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: سعید صدیق زاده
آدرس گیت: آدرس نوتبوک در گیت ، آدرس سند در گیت

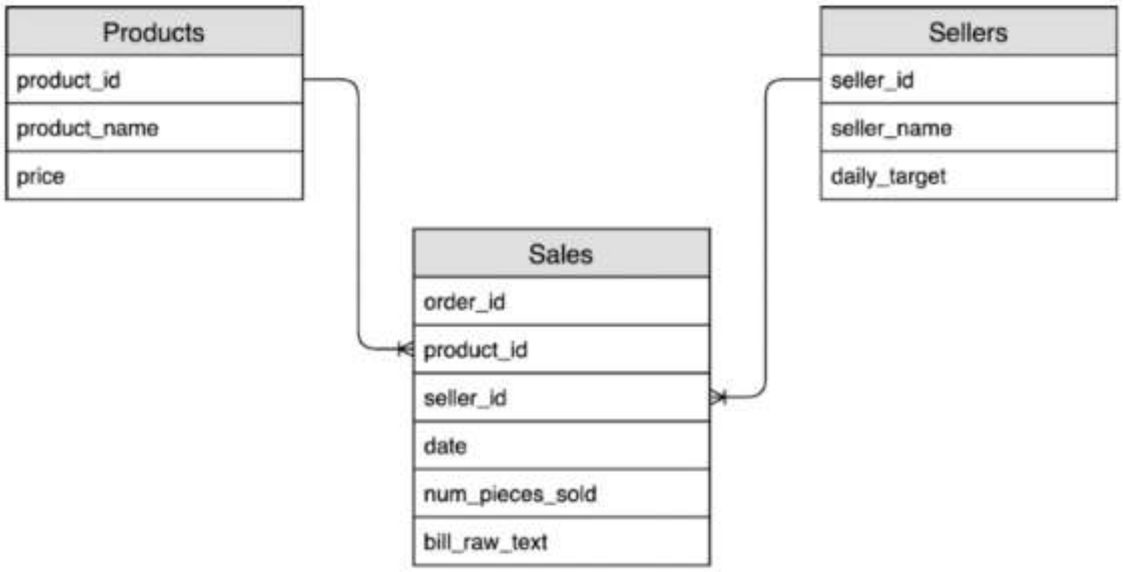
- **لطفا پاسخ تمارین حتما در سامانه‌ی کوئرا ارسال شود.**
- **لطفا پاسخ‌های خود را در خود سند سوال نوشته و در قالب یک فایل PDF ارسال کنید.**
- **نام سند ارسالی HW-{homework number}-{Name Family}-{student number}**
- **تمامی فایل‌های مورد نیاز این تمرین در این لینک قابل دسترس است.**
- **خروجی از هر مرحله‌ی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

سوال ۱)

قطعه کد:

```
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.sql.functions import *

sc = SparkContext.getOrCreate();('local')
spark = SparkSession(sc)
products_data = spark.read.option("mergeSchema", "true").parquet("/con-
tent/products_parquet")
sellers_data = spark.read.option("mergeSchema", "true").parquet("/con-
tent/sellers_parquet")
sales_data = spark.read.option("mergeSchema", "true").parquet("/con-
tent/sales_parquet")
```

```
products_data.printSchema()
sellers_data.printSchema()
sales_data.printSchema()
```

خروجی:

```
root
 |-- product_id: string (nullable = true)
 |-- product_name: string (nullable = true)
 |-- price: string (nullable = true)

root
 |-- seller_id: string (nullable = true)
 |-- seller_name: string (nullable = true)
 |-- daily_target: string (nullable = true)

root
 |-- order_id: string (nullable = true)
 |-- product_id: string (nullable = true)
 |-- seller_id: string (nullable = true)
 |-- date: string (nullable = true)
 |-- num_pieces_sold: string (nullable = true)
 |-- bill_raw_text: string (nullable = true)
```

الف)

قطعه کد:

```
print('Products Number:',products_data.count())
print('Sellers Number:',sellers_data.count())
print('Sales Number:',sales_data.count())
```

خروجی:

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

ب)

قطعه کد:

```
sales_data.select(countDistinct("product_id")).show()
```

خروجی:

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

ج)

