In [134…

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [135…

```python
customer = pd.read_csv("Customer_Master.csv")
transaction = pd.read_csv("transactions.csv")
```

In [136…

```python
customer.head(2)
```

Out[136…

	customer_id	age_group	home_location	credit_score	account_age_years	account_type
0	1	26-35	Palpa	714	13	Savings
1	2	26-35	Kathmandu	607	7	Savings

In [137…

```python
transaction.head(2)
```

Out[137…

	transaction_id	customer_id	transaction_date	transaction_type	amount	locati
0	TXN20241124104326	727	11/24/2024 15:29	Inward Remittance	13925.72	Nawalpar
1	TXN20241204130277	539	12/4/2024 5:26	ATM Withdrawal	25037.35	Ka

2 rows × 28 columns

Find intersecting columns

In [138…

```python
duplicate_cols = set(customer.columns).intersection(set(transaction.columns))
duplicate_cols
```

Out[138…

```
{'account_age_years',
 'account_type',
 'age_group',
 'avg_monthly_income',
 'credit_score',
 'customer_id',
 'employment_status',
 'home_location',
 'international_activity',
 'mobile_banking_user',
 'risk_score',
 'transaction_frequency'}
```

In [139…

```python
duplicate_cols = duplicate_cols - {'customer_id'}
duplicate_cols
```

Out[139…

```
{'account_age_years',
 'account_type',
 'age_group',
 'avg_monthly_income',
 'credit_score',
 'employment_status',
 'home_location',
 'international_activity',
 'mobile_banking_user',
 'risk_score',
 'transaction_frequency'}
```

In [140…

```
transaction = transaction.drop(columns= duplicate_cols)
```

In [141…

```
#check duplicate again
duplicate_cols = set(customer.columns).intersection(set(transaction.columns))
duplicate_cols
```

Out[141…

```
{'customer_id'}
```

merge these datasets

In [142…

```
dataset = customer.merge(transaction, on="customer_id", how="inner")
```

In [143…

```
dataset.head(5)
```

Out[143…

	customer_id	age_group	home_location	credit_score	account_age_years	account_type
0	1	26-35	Palpa	714	13	Savings
1	1	26-35	Palpa	714	13	Savings
2	1	26-35	Palpa	714	13	Savings
3	1	26-35	Palpa	714	13	Savings
4	1	26-35	Palpa	714	13	Savings

5 rows × 33 columns

In [144…

```
#check for null
dataset.isna().sum()
```

Out[144…

```
customer_id                    0
age_group                      0
home_location                  0
credit_score                   0
account_age_years              0
account_type                   0
avg_monthly_income             0
mobile_banking_user            0
primary_device             14202
primary_os                 14202
primary_browser            14202
avg_transaction_amount         0
transaction_frequency          0
employment_status              0
preferred_transaction_types    0
international_activity         0
risk_score                     0
transaction_id                 0
transaction_date               0
transaction_type               0
amount                         0
location                       0
ip_address                     0
device                     79991
os                         79963
browser                    79967
attempt_sequence               0
time_of_day                    0
transaction_velocity           0
status                         0
auth_method                    0
amount_deviation               0
is_suspicious                  0
dtype: int64
```

In [145…

```
dataset.shape
```

Out[145…

```
(103500, 33)
```

In [146…

```
#device, os and browser have about 77% of null data so we drop those columns
dataset.drop(columns=["device", "os", "browser", "attempt_sequence"], inplace= True
```

In [147…

```
dataset.duplicated().sum()
```

Out[147…

```
np.int64(0)
```

In [148…

```
dataset.isna().sum()
```

Out[148…

```
customer_id                        0
age_group                          0
home_location                      0
credit_score                       0
account_age_years                  0
account_type                       0
avg_monthly_income                 0
mobile_banking_user                0
primary_device                 14202
primary_os                     14202
primary_browser                14202
avg_transaction_amount             0
transaction_frequency              0
employment_status                  0
preferred_transaction_types        0
international_activity             0
risk_score                         0
transaction_id                     0
transaction_date                   0
transaction_type                   0
amount                             0
location                           0
ip_address                         0
time_of_day                        0
transaction_velocity               0
status                             0
auth_method                        0
amount_deviation                   0
is_suspicious                      0
dtype: int64
```

