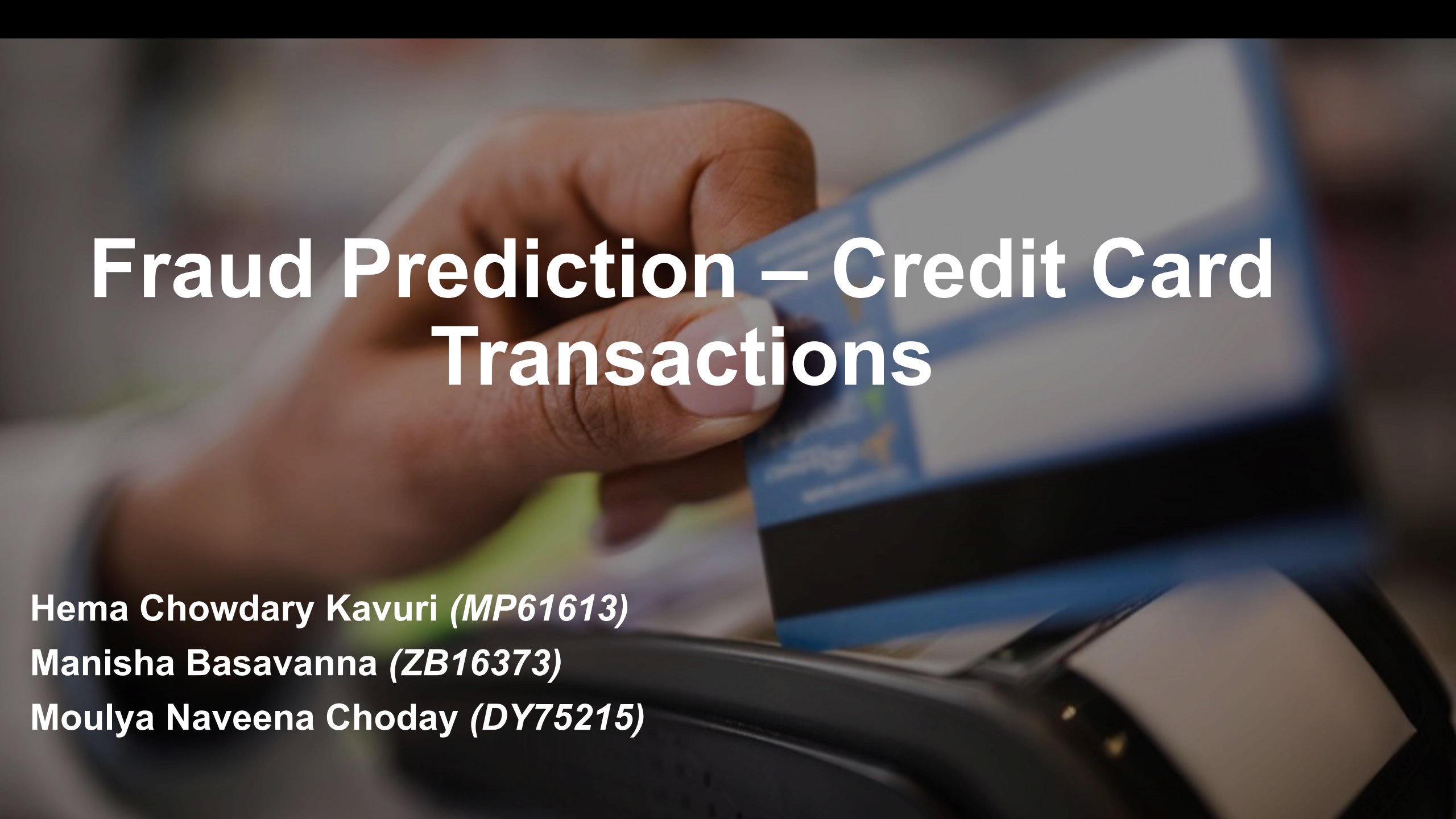


Fraud Prediction – Credit Card Transactions

Hema Chowdary Kavuri ***(MP61613)***

Manisha Basavanna ***(ZB16373)***

Moulya Naveena Choday ***(DY75215)***

Project Description

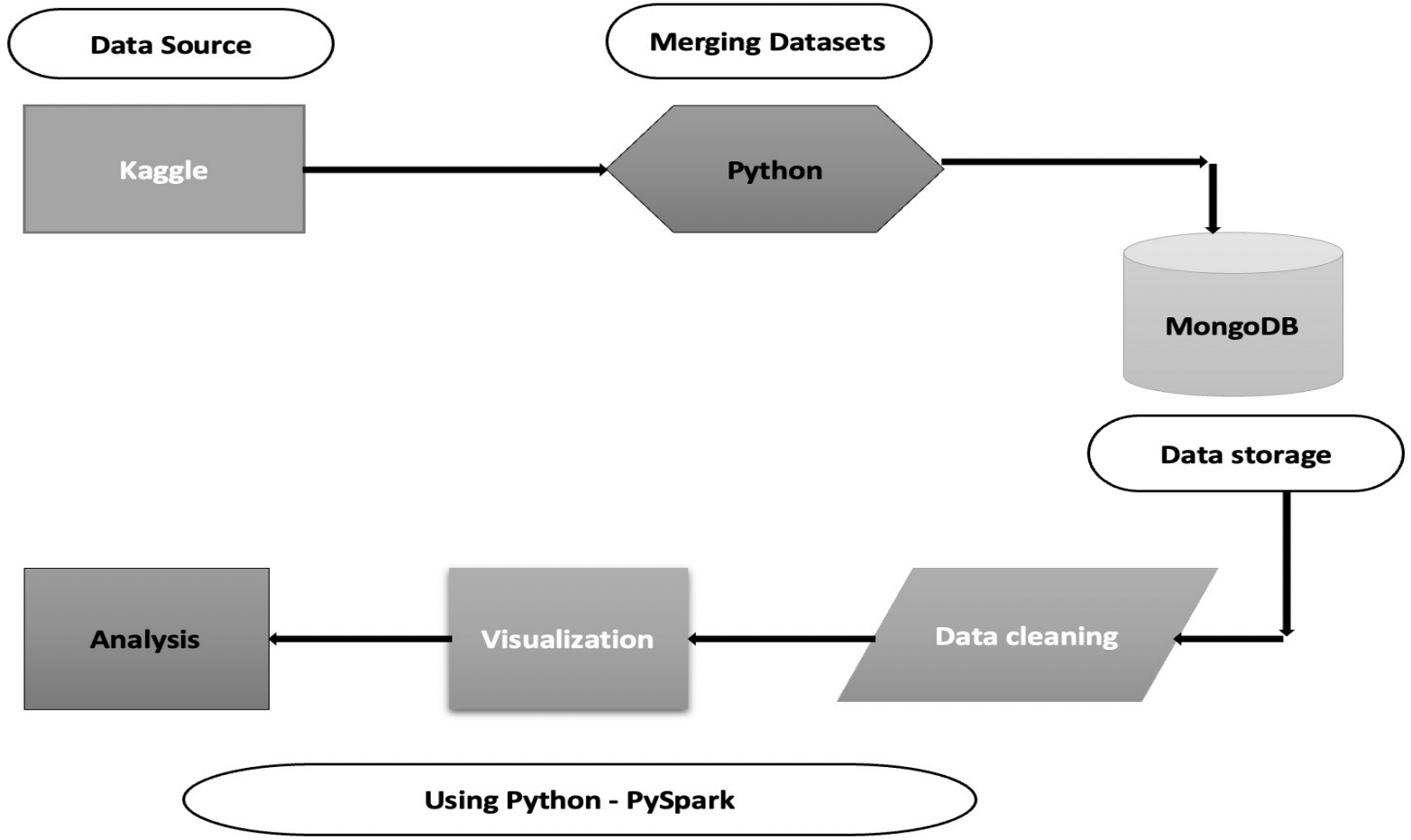
- The theft in which someone else uses your credit card to make an unauthorized purchase is credit card fraud.
- There is a rapid increase in credit card frauds with the increase in E-payment systems.
- Out of 22 percent of total card volume in the United States, 36 percent was accounted as card fraud in 2021.
- credit card fraud increased by 44.6% from 271,927 in 2019 to 393,207 in 2020.
- According to Nilson's research's annual report, Global Card frauds might cost approximately $408.50 billion over the next decade.