قطعه کد:

```
sales_data.groupBy("product_id").agg(count("order_id").alias("or-
der_count")).orderBy(desc("order_count")).take(1)
```

خروجی:

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

سوال ۲)

قطعه کد:

```
sales_data.groupBy("date").agg(countDistinct("prod-
uct_id").alias("sale_count")).orderBy(desc("sale_count")).show()
```

خروجی:

```
+----------+----------+
|      date|sale_count|
+----------+----------+
|2020-07-06|    100765|
|2020-07-09|    100501|
|2020-07-01|    100337|
|2020-07-03|    100017|
|2020-07-02|     99807|
|2020-07-05|     99796|
|2020-07-04|     99791|
|2020-07-07|     99756|
|2020-07-08|     99662|
|2020-07-10|     98973|
+----------+----------+
```

سوال ۳)

قطعه کد:

```
saled_products=products_data.join(sales_data,["product_id"])
SP_with_income=saled_products.withColumn('in-
come', col("price")*col("num_pieces_sold"))
SP_with_income.agg(avg(col("income"))).show()
```

خروجی:

```
+------------------+
|       avg(income)|
+------------------+
|3808.9944235473604|
+------------------+
```

توجه: در این سوال که به انجام عملیات join بین جداول sales و product (که جداول حجیمی هستند) نیاز است، متا سفانه خطاهای مختلفی رخ می‌داد که نمونه‌ای از این ارورها پس از اجرای عملیات join، در پیو ست این تکلیف، قید شده است. این در حالی است که در صورت حذف برخی از رکوردهای هر یک از این دو جدول (که از طریق حذف برخی فایل پارتیشن‌های دیتاست در دو جدول مذکور انجام شد)، با هدف کاهش حجم دیتاست، قطعه کد مذکور به درستی و بدون خطا اجرا می‌شـد و نتیجه مربوطه را تولید می‌کرد. لذا خروجی نمایش داده شـده بالا، مربوط به اجرای کد بر روی زیرمجموعه‌ای از دیتاست مورد بررسی در دو جدول مذکور است. این مشکل عینا در مورد سوال ۵-الف هم وجود دارد و در آن سوال نیز که نیاز به join کلیه جداول دیتا ست وجود دارد، کدها روی زیرمجموعه‌ای کم حجم‌تر از نسخه ا صلی دیتاست اجرا شده‌اند. بنابراین در هر دوی این دو سوال، تفاوت مقادیر و نتایج با نتایج مورد انتظار، به این علت ا ست که کدها روی مجموعه داده کامل اجرا نشـــده‌اند و برخی از رکوردها در دو جدول حذف شـــده‌اند تا خطای مورد اشـــاره رخ ندهد.

در مورد خطاهای دریافتی در هنگام اجرای join بر روی کل جداول ذکر این نکته ضروری ا ست که پارامترهای مختلفی که در این نمونه خطا قابل مشاهده است، هر بار و در اجراهای مختلف متفاوت بود. جستجوهای مختلف بر روی این خطا نیز، نشان داد که چنین خطایی منشاء نامشخصی دارد که ظاهرا به محدودیت‌های پردازشی و حافظه مرتبط ا ست (که اجرای صـحیح کدها در نسـخه کم‌حجم‌تر مجموعه داده نیز موید همین موضـوع اسـت). با این حال سـعی شـد تغییرات مختلفی از جمله آزمودن روش‌های مختلف جهت load داده‌ها از فایل‌ها، آزمودن روش‌های مختلف join، آزمودن نصب نسخه‌های مختلف اسپارک، پای‌اسپارک و jdk (جهت بررسی ناسازگاری نسخه‌های مختلف این پکیج‌ها با یکدیگر) و غیره در کدها صـــورت گیرد تا منجر به رفع این مشـــکل شـــود، که هیچ یک از آنها باعث اجرای بدون خطای توابع aggregate بر روی جداول حاصل از join مجموعه داده کامل نشد.

سوال ۴)

نکته: در این سوال نسبت تعداد محصولات فروخته شده (ستون num_pieces_sold) در جدول سفارشات به مقدار کل فروش مورد انتظار هر فروشنده (ستون daily_target) در جدول فروشندگان محاسبه شده و به عنوان درصد سهم سفارش برای هر فروشنده در نظر گرفته شده است.

