Week 12 : Big Data Term Project

****

College of Science and Technology, Bellevue University

DSC650-T301 Big Data (2243-1)

Sashidhar Bezawada

February 29 , 2024

Finance - Transaction Log Analysis

Objective: Streamline large volumes of transaction logs, identifying transaction volumes, peak transaction times, and other important metrics.

Dataset : Bank Transactions Dataset (Kaggle)

The dataset used for the project is presently only one of few on Kaggle with information on the rising risk of digital financial fraud, emphasizing the difficulty in obtaining such data. This synthetically generated dataset consists of payments from various customers made in different time periods and with different amounts.
Columns:

- Step: This feature represents the day from the start of simulation. It has 180 steps so simulation ran for virtually 6 months.
- Customer: This feature represents the customer id
- zipCodeOri: The zip code of origin/source.
- Merchant: The merchant's id
- zipMerchant: The merchant's zip code
- Age: Categorized age
    - 0: <= 18,
    - 1: 19-25,
    - 2: 26-35,
    - 3: 36-45,
    - 4: 46:55,
    - 5: 56:65,
    - 6: > 65
    - U: Unknown
- Gender: Gender for customer
    - E : Enterprise,
    - F: Female,
    - M: Male,
    - U: Unknown
- Category: Category of the purchase. I won't write all categories here, we'll see them later in the analysis.
- Amount: Amount of the purchase
- Fraud: Target variable which shows if the transaction fraudulent(1) or benign(0)

Exploratory Data Analysis

Firstly, we shall import the dataset into Hadoop and gain insights of metadata & data.

bash-5.0# hdfs dfs -put /data/bs140513_032310.csv /

bash-5.0# hdfs dfs -head /bs140513_032310.csv
step,customer,age,gender,zipcodeOri,merchant,zipMerchant,category,amount,fraud
0,'C1093826151','4','M','28007','M348934600','28007','es_transportation',4.55,0
0,'C352968107','2','M','28007','M348934600','28007','es_transportation',39.68,0
0,'C2054744914','4','F','28007','M1823072687','28007','es_transportation',26.89,0
0,'C1760612790','3','M','28007','M348934600','28007','es_transportation',17.25,0
0,'C757503768','5','M','28007','M348934600','28007','es_transportation',35.72,0
0,'C1315400589','3','F','28007','M348934600','28007','es_transportation',25.81,0
0,'C765155274','1','F','28007','M348934600','28007','es_transportation',9.1,0
0,'C202531238','4','F','28007','M348934600','28007','es_transportation',21.17,0
0,'C105845174','3','M','28007','M348934600','28007','es_transportation',32.4,0
0,'C39858251','5','F','28007','M348934600','28007','es_transportation',35.4,0
0,'C98707741','4','F','28007','M348934600','28007','es_transportation',14.95,0

Based on the descriptions, I have renamed the Columns to give more value add:

step	==> step_date
customer	==> customer_id
age	==> age_category
gender	==> customer_gender
zipcodeOri	==> origin_zipcode
merchant	==> merchant_id
zipMerchant	==> merchant_zipcode
category	==> purchase_category
amount	==> purchase_amount
fraud	==> fraud_ind

Data Analysis (using Hive)

1. Created the Hive Table bank_tran.
2. Loaded the hdfs file into the Hive table – bank_tran
3. Check total count , sample data, frequency on columns – age, gender , category, fraud and Min,max,avg of amount.

hive>

```
CREATE TABLE bank_tran(
`step` INT,
`customer` STRING,
`age` STRING,
`gender` STRING,
`zipcodeOrigin` STRING,
`merchant` STRING,
`zipMerchant` STRING,
`category` STRING,
`amount` decimal(13,2),
`fraud` INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
tblproperties("skip.header.line.count"="1");
```

hive> LOAD DATA INPATH '/bs140513_032310.csv' INTO TABLE bank_tran;

hive> select count(*) from bank_tran;