In [149…

```
#now filling values with mode for 3 remaining columns
dataset["primary_device"].fillna(dataset["primary_device"].mode()[0], inplace= True
dataset["primary_os"].fillna(dataset["primary_os"].mode()[0], inplace= True)
dataset["primary_browser"].fillna(dataset["primary_browser"].mode()[0], inplace= Tr
```

```
C:\Users\Sandesh Khatiwada\AppData\Local\Temp\ipykernel_11972\1276973883.py:2: Futur
eWarning: A value is trying to be set on a copy of a DataFrame or Series through cha
ined assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  dataset["primary_device"].fillna(dataset["primary_device"].mode()[0], inplace= Tru
e)
C:\Users\Sandesh Khatiwada\AppData\Local\Temp\ipykernel_11972\1276973883.py:3: Futur
eWarning: A value is trying to be set on a copy of a DataFrame or Series through cha
ined assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  dataset["primary_os"].fillna(dataset["primary_os"].mode()[0], inplace= True)
C:\Users\Sandesh Khatiwada\AppData\Local\Temp\ipykernel_11972\1276973883.py:4: Futur
eWarning: A value is trying to be set on a copy of a DataFrame or Series through cha
ined assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  dataset["primary_browser"].fillna(dataset["primary_browser"].mode()[0], inplace= T
rue)
```

In [150…

```
dataset.isna().sum().sum()
```

Out[150…

```
np.int64(0)
```

In [151…

```
dataset.head(3)
```

Out[151...

	customer_id	age_group	home_location	credit_score	account_age_years	account_type
0	1	26-35	Palpa	714	13	Savings
1	1	26-35	Palpa	714	13	Savings
2	1	26-35	Palpa	714	13	Savings

3 rows × 29 columns

Comparing different sorts of features

In [152...

```
# visualizing which transaction type fraud is the most
count_and_rate = (
    dataset.groupby('transaction_type')['is_suspicious']
    .agg(['sum', 'count', 'mean'])
    .sort_values(by='mean', ascending=False)
)

print(count_and_rate)
```

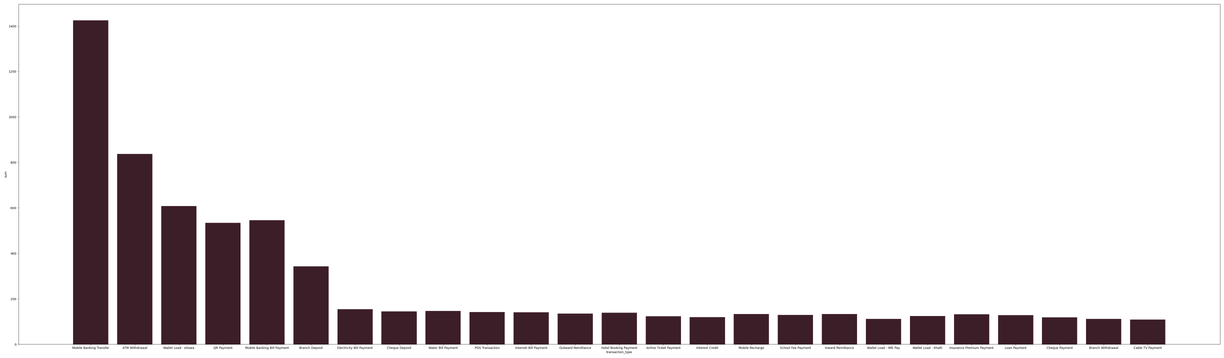
```
                              sum  count      mean
transaction_type
Mobile Banking Transfer      1425   4775  0.298429
ATM Withdrawal                837   4947  0.169193
Wallet Load - eSewa           608   4514  0.134692
QR Payment                    534   4382  0.121862
Mobile Banking Bill Payment   546   4652  0.117369
Branch Deposit                343   4161  0.082432
Electricity Bill Payment      154   3843  0.040073
Cheque Deposit                144   3626  0.039713
Water Bill Payment            146   3822  0.038200
POS Transaction               141   3836  0.036757
Internet Bill Payment         140   3877  0.036110
Outward Remittance            135   3849  0.035074
Hotel Booking Payment         139   4021  0.034569
Airline Ticket Payment        123   3840  0.032031
Interest Credit               119   3722  0.031972
Mobile Recharge               133   4245  0.031331
School Fee Payment            129   4159  0.031017
Inward Remittance             133   4404  0.030200
Wallet Load - IME Pay         111   3687  0.030106
Wallet Load - Khalti          124   4166  0.029765
Insurance Premium Payment     132   4437  0.029750
Loan Payment                  128   4361  0.029351
Cheque Payment                118   4116  0.028669
Branch Withdrawal             111   4042  0.027462
Cable TV Payment              108   4016  0.026892
```

In [153...

```
plt.figure(figsize=(70,20))
sns.barplot(
    data=count_and_rate,
```

```
    x = 'transaction_type',
    y = 'sum',
    color = '#451828'
)
plt.plot()
```

Out[153… []