قطعه کد:

```
sellers_orders=sellers_data.join(sales_data,["seller_id"])
SO_with_share=sellers_orders.withCol-
umn('share', (col("num_pieces_sold")/col("daily_target"))*100)
SO_with_share.groupBy("seller_id").agg(avg("share").alias("or-
der_share")).show()
```

خروجی:

```
+---------+--------------------+
|seller_id|         order_share|
+---------+--------------------+
|        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|
+---------+--------------------+
```

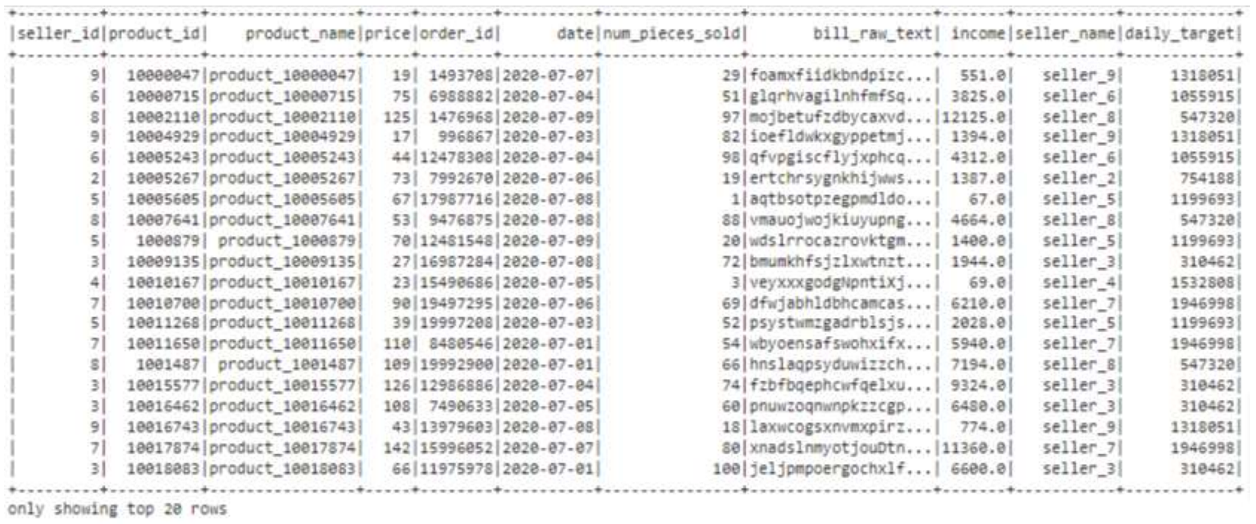
سوال ۵)

نکته: در این سوال فرض بر این بوده است که منظور از پرفروش‌ترین فروشنده، فروشنده‌ای با بیشترین درآمد از فروش محصولات خود است. در واقع معیار پرفروش بودن، درآمد بالاتر فرض شده و نه تعداد کالاها و غیره. به همین جهت از جدول تشکیل شده در سوال ۳ (SP_with_income) که حاصل ضرب قیمت در تعداد فروش به عنوان درآمد در یک ستون اضافی محاسبه شده بود، استفاده شده است.

