BIG DATA WITH SPARK AND HIVE

ANALYSIS OF TELECOM CALL DETAIL RECORDS FOR FRAUD DETECTION USING PYSPARK

USECSAE-ANALYSIS OF TELECOM CALL DETAIL RECORDS FOR FRAUD DETECTION

INTRODUCTION

A telecom operator wants to study call detail records (CDRs) to detect fraud and protect its network from misuse. By analyzing caller and receiver IDs, call duration, call type, SIM and device IDs, transaction status, and fraud type, the company can identify unusual behaviors such as very short or long calls, a high number of night calls, frequent call failures, or mismatches in location and country. Looking at these patterns across regions and time periods helps to uncover fraud hotspots, highlight risky SIMs or devices, and detect suspicious activities in real time. With these insights, the operator can take quick action to prevent revenue loss, improve service quality, protect customer trust, and ensure that communication services remain reliable and secure.

KEY OBJECTIVES

1. Analyze fraud vs non-fraud call distribution to understand the overall occurrence of fraudulent activities.
2. Examine call duration patterns to detect unusually short or long calls that may indicate fraud.
3. Study day-time and night-time calling trends to identify when fraudulent calls are most frequent.
4. Visualize fraud distribution across locations and countries to highlight high-risk regions.
5. Explore the relationship between SIM IDs, device IDs, and fraud cases to detect risky users or devices.
6. Classify fraud types to understand their contribution and frequency in the dataset.
7. Provide actionable insights for telecom operators to detect fraud in real time, reduce losses, and improve service reliability.

GOOGLE COLAB

Google Colab is a free, cloud-based platform provided by Google that allows users to write and execute code in a web-based environment without requiring any local setup. It is built on top of Jupyter Notebook and supports multiple programming languages, with Python being the most commonly used. Colab provides free access to computing resources, including CPUs, GPUs, and TPUs, making it especially useful for tasks involving data analysis, machine

learning, and deep learning. The interface consists of cells where users can write and run code, as well as markdown cells for adding text, images, and equations. Since it runs entirely in the cloud, users can easily share notebooks via Google Drive and collaborate in real time. Google Colab is widely used in both academia and industry because of its accessibility, powerful computing support, and seamless integration with Google services.

KEY FEATURES

1. **Free Cloud Platform** – Allows users to run code without installing software locally.
2. **Pre-installed Libraries** – Provides built-in support for popular Python libraries such as NumPy, Pandas, Matplotlib, and PySpark.
3. **Hardware Acceleration** – Offers free access to CPUs, GPUs, and TPUs for faster execution of heavy computations.
4. **Google Drive Integration** – Enables easy saving, uploading, and sharing of datasets and notebooks.
5. **Collaboration –** Multiple users can work on the same notebook in real time.
6. **Interactive Coding** – Supports code cells, text cells, and visualizations for a smooth workflow.
7. **Big Data & Machine Learning Support –** Well-suited for data analysis, deep learning, and large-scale data processing.

PROBLEM STATEMENT

Telecom companies face many problems because of fraudulent calls and misuse of their network. These activities cause money loss, poor service quality, and reduce customer trust. Fraud can happen in many ways, such as very short or long calls, calls made at unusual times, location mismatches, or suspicious SIM and device usage. Since millions of call records are generated every day, it is very difficult to detect fraud using traditional methods. To solve this issue, we can use Spark in Google Colab to process large call detail records (CDRs) quickly. This helps to find fraud patterns, study their distribution, analyze locations with high fraud, and classify fraud types. With these insights, telecom operators can prevent fraud in real time, protect their network, and improve service quality.

DATASET DESCRIPTION

Attribute 1 – caller_id (int64): Unique identifier of the caller making the call.

Attribute 2 – receiver_id (int64): Unique identifier of the receiver of the call.

Attribute 3 – start_time (object): The timestamp when the call was initiated.

Attribute 4 – duration_sec (int64): Duration of the call in seconds.

Attribute 5 – call_type (object): Type of the call such as local or international.

Attribute 6 – sim_id (object): Identifier of the SIM card used for the call.

Attribute 7 – device_id (object): Identifier of the device from which the call was made.

Attribute 8 – location_origin (object): The originating location of the call.

Attribute 9 – country_origin (object): The country where the call originated (e.g., UG, EG)

Attribute 10 – location_dest (object): The destination location of the call.

