Banking Data Analytics Report

1. Dataset Overview

Total Records: 3000

Total Columns: 25

2. Methodology

- Loaded CSV dataset into Jupyter Notebook using Pandas.
- Performed data inspection (.info(), .describe(), null checks).
- Handled missing values and corrected data types.
- Conducted Exploratory Data Analysis (EDA).
- Built interactive Power BI dashboard.
- Generated business insights and recommendations.

3. Exploratory Data Analysis (Queries & Results)

EDA Step 1 - Query:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

df = pd.read_csv('Banking.csv')
```

EDA Step 2 - Query:

```
df.head(5)
```

Result:

```
  Client ID             Name  Age  Location ID Joined Bank   Banking Contact  \
0  IND81288    Raymond Mills   24        34324  06-05-2019    Anthony Torres
1  IND65833    Julia Spencer   23        42205  10-12-2001  Jonathan Hawkins
2  IND47499   Stephen Murray   27         7314  25-01-2010     Anthony Berry
3  IND72498   Virginia Garza   40        34594  28-03-2019        Steve Diaz
4  IND60181  Melissa Sanders   46        41269  20-07-2012        Shawn Long

  Nationality            Occupation Fee Structure Loyalty Classification  ...  \
0    American  Safety Technician IV          High                   Jade  ...
1     African   Software Consultant          High                   Jade  ...
2    European    Help Desk Operator          High                   Gold  ...
3    American          Geologist II           Mid                 Silver  ...
```

```
4    American   Assistant Professor           Mid               Platinum  ...

   Bank Deposits  Checking Accounts  Saving Accounts  \
0     1485828.64          603617.88        607332.46   
1      641482.79          229521.37        344635.16   
2     1033401.59          652674.69        203054.35   
3     1048157.49         1048157.49        234685.02   
4      487782.53          446644.25        128351.45   

   Foreign Currency Account  Business Lending  Properties Owned  \
0                  12249.96        1134475.30                 1   
1                  61162.31        2000526.10
```

EDA Step 3 - Query:

```
df.shape
```

Result:

```
(3000, 25)
```

EDA Step 4 - Query:

```
df.info()
```

Result:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 25 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Client ID                 3000 non-null   object 
 1   Name                      3000 non-null   object 
 2   Age                       3000 non-null   int64  
 3   Location ID               3000 non-null   int64  
 4   Joined Bank               3000 non-null   object 
 5   Banking Contact           3000 non-null   object 
 6   Nationality               3000 non-null   object 
 7   Occupation                3000 non-null   object 
 8   Fee Structure             3000 non-null   object 
 9   Loyalty Classification    3000 non-null   object 
 10  Estimated Income          3000 non-null   float64
 11  Superannuation Savings    3000 non-null   float64
 12  Amount of Credit Cards    3000 non-null   int64  
 13  Credit Card Balance       3000 non-null   float64
 14  Bank Loans                3000 non-null   float64
 15  Bank Deposits             3000 non-null   float64
 16  Checking Accounts         3000 non-null   float64
 17  Saving Accounts           3000 non-null   float64
 18  Foreign Currency Account  3000 non-null   float64
 19  Business Lending          3000 non-null   float64
 20  Properties Owned          3000 non-null   int64  
 21  Risk Weighting            3000 non-null   int64  
 22  BRId                      3000 non-null   int64  
 23  GenderId
```

EDA Step 5 - Query:

```
#Generating descriptive staticss for the dataframe
df.describe()
```

Result:

```
               Age   Location ID  Estimated Income  Superannuation Savings  \
count  3000.000000   3000.000000       3000.000000             3000.000000
```

```
mean     51.039667  21563.323000     171305.034263            25531.599673
std      19.854760  12462.273017     111935.808209            16259.950770
min      17.000000     12.000000      15919.480000             1482.030000
25%      34.000000  10803.500000      82906.595000            12513.775000
50%      51.000000  21129.500000     142313.480000            22357.355000
75%      69.000000  32054.500000     242290.305000            35464.740000
max      85.000000  43369.000000     522330.260000            75963.900000

       Amount of Credit Cards  Credit Card Balance    Bank Loans  \
count             3000.000000          3000.000000  3.000000e+03
mean                 1.463667          3176.206943  5.913862e+05
std                  0.676387          2497.094709  4.575570e+05
min                  1.000000             1.170000  0.000000e+00
25%                  1.000000          1236.630000  2.396281e+05
50%                  1.000000          2560.805000  4.797934e+05
75%                  2.000000          4522.632500  8.258130e+05
max                  3.000000         13991.990000  2.667557e+06

       Bank Deposits  Checking Accounts  Saving Accounts  \
count   3.000000e+03       3.000000e+03     3.000000e+03
mean    6.715602e+05       3.210929e+05     2.329084e+05
std
```

EDA Step 6 - Query:

```
df.columns = df.columns.str.lower()
df.columns = df.columns.str.replace(' ','_')
```