Problem statement

Credit card fraud transactions is the well-known online theft. Predicting the customers transactions data and finding the ***fraud transactions patterns*** will help in ***bringing awareness*** to the customers and in ***preventing*** the fraud transactions in future.

Technical Environment

- Kaggle: https://www.kaggle.com/kartik2112/fraud-detection
- Using the dataset from Kaggle containing legitimate and fraud transactions of 1000 customers.
- Size: 502MB
- Rows: 1296675 and Columns: 23
- Libraries – Pandas, Matplotlib, Seaborn, Pyspark
- Machine Learning models – Logistic Regression, NaiveBayes and DecisionTreeClassifier
- MongoDB and Python



Project Architecture

Project Implementation

Data Merging using Python

Data Storage in MongoDB

Data Cleaning using Python

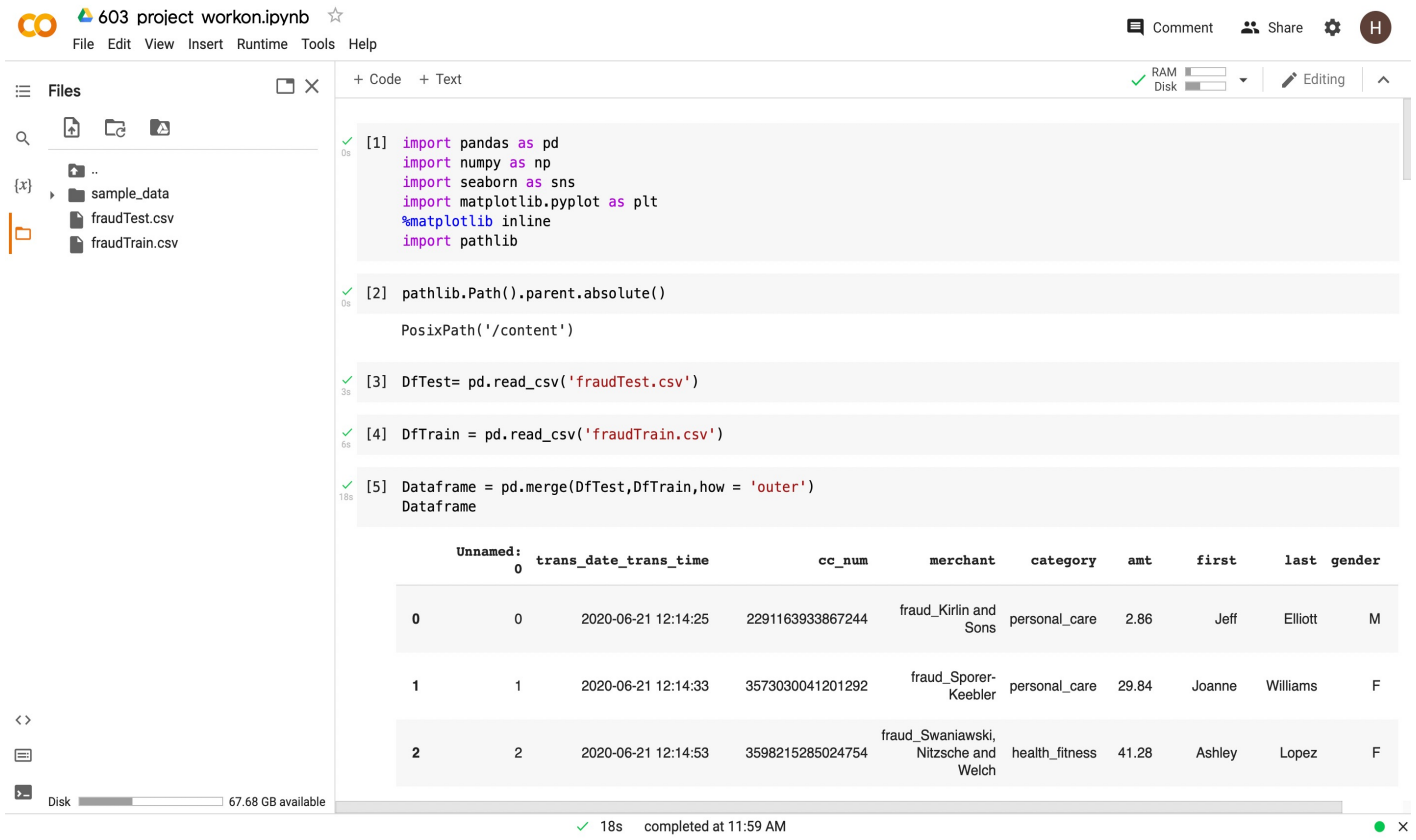
Connection between MongoDB and PySpark

Exploratory Data Analysis using PySpark

Analysis of Machine Learning models using MLlib

Screenshots of Implementation

Merging of two data sets into a single Dataframe using Python



	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care	2.86	Jeff	Elliott	M
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care	29.84	Joanne	Williams	F
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley	Lopez	F