قطعه کد:

```
all_tables=SP_with_income.join(sellers_data,["seller_id"])
```

خروجی:



```
+---------+----------+----------------+-----+--------+----------+---------------+--------------------+-------+-----------+------------+
|seller_id|product_id|    product_name|price|order_id|      date|num_pieces_sold|       bill_raw_text| income|seller_name|daily_target|
+---------+----------+----------------+-----+--------+----------+---------------+--------------------+-------+-----------+------------+
|        9|  10000047|product_10000047|   19| 1493708|2020-07-07|             29|foamxfiidkbndpizc...|  551.0|   seller_9|     1318051|
|        6|  10000715|product_10000715|   75| 6988882|2020-07-04|             51|glqrhvagilnhfmfSq...| 3825.0|   seller_6|     1055915|
|        8|  10002110|product_10002110|  125| 1476968|2020-07-09|             97|mojbetufzdbycaxvd...|12125.0|   seller_8|      547320|
|        9|  10004929|product_10004929|   17|  996867|2020-07-03|             82|ioefldwkxgyppetmj...| 1394.0|   seller_9|     1318051|
|        6|  10005243|product_10005243|   44|12478308|2020-07-04|             98|qfvpgiscflyjxphcq...| 4312.0|   seller_6|     1055915|
|        2|  10005267|product_10005267|   73| 7992670|2020-07-06|             19|ertchrsygnkhijwws...| 1387.0|   seller_2|      754188|
|        5|  10005605|product_10005605|   67|17987716|2020-07-08|              1|aqtbsotpzegpmdldo...|   67.0|   seller_5|     1199693|
|        8|  10007641|product_10007641|   53| 9476875|2020-07-08|             88|vmauojwojkiuyupng...| 4664.0|   seller_8|      547320|
|        5|   1000879| product_1000879|   70|12481548|2020-07-09|             20|wdslrrocazrovktgm...| 1400.0|   seller_5|     1199693|
|        3|  10009135|product_10009135|   27|16987284|2020-07-08|             72|bmumkhfsjzlxwtnzt...| 1944.0|   seller_3|      310462|
|        4|  10010167|product_10010167|   23|15490686|2020-07-05|              3|veyxxxgodgNpntiXj...|   69.0|   seller_4|     1532808|
|        7|  10010700|product_10010700|   90|19497295|2020-07-06|             69|dfwjabhldbhcamcas...| 6210.0|   seller_7|     1946998|
|        5|  10011268|product_10011268|   39|19997208|2020-07-03|             52|psystwmzgadrblsjs...| 2028.0|   seller_5|     1199693|
|        7|  10011650|product_10011650|  110| 8480546|2020-07-01|             54|wbyoensafswohxifx...| 5940.0|   seller_7|     1946998|
|        8|   1001487| product_1001487|  109|19992900|2020-07-01|             66|hnslaqpsyduwizzch...| 7194.0|   seller_8|      547320|
|        3|  10015577|product_10015577|  126|12986886|2020-07-04|             74|fzbfbqephcwfqelxu...| 9324.0|   seller_3|      310462|
|        3|  10016462|product_10016462|  108| 7490633|2020-07-05|             60|pnuwzoqnwnpkzzcgp...| 6480.0|   seller_3|      310462|
|        9|  10016743|product_10016743|   43|13979603|2020-07-08|             18|laxwcogsxnvmxpirz...|  774.0|   seller_9|     1318051|
|        7|  10017874|product_10017874|  142|15996052|2020-07-07|             80|xnadslnmyotjouDtn...|11360.0|   seller_7|     1946998|
|        3|  10018083|product_10018083|   66|11975978|2020-07-01|            100|jeljpmpoergochxlf...| 6600.0|   seller_3|      310462|
+---------+----------+----------------+-----+--------+----------+---------------+--------------------+-------+-----------+------------+
only showing top 20 rows
```

الف)

قطعه کد:

```
best_sellers=all_tables.groupBy("seller_id").agg(sum("income").alias("in-
come_sum")).orderBy(desc("income_sum"))
best_sellers.collect()[1]
best_sellers.collect()[-2]
```

خروجی:

دومین پرفروش‌ترین فروشنده:

```
Row(seller_id='6', income_sum=16461045.0)
```

دومین کم‌فروش‌ترین فروشنده:

```
Row(seller_id='9', income_sum=15769962.0)
```

ب)

قطعه کد:

```
sellers_orders.filter(sellers_orders.product_id ==0).se-
lect("seller_name").distinct().show()
```

خروجی:

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

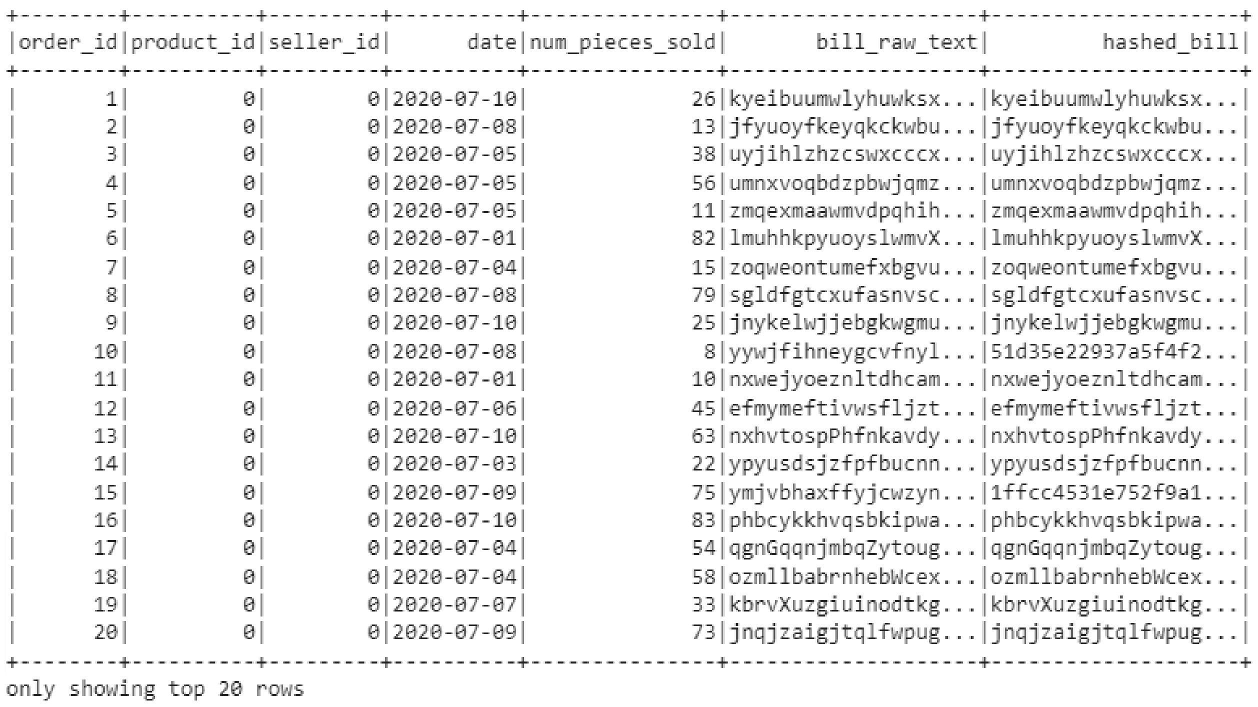
سوال ۶)

قطعه کد:

```
import hashlib
from pyspark.sql.types import IntegerType
def encryption_function(bill_raw_text,order_id):
    order_id=int(order_id)
    a_count=bill_raw_text.count('A')
    hashed_str=bill_raw_text
    if (order_id % 2) == 0:
        for x in range(a_count):
            hashed_str=hashlib.md5(hashed_str.encode('utf-8')).hexdigest()
    else:
        for x in range(a_count):
            hashed_str=hashlib.sha256(hashed_str.encode('utf-
8')).hexdigest()

    return hashed_str
encryption_function_UDF = udf( encryption_function)
sales_data_with_hash=sales_data.withColumn("hashed_bill", encryption_func-
tion_UDF(col("bill_raw_text"),col("order_id")))
sales_data_with_hash.show()
```