594643
Time taken: 13.149 seconds, Fetched: 1 row(s)

hive> set hive.cli.print.header=true;
hive> select step, customer ,age ,gender , zipcodeOrigin ,merchant ,zipMerchant , category , amount , fraud from bank_tran limit 10;
OK

step	customer	age	gender	zipcodeori	merchant	zipmerchant	category	amount	fraud
0	'C1093826151'	'4'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	4.55	0
0	'C352968107'	'2'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	39.68	0
0	'C2054744914'	'4'	'F'	'28007'	'M1823072687'	'28007'	'es_transportation'	26.89	0
0	'C1760612790'	'3'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	17.25	0
0	'C757503768'	'5'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	35.72	0
0	'C1315400589'	'3'	'F'	'28007'	'M348934600'	'28007'	'es_transportation'	25.81	0
0	'C765155274'	'1'	'F'	'28007'	'M348934600'	'28007'	'es_transportation'	9.10	0
0	'C202531238'	'4'	'F'	'28007'	'M348934600'	'28007'	'es_transportation'	21.17	0
0	'C105845174'	'3'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	32.40	0
0	'C39858251'	'5'	'F'	'28007'	'M348934600'	'28007'	'es_transportation'	35.40	0

Time taken: 0.185 seconds, Fetched: 10 row(s)

hive> select fraud,count(*) as count,min(amount) as min_amount ,max(amount) as max_amount , avg(amount) as mean_amount from bank_tran group by fraud ;

fraud	count	min_amount	max_amount	mean_amount
1	**7200**	**0.03**	**8329.96**	**530.926551388888889**
0	587443	0.00	2144.86	31.84723038660772194

Time taken: 8.137 seconds, Fetched: 2 row(s)

hive> select gender,age,count(*) as count from bank_tran where fraud = 1 group by gender,age order by count(*);

gender	age	count
'E'	'U'	7
'M'	'0'	9
'F'	'0'	39
'M'	'6'	117
'F'	'6'	144
'M'	'5'	159
'M'	'1'	193
'F'	'1'	496
'F'	'5'	527
'M'	'4'	599
'M'	'3'	656
'M'	'2'	702
'F'	**'4'**	**811**
'F'	**'3'**	**1099**
'F'	**'2'**	**1642**

Time taken: 9.146 seconds, Fetched: 15 row(s)

hive> select gender,category,count(*) as count from bank_tran where fraud = 1 group by gender,category order by count(*);

category	count
'es_fashion'	116
'es_barsandrestaurants'	120
'es_tech'	158
'es_otherservices'	228
'es_hyper'	280
'es_home'	302
'es_leisure'	474
'es_hotelservices'	548
'es_travel'	578
'es_wellnessandbeauty'	718
'es_health'	**1696**
'es_sportsandtoys'	**1982**

Time taken: 16.187 seconds, Fetched: 12 row(s)

Based on the above Data analysis using Hive, my observations are as follows:

1. Dataset has 7200 fraud transactions out of total 594643 transactions. (i.e. **1.21%)**
2. Purchases made by Females spanning in the age groups category 2,3,4 (I.e age groups between 25 thru 55) contribute - (3522/ 7200 *100) = **49.33 %** of fraud purchases.
3. Purchases made in the two categories – es_health & es_sportsandtoys contribute to more than (3678/7200) = **51.08%** of fraud purchases.