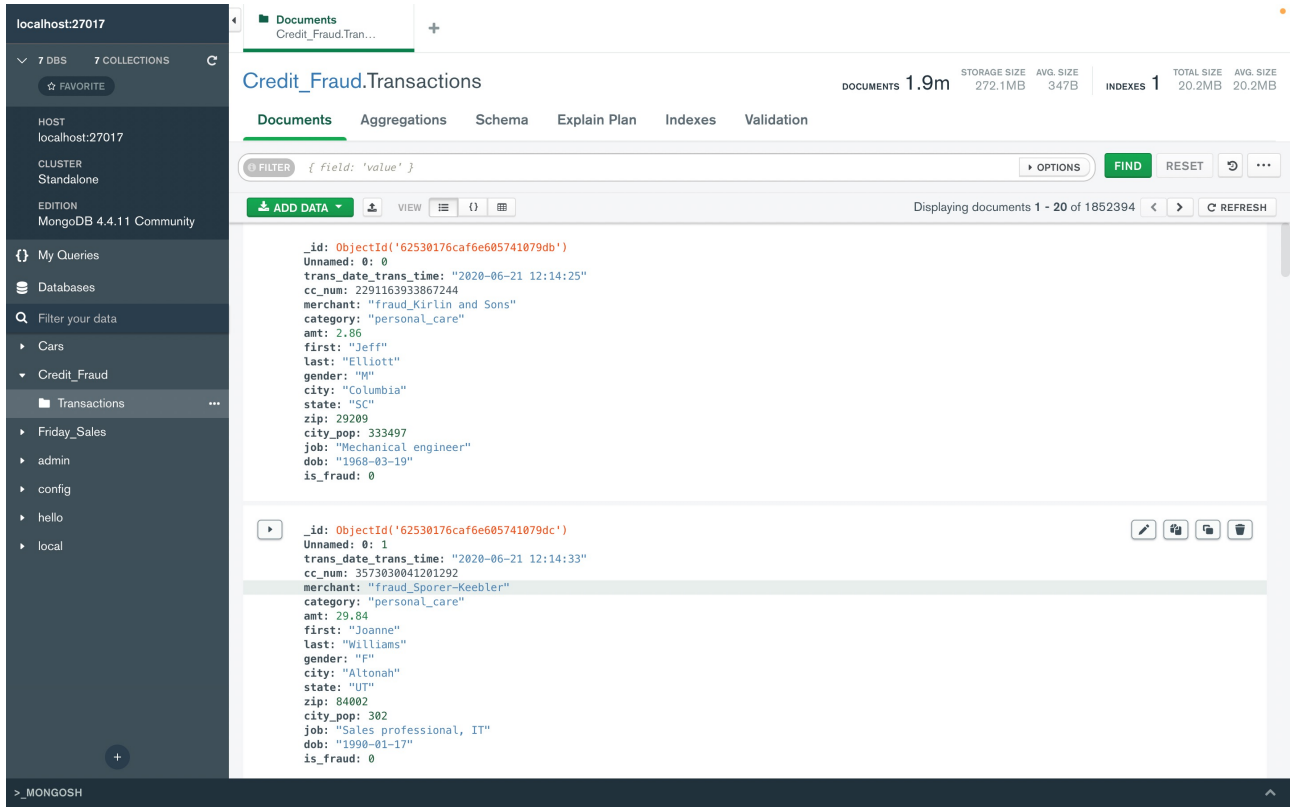
Data Storage

Used mongoDB compass to stored data set

In [1]:
```
import pymongo
from pymongo import MongoClient
import json
import pandas as pd
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [2]:
```
try:
    import pymongo
    from pymongo import MongoClient
    import json
except Exception as e:
    print("Some Modules are Missing ")

class MongoDB(object):

    def __init__(self, dBName=None, collectionName=None):

        self.dBName = dBName
        self.collectionName = collectionName

        self.client = MongoClient("localhost", 27017, maxPoolSize=50)

        self.DB = self.client[self.dBName]
        self.collection = self.DB[self.collectionName]


    def InsertData(self, path=None):
        """
        :param path: Path os csv File
        :return: None
        """

        df = pd.read_csv(path)
        data = df.to_dict('records')

        self.collection.insert_many(data, ordered=False)
        print("All the Data has been Exported to Mongo DB Server .... ")

if __name__ == "__main__":
    mongodb = MongoDB(dBName = 'Credit_Fraud', collectionName='Transactions')
    mongodb.InsertData(path="credit_cleaned.csv")
```

```
All the Data has been Exported to Mongo DB Server ....
```



Pyspark and mongodb Connection

```
In [3]: import pyspark
        from pyspark import SparkContext
        from pyspark.sql import SparkSession
        from pyspark.sql import SQLContext

In [4]: print(pyspark.__version__)

        3.2.1

In [5]: conf = pyspark.SparkConf().set("spark.jars.packages",
                                       "org.mongodb.spark:mongo-spark-connector_2.12:3.0.1").setMaster("local").setAppName("

In [6]: sc = SparkContext(conf=conf)

In [7]: sqlC = SQLContext(sc)

In [14]: mongo_ip = "mongodb://localhost:27017/Credit_Fraud."

In [15]: print(mongo_ip)

         mongodb://localhost:27017/Credit_Fraud.

In [16]: cf = sqlC.read.format("com.mongodb.spark.sql.DefaultSource").option("uri", mongo_ip + "Transactions").load()

In [17]: cf.createOrReplaceTempView("cf")

In [18]: cf = sqlC.sql("SELECT *FROM cf")

In [19]: cf.show()

+----------+--------------------+------+-------------+--------------------+-------------+--------+----------+-----
-----+------+--------+--------------------+--------+-------------------+-----+
|Unnamed: 0|                 _id|   amt|     category|              cc_num|         city|city_pop|       dob|    f
irst|gender|is_fraud|                 job|    last|           merchant|state|trans_date_trans_time|  zip|
+----------+--------------------+------+-------------+--------------------+-------------+--------+----------+-----
-----+------+--------+--------------------+--------+-------------------+-----+
|         0|{62530176caf6e605...|  2.86| personal_care|   2291163933867244|     Columbia|  333497|1968-03-19|
Jeff|     M|       0| Mechanical engineer| Elliott|fraud_Kirlin and ...|   SC|  2020-06-21 12:14:25|29209|
|         1|{62530176caf6e605...| 29.84| personal_care|   3573030041201292|      Altonah|     302|1990-01-17|   Jo
anne|     F|       0|Sales professiona...|Williams|fraud_Sporer-Keebler|   UT|  2020-06-21 12:14:33|84002|
|         2|{62530176caf6e605...| 41.28|health_fitness|   3598215285024754|     Bellmore|   34496|1970-10-21|   As
hley|     F|       0|   Librarian, public|   Lopez|fraud_Swaniawski,...|   NY|  2020-06-21 12:14:53|11710|
|         3|{62530176caf6e605...| 60.05|      misc_pos|   3591919803438423|   Titusville|   54767|1987-07-25|    B
rian|     M|       0|        Set designer|Williams|   fraud_Haley Group|   FL|  2020-06-21 12:15:15|32780|
```