In [154…

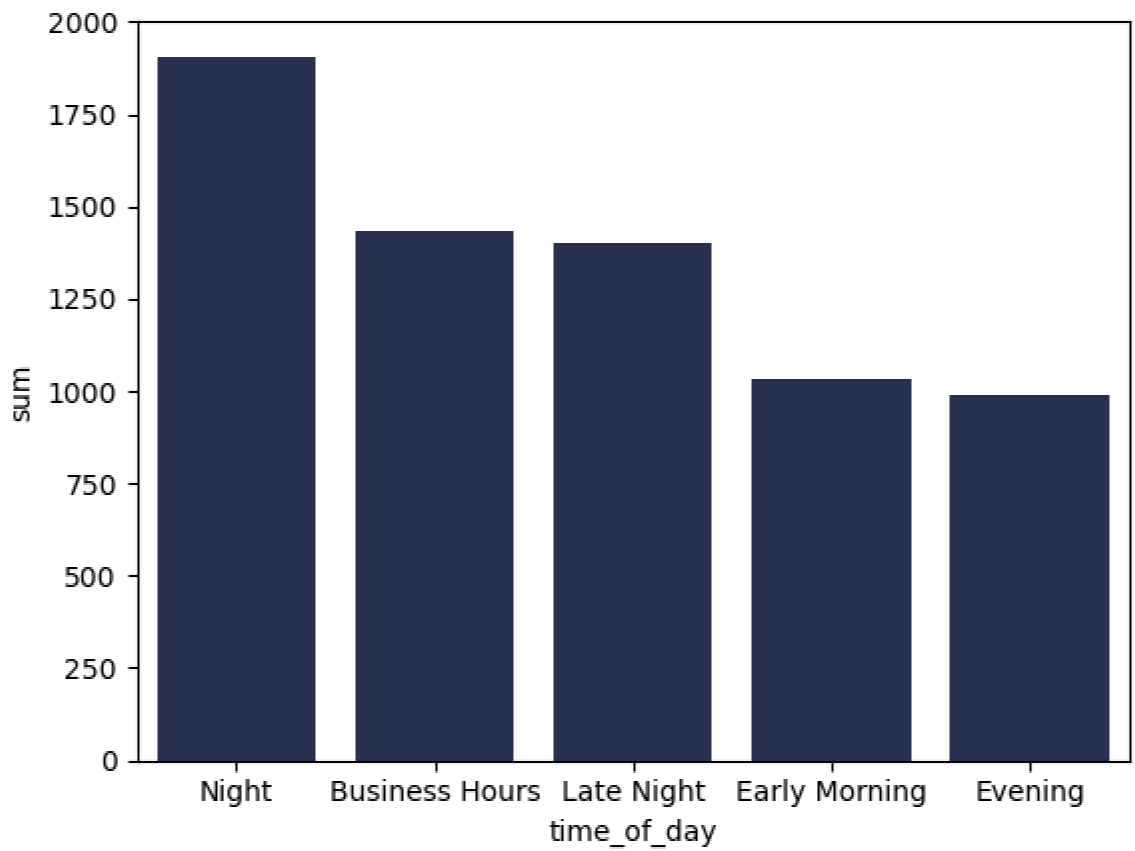
```
# time of day vs fraud
fraud_by_time = (
    dataset.groupby('time_of_day')['is_suspicious']
    .agg(['sum', 'mean'])
    .sort_values(by = 'sum', ascending=False)
    .reset_index()
)

print(fraud_by_time)
```

```
      time_of_day   sum      mean
0           Night  1907  0.137115
1  Business Hours  1434  0.042668
2      Late Night  1401  0.054545
3   Early Morning  1031  0.077864
4         Evening   988  0.057920
```

In [155…

```
sns.barplot(
    data = fraud_by_time,
    x = 'time_of_day',
    y = 'sum',
    color= '#252d59'
)
```

Out[155… <Axes: xlabel='time_of_day', ylabel='sum'>



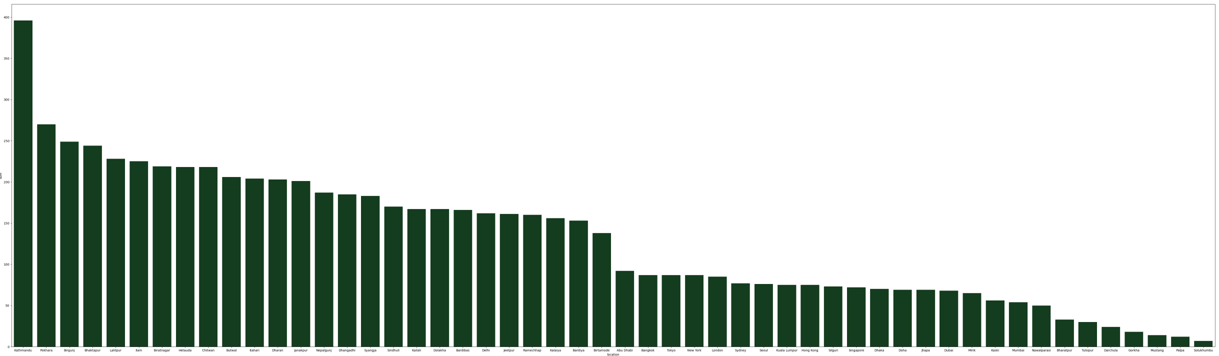
In [156…

```
#location vs fraud
fraud_by_location = (
    dataset.groupby('location')['is_suspicious']
    .agg(['sum', 'mean'])
    .sort_values(by = 'sum', ascending= False)
)

print(fraud_by_location)
```

