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: عطیه نیکبخت
آدرس گیت: https://github.com/AtiyehNikbakht/DataFrameSparkPractice.git

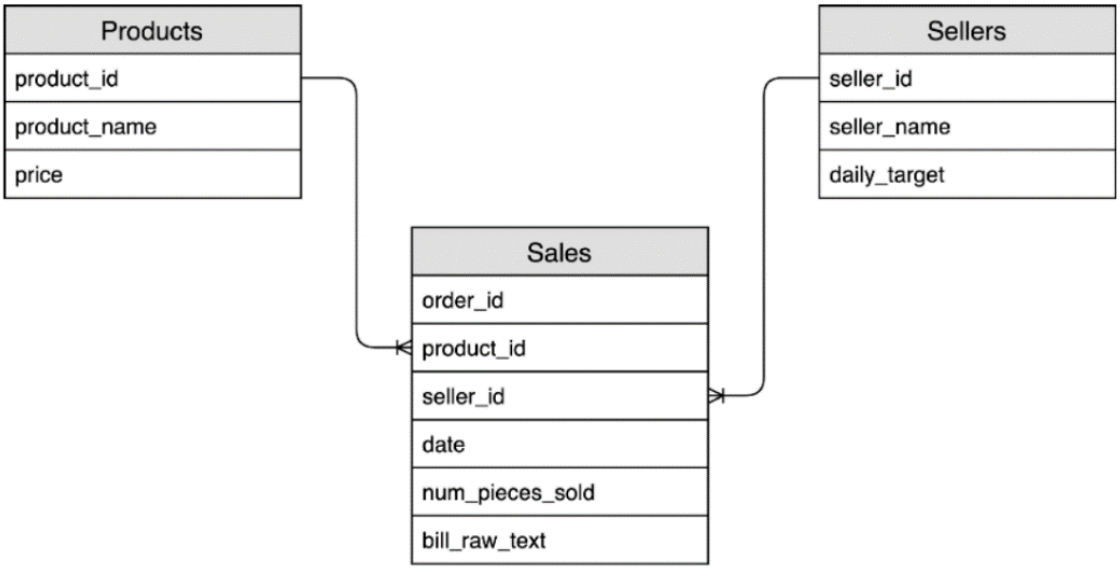
- **لطفا پاسخ تمارین حتما در سامانه‌ی کوئرا ارسال شود.**
- **لطفا پاسخ‌های خود را در خود سند سوال نوشته و در قالب یک فایل PDF ارسال کنید.**
- **نام سند ارسالی HW-{homework number}-{Name Family}-{student number}**
- **تمامی فایل‌های مورد نیاز این تمرین در این لینک قابل دسترس است.**
- **خروجی از هر مرحله‌ی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

کتابخانه‌های مورد استفاده:

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql import functions as F
import os
```

سوال ۱)

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

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

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

ابتدا با دستورات زیر جداول را به DataFrame تبدیل می‌کنیم:

```
spark = SparkSession.builder.appName("Dataframe_practice").getOrCreate()
```

```
productDf = spark.read.parquet("/content/products_parquet")
saleDf = spark.read.parquet("/content/sales_parquet")
sellerDf = spark.read.parquet("/content/sellers_parquet")
```

الف) با دستور count() تعداد سفارشات، فروشندگان و محصولات را به‌دست می‌آوریم.

```
print("Number of Products:", productDf.count())
print("Number of Sales:", saleDf.count())
print("Number of Sellers:", sellerDf.count())
```

```
Number of Products: 75000000
Number of Sales: 20000040
Number of Sellers: 10
```

ب) تعداد محصولاتی که حداقل یکبار به فروش رسیده‌اند را در زیر مشاهده می‌کنید.

```
saleDf.select('product_id').distinct().count()
```

```
993429
```

ج) محصول 0 در 19000000 سفارش تکرار شده است و بیشترین تکرار در سفارشات را دارد.

```
NumberOfsale = saleDf.groupBy('product_id').agg(F.count('order_id').alias('Number'))
NumberOfsale.filter(NumberOfsale.Number == NumberOfsale.agg(F.max('Number').alias('max_number')).collect()[0][0]).show()
```

```
+----------+--------+
|product_id|  Number|
+----------+--------+
|         0|19000000|
+----------+--------+
```

سوال ۲)

چند محصول متمایز در هر روز به فروش می‌رسد؟

براساس تاریخ محصولات را دسته‌بندی می‌کنیم و موارد تکراری را با دستور count_distinct() حذف می‌کنیم.

```
saleDf.groupBy('date').agg(F.count_distinct('product_id')).collect()
```

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

سوال ۳)

میانگین درآمد سفارشات در این دیتاست را محاسبه کنید.

میانگین درآمد سفارشات به‌صورت زیر می‌باشد.