```
2022-04-07T19:19:37.433-0400    connected to: mongodb://localhost/
2022-04-07T19:19:40.434-0400    [#.......................] credit_fraud.Transactions    18.8MB/284MB (6.6%)
2022-04-07T19:19:43.433-0400    [###.....................] credit_fraud.Transactions    38.8MB/284MB (13.7%)
2022-04-07T19:19:46.433-0400    [#####...................] credit_fraud.Transactions    59.5MB/284MB (21.0%)
2022-04-07T19:19:49.433-0400    [######..................] credit_fraud.Transactions    79.9MB/284MB (28.2%)
2022-04-07T19:19:52.432-0400    [########................] credit_fraud.Transactions    102MB/284MB (36.0%)
2022-04-07T19:19:55.432-0400    [##########..............] credit_fraud.Transactions    126MB/284MB (44.4%)
2022-04-07T19:19:58.432-0400    [############............] credit_fraud.Transactions    149MB/284MB (52.6%)
2022-04-07T19:20:01.432-0400    [##############..........] credit_fraud.Transactions    173MB/284MB (61.0%)
2022-04-07T19:20:04.432-0400    [################........] credit_fraud.Transactions    196MB/284MB (69.1%)
2022-04-07T19:20:07.432-0400    [##################......] credit_fraud.Transactions    219MB/284MB (77.2%)
2022-04-07T19:20:10.432-0400    [####################....] credit_fraud.Transactions    243MB/284MB (85.7%)
2022-04-07T19:20:13.432-0400    [######################..] credit_fraud.Transactions    266MB/284MB (93.8%)
2022-04-07T19:20:15.790-0400    [########################] credit_fraud.Transactions    284MB/284MB (100.0%)
2022-04-07T19:20:15.791-0400    1852394 document(s) imported successfully. 0 document(s) failed to import.
manisha@Manishas-MacBook-Pro ~ % mongo
MongoDB shell version v4.4.13
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("75da0f96-90dc-4b1a-bec4-d5fe9c7a1126") }
MongoDB server version: 4.4.13
---
The server generated these startup warnings when booting:
        2022-04-07T19:13:30.833-04:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin          0.000GB
car_make       0.000GB
config         0.000GB
credit_fraud   0.260GB
local          0.000GB
umbc_is_mongo  0.000GB
> use credit_fraud
switched to db credit_fraud
> show collections
Transactions
> db.Transactions.find().limit(5)
{ "_id" : ObjectId("624f71895a41f3f249efe6db"), "" : 0, "trans_date_trans_time" : "2020-06-21 12:14:25", "cc_num" : NumberLong("2291163933867244"), "merchant" : "fraud_Kirlin and Sons", "category" : "pers
onal_care", "amt" : 2.86, "first" : "Jeff", "last" : "Elliott", "gender" : "M", "city" : "Columbia", "state" : "SC", "zip" : 29209, "city_pop" : 333497, "job" : "Mechanical engineer", "dob" : "1968-03-19"
, "is_fraud" : 0 }
{ "_id" : ObjectId("624f71895a41f3f249efe6dc"), "" : 7, "trans_date_trans_time" : "2020-06-21 12:15:50", "cc_num" : NumberLong("3589289942931264"), "merchant" : "fraud_Reichel LLC", "category" : "personal
_care", "amt" : 10.37, "first" : "Paula", "last" : "Estrada", "gender" : "F", "city" : "Spencer", "state" : "SD", "zip" : 57374, "city_pop" : 343, "job" : "Development worker, international aid", "dob" :
"1972-03-05", "is_fraud" : 0 }
{ "_id" : ObjectId("624f71895a41f3f249efe6dd"), "" : 8, "trans_date_trans_time" : "2020-06-21 12:16:10", "cc_num" : NumberLong("3596357274378601"), "merchant" : "fraud_Goyette, Howell and Collier", "categ
ory" : "shopping_pos", "amt" : 4.37, "first" : "David", "last" : "Everett", "gender" : "M", "city" : "Morrisdale", "state" : "PA", "zip" : 16858, "city_pop" : 3688, "job" : "Advice worker", "dob" : "1973-
05-27", "is_fraud" : 0 }
{ "_id" : ObjectId("624f71895a41f3f249efe6de"), "" : 9, "trans_date_trans_time" : "2020-06-21 12:16:11", "cc_num" : NumberLong("3546897637165774"), "merchant" : "fraud_Kilback Group", "category" : "food_d
ining", "amt" : 66.54, "first" : "Kayla", "last" : "Obrien", "gender" : "F", "city" : "Prairie Hill", "state" : "TX", "zip" : 76678, "city_pop" : 263, "job" : "Barrister", "dob" : "1956-05-30", "is_fraud"
 : 0 }
{ "_id" : ObjectId("624f71895a41f3f249efe6df"), "" : 10, "trans_date_trans_time" : "2020-06-21 12:16:20", "cc_num" : NumberLong("2242542703101233"), "merchant" : "fraud_Feil, Hilpert and Koss", "category"
 : "food_dining", "amt" : 7.01, "first" : "Samuel", "last" : "Jenkins", "gender" : "M", "city" : "Westport", "state" : "KY", "zip" : 40077, "city_pop" : 564, "job" : "Pensions consultant", "dob" : "1996-0
4-10", "is_fraud" : 0 }
>
```

Data Cleaning

Removed unwanted columns using Python

603 project workon.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share H

+ Code + Text

RAM Disk Editing

```
[6] print("Number rows and columns in a dataframe:",Dataframe.shape)
    print("dimensions of dataset:",Dataframe.ndim)
    print('-----------------------------------------------------------------')
    Dataframe.info()
```

```
Number rows and columns in a dataframe: (1852394, 23)
dimensions of dataset: 2
-----------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1852394 entries, 0 to 1852393
Data columns (total 23 columns):
 #   Column                 Dtype
---  ------                 -----
 0   Unnamed: 0             int64
 1   trans_date_trans_time  object
 2   cc_num                 int64
 3   merchant               object
 4   category               object
 5   amt                    float64
 6   first                  object
 7   last                   object
 8   gender                 object
 9   street                 object
 10  city                   object
 11  state                  object
 12  zip                    int64
 13  lat                    float64
 14  long                   float64
 15  city_pop               int64
 16  job                    object
 17  dob                    object
 18  trans_num              object
 19  unix_time              int64
 20  merch_lat              float64
 21  merch_long             float64
 22  is_fraud               int64
dtypes: float64(5), int64(6), object(12)
memory usage: 339.2+ MB
```

1s completed at 12:45 PM

