‌دهد.

```
#num_sold = sales.groupBy('product_id').count().withColumnRenamed("count","frequency")
num_sold = sales.groupBy('product_id').agg({'num_pieces_sold':'sum'}).withColumnRenamed("sum(num_pieces_sold)","frequency")
num_sold.filter(num_sold.frequency == num_sold.agg({'frequency':'max'}).collect()[0][0]).show()
```

```
+----------+-----------+
|product_id|  frequency|
+----------+-----------+
|         0|9.59445802E8|
+----------+-----------+
```

**شکل ۱۰: محصول با بیشترین تکرار**

## سوال ۲

شکل ۱۱، پرسش تعداد محصول متمایز فروخته شده در هر روز و خروجی آن را نشان می‌دهد. برای محاسبه این تعداد، پس از گروه‌بندی data frame سفارشات براساس تاریخ، تعداد محصول متمایز هر گروه را به‌کمک تابع count و عبارت distinct به‌دست می‌آوریم.

```
ps.sql("select date,count(distinct product_id) from {sales_pd} group by date")
```

| | date | count(DISTINCT product_id) |
|---|---|---|
| 0 | 2020-07-03 | 100017 |
| 1 | 2020-07-07 | 99756 |
| 2 | 2020-07-01 | 100337 |
| 3 | 2020-07-08 | 99662 |
| 4 | 2020-07-04 | 99791 |
| 5 | 2020-07-10 | 98973 |
| 6 | 2020-07-09 | 100501 |
| 7 | 2020-07-06 | 100765 |
| 8 | 2020-07-02 | 99807 |
| 9 | 2020-07-05 | 99796 |

**شکل ۱۱: تعداد محصول متمایز فروخته شده در هر روز**

شکل ۱۲، پرسش قبل را با به‌کارگیری عملیات‌های spark نشان می‌دهد.

```
sales.select('product_id','date').distinct().groupBy('date').count().show()
```

```
+----------+------+
|      date| count|
+----------+------+
|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 ۱۲: تعداد محصول متمایز فروخته شده در هر روز**

## سوال ۳

شکل ۱۳، پرسش میانگین درآمد سفارشات و خروجی آن را نشان می‌دهد. برای محاسبه میانگین، از data frameهای سفارشات و محصولات و هم‌چنین از تابع avg() استفاده شده است.

```
ps.sql("select avg(price) from {sales_pd},{products_pd} where {sales_pd}.product_id={products_pd}.product_id")
```

| | avg(price) |
|---|---|
| 0 | 24.67622 |

**شکل ۱۳: میانگین درآمد سفارشات**

## سوال ۴

شکل ۱۴، پرسش میانگین درصد سهم یک سفارش در سهمیه روزانه فروشندگان و خروجی آن را نشان می‌دهد. ابتدا درصد تعداد محصولات فروخته شده در مقدار فروش روزانه هر فروشنده را محاسبه می‌کنیم. سپس میانگین این مقادیر را به‌دست آورده و براساس کد فروشنده گروه‌بندی می‌کنیم.

```
sales_sellers = ps.sql("select sa.seller_id,sa.num_pieces_sold*100/se.daily_target as order_percent from {sales_pd} as sa,{sellers_pd} as se where sa.seller_id==se.seller_id")
ps.sql("select seller_id,avg(order_percent) from {sales_sellers} group by seller_id")
```

| | seller_id | avg(order_percent) |
|---|---|---|
| 0 | 0 | 0.002020 |
| 1 | 7 | 0.002595 |
| 2 | 3 | 0.016289 |
| 3 | 8 | 0.009213 |
| 4 | 5 | 0.004211 |
| 5 | 6 | 0.004782 |
| 6 | 9 | 0.003838 |
| 7 | 1 | 0.019642 |
| 8 | 4 | 0.003296 |
| 9 | 2 | 0.006690 |

**شکل ۱۴: میانگین درصد سهم یک سفارش در سهمیه روزانه فروشندگان**

## سوال ۵

الف) شکل ۱۵، پرسش دومین پرفروش‌ترین و دومین کم‌فروش‌ترین فروشنده و خروجی آن را نشان می‌دهد. برای پاسخ به این پرسش، ابتدا با گروه‌بندی data frame سفارشات براساس seller_id، میزان فروش هر فروشنده را به‌دست می‌آوریم. خروجی این پرسش که شامل تعداد محصول فروخته شده، کد و نام فروشنده است در متغیر num_sold_pro ذخیره شده است. سپس برای به‌دست آوردن پرفروش‌ترین و کم‌فروش‌ترین فروشندگان، این متغیر را به‌ترتیب به‌صورت نزولی و صعودی مرتب کرده و هربار دومین ردیف را بازیابی می‌کنیم.