	sum	mean
location		
Kathmandu	396	0.020643
Pokhara	270	0.032032
Birgunj	249	0.057546
Bhaktapur	244	0.051597
Lalitpur	228	0.051213
Ilam	225	0.069103
Biratnagar	219	0.045540
Hetauda	218	0.062662
Chitwan	218	0.069338
Butwal	206	0.076637
Itahari	204	0.080473
Dharan	203	0.058233
Janakpur	201	0.076021
Nepalgunj	187	0.074118
Dhangadhi	185	0.088900
Syangja	183	0.197198
Sindhuli	170	0.184783
Kailali	167	0.170234
Dolakha	167	0.178419
Bardibas	166	0.180043
Delhi	162	1.000000
Jeetpur	161	0.177508
Ramechhap	160	0.170576
Kalaiya	156	0.163522
Bardiya	153	0.160042
Birtamode	138	0.154190
Abu Dhabi	92	1.000000
Bangkok	87	1.000000
Tokyo	87	1.000000
New York	87	1.000000
London	85	1.000000
Sydney	77	1.000000
Seoul	76	1.000000
Kuala Lumpur	75	1.000000
Hong Kong	75	1.000000
Silguri	73	1.000000
Singapore	72	1.000000
Dhaka	70	1.000000
Doha	69	1.000000
Jhapa	69	0.015732
Dubai	68	1.000000
Mirik	65	1.000000
Kaski	56	0.014308
Mumbai	54	1.000000
Nawalparasi	50	0.016345
Bharatpur	33	0.016658
Tulsipur	30	0.012077
Darchula	24	0.016690
Gorkha	18	0.014458
Mustang	14	0.018767
Palpa	12	0.008708
Solukhumbu	7	0.017812

In [157…

```python
#figure for this
plt.figure(figsize=(70,20))
sns.barplot(data=fraud_by_location, x= 'location', y = 'sum', color='#0f451d')
plt.show()
```



In [158…

```python
#fraud by age
dataset['age_group'] = pd.Categorical(
    dataset['age_group'],
    ordered=True,
    categories=['18-25', '26-35', '36-45', '46-55', '56+']
)

dataset.groupby('age_group')['is_suspicious'].sum()
```

```
C:\Users\Sandesh Khatiwada\AppData\Local\Temp\ipykernel_11972\3527396111.py:8: Futur
eWarning: The default of observed=False is deprecated and will be changed to True in
a future version of pandas. Pass observed=False to retain current behavior or observ
ed=True to adopt the future default and silence this warning.
  dataset.groupby('age_group')['is_suspicious'].sum()
```

Out[158…

```
age_group
18-25    1070
26-35    2032
36-45    1526
46-55    1104
56+         0
Name: is_suspicious, dtype: int64
```