```
Dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1852394 entries, 0 to 1852393
Data columns (total 15 columns):
 #   Column                 Dtype
---  ------                 -----
 0   trans_date_trans_time  object
 1   cc_num                 int64
 2   merchant               object
 3   category               object
 4   amt                    float64
 5   first                  object
 6   last                   object
 7   gender                 object
 8   city                   object
 9   state                  object
 10  zip                    int64
 11  city_pop               int64
 12  job                    object
 13  dob                    object
 14  is_fraud               int64
dtypes: float64(1), int64(4), object(10)
memory usage: 226.1+ MB
```

Checked for missing values and duplicate records

603 Pyspark-2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share H

+ Code + Text RAM Disk Editing

```
df.printSchema()
```

```
root
 |-- _c0: integer (nullable = true)
 |-- trans_date_trans_time: string (nullable = true)
 |-- cc_num: long (nullable = true)
 |-- merchant: string (nullable = true)
 |-- category: string (nullable = true)
 |-- amt: double (nullable = true)
 |-- first: string (nullable = true)
 |-- last: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- zip: integer (nullable = true)
 |-- city_pop: integer (nullable = true)
 |-- job: string (nullable = true)
 |-- dob: string (nullable = true)
 |-- is_fraud: integer (nullable = true)
```

```
#df.describe().toPandas()
```

```
#checking for missing values
```

```
[9] df.select([count(when(isnan(c),c)).alias(c) for c in df.columns]).toPandas().head()
```

	_c0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	city	state	zip	city_pop	job	dob	is_fraud
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

There are no missing values

603 project workon.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share H

+ Code + Text RAM Disk Editing

```
[11] Dataframe_=Dataframe.drop_duplicates(keep='first')
```

```
Dataframe_.head()
```

	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	city	state	zip	city_pop	job	dob	is_fraud
0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care	2.86	Jeff	Elliott	M	Columbia	SC	29209	333497	Mechanical engineer	1968-03-19	0
1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care	29.84	Joanne	Williams	F	Altonah	UT	84002	302	Sales professional, IT	1990-01-17	0
2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley	Lopez	F	Bellmore	NY	11710	34496	Librarian, public	1970-10-21	0
3	2020-06-21 12:15:15	3591919803438423	fraud_Haley Group	misc_pos	60.05	Brian	Williams	M	Titusville	FL	32780	54767	Set designer	1987-07-25	0
4	2020-06-21 12:15:17	3526826139003047	fraud_Johnston-Casper	travel	3.19	Nathan	Massey	M	Falmouth	MI	49632	1126	Furniture designer	1955-07-06	0

```
print(Dataframe_.shape) #After checking for duplicates
print(Dataframe.shape) #Before checking for duplicates
```

```
(1852394, 15)
(1852394, 15)
```

Visualization using PySpark

How many fraudulent transactions are there?

What is the severity of the fraud transaction?

603 Pyspark-2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share H

+ Code + Text

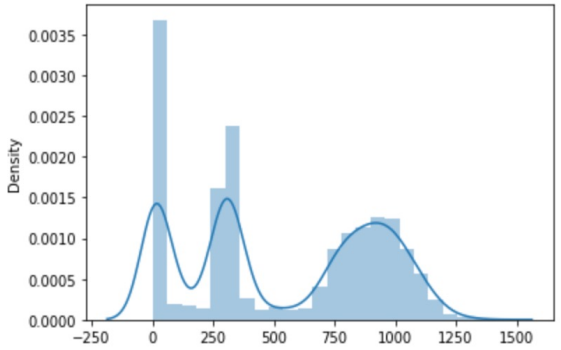
RAM Disk Editing

What is the severity of the fraud transaction?

```
[ ] dist_df = Fraud_df.select(['amt'])
```

```
#converting dist df to pandas df
pandas_df = dist_df.toPandas()
sns.distplot(pandas_df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future vers
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f63a75067d0>
```

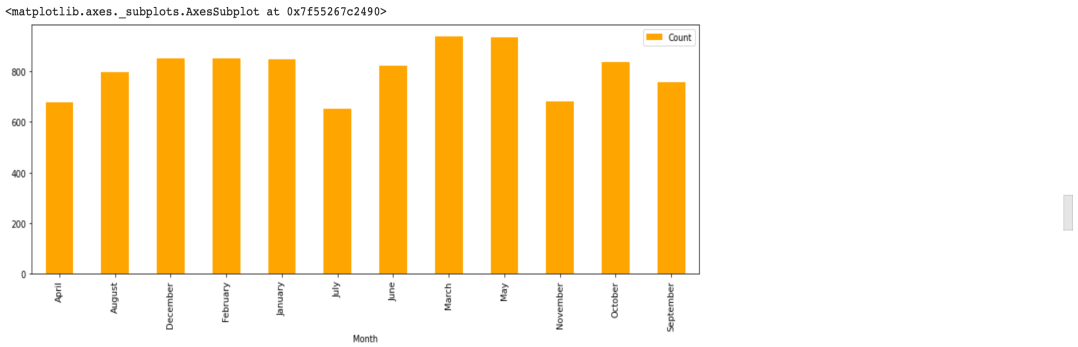


Plotted a distribution plot.

How frequently do fraud transactions occur?

```
[33] import pandas as pd
     month = pd.DataFrame({'Month': ['April', 'August', 'December', 'February', 'January', 'July', 'June', 'March', 'May', 'November', 'October', 'September'],
     month.plot.bar(x='Month', y='Count', figsize=(15,5), color='orange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f55267c2490>
```




```
month = newdf.groupBy('date').count().orderBy('count').sort(asc("date"))
month.show()
```

```
+---------+-----+
|     date|count|
+---------+-----+
|    April|  678|
|   August|  797|
| December|  850|
| February|  853|
|  January|  849|
|     July|  652|
|     June|  821|
|    March|  938|
|      May|  935|
| November|  682|
|  October|  838|
|September|  758|
+---------+-----+
```

- Tried to find the pattern of data's frequency based on months
- Converted "trans_date_trans_time" column data type from string to date and extracted month from the date and time.

Merchants in shopping category are more vulnerable to fraud

Analyzed vulnerability based on maximum fraud transactions in category of merchants

603 Pyspark Updated.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share H

+ Code + Text

RAM Disk Editing

Which merchants are more vulnerable to fraud?