Additional Data Preparation, Transformation and Analysis （Using pyspark）

1. **Renaming the Columns and Changing datatypes**

```
banktran_df = banktran_df.withColumn("step", banktran_df.step.cast('int')) \
        .withColumn("amount", banktran_df.amount.cast('decimal(13,2)')) \
        .withColumn("fraud", banktran_df.fraud.cast('int'))

banktran_df= banktran_df.withColumnRenamed("step", "step_day_num") \
        .withColumnRenamed("customer", "customer_id") \
        .withColumnRenamed("age", "age_category") \
        .withColumnRenamed("gender", "customer_gender") \
        .withColumnRenamed("zipcodeOri", "origin_zipcode") \
        .withColumnRenamed("merchant", "merchant_id") \
        .withColumnRenamed("zipMerchant", "merchant_zipcode") \
        .withColumnRenamed("category", "purchase_category") \
        .withColumnRenamed("amount", "purchase_amount") \
        .withColumnRenamed("fraud", "fraud_ind")
```

```
>>> banktran_df.printSchema()
root
 |-- step_day_num: integer (nullable = true)
 |-- customer_id: string (nullable = true)
 |-- age_category: string (nullable = true)
 |-- customer_gender: string (nullable = true)
 |-- origin_zipcode: string (nullable = true)
 |-- merchant_id: string (nullable = true)
 |-- merchant_zipcode: string (nullable = true)
 |-- purchase_category: string (nullable = true)
 |-- purchase_amount: decimal(13,2) (nullable = true)
 |-- fraud_ind: integer (nullable = true)
```

2. **Checking for Nulls in the dataset**

```
>>> from pyspark.sql.functions import col,isnan,when,count
>>> bank_tran_columns=banktran_df.columns
>>> banktran_df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in
bank_tran_columns]).show()
+------------+-----------+------------+---------------+--------------+-----------+----------------+-----------------+---------------+---------+
|step_day_num|customer_id|age_category|customer_gender|origin_zipcode|merchant_id|merchant_zipcode|purchase_category|purchase_amount|fraud_ind|
+------------+-----------+------------+---------------+--------------+-----------+----------------+-----------------+---------------+---------+
|           0|          0|           0|              0|             0|          0|               0|                0|              0|        0|
+------------+-----------+------------+---------------+--------------+-----------+----------------+-----------------+---------------+---------+
```

3. **Data Transformations and Further Analysis :**

```
banktran_df_fraud=banktran_df.filter(banktran_df.fraud_ind == 1)
banktran_df_nonfraud=banktran_df.filter(banktran_df.fraud_ind == 0)
banktran_df.createOrReplaceTempView('banktran_df')
banktran_df_fraud.createOrReplaceTempView('banktran_df_fraud')
banktran_df_nonfraud.createOrReplaceTempView('banktran_df_nonfraud')
spark.sql('SHOW TABLES').show()
```

```
+--------+-------------------+-----------+
|database|          tableName|isTemporary|
+--------+-------------------+-----------+
| default|          bank_tran|      false|
|        |         banktran_df|       true|
|        |   banktran_df_fraud|       true|
|        |banktran_df_nonfraud|       true|
+--------+-------------------+-----------+
```

4. Fraud Frequency :

a.Fraud Frequency for purchase_category

```
+-------------------+---------+-------------+----------------+
|  purchase_category|total_cnt|fraud_percent|nonfraud_percent|
+-------------------+---------+-------------+----------------+
|        'es_leisure'|      499|           95|               5|
|         'es_travel'|      728|           79|              21|
|       'es_contents'|      885|         null|             100|
|  'es_otherservices'|      912|           25|              75|
|  'es_hotelservices'|     1744|           31|              69|
|           'es_home'|     1986|           15|              85|
|           'es_tech'|     2370|            7|              93|
|  'es_sportsandtoys'|     4002|           50|              50|
|          'es_hyper'|     6098|            5|              95|
|'es_barsandrestau...|     6373|            2|              98|
|        'es_fashion'|     6454|            2|              98|
|'es_wellnessandbe...|    15086|            5|              95|
|         'es_health'|    16133|           11|              89|
|           'es_food'|    26254|         null|             100|
| 'es_transportation'|   505119|         null|             100|
+-------------------+---------+-------------+----------------+
```

```
spark.sql('select total.purchase_category,total_cnt,decimal(fraud_cnt/total_cnt*100 )  as
fraud_percent,decimal(nonfraud_cnt/total_cnt*100) as nonfraud_percent \
            from ( SELECT purchase_category,count(*) as total_cnt FROM banktran_df group by
purchase_category) as total \
            left join ( SELECT purchase_category,count(*) as fraud_cnt FROM banktran_df_fraud group by
purchase_category) as  fraud      on total.purchase_category=fraud.purchase_category \
            left join ( SELECT purchase_category,count(*) as nonfraud_cnt FROM banktran_df_nonfraud group
by purchase_category) as nonfraud on total.purchase_category=nonfraud.purchase_category order by
total_cnt').show()
```