خروجی (۲۰ ردیف ابتدایی جدول sales پس از افزودن ستون جدید):



```
+--------+----------+---------+----------+---------------+--------------------+--------------------+
|order_id|product_id|seller_id|      date|num_pieces_sold|       bill_raw_text|         hashed_bill|
+--------+----------+---------+----------+---------------+--------------------+--------------------+
|       1|         0|        0|2020-07-10|             26|kyeibuumwlyhuwksx...|kyeibuumwlyhuwksx...|
|       2|         0|        0|2020-07-08|             13|jfyuoyfkeyqkckwbu...|jfyuoyfkeyqkckwbu...|
|       3|         0|        0|2020-07-05|             38|uyjihlzhzcswxcccx...|uyjihlzhzcswxcccx...|
|       4|         0|        0|2020-07-05|             56|umnxvoqbdzpbwjqmz...|umnxvoqbdzpbwjqmz...|
|       5|         0|        0|2020-07-05|             11|zmqexmaawmvdpqhih...|zmqexmaawmvdpqhih...|
|       6|         0|        0|2020-07-01|             82|lmuhhkpyuoyslwmvX...|lmuhhkpyuoyslwmvX...|
|       7|         0|        0|2020-07-04|             15|zoqweontumefxbgvu...|zoqweontumefxbgvu...|
|       8|         0|        0|2020-07-08|             79|sgldfgtcxufasnvsc...|sgldfgtcxufasnvsc...|
|       9|         0|        0|2020-07-10|             25|jnykelwjjebgkwgmu...|jnykelwjjebgkwgmu...|
|      10|         0|        0|2020-07-08|              8|yywjfihneygcvfnyl...|51d35e22937a5f4f2...|
|      11|         0|        0|2020-07-01|             10|nxwejyoeznltdhcam...|nxwejyoeznltdhcam...|
|      12|         0|        0|2020-07-06|             45|efmymeftivwsfljzt...|efmymeftivwsfljzt...|
|      13|         0|        0|2020-07-10|             63|nxhvtospPhfnkavdy...|nxhvtospPhfnkavdy...|
|      14|         0|        0|2020-07-03|             22|ypyusdsjzfpfbucnn...|ypyusdsjzfpfbucnn...|
|      15|         0|        0|2020-07-09|             75|ymjvbhaxffyjcwzyn...|1ffcc4531e752f9a1...|
|      16|         0|        0|2020-07-10|             83|phbcykkhvqsbkipwa...|phbcykkhvqsbkipwa...|
|      17|         0|        0|2020-07-04|             54|qgnGqqnjmbqZytoug...|qgnGqqnjmbqZytoug...|
|      18|         0|        0|2020-07-04|             58|ozmllbabrnhebWcex...|ozmllbabrnhebWcex...|
|      19|         0|        0|2020-07-07|             33|kbrvXuzgiuinodtkg...|kbrvXuzgiuinodtkg...|
|      20|         0|        0|2020-07-09|             73|jnqjzaigjtqlfwpug...|jnqjzaigjtqlfwpug...|
+--------+----------+---------+----------+---------------+--------------------+--------------------+
only showing top 20 rows
```

بررسی موارد تکراری در مقادیر hash شده)

قطعه کد:

```
sales_data_with_hash.groupBy("hashed_bill").agg(countDistinct("or-
der_id").alias("equal_hash_cnt")).orderBy(desc("equal_hash_cnt")).show()
```

خروجی:

```
+--------------------+--------------+
|         hashed_bill|equal_hash_cnt|
+--------------------+--------------+
|bkchmpqmiadolsdwd...|             1|
|idanrsrwjhipkhhzu...|             1|
|nywertMvphvjmsfil...|             1|
|4a78c7d20e9dd105b...|             1|
|1e1831c672abc7917...|             1|
|ehitfhsuqvljmtzcs...|             1|
|2befe233b535e4277...|             1|
|0a090c4803de29d51...|             1|
|eswtetrqgzmdftlmg...|             1|
|jupsczladthcybvkr...|             1|
|cmblnstqyaradxlra...|             1|
|zllrlwxiprlwzfssz...|             1|
|iqsitnlbgjrkhbkma...|             1|
|gplkhzojziliViwxq...|             1|
|tlcepjjuuipboynyy...|             1|
|vysprexsxqehpsapc...|            1|
|nhkchtchhqxrrdpjq...|            1|
|7bd0faeddf6a77a66...|            1|
|dksfvzyggcobitkvi...|            1|
|kbuvibzyfrjqocvlj...|            1|
+--------------------+-------------+
only showing top 20 rows
```

در این قطعه کد بررسی شد که هر یک از مقادیر hash تولید شده، دارای چه تعداد رکورد سفارش هستند. در واقع اگر این مقدار بیش از عدد یک با شد ن شان‌دهنده این ا ست که رکوردهایی وجود دارد که مقدار ستون hashed_bill برای آنها یکسان است. نتایج نشان می‌دهد که هیچ یک از این مقادیر بیشتر از عدد ۱ نشده است، بنابراین hash هیچ یک از ستون‌ها با یکدیگر یکسان نیست.