Attribute 11 – country_dest (object): The country where the call was received (e.g., ZA, NG)

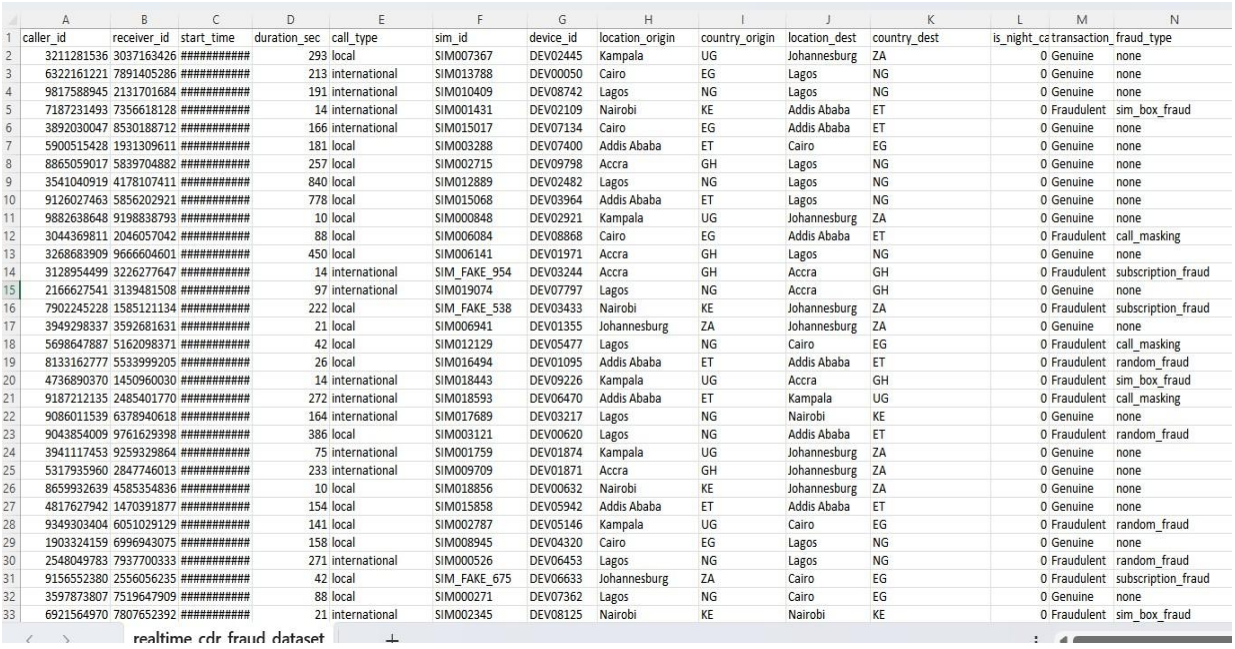
Attribute 12 – is_night_call (int64): Indicates whether the call took place during the night (0 = No, 1 = Yes).

Attribute 13 – transaction_status (object): Status of the call, such as Genuine or Fraudulent.

Attribute 14 – fraud_type (object): Type of fraud detected (e.g., sim_box_fraud) or none if genuine.