b.Fraud Frequency customer_gender

```
+---------------+---------+-------------+----------------+
|customer_gender|total_cnt|fraud_percent|nonfraud_percent|
+---------------+---------+-------------+----------------+
|            'U'|      515|         null|             100|
|            'E'|     1178|            1|              99|
|            'M'|   268385|            1|              99|
|            'F'|   324565|            1|              99|
+---------------+---------+-------------+----------------+
```

```
spark.sql('select total.customer_gender,total_cnt ,decimal(fraud_cnt/total_cnt*100 )  as
fraud_percent,decimal(nonfraud_cnt/total_cnt*100) as nonfraud_percent \
            from ( SELECT customer_gender,count(*) as total_cnt FROM banktran_df group by customer_gender)
as total \
            left join ( SELECT customer_gender,count(*) as fraud_cnt FROM banktran_df_fraud group by
customer_gender) as  fraud      on total.customer_gender=fraud.customer_gender \
            left join ( SELECT customer_gender,count(*) as nonfraud_cnt FROM banktran_df_nonfraud group by
customer_gender) as nonfraud on total.customer_gender=nonfraud.customer_gender order by total_cnt').show()
```

c. Fraud Frequency for age_category

age_category	total_cnt	fraud_percent	nonfraud_percent
'U'	1178	1	99
'0'	**2452**	**2**	**98**
'6'	26774	1	99
'1'	58131	1	99
'5'	62642	1	99
'4'	109025	1	99
'3'	147131	1	99
'2'	187310	1	99

```
spark.sql('select total.age_category,total_cnt ,decimal(fraud_cnt/total_cnt*100 )  as
fraud_percent,decimal(nonfraud_cnt/total_cnt*100) as nonfraud_percent \
            from ( SELECT age_category,count(*) as total_cnt FROM banktran_df group by age_category) as
total \
            left join ( SELECT age_category,count(*) as fraud_cnt FROM banktran_df_fraud group by
age_category) as  fraud       on total.age_category=fraud.age_category \
            left join ( SELECT age_category,count(*) as nonfraud_cnt FROM banktran_df_nonfraud group by
age_category) as nonfraud on total.age_category=nonfraud.age_category order by total_cnt').show()
```

d. Fraud Frequency for purchase_amount_category

fraud_ind	purchase_amount_category	count(1)
1	Amt_801_to_1000	283
1	null	637
1	Amt_601_to_800	643
1	Amt_401_to_600	1196
1	**Amt_201_to_400**	**2196**
1	**Amt_0_to_200**	**2245**
0	Amt_801_to_1000	19
0	Amt_601_to_800	24
0	Amt_401_to_600	53
0	null	84
0	Amt_201_to_400	2509
0	Amt_0_to_200	584754

```
banktran_df_fraud_bucket = spark.sql(f'''
SELECT a.*
      ,(case when purchase_amount between   0 and 200 Then 'Amt_0_to_200'
             when purchase_amount between 201 and 400 Then 'Amt_201_to_400'
             when purchase_amount between 401 and 600 Then 'Amt_401_to_600'
             when purchase_amount between 601 and 800 Then 'Amt_601_to_800'
             when purchase_amount between 801 and 1000 Then 'Amt_801_to_1000'
         END ) as purchase_amount_category
      ,(case when step_day_num between   0 and 29 Then 'day_Bucket_30'
             when step_day_num between  30 and 59 Then 'day_Bucket_60'
             when step_day_num between  60 and 89 Then 'day_Bucket_90'
             when step_day_num between  90 and 119 Then 'day_Bucket_120'
             when step_day_num between 120 and 149 Then 'day_Bucket_150'
             when step_day_num between 150 and 179 Then 'day_Bucket_180'
         END ) as day_Bucket_category
FROM banktran_df a
''')
```

```
banktran_df_fraud_bucket.createOrReplaceTempView('banktran_df_fraud_bucket')


spark.sql('select total.purchase_amount_category,total_cnt ,decimal(fraud_cnt/total_cnt*100 )  as
fraud_percent,decimal(nonfraud_cnt/total_cnt*100) as nonfraud_percent \
            from ( SELECT purchase_amount_category,count(*) as total_cnt FROM banktran_df_fraud_bucket
group by purchase_amount_category) as total \
            left join ( SELECT purchase_amount_category,count(*) as fraud_cnt FROM banktran_df_fraud_bucket
where fraud_ind = 1 group by purchase_amount_category) as  fraud       on
total.purchase_amount_category=fraud.purchase_amount_category \
            left join ( SELECT purchase_amount_category,count(*) as nonfraud_cnt FROM
banktran_df_fraud_bucket where fraud_ind = 0 group by purchase_amount_category) as nonfraud on
total.purchase_amount_category=nonfraud.purchase_amount_category order by total_cnt').show()
```