پیوست:

نمونه‌ای از خطای مذکور در سوال ۳، که در صورت کاهش حجم دیتاست در دو جدول sales و products این خطا برطرف می‌شد و نتایج قید شده در سوال ۳ و سوال ۵- الف به دست می‌آمد. مجددا تاکید می‌گردد که پارامترهایی که در متن خطای زیر نشان داده شده ثابت نبود و در اجراهای مختلف، مقادیر متفاوتی را نشان می‌داد. ضمن اینکه هر بار پس از اینکه این خطا به وقوع می‌پیوست، اتصال دیباگ گوگل کولب قطع می‌شد و نیاز به راه‌اندازی مجدد داشت.

```
---------------------------------------------------------------------------
Py4JJavaError                             Traceback (most recent call last)
<ipython-input-6-ef457b71f55f> in <module>()
      1 saled_products=products_data.join(sales_data,["product_id"])
----> 2 print('Saled product Count:',saled_products.count())

________________________________________
3 frames
________________________________________
/usr/local/lib/python3.7/dist-packages/py4j/protocol.py in get_return_value(answer, gateway_client, target_id, name)
    326                 raise Py4JJavaError(
    327                     "An error occurred while calling {0}{1}{2}.\n".
--> 328                     format(target_id, ".", name), value)
    329             else:
    330                 raise Py4JError(

Py4JJavaError: An error occurred while calling o40.count.
: org.apache.spark.SparkException: Job aborted due to stage failure: Task 1 in stage 18.0 failed 1 times, most recent failure: Lost task 1.0 in stage 18.0 (TID 196) (d943f83ee673 executor driver): java.lang.OutOfMemoryError: GC overhead limit exceeded
	at java.util.LinkedList.linkLast(LinkedList.java:142)
	at java.util.LinkedList.add(LinkedList.java:338)
	at org.apache.spark.sql.execution.BufferedRowIterator.append(BufferedRowIterator.java:73)
	at
org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage4.processNext(Unknown
Source)
        at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)
        at
org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)
        at
org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage5.smj_findNextJoinRows
_0$(Unknown Source)
        at
org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage5.agg_doAggregateWith
outKey_0$(Unknown Source)
        at
org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage5.processNext(Unknown
Source)
        at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)
        at
org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$2.hasNext(WholeStageCodegenExec.scala:778)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.write(BypassMergeSortShuffleWriter.java:140)
        at org.apache.spark.shuffle.ShuffleWriteProcessor.write(ShuffleWriteProcessor.scala:59)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:52)
        at org.apache.spark.scheduler.Task.run(Task.scala:131)
        at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)
        at org.apache.spark.executor.Executor$TaskRunner$$Lambda$1409/1401541426.apply(Unknown Source)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)

Driver stacktrace:
        at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)
        at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)
        at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)
        at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)
        at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)
        at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)
        at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)
        at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)
        at
org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)
        at scala.Option.foreach(Option.scala:407)
        at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)
        at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)
Caused by: java.lang.OutOfMemoryError: GC overhead limit exceeded
        at java.util.LinkedList.linkLast(LinkedList.java:142)
        at java.util.LinkedList.add(LinkedList.java:338)
        at org.apache.spark.sql.execution.BufferedRowIterator.append(BufferedRowIterator.java:73)
        at
org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage4.processNext(Unknown
Source)
```

```
        at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)
        at
org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)
        at
org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage5.smj_findNextJoinRows
_0$(Unknown Source)
        at
org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage5.agg_doAggregateWith
outKey_0$(Unknown Source)
        at
org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage5.processNext(Unknown
Source)
        at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)
        at
org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$2.hasNext(WholeStageCodegenExec.scala:778)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.write(BypassMergeSortShuffleWriter.java:140)
        at org.apache.spark.shuffle.ShuffleWriteProcessor.write(ShuffleWriteProcessor.scala:59)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:52)
        at org.apache.spark.scheduler.Task.run(Task.scala:131)
        at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)
        at org.apache.spark.executor.Executor$TaskRunner$$Lambda$1409/1401541426.apply(Unknown Source)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
SEARCH STACK OVERFLOW
ERROR:root:Exception while sending command.
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/py4j/clientserver.py", line 475, in send_command
    answer = smart_decode(self.stream.readline()[:-1])
  File "/usr/lib/python3.7/socket.py", line 589, in readinto
    return self._sock.recv_into(b)
ConnectionResetError: [Errno 104] Connection reset by peer

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/py4j/java_gateway.py", line 1038, in send_command
    response = connection.send_command(command)
  File "/usr/local/lib/python3.7/dist-packages/py4j/clientserver.py", line 504, in send_command
    "Error while sending or receiving", e, proto.ERROR_ON_RECEIVE)
py4j.protocol.Py4JNetworkError: Error while sending or receiving
```