DATASET

Link: https://www.kaggle.com/datasets/mehmedumer0028/telecom-cdr-fraud-dataset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	caller_id	receiver_id	start_time	duration_sec	call_type	sim_id	device_id	location_origin	country_origin	location_dest	country_dest	is_night_ca	transaction_	fraud_type
2	3211281536	3037163426	##########	293	local	SIM007367	DEV02445	Kampala	UG	Johannesburg	ZA	0	Genuine	none
3	6322161221	7891405286	##########	213	international	SIM013788	DEV00050	Cairo	EG	Lagos	NG	0	Genuine	none
4	9817588945	2131701684	##########	191	international	SIM010409	DEV08742	Lagos	NG	Lagos	NG	0	Genuine	none
5	7187231493	7356618128	##########	14	international	SIM001431	DEV02109	Nairobi	KE	Addis Ababa	ET	0	Fraudulent	sim_box_fraud
6	3892030047	8530188712	##########	166	international	SIM015017	DEV07134	Cairo	EG	Addis Ababa	ET	0	Genuine	none
7	5900515428	1931309611	##########	181	local	SIM003288	DEV07400	Addis Ababa	ET	Cairo	EG	0	Genuine	none
8	8865059017	5839704882	##########	257	local	SIM002715	DEV09798	Accra	GH	Lagos	NG	0	Genuine	none
9	3541040919	4178107411	##########	840	local	SIM012889	DEV02482	Lagos	NG	Lagos	NG	0	Genuine	none
10	9126027463	5856202921	##########	778	local	SIM015068	DEV03964	Addis Ababa	ET	Lagos	NG	0	Genuine	none
11	9882638648	9198838793	##########	10	local	SIM000848	DEV02921	Kampala	UG	Johannesburg	ZA	0	Genuine	none
12	3044369811	2046057042	##########	88	local	SIM006084	DEV08868	Cairo	EG	Addis Ababa	ET	0	Fraudulent	call_masking
13	3268683909	9666604601	##########	450	local	SIM006141	DEV01971	Accra	GH	Lagos	NG	0	Genuine	none
14	3128954499	3226277647	##########	14	international	SIM_FAKE_954	DEV03244	Accra	GH	Accra	GH	0	Fraudulent	subscription_fraud
15	2166627541	3139481508	##########	97	international	SIM019074	DEV07797	Lagos	NG	Accra	GH	0	Genuine	none
16	7902245228	1585121134	##########	222	local	SIM_FAKE_538	DEV03433	Nairobi	KE	Johannesburg	ZA	0	Fraudulent	subscription_fraud
17	3949298337	3592681631	##########	21	local	SIM006941	DEV01355	Johannesburg	ZA	Johannesburg	ZA	0	Genuine	none
18	5698647887	5162098371	##########	42	local	SIM012129	DEV05477	Lagos	NG	Cairo	EG	0	Fraudulent	call_masking
19	8133162777	5533999205	##########	26	local	SIM016494	DEV01095	Addis Ababa	ET	Addis Ababa	ET	0	Fraudulent	random_fraud
20	4736890370	1450960030	##########	14	international	SIM018443	DEV09226	Kampala	UG	Accra	GH	0	Fraudulent	sim_box_fraud
21	9187212135	2485401770	##########	272	international	SIM018593	DEV06470	Addis Ababa	ET	Kampala	UG	0	Fraudulent	call_masking
22	9086011539	6378940618	##########	164	international	SIM017689	DEV03217	Lagos	NG	Nairobi	KE	0	Genuine	none
23	9043854009	9761629398	##########	386	local	SIM003121	DEV00620	Lagos	NG	Addis Ababa	ET	0	Fraudulent	random_fraud
24	3941117453	9259329864	##########	75	international	SIM001759	DEV01874	Kampala	UG	Johannesburg	ZA	0	Genuine	none
25	5317935960	2847746013	##########	233	international	SIM009709	DEV01871	Accra	GH	Johannesburg	ZA	0	Genuine	none
26	8659932639	4585354836	##########	10	local	SIM018856	DEV00632	Nairobi	KE	Johannesburg	ZA	0	Genuine	none
27	4817627942	1470391877	##########	154	local	SIM015858	DEV05942	Addis Ababa	ET	Addis Ababa	ET	0	Genuine	none
28	9349303404	6051029129	##########	141	local	SIM002787	DEV05146	Kampala	UG	Cairo	EG	0	Fraudulent	random_fraud
29	1903324159	6996943075	##########	158	local	SIM008945	DEV04320	Cairo	EG	Lagos	NG	0	Genuine	none
30	2548049783	7937700333	##########	271	international	SIM000526	DEV06453	Lagos	NG	Lagos	NG	0	Fraudulent	random_fraud
31	9156552380	2556056235	##########	42	local	SIM_FAKE_675	DEV06633	Johannesburg	ZA	Cairo	EG	0	Fraudulent	subscription_fraud
32	3597873807	7519647909	##########	88	local	SIM000271	DEV07362	Lagos	NG	Cairo	EG	0	Genuine	none
33	6921564970	7807652392	##########	21	international	SIM002345	DEV08125	Nairobi	KE	Nairobi	KE	0	Fraudulent	sim_box_fraud

realtime_cdr_fraud_dataset

SOURCE CODE

 # INSTALL PYSPARK

```
!pip install pyspark
```

 # IMPORT LIBRARIES

```
from pyspark.sql import SparkSession

from pyspark.sql.functions import col, count, avg
```

 # CREATE SPARK SESSION

```
spark = SparkSession.builder.appName("CDR Fraud Analysis").getOrCreate()
```

 # Load Dataset

```
df=spark.read.csv("/content/realtime cdr_fraud_dataset.csv",header=True,inferSchema=True)
```

 # Check Schema

```
df.printSchema()
```

```
root
 |-- caller_id: long (nullable = true)
 |-- receiver_id: long (nullable = true)
 |-- start_time: timestamp (nullable = true)
 |-- duration_sec: integer (nullable = true)
 |-- call_type: string (nullable = true)
 |-- sim_id: string (nullable = true)
 |-- device_id: string (nullable = true)
 |-- location_origin: string (nullable = true)
 |-- country_origin: string (nullable = true)
 |-- location_dest: string (nullable = true)
 |-- country_dest: string (nullable = true)
 |-- is_night_call: integer (nullable = true)
 |-- transaction_status: string (nullable = true)
 |-- fraud_type: string (nullable = true)
```

#SHOW SAMPLE ROWS

```
df.show(5)
```

caller_id	receiver_id	start_time	duration_sec	call_type	sim_id	device_id	location_origin	country_origin	location_dest	country_dest	is_night_call	transaction_status	fraud_type
3211281536	3037163426	2025-05-31 16:09:56	293	local	SIM007367	DEV02445	Kampala	UG	Johannesburg	ZA	0	Genuine	none
6322161221	7891405286	2025-05-31 16:09:56	213	international	SIM013788	DEV00050	Cairo	EG	Lagos	NG	0	Genuine	none
9817588945	2131701684	2025-05-31 16:09:56	191	international	SIM010409	DEV08742	Lagos	NG	Lagos	NG	0	Genuine	none
7187231493	7356618128	2025-05-31 16:09:56	14	international	SIM001431	DEV02109	Nairobi	KE	Addis Ababa	ET	0	Fraudulent	sim_box_fraud
3892030047	8530188712	2025-05-31 16:09:56	166	international	SIM015017	DEV07134	Cairo	EG	Addis Ababa	ET	0	Genuine	none