```
saleDf = saleDf.withColumn('num_pieces_sold', saleDf['num_pieces_sold'].cast('Integer'))
productDf = productDf.withColumn('price', productDf['price'].cast('Integer'))

numProSaleDf = saleDf.groupBy('product_id').agg(F.sum('num_pieces_sold').alias('sum_pieces'))

income = numProSaleDf.join(productDf, 'product_id')
income = income.withColumn('Total', income.sum_pieces * income.price)
income.agg(F.avg('Total').alias('average_of_income')).show()
```

```
+-----------------+
|average_of_income|
+-----------------+
|25087.57743834738|
+-----------------+
```

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.)

برای هر فروشنده میانگین درصد سهم یک سفارش به‌صورت زیر می‌باشد.

```
sellerDf = sellerDf.withColumn('daily_target', sellerDf['daily_target'].cast('Integer'))

precentOfOrder = sellerDf.join(saleDf, 'seller_id', 'inner')
precentOfOrder = precentOfOrder.withColumn('precent_order', (precentOfOrder['num_pieces_sold']*100)/precentOfOrder['daily_target'])
precentOfOrder.groupBy('seller_id').agg({'precent_order': 'avg'}).show()
```

```
+---------+--------------------+
|seller_id|  avg(precent_order)|
+---------+--------------------+
|        0|0.002019885898947...|
|        7|0.002595228787788...|
|        3|0.016288853705659134|
|        8|0.009213030375408902|
|        5|0.004211073965904032|
|        6|0.004782147194369067|
|        9|0.003837913136180...|
|        1|  0.0196423336646103|
|        4|0.003296428039825...|
|        2|0.006690408001060533|
+---------+--------------------+
```

سوال ٥)

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

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

توجه:

- ✓ در حین بررسی ممکن است به محصولی برخوردکنید که تنها توسط یک فروشنده به فروش رسیده باشد، در نتیجه این محصول به عنوان یک گروه یک گروه مجزا درنظر گرفته می‌شود.
- ✓ به عنوان مثال ممکن است، "product_0"، توسط بیش از یک فروشنده به فروش رسیده باشد ولی همه فروشندگان به مقدار مساوی از این محصول را فروخته‌اند، بنابراین همه فروشندگان را در یک گروه قرار داده و فرض می‌کنیم این محصول فقط توسط یک فروشنده به فروش رسیده است.
- ✓ حتی ممکن است در این بررسی فروشنده با کمترین میزان فروش، همان دومین فروشنده باشد، آنگاه این فروشنده به‌عنوان دومین فروشنده با کمترین میزان فروش معرفی می‌شود.

الف) دومین پرفروش‌ترین و کم‌فروش‌ترین فروشنده:

```
saleOfSeller = saleDf.groupBy('seller_id').agg(F.sum('num_pieces_sold').alias('num_sale_seller')).sort('num_sale_seller')
print('The second best seller:', saleOfSeller.collect()[-2])
print('The second lowest selling seller:', saleOfSeller.collect()[2])
```

```
The second best seller: Row(seller_id='9', num_sale_seller=5634837)
The second lowest selling seller: Row(seller_id='5', num_sale_seller=5601350)
```

ب) نام فروشنده‌ای که محصول 0 را می‌فروشد به‌صورت زیر است.

```
sellerProZero = saleDf.filter('product_id == 0').select('seller_id').distinct()
sellerProZeroName = sellerProZero.join(sellerDf, 'seller_id')
sellerProZeroName.select('seller_name').show()
```

```
+-----------+
|seller_name|
+-----------+
|   seller_0|
+-----------+
```

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.

در صورتی که شماره سفارش زوج باشد در تابع تعریف شده f()، تابع md5 به تعداد کاراکترهای A روی bill_raw_text اعمال می‌شود و در غیراین‌صورت تابع sha256 اعمال می‌شود و مقادیر در ستون جدید قرار می‌گیرد. در آخر تعداد سطرها براساس تکراری بودن و یا نبودن ستون hashed_bill مورد بررسی قرار گرفته است که با توجه به نتیجه، موارد تکراری در این ستون قرار ندارد.

```
from pyspark.sql.types import StringType
import hashlib

@F.udf(returnType=StringType())
def f(billCol, countACol):
  hashBill = billCol
  a = countACol
  for i in range(0, int(a)):
    try:
      hashBill =  hashlib.md5(hashBill.encode()).hexdigest()
    except:
      break
  return hashBill

saleDf = saleDf.withColumn('order_id', saleDf['order_id'].cast('Integer'))

r = saleDf.filter(saleDf.order_id %2 == 0).select('order_id', 'bill_raw_text')
r = r.rdd.map(lambda x: (x[1].count('A'), x[0])).filter(lambda x: x[1] != '0')
r = r.toDF()

saleDf = saleDf.join(r , saleDf.order_id == r._2, 'leftouter')
saleDf = saleDf.na.fill({'_1': 0})
saleDf = saleDf.withColumn('hashed_bill', F.when(saleDf.order_id %2 == 0, f('bill_raw_text', '_1')).otherwise(F.sha2(saleDf.bill_raw_text, 256)))

print("Number of all hashed bill:", saleDf.select('hashed_bill').count())
print("Number of hashed bill without distinct:", saleDf.select('hashed_bill').distinct().count())
```

```
Number of all hashed bill: 20000040
Number of hashed bill without distinct: 20000040
```