```
num_sold_pro = ps.sql("select sp.seller_id,sep.seller_name,count(sp.seller_id) as sold_p from {sales_pd} as sp,{sellers_pd} as sep where sp.seller_id=sep.seller_id group by sp.seller_id,sep.seller_name")
second_max = ps.sql("select * from(select row_number() over(order by sold_p desc) as row_num,seller_id,seller_name,sold_p as num_sold_products from {num_sold_pro}) as sub where row_num=2")
second_min = ps.sql("select * from(select row_number() over(order by sold_p) as row_num,seller_id,seller_name,sold_p as num_sold_products from {num_sold_pro}) as sub where row_num=2")
print("The Second Best Seller:\n\n" + str(second_max) + "\n\n\nThe Second Worst Seller:\n\n" + str(second_min))
```

```
The Second Best Seller:

   row_num seller_id seller_name  num_sold_products
0        2         9    seller_9             111392


The Second Worst Seller:

   row_num seller_id seller_name  num_sold_products
0        2         5    seller_5             110874
```

**شکل ۱۵: دومین پرفروش‌ترین و دومین کم‌فروش‌ترین فروشنده**

ب) شکل ۱۶، پرسش فروشندگان محصول product_0 و خروجی آن را نشان می‌دهد. به‌این منظور، ابتدا سفارشات را براساس فروشندگان گروه‌بندی کرده و فروشندگانی که محصول product_0 را فروخته‌اند، بازیابی می‌کنیم.

```
ps.sql("select seller_id from {sales_pd} where product_id=0 group by seller_id")
```

| | seller_id |
|---|---|
| 0 | 0 |

**شکل ۱۶: فروشندگان محصول product_0**

شکل ۱۷، پرسش قبل را با به‌کارگیری عملیات‌های spark نشان می‌دهد.

```
sales.filter(sales.product_id=='0').groupBy('seller_id').agg({'num_pieces_sold':'sum'}).show()
```

```
+---------+--------------------+
|seller_id|sum(num_pieces_sold)|
+---------+--------------------+
|        0|        9.59445802E8|
+---------+--------------------+
```

**شکل ۱۷: فروشندگان محصول product_0**

## سوال ۶

شکل ۱۸، پرسش افزودن ستون مربوط به تابع رمزنگار و خروجی آن را نشان می‌دهد. تابع hashing() برای بررسی زوج یا فرد بودن کد سفارش و اعمال تابع رمزنگار مناسب به‌ازای هر مقدار "A" موجود در متن صورت‌حساب، تعریف شده است. از تابع udf() برای فراخوانی تابع hashing() و از تابع withColumn() برای ایجاد ستون جدید hashed_bill و مقداردهی آن استفاده شده است. هم‌چنین از تابع count() برای شمارش تعداد موارد گروه‌بندی شده براساس hashed_bill استفاده شده است.

```
from pyspark.sql.functions import sha2,udf,col,lit
from hashlib import md5
from pyspark.sql.types import StringType

@udf
def hashing(order_id,bill_raw_text):
  if order_id % 2 ==0:
    for l in bill_raw_text:
      if l == 'A':
        return md5(bill_raw_text.encode("utf-8") ).hexdigest()
  else:
    for l in bill_raw_text:
      if l == 'A':
        return sha2(bill_raw_text,256)

hashed = udf(lambda x: hashing(x))

sales1 = sales.withColumn("hashed_bill",lit(hashing(sales["order_id"], sales["bill_raw_text"])))
sales2 = sales1.groupBy("hashed_bill").count()
```

**شکل ۱۸: افزودن ستون تابع رمزنگاری**