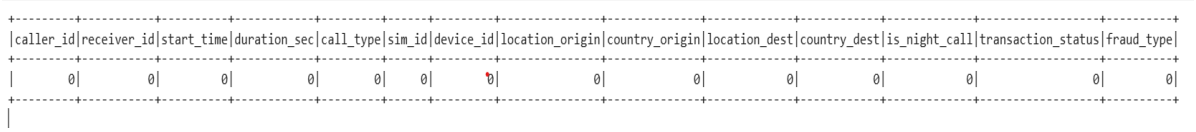
only showing top 5 rows

from pyspark.sql.functions import col, sum

 # COUNT NULLS IN EACH COLUMN

null_counts = df.select([sum(col(c).isNull().cast("int")).alias(c) for c in df.columns])

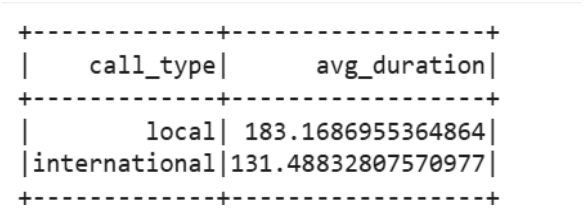
null_counts.show()



```
+---------+-----------+----------+------------+---------+------+---------+---------------+--------------+-------------+------------+---------------+------------------+----------+
|caller_id|receiver_id|start_time|duration_sec|call_type|sim_id|device_id|location_origin|country_origin|location_dest|country_dest|is_night_call|transaction_status|fraud_type|
+---------+-----------+----------+------------+---------+------+---------+---------------+--------------+-------------+------------+---------------+------------------+----------+
|        0|          0|         0|           0|        0|     0|        0|              0|             0|            0|           0|              0|                 0|         0|
+---------+-----------+----------+------------+---------+------+---------+---------------+--------------+-------------+------------+---------------+------------------+----------+
```

 # AVERAGE CALL DURATION BY CALL TYPE

df.groupBy("call_type").agg(avg("duration_sec").alias("avg_duration")).show()



```
+-------------+------------------+
|    call_type|      avg_duration|
+-------------+------------------+
|        local| 183.1686955364864|
|international|131.48832807570977|
+-------------+------------------+
```

 # FRAUD PERCENTAGE BY CALL TYPE

total_calls = df.count()

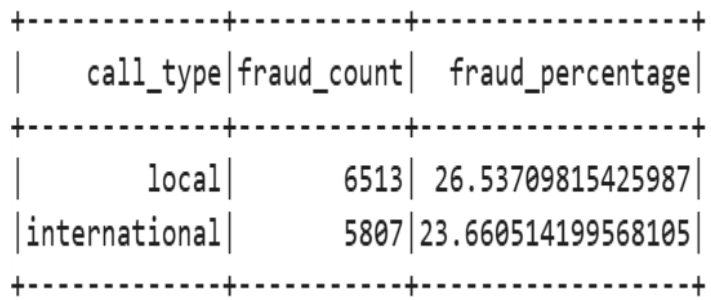
fraud_by_type = df.filter(col("transaction_status")=="Fraudulent") \

.groupBy("call_type").count() \

.withColumnRenamed("count","fraud_count")

fraud_by_type = fraud_by_type.withColumn("fraud_percentage", (col("fraud_count") / total_calls) * 100)

fraud_by_type.show()



```
+-------------+-----------+------------------+
|    call_type|fraud_count|  fraud_percentage|
+-------------+-----------+------------------+
|        local|       6513| 26.53709815425987|
|international|       5807|23.660514199568105|
+-------------+-----------+------------------+
```

 # FRAUD RATE BY COUNTRY (ORIGIN)

```
df.groupBy("country_origin", "transaction_status").count().show(20)
```

```
+--------------+------------------+-----+
|country_origin|transaction_status|count|
+--------------+------------------+-----+
|            GH|        Fraudulent| 1700|
|            ET|           Genuine| 1762|
|            ET|        Fraudulent| 1718|
|            UG|        Fraudulent| 1814|
|            ZA|        Fraudulent| 1770|
|            ZA|           Genuine| 1682|
|            UG|           Genuine| 1782|
|            NG|        Fraudulent| 1732|
|            GH|           Genuine| 1754|
|            EG|           Genuine| 1771|
|            EG|        Fraudulent| 1786|
|            KE|           Genuine| 1749|
|            NG|           Genuine| 1723|
|            KE|        Fraudulent| 1800|
+--------------+------------------+-----+
```