```
C_df = newdf.groupBy('category').count().orderBy('count')
```

```
C_df.show()
```

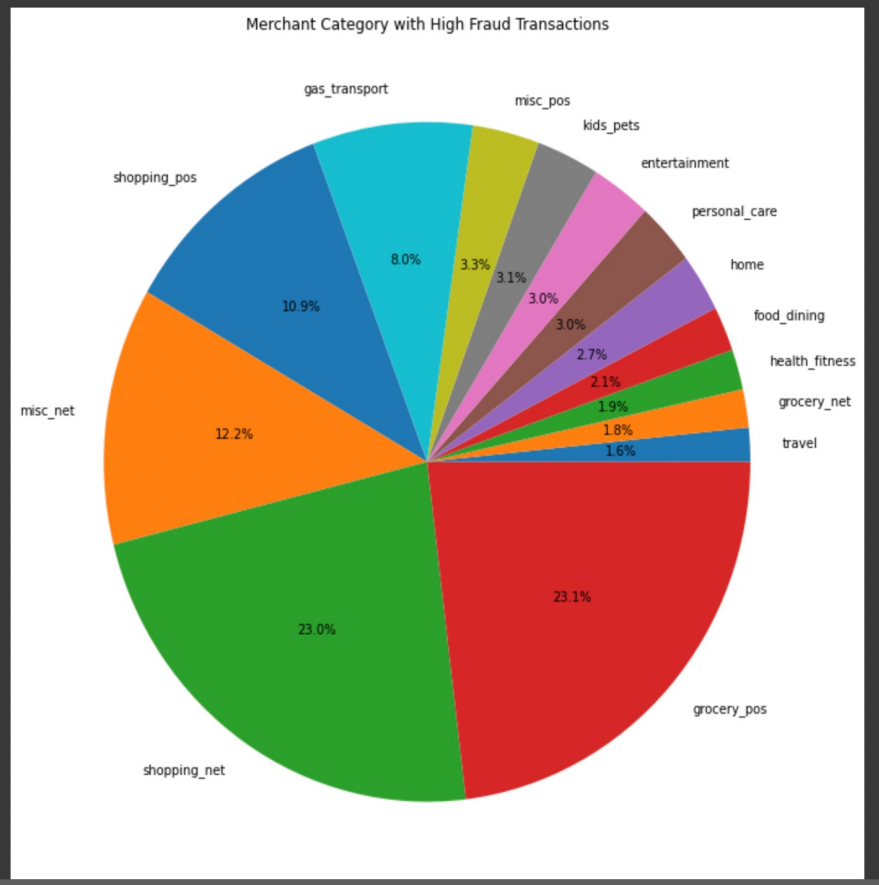
```
+--------------+-----+
|      category|count|
+--------------+-----+
|        travel|  156|
|   grocery_net|  175|
|health_fitness|  185|
|   food_dining|  205|
|          home|  265|
| personal_care|  290|
| entertainment|  292|
|     kids_pets|  304|
|      misc_pos|  322|
| gas_transport|  772|
|  shopping_pos| 1056|
|      misc_net| 1182|
|  shopping_net| 2219|
|   grocery_pos| 2228|
+--------------+-----+
```

```
[36] y = np.array([156, 175, 185, 205, 265, 290, 292, 304, 322, 772, 1056, 1182, 2219, 2228])
     mylabels = ["travel", "grocery_net", "health_fitness", "food_dining", "home", "personal_care", "entertainment", "kids_pets", "misc_pos", "gas_transport", "s
     plt.pie(y, labels = mylabels, autopct='%1.1f%%')
     fig = plt.gcf()
     fig.set_size_inches(12,12)
     plt.title("Merchant Category with High Fraud Transactions")
     plt.show()
```

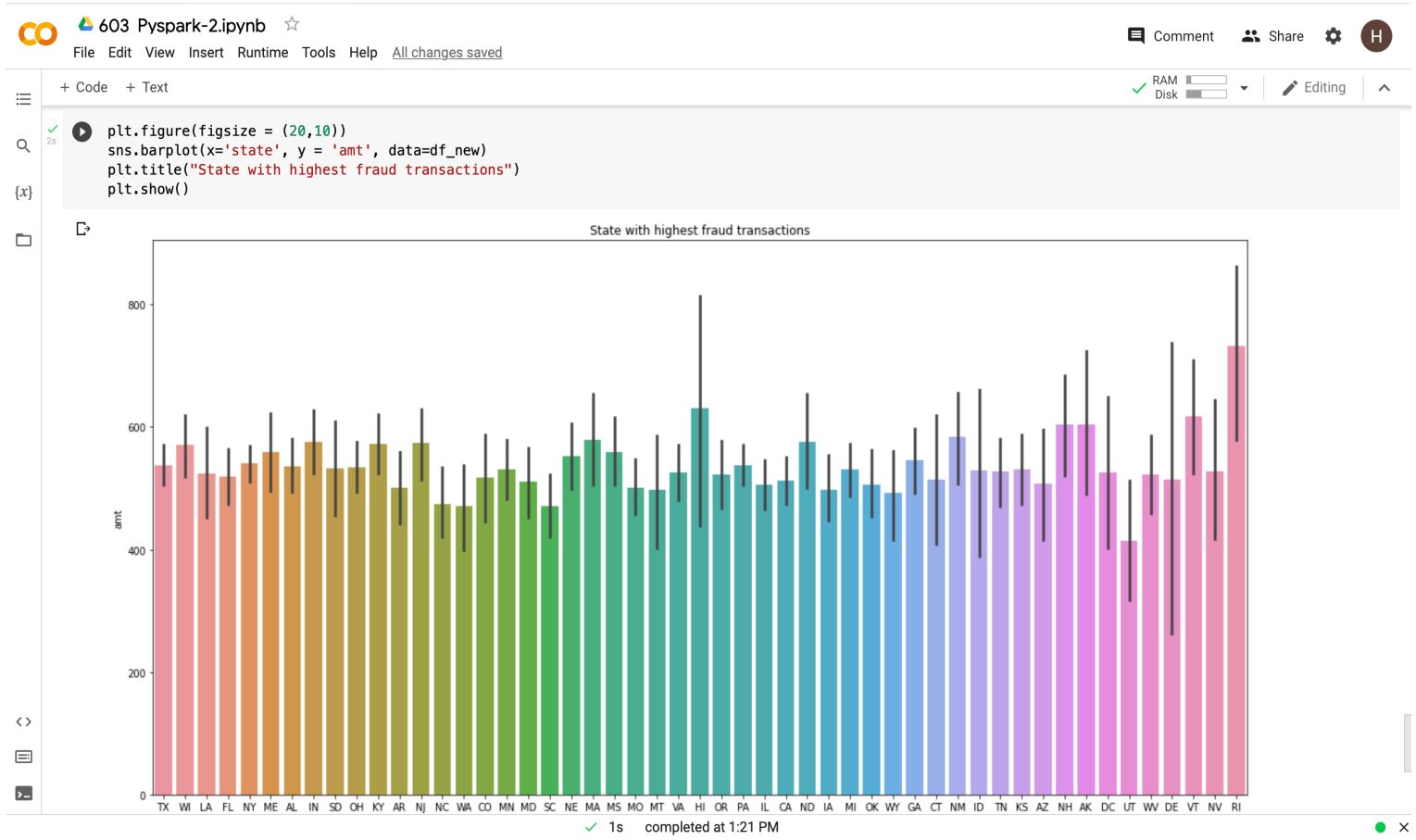
603 Pyspark Updated.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text



Which location has the highest number of fraud transactions?



Plotted Bar graph for fraud transactions in each state.

SPARK MILib -

It is Machine Learning Library in Spark consisting of common Machine Learning algorithms including Classification, Regression, Clustering, etc.

Machine Learning Models Used

1. Logistic Regression
2. Naïve Bayes Classifier
3. Decision Tree Classifier

Implementation

LogisticRegression

In [132]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

In [133]:
```
from pyspark.ml.classification import LogisticRegression
lr=LogisticRegression(featuresCol='features',labelCol='is_fraud_index', maxIter=5)
lrModel = lr.fit(train_df2)
evaluator = BinaryClassificationEvaluator(labelCol="is_fraud_index")
logpred = lrModel.transform(test_df2)
print('Accuracy of Logistic Regression Model:', evaluator.evaluate(logpred))
```