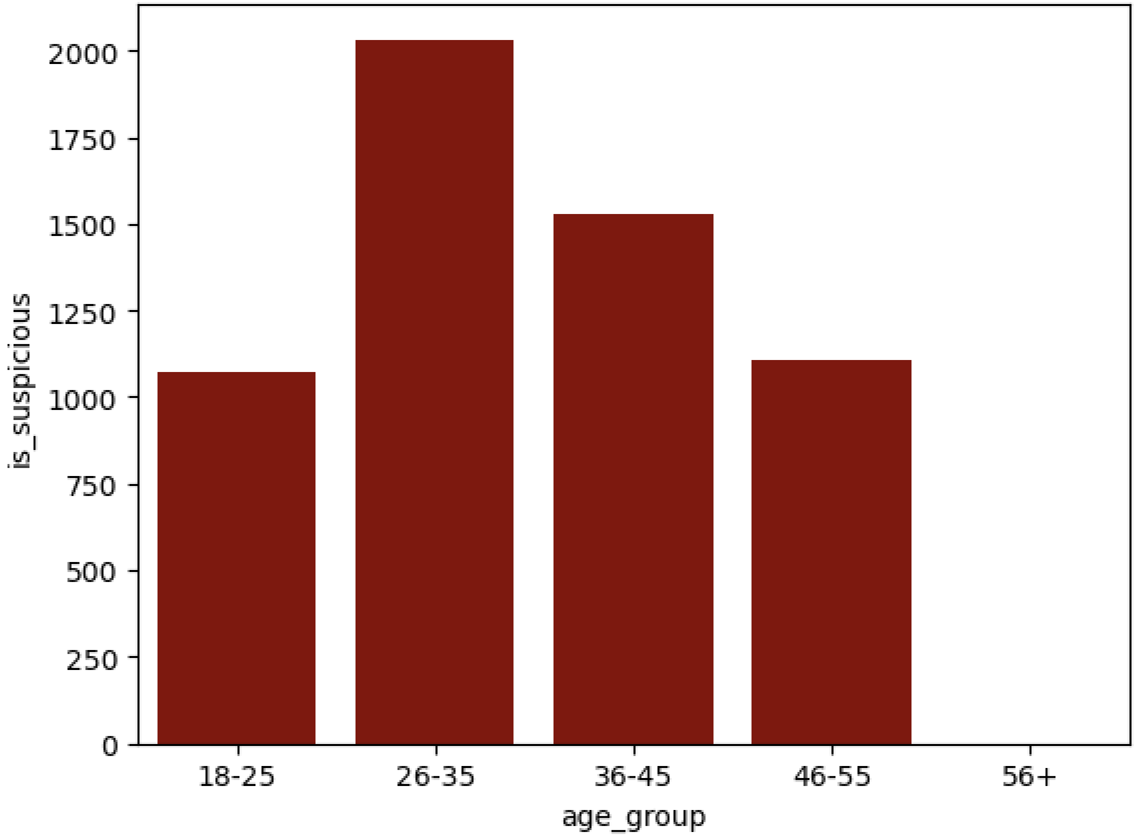
In [159…

```python
#make graph for this
# aggregate first
age_fraud = dataset.groupby('age_group', as_index=False)['is_suspicious'].sum()

# plot
sns.barplot(data=age_fraud, x='age_group', y='is_suspicious', color='#8f0c00')
```

```
C:\Users\Sandesh Khatiwada\AppData\Local\Temp\ipykernel_11972\143820651.py:3: Future
Warning: The default of observed=False is deprecated and will be changed to True in
a future version of pandas. Pass observed=False to retain current behavior or observ
ed=True to adopt the future default and silence this warning.
  age_fraud = dataset.groupby('age_group', as_index=False)['is_suspicious'].sum()
```

Out[159…

```
<Axes: xlabel='age_group', ylabel='is_suspicious'>
```




```
In [ ]:
```

```
In [160…
#comparing fraud against various features
combined = (
    dataset.groupby(['transaction_type', 'time_of_day', 'location', 'age_group'])['
    .agg(['sum', 'count'])
    .sort_values(by='sum', ascending=False)
)

print(combined)
```

```
C:\Users\Sandesh Khatiwada\AppData\Local\Temp\ipykernel_11972\318046994.py:3: Future
Warning: The default of observed=False is deprecated and will be changed to True in
a future version of pandas. Pass observed=False to retain current behavior or observ
ed=True to adopt the future default and silence this warning.
  dataset.groupby(['transaction_type', 'time_of_day', 'location', 'age_group'])['is_
suspicious']
```

```
                                                                          sum  count
transaction_type        time_of_day    location  age_group
Mobile Banking Transfer Night          Butwal    26-35                     13     18
                                       Syangja   26-35                     13     14
Wallet Load - eSewa     Night          Lalitpur  26-35                     12     18
ATM Withdrawal          Night          Birgunj   36-45                     12     13
Mobile Banking Transfer Night          Dhangadhi 26-35                     10     12
...                                                                       ...    ...
Airline Ticket Payment  Night          London    26-35                      0      0
                                                 36-45                      0      0
ATM Withdrawal          Business Hours Butwal    56+                        0      0
                                       Chitwan   18-25                      0      6
Water Bill Payment      Night          Sindhuli  36-45                      0      0

[32500 rows x 2 columns]
```

In [161…

```
top5 = combined.sort_values(by='sum', ascending=False).head(5)
top5
```

Out[161…

transaction_type	time_of_day	location	age_group	sum	count
Mobile Banking Transfer	**Night**	**Butwal**	**26-35**	13	18
		Syangja	**26-35**	13	14
Wallet Load - eSewa	**Night**	**Lalitpur**	**26-35**	12	18
ATM Withdrawal	**Night**	**Birgunj**	**36-45**	12	13
Mobile Banking Transfer	**Night**	**Dhangadhi**	**26-35**	10	12