 # TRANSACTIONS BY DURATION & STATUS

```
df.withColumn("duration_category",
            when(col("duration_sec") < 60, "Short (<1min)")
            .when((col("duration_sec") >= 60)&(col("duration_sec") < 300),"Medium (1-5min)")
            .otherwise("Long (>5min)")
           ) \
  .groupBy("duration_category", "transaction_status") \
  .count().orderBy("duration_category").show()
```

```
+-----------------+------------------+-----+
|duration_category|transaction_status|count|
+-----------------+------------------+-----+
|     Long (>5min)|           Genuine| 2371|
|     Long (>5min)|        Fraudulent| 1741|
|  Medium (1-5min)|           Genuine| 6457|
|  Medium (1-5min)|        Fraudulent| 4906|
|    Short (<1min)|        Fraudulent| 5673|
|    Short (<1min)|           Genuine| 3395|
+-----------------+------------------+-----+
```

 # TOP FRAUDULENT ORIGIN → DESTINATION COUNTRY PAIRS

```
df.filter(col("transaction_status")=="Fraudulent")  \
  .groupBy("country_origin", "country_dest") \
  .count().orderBy(desc("count")).show(15)
```

```
+--------------+------------+-----+
|country_origin|country_dest|count|
+--------------+------------+-----+
|            KE|          NG|  309|
|            EG|          ET|  289|
|            ZA|          KE|  287|
|            KE|          ET|  286|
|            EG|          NG|  284|
|            UG|          GH|  283|
|            EG|          ZA|  282|
|            ZA|          EG|  282|
|            UG|          NG|  281|
|            UG|          ET|  281|
|            ET|          EG|  280|
|            GH|          ZA|  275|
|            GH|          UG|  274|
|            UG|          KE|  269|
|            ET|          KE|  268|
+--------------+------------+-----+
only showing top 15 rows
```

 # DISTRIBUTION OF FRAUD TYPES

df.groupBy("fraud_type").count().orderBy(col("count").desc()).show()

```
+-------------------+-----+
|         fraud_type|count|
+-------------------+-----+
|               none|12223|
|      sim_box_fraud| 3096|
|subscription_fraud| 3089|
|       random_fraud| 3083|
|       call_masking| 3052|
+-------------------+-----+
```

 # DISTRIBUTION OF 4 FRAUD TYPES WITH SIM ID

```
from pyspark.sql.functions import col, count

fraud_types = df.select("fraud_type").distinct().rdd.flatMap(lambda x: x).collect()

print(f"Unique fraud types: {fraud_types}")

def fraud_summary(dataframe, fraud_type):

 subset = dataframe.filter(col("fraud_type") == fraud_type)
```

Group by sim_id and call_type and count using Spark operations

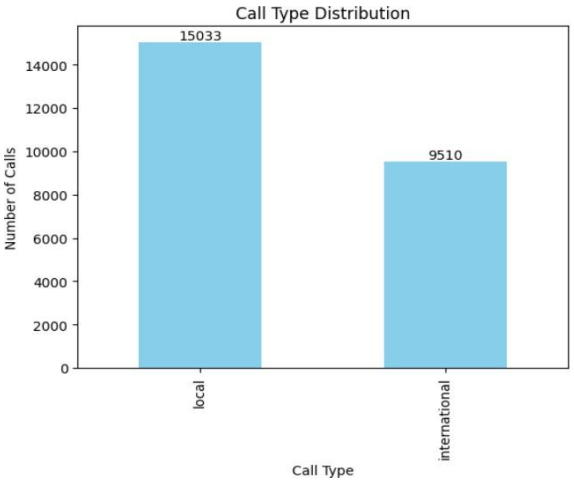
```
result = subset.groupBy('sim_id', 'call_type').agg(count("*").alias("fraud_call_count"))

  print(f"\n--- {fraud_type.upper()} ---")

 result.show(10) # show first 10 for preview

 return resultsim_box = fraud_summary(df, "sim_box_fraud")

subscription = fraud_summary(df, "subscription_fraud")

random_f = fraud_summary(df, "random_fraud")

call_mask = fraud_summary(df, "call_masking")
```

```
+---------+-------------+----------------+
|   sim_id|    call_type|fraud_call_count|
+---------+-------------+----------------+
|SIM013306|international|               1|
|SIM017244|international|               1|
|SIM012955|international|               1|
|SIM006526|international|               1|
|SIM008610|international|               1|
|SIM012529|international|               1|
|SIM009119|international|               1|
|SIM003549|international|               1|
|SIM019163|international|               1|
|SIM005907|international|               1|
+---------+-------------+----------------+
only showing top 10 rows

--- SUBSCRIPTION_FRAUD ---
+------------+-------------+----------------+
|      sim_id|    call_type|fraud_call_count|
+------------+-------------+----------------+
|SIM_FAKE_119|international|               2|
|SIM_FAKE_517|        local|               3|
|SIM_FAKE_614|        local|               3|
|SIM_FAKE_228|international|               2|
|SIM_FAKE_232|        local|               2|
|SIM_FAKE_971|international|               2|
|SIM_FAKE_240|international|               1|
|SIM_FAKE_385|        local|               2|
|SIM_FAKE_858|        local|               3|
|SIM_FAKE_776|        local|               7|
+------------+-------------+----------------+
only showing top 10 rows
```

```
--- RANDOM_FRAUD ---
+---------+-------------+----------------+
|   sim_id|    call_type|fraud_call_count|
+---------+-------------+----------------+
|SIM015794|        local|               1|
|SIM004289|international|               1|
|SIM013130|international|               1|
|SIM000765|international|               1|
|SIM019544|        local|               1|
|SIM019620|international|               1|
|SIM013306|international|               1|
|SIM017829|        local|               1|
|SIM016732|international|               1|
|SIM015548|        local|               1|
+---------+-------------+----------------+
only showing top 10 rows


--- CALL_MASKING ---
+---------+-------------+----------------+
|   sim_id|    call_type|fraud_call_count|
+---------+-------------+----------------+
|SIM012101|        local|               1|
|SIM015211|international|               1|
|SIM003348|        local|               2|
|SIM017829|        local|               1|
|SIM000765|international|               1|
|SIM003025|        local|               1|
|SIM019501|        local|               1|
|SIM001904|international|               1|
|SIM010380|        local|               1|
|SIM003177|        local|               1|
+---------+-------------+----------------+
only showing top 10 rows
```