```
Accuracy of Logistic Regression Model: 0.8543367392521881
```

NaiveBayes

In [135]:
```
from pyspark.ml.classification import NaiveBayes
nb = NaiveBayes(modelType="multinomial", labelCol="is_fraud_index")
nbmodel = nb.fit(train_df2)
predictions = nbmodel.transform(test_df2)
evaluator2 = MulticlassClassificationEvaluator(labelCol="is_fraud_index")
nbaccuracy = evaluator2.evaluate(predictions)
print("Accuracy of Naive Bayes Model:" + str(nbaccuracy))
```

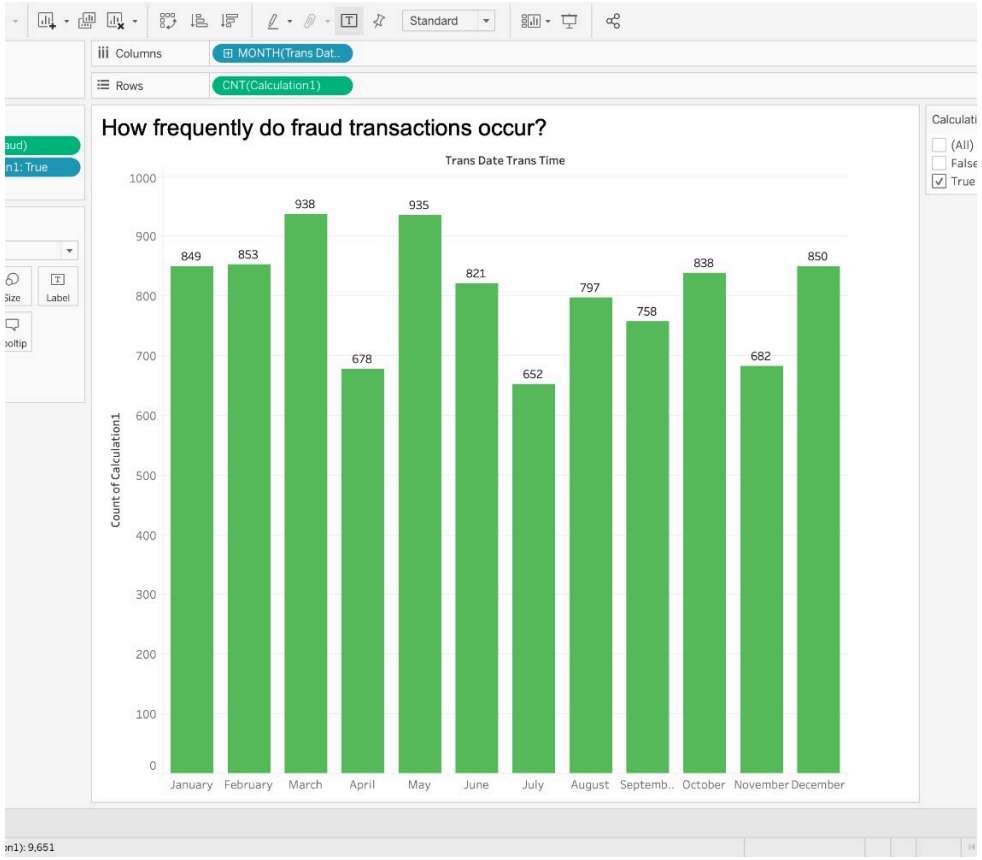
```
Accuracy of Naive Bayes Model:0.8276555047392877
```

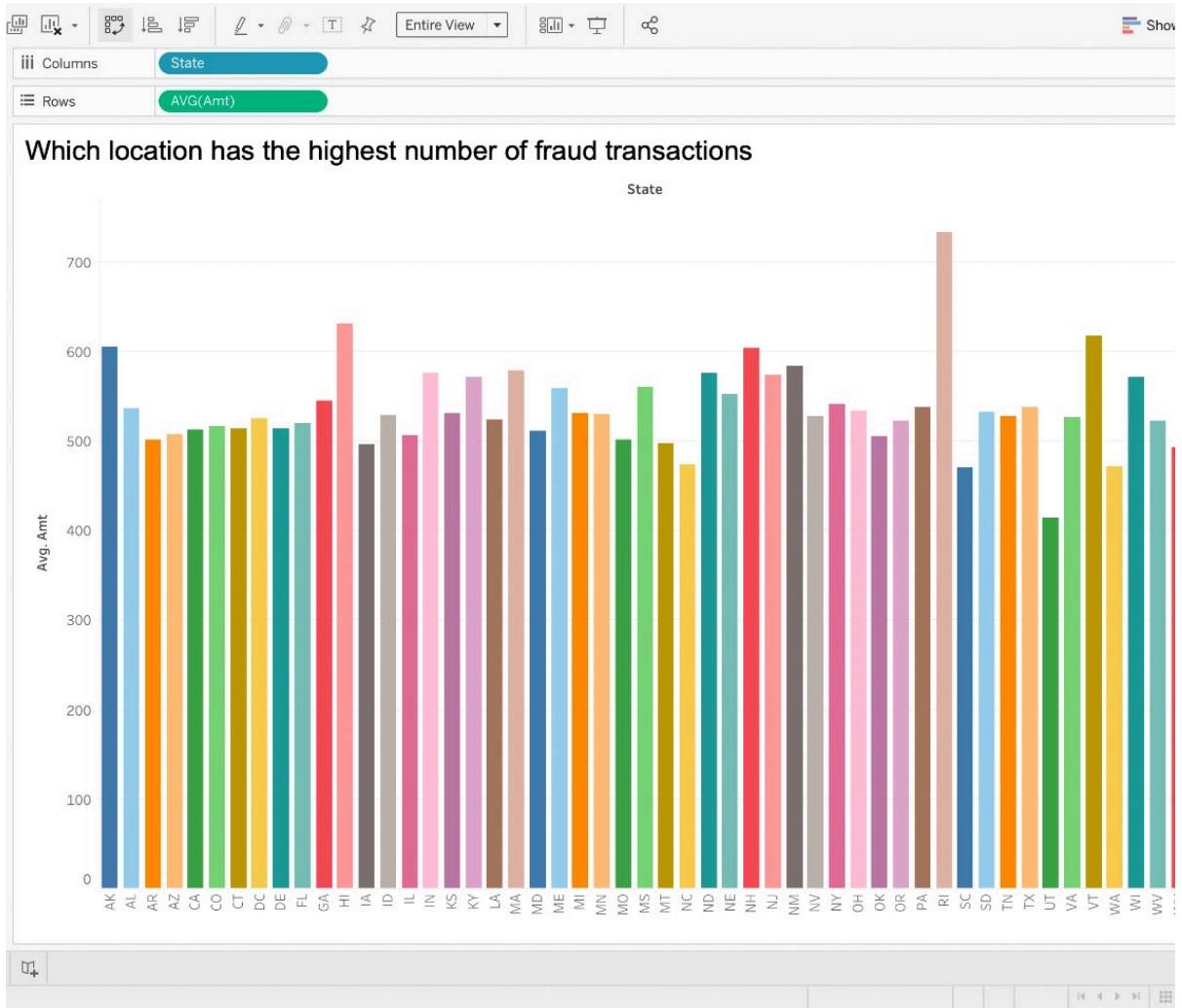
DecisionTreeClassifier

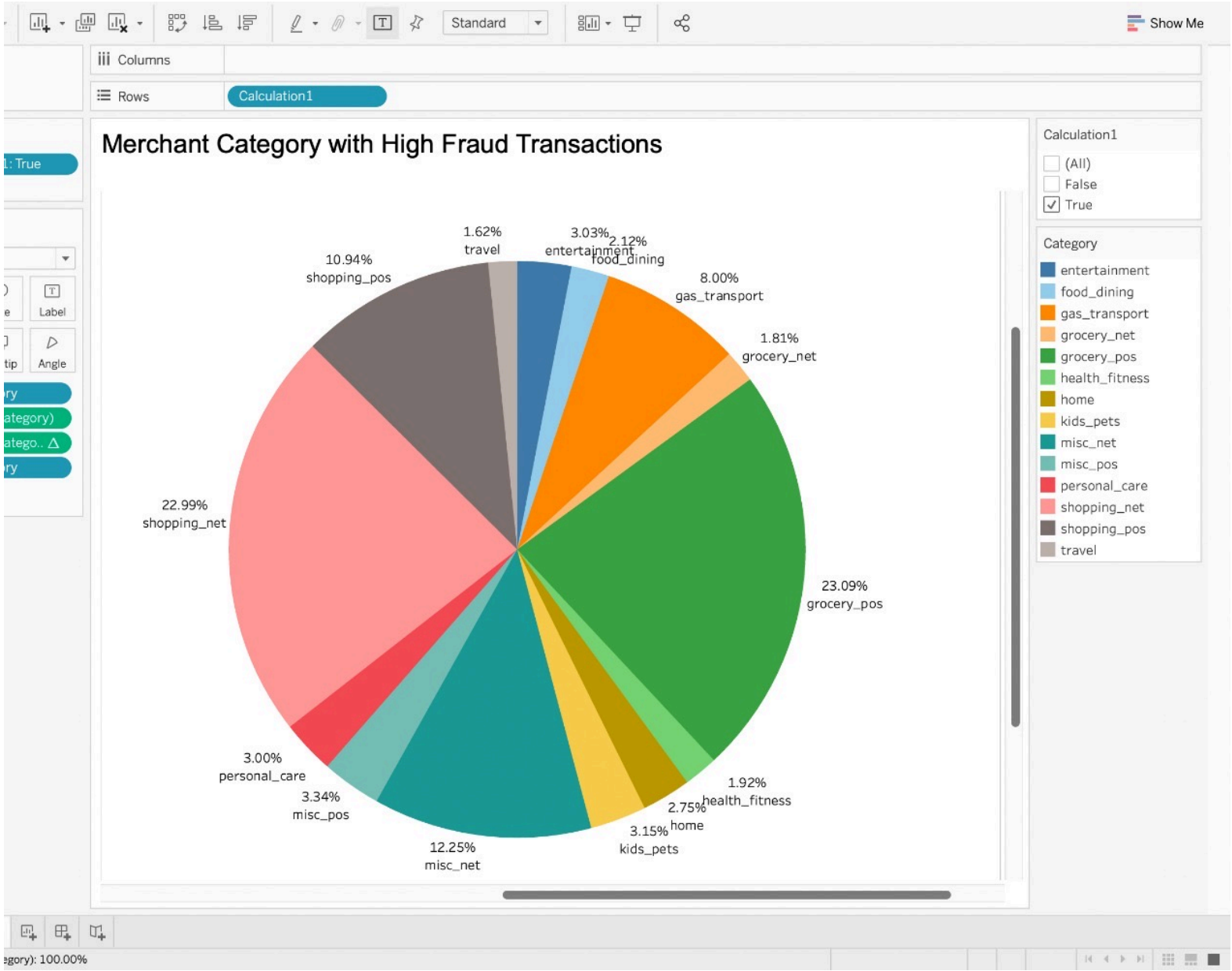
[53]:
```
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'is_fraud_index', impurity =
dtModel = dt.fit(train_df2)
evaluator1 = BinaryClassificationEvaluator(labelCol="is_fraud_index")
predictions = dtModel.transform(test_df2)
print("Accuracy of Decision Tree Model: " + str(evaluator1.evaluate(predictions)))
```