In [162…

```
top5 = combined.head(5).reset_index()
top5
```

Out[162…

	transaction_type	time_of_day	location	age_group	sum	count
0	Mobile Banking Transfer	Night	Butwal	26-35	13	18
1	Mobile Banking Transfer	Night	Syangja	26-35	13	14
2	Wallet Load - eSewa	Night	Lalitpur	26-35	12	18
3	ATM Withdrawal	Night	Birgunj	36-45	12	13
4	Mobile Banking Transfer	Night	Dhangadhi	26-35	10	12

In [163…

```
# convert everything to string before concatenation
top5 = top5.astype({
    "transaction_type": "string",
    "time_of_day": "string",
    "location": "string",
    "age_group": "string"
})
```

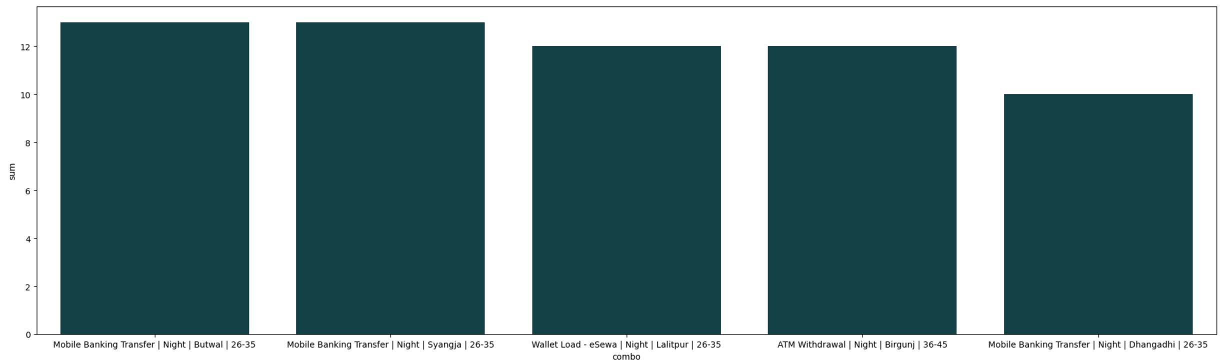
```python
top5["combo"] = (
    top5["transaction_type"] + " | " +
    top5["time_of_day"] + " | " +
    top5["location"] + " | " +
    top5["age_group"]
)
```

In [164...

```python
plt.figure(figsize=(25,7))
sns.barplot(data=top5, x="combo", y="sum", color= "#0f4a4f")
# plt.xticks(rotation=45)
```

Out[164...

```
<Axes: xlabel='combo', ylabel='sum'>
```



In [ ]:

```python

```

In [165...

```python
for col in dataset.columns:
    print("unique values of dataset[",col, "]: ", dataset[col].unique(), "\n")
```

```
 'Manual Verification']

unique values of dataset[ amount_deviation ]:  [1.49371800e+00 7.76204086e-01 1.2119
5564e+00 ... 7.58999995e+00
 7.10757000e-04 8.92150926e+00]

unique values of dataset[ is_suspicious ]:  [False  True]
```

In [166…

```python
#normalizing avg_monthly_income , amount, credit_score , avg_transaction_amount
from sklearn.preprocessing import MinMaxScaler
ms_avg_monthly_income = MinMaxScaler(feature_range=(0, 1))
ms_amount = MinMaxScaler(feature_range=(0, 1))
ms_credit_score = MinMaxScaler(feature_range=(0, 1))
ms_avg_transaction_amount = MinMaxScaler(feature_range=(0, 1))
ms_amount_deviation = MinMaxScaler(feature_range=(0, 1))
```

In [167…

```python
dataset["avg_monthly_income"] = ms_avg_monthly_income.fit_transform(dataset[["avg_m
dataset["amount"] = ms_amount.fit_transform(dataset[["amount"]])
dataset["credit_score"] = ms_credit_score.fit_transform(dataset[["credit_score"]])
dataset["avg_transaction_amount"] = ms_avg_transaction_amount.fit_transform(dataset
dataset["amount_deviation"] = ms_amount_deviation.fit_transform(dataset[["amount_de
```

In [ ]:

```python

```

In [168…

```python
#label encoding age_group, home_location ,account_type , mobile_banking_user ,  pri
# preferred_transaction_types , location , time_of_day , status , auth_method , is_
```