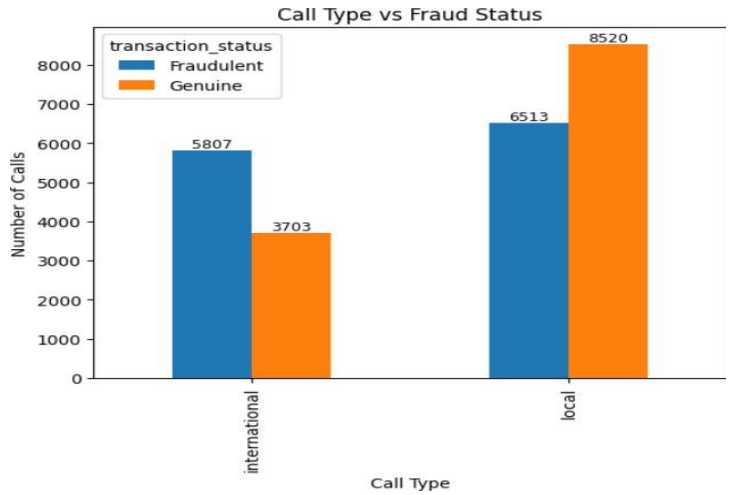
 # CALL TYPE DISTRIBUTION

```
import matplotlib.pyplot as plt

call_type = df.groupBy("call_type").count().toPandas()

call_type.plot(kind="bar", x="call_type", y="count", legend=False, color="skyblue")

plt.title("Call Type Distribution")

plt.xlabel("Call Type")

plt.ylabel("Number of Calls")

plt.show()
```



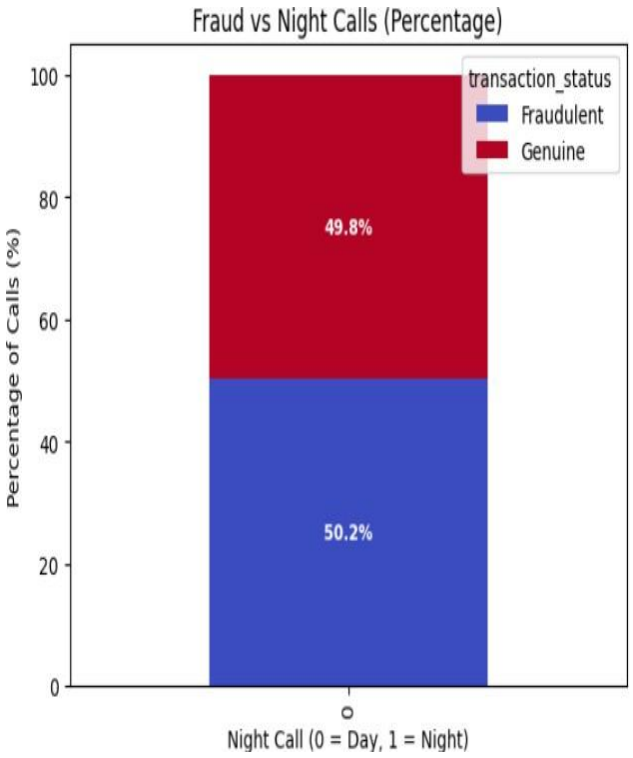
 # CALL TYPE VS FRAUD STATUS

```
call_type_fraud = df.groupBy("call_type", "transaction_status").count().toPandas()

pivot_df = call_type_fraud.pivot(index="call_type", columns="transaction_status", values="count")

ax = pivot_df.plot(kind="bar")

plt.title("Call Type vs Fraud Status")

plt.ylabel("Number of Calls")

plt.xlabel("Call Type")

for container in ax.containers:

    ax.bar_label(container, fmt='%d', label_type='edge', fontsize=9)

plt.show()
```