purchase_amount_category	total_cnt	fraud_percent	nonfraud_percent
Amt_801_to_1000	302	94	6
Amt_601_to_800	667	96	4
null	721	null	null
Amt_401_to_600	1249	96	4
Amt_201_to_400	4705	47	53
Amt_0_to_200	586999	0	100

e. Fraud Frequency for day_Bucket_category

fraud_ind	day_Bucket_category	count(1)
1	day_Bucket_90	1200
1	day_Bucket_150	1200
1	day_Bucket_60	1200
1	day_Bucket_180	1200
1	day_Bucket_120	1200
1	**day_Bucket_30**	**1200**
0	**day_Bucket_30**	**79279**
0	day_Bucket_60	90119
0	day_Bucket_90	97490
0	day_Bucket_120	103226
0	day_Bucket_150	107223
0	day_Bucket_180	110106

Based on the above Data analysis using pyspark, my observations are as follows:

1. Dataset does not have any Null values.
2. Fraud Frequency for purchase_category - 'es_leisure' , 'es_travel' are top 2 category
3. Fraud Frequency for customer_gender - Female gender has the highest category
4. Fraud Frequency for age_category - Age group less than 18 has the highest fraud category.
5. Fraud Frequency for purchase_amount_category - 'Amt_401_to_600' & 'Amt_601_to_800' both have 96 % of fraud transactions categories
6. Fraud Frequency for day_Bucket_category - 'day_Bucket_30' has the highest fraud transactions category.

spark.sql('select fraud_ind,customer_gender,purchase_category,count(*) as total_cnt from banktran_df where fraud_ind = 1   group by fraud_ind,customer_gender,purchase_category order by count(*) desc ').show(50)

```
+---------+---------------+--------------------+---------+
|fraud_ind|customer_gender|   purchase_category|total_cnt|
+---------+---------------+--------------------+---------+
|        1|            'F'|  'es_sportsandtoys'|     1305|
|        1|            'F'|         'es_health'|     1111|
|        1|            'M'|  'es_sportsandtoys'|      677|
|        1|            'M'|         'es_health'|      584|
|        1|            'F'|'es_wellnessandbe...|      503|
|        1|            'F'|         'es_travel'|      378|
|        1|            'F'|  'es_hotelservices'|      360|
|        1|            'F'|        'es_leisure'|      309|
|        1|            'M'|'es_wellnessandbe...|      213|
|        1|            'M'|         'es_travel'|      200|
+---------+---------------+--------------------+---------+
```

Conclusion :

Top fraud categories are see from below :

a. Purchase_category → 'es_sportsandtoys' & 'es_health'
b. Customer_gender → It is shared between F & M
c. Age_category → less than 18 year contribute to more frauds than other age groups.
d. purchase_amount_category → Purchase amount made less than 400 contribute to nearly 62 % of the fraud transactions.