In [169…

```python
from sklearn.preprocessing import LabelEncoder
le_age_group = LabelEncoder()
le_home_location = LabelEncoder()
le_account_type = LabelEncoder()
le_mobile_banking_user = LabelEncoder()
le_primary_device = LabelEncoder()
le_primary_os = LabelEncoder()
le_primary_browser = LabelEncoder()
le_employment_status = LabelEncoder()
le_preferred_transaction_types = LabelEncoder()
le_location = LabelEncoder()
le_time_of_day = LabelEncoder()
le_status = LabelEncoder()
le_auth_method = LabelEncoder()
le_is_suspicious = LabelEncoder()
le_transaction_type = LabelEncoder()
```

In [170…

```python
dataset["age_group"] = le_age_group.fit_transform(dataset["age_group"])
dataset["home_location"] = le_home_location.fit_transform(dataset["home_location"])
dataset["account_type"] = le_account_type.fit_transform(dataset["account_type"])
dataset["mobile_banking_user"] = le_mobile_banking_user.fit_transform(dataset["mobi
dataset["primary_device"] = le_primary_device.fit_transform(dataset["primary_device
dataset["primary_os"] = le_primary_os.fit_transform(dataset["primary_os"])
dataset["primary_browser"] = le_primary_browser.fit_transform(dataset["primary_brow
dataset["employment_status"] = le_employment_status.fit_transform(dataset["employme
dataset["preferred_transaction_types"] = le_preferred_transaction_types.fit_transfo
dataset["location"] = le_location.fit_transform(dataset["location"])
```

```python
dataset["time_of_day"] = le_time_of_day.fit_transform(dataset["time_of_day"])
dataset["status"] = le_status.fit_transform(dataset["status"])
dataset["auth_method"] = le_auth_method.fit_transform(dataset["auth_method"])
dataset["is_suspicious"] = le_is_suspicious.fit_transform(dataset["is_suspicious"])
dataset["transaction_type"] = le_transaction_type.fit_transform(dataset["transactio
```

In [171…

```python
label_maps = {
    "age_group": dict(zip(le_age_group.classes_, le_age_group.transform(le_age_grou
    "home_location": dict(zip(le_home_location.classes_, le_home_location.transform
    "account_type": dict(zip(le_account_type.classes_, le_account_type.transform(le
    "mobile_banking_user": dict(zip(le_mobile_banking_user.classes_, le_mobile_bank
    "primary_device": dict(zip(le_primary_device.classes_, le_primary_device.transf
    "primary_os": dict(zip(le_primary_os.classes_, le_primary_os.transform(le_prima
    "primary_browser": dict(zip(le_primary_browser.classes_, le_primary_browser.tra
    "employment_status": dict(zip(le_employment_status.classes_, le_employment_stat
    "preferred_transaction_types": dict(zip(le_preferred_transaction_types.classes_
    "location": dict(zip(le_location.classes_, le_location.transform(le_location.cl
    "time_of_day": dict(zip(le_time_of_day.classes_, le_time_of_day.transform(le_ti
    "status": dict(zip(le_status.classes_, le_status.transform(le_status.classes_))
    "auth_method": dict(zip(le_auth_method.classes_, le_auth_method.transform(le_au
    "is_suspicious": dict(zip(le_is_suspicious.classes_, le_is_suspicious.transform
    "transaction_type": dict(zip(le_transaction_type.classes_, le_transaction_type.
}
label_maps
```

```
 'School Fee Payment': np.int64(20),
 'Wallet Load - IME Pay': np.int64(21),
 'Wallet Load - Khalti': np.int64(22),
 'Wallet Load - eSewa': np.int64(23),
 'Water Bill Payment': np.int64(24)}}
```

In [172…

```
dataset.head()
```

Out[172…

	customer_id	age_group	home_location	credit_score	account_age_years	account_type
0	1	1	21	0.745794	13	3
1	1	1	21	0.745794	13	3
2	1	1	21	0.745794	13	3
3	1	1	21	0.745794	13	3
4	1	1	21	0.745794	13	3

5 rows × 29 columns

In [173…

```
dataset.dtypes
```

Out[173…

```
customer_id                      int64
age_group                        int64
home_location                    int64
credit_score                   float64
account_age_years                int64
account_type                     int64
avg_monthly_income             float64
mobile_banking_user              int64
primary_device                   int64
primary_os                       int64
primary_browser                  int64
avg_transaction_amount         float64
transaction_frequency            int64
employment_status                int64
preferred_transaction_types      int64
international_activity            bool
risk_score                       int64
transaction_id                  object
transaction_date                object
transaction_type                 int64
amount                         float64
location                         int64
ip_address                      object
time_of_day                      int64
transaction_velocity             int64
status                           int64
auth_method                      int64
amount_deviation               float64
is_suspicious                    int64
dtype: object
```