 # FRAUD VS NIGHT CALLS

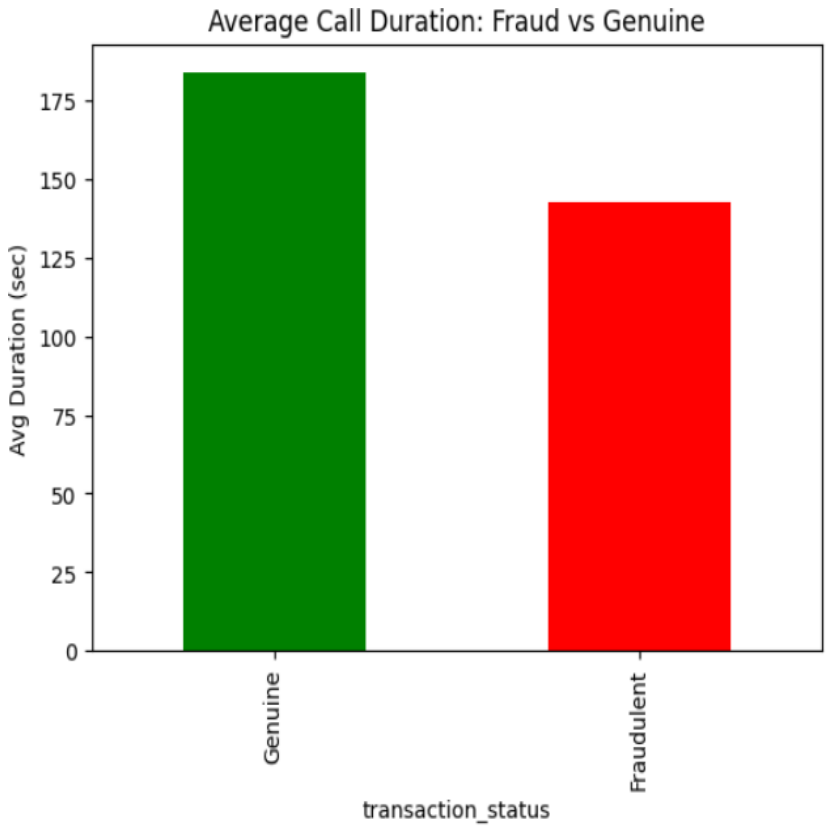
```
fraud_night = df.groupBy("is_night_call", "transaction_status").count().toPandas()
pivot_df=fraud_night.pivot(index="is_night_call",columns="transaction_status",
values="count")
percent_df = pivot_df.div(pivot_df.sum(axis=1), axis=0) * 100
ax = percent_df.plot(kind="bar", stacked=True, cmap="coolwarm")
plt.title("Fraud vs Night Calls (Percentage)")
plt.xlabel("Night Call (0 = Day, 1 = Night)")
plt.ylabel("Percentage of Calls (%)")
for container in ax.containers:
   ax.bar_label(container, fmt="%.1f%%", label_type='center', fontsize=9, color="white",
weight="bold")
plt.show()
```