```
Accuracy of Decision Tree Model: 0.7684775513070308
```

Visualization using Tableau






Standard
Show Me
Columns
Rows
Calculation1
Merchant Category with High Fraud Transactions
1.62%
travel
3.03%
entertainment
2.12%
food_dining
10.94%
shopping_pos
8.00%
gas_transport
1.81%
grocery_net
22.99%
shopping_net
23.09%
grocery_pos
3.00%
personal_care
3.34%
misc_pos
1.92%
health_fitness
2.75%
home
3.15%
kids_pets
12.25%
misc_net
Calculation1
(All)
False
True
Category
entertainment
food_dining
gas_transport
grocery_net
grocery_pos
health_fitness
home
kids_pets
misc_net
misc_pos
personal_care
shopping_net
shopping_pos
travel

Conclusion

- Stored and retrieved data from MongoDB
- Performed data cleaning and visualization using Pyspark – Jupyter Notebook
- Predicted the outcomes using Machine Learning Algorithms – MLlib

Learned MongoDB and implementing analysis and prediction using Spark.

Future Work

We plan to work on creating a user interactive website or application to predict if the transaction is legit or fraud for their credit transactions before payments.

References

- https://mint.intuit.com/blog/planning/credit-card-fraud-statistics/
- https://spark.apache.org/docs/latest/api/python/
- https://spark.apache.org/docs/latest/ml-guide.html
- https://towardsdatascience.com/machine-learning-decision-tree-using-spark-for-layman-8eca054c8843

Thank you