In [174…

```
dataset['transaction_id'].nunique()
```

Out[174… 103492

In [175…
```
dataset['transaction_date'].nunique()
```
Out[175… 98133

In [176…
```
dataset.drop(columns=['ip_address', 'transaction_id', 'transaction_date'], inplace=
```
In [177…
```
#now for algorithm seperate the data
X = dataset.drop(columns=['is_suspicious'])
y = dataset['is_suspicious']
X.ndim
```
Out[177… 2

In [178…
```
from sklearn.model_selection import train_test_split
```
In [179…
```
# help(train_test_split)
```
In [180…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_st
```
In [181…
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
```
In [186…
```
rf = LinearRegression()
lg = LogisticRegression()
```
In [183…
```
rf.fit(X_train, y_train)
```
Out[183…

LinearRegression

▶ Parameters

In [185…
```
rf.score(X_test, y_test)
```
Out[185… 0.6133427224337857

In [ ]:
```

```
In [187…
```
lg.fit(X_train, y_train)
```

```
D:\Installations\Miniconda\envs\dsml\Lib\site-packages\sklearn\linear_model\_logisti
c.py:473: ConvergenceWarning: lbfgs failed to converge after 100 iteration(s) (statu
s=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=100).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[187…

```
▾ LogisticRegression
▶ Parameters
```

In [188…

```
lg.score(X_test, y_test)
```

Out[188…

```
0.9717171717171718
```

In [ ]:

```
```

In [190…

```
y_pred = lg.predict(X_test)
```

In [192…

```
# finding confusion matrix for this
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
confusion_logistic  = confusion_matrix(y_test, y_pred)
confusion_logistic
```

Out[192…

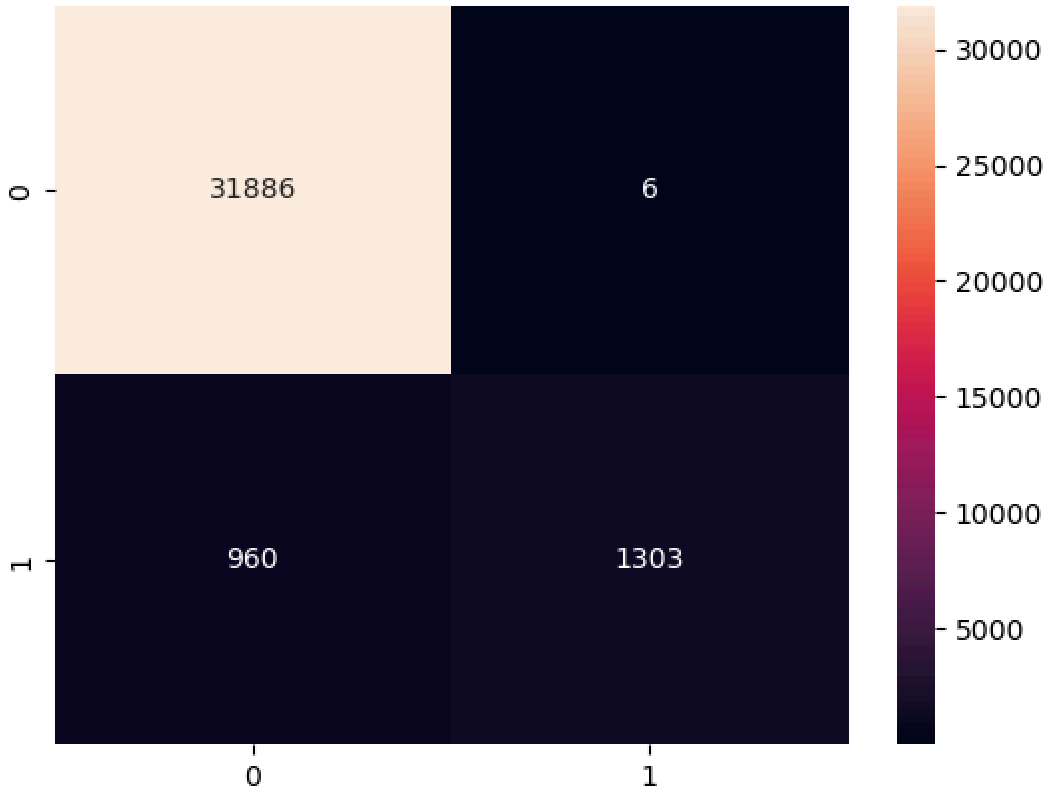
```
array([[31886,     6],
       [  960,  1303]])
```

In [193…

```
sns.heatmap(confusion_logistic, annot=True, fmt="d")
plt.plot()
```

Out[193…

```
[]
```



In [196…

```
# convert (run in a cell)
!jupyter nbconvert --to webpdf "D:/Github/Data-Science-And-Machine-Learning-Course/
```