 # AVERAGE CALL DURATION: FRAUD VS GENUINE

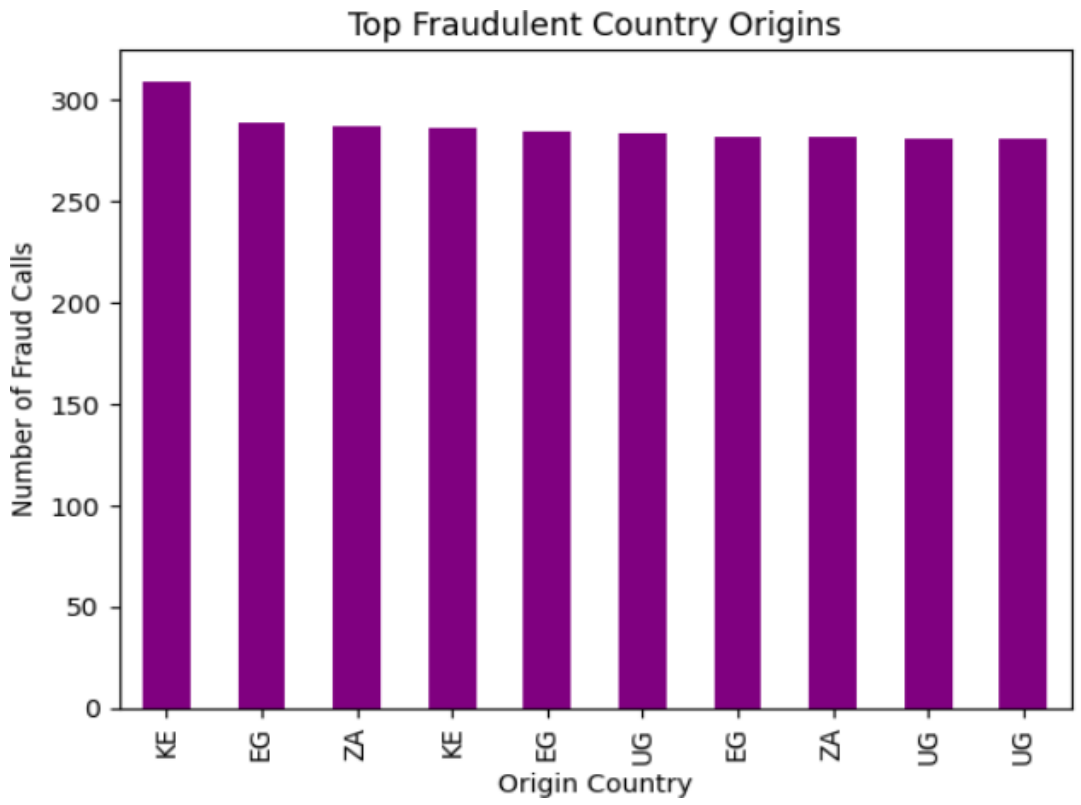
```
duration=df.groupBy("transaction_status").agg(avg("duration_sec").alias("avg_duration")).to
Pandas()
```

```
duration.plot(kind="bar", x="transaction_status", y="avg_duration", color=["green","red"],
legend=False)
plt.title("Average Call Duration: Fraud vs Genuine")
plt.ylabel("Avg Duration (sec)")
plt.show()
```



 # TOP FRAUDULENT COUNTRY ORIGINS

```
fraud_country = df.filter(col("transaction_status")=="Fraudulent") \
        .groupBy("country_origin", "country_dest") \
        .count().orderBy(col("count").desc()).limit(10).toPandas()
fraud_country.plot(kind="bar", x="country_origin", y="count", color="purple",
legend=False)
plt.title("Top Fraudulent Country Origins")
plt.xlabel("Origin Country")
plt.ylabel("Number of Fraud Calls")
plt.show()
```



CONCLUSION

The analysis of telecom call detail records (CDRs) using PySpark has shown how large datasets can be used effectively to detect fraudulent activities in real time. By examining various features such as call type, transaction status, fraud type, call duration, time of day, and location details, important patterns of fraud were uncovered. The study revealed that fraudulent calls often occur during unusual time intervals, show abnormal call durations, and are concentrated in specific locations or countries. Fraud types like sim box fraud were identified as major contributors, highlighting areas where telecom operators must pay close attention. The comparison between genuine and fraudulent calls also showed clear differences in behavior, making it possible to design stronger fraud detection systems. These findings can help telecom operators move from reactive measures to proactive fraud prevention, reduce revenue losses, improve service quality, and strengthen customer trust by ensuring reliable and secure communication services.

REFERENCES

1. Aslam, M., Khan, A. A., Iqbal, Z., & Aslam, A. (2020). Fraud Detection Call Detail Record Using Machine Learning in Telecommunications Company. International Journal of Computer Applications, 175(3), 22–28. https://doi.org/10.5120/ijca2020920979
2. Rathore, M. M., Ahmad, A., Paul, A., & Rho, S. (2016). Detecting fraudulent activities in telecom networks through call detail analysis and big data techniques. International Journal of Communication Systems, 29(5), e2935. https://doi.org/10.1002/dac.2935
3. Kaur, P., & Sharma, A. (2022). Telecom Fraud Detection Using Machine Learning. KTH Royal Institute of Technology, Master's Thesis. Retrieved from https://kth.diva-portal.org/smash/get/diva2%3A1802838/FULLTEXT01.pdf
4. Mahmood, A., Shah, S. A., & Khan, M. A. (2024). Insight into anomaly detection and prediction and mobile network security. Journal of Applied Sciences, 14(2), 45–55. https://doi.org/10.3390/app142215
5. Zhao, Y., Li, J., & Chen, X. (2023). GAT-COBO: Cost-Sensitive Graph Neural Network for Telecom Fraud Detection. arXiv preprint. https://arxiv.org/abs/2303.17334