EDA Step 7 - Query:

```
df.columns
```

Result:

```
Index(['client_id', 'name', 'age', 'location_id', 'joined_bank',
       'banking_contact', 'nationality', 'occupation', 'fee_structure',
       'loyalty_classification', 'estimated_income', 'superannuation_savings',
       'amount_of_credit_cards', 'credit_card_balance', 'bank_loans',
       'bank_deposits', 'checking_accounts', 'saving_accounts',
       'foreign_currency_account', 'business_lending', 'properties_owned',
       'risk_weighting', 'brid', 'genderid', 'iaid'],
      dtype='object')
```

EDA Step 8 - Query:

```
bins = [0,100000,300000,float('inf')]
labels = ['Low','Med','High']
df['income_band'] = pd.cut(df['estimated_income'],bins=bins,labels=labels,right=False)
```

EDA Step 9 - Query:

```
df['income_band'].value_counts().plot(kind='bar')
```

Result:

```
<Axes: xlabel='income_band'><Figure size 640x480 with 1 Axes>
```

EDA Step 10 - Query:

```
#Examine the distribution of unique categories in categorical columns
categorical_cols = df[["brid","genderid","iaid","amount_of_credit_cards","nationality","occupation","

for col in categorical_cols:
    print(f"value counts for '{col}':")
    display(df[col].value_counts())
```

Result:

```
value counts for 'brid':
brid
3    1352
1     660
2     495
4     493
Name: count, dtype: int64value counts for 'genderid':
genderid
2    1512
1    1488
Name: count, dtype: int64value counts for 'iaid':
iaid
1     177
2     177
3     177
4     177
8     177
9     176
13    176
12    176
10    176
11    176
14    176
15    176
6      89
5      89
7      89
16     88
17     88
18     88
19     88
20     88
21     88
22     88
Name: count, dtype: int64value counts for 'amount_of_credit_cards':
amount_of_credit_cards
1    1922
2     765
3     313
Name: count, dtype: int64value counts for 'nationality':
nationality
European      1309
Asian          754
American       507
Australian     254
African        176
Name: count, dtype: int64value counts for 'occupation':
occupation
Associate Professor             28
Structural Analysis Engineer    28
Recruiter                       25
Account Coordinator             24
Human Resources Manager         24
                                ..
Office Assistant IV              8
Automation Specialist I          7
Computer Systems Analyst I       6
Developer III                    5
Senior Sales Associate           4
Name: count, Length: 195, dtype: int64value counts for 'fee_structure':
fee_structure
High    1476
Mid      962
Low      562
Name: count, dtype: int64value counts for 'loyalty_classification':
loyalty_classification
Jade        1331
Silver       767
```

```
Gold         585
Platinum     317
Name: count, dtype: int64value counts for 'properti
```

EDA Step 11 - Query:

```
# Histplot of the value counts for different occupations

for col in categorical_cols:
    if col == "occupation":
        continue
    plt.figure(figsize=(8,4))
    sns.histplot(df[col])
    plt.title('Histogram of occupation count')
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.show()
```

Result:

```
<Figure size 800x400 with 1 Axes><Figure size 800x400 with 1 Axes><Figure size 800x400 with 1 Axes><F
```

EDA Step 12 - Query:

```
# Numerical Analysis

numerical_cols = ['estimated_income','superannuation_savings','credit_card_balance','bank_loans','ban

#Univariate analysis and visualization
plt.figure(figsize=(15,10))
for i,col in enumerate(numerical_cols):
    plt.subplot(4,3,i+1)
    sns.histplot(df[col],kde=True)
    plt.title(col)
plt.show()
```

Result:

```
<Figure size 1500x1000 with 9 Axes>
```

EDA Step 13 - Query:

```
#Heatmaps

correlation_matrix = df[numerical_cols].corr()

plt.figure(figsize=(12,12))
sns.heatmap(correlation_matrix,annot=True,cmap='crest',fmt=".2f")
plt.title("Correlation Matrix ")
plt.show()
```

Result:

```
<Figure size 1200x1200 with 2 Axes>
```

EDA Step 14 - Query:

```
#1. The strongest positive correlation occur among "bank deposit" with "checking accounts","saving ac
#account" indicating that customers who maintain high balances in one account type often hold substa
#accounts as well.
```

EDA Step 15 - Query:

```
from sqlalchemy import create_engine

username = "postgres"
password = "system"
host = "localhost"
port = "5432"
database = "bank_data"

engine = create_engine(f"postgresql+psycopg2://{username}:{password}@{host}:{port}/{database}")

table_name = "customer"
df.to_sql(table_name,engine,if_exists="replace",index=False)

print(f"Data successfully loaded into table '{table_name}' in database '{database}'.")
```

Result:

```
Data successfully loaded into table 'customer' in database 'bank_data'.
```

4. Power BI Dashboard Explanation

The Power BI dashboard presents key banking performance indicators in an interactive format. It includes KPI cards for total customers, account distribution, loan metrics, and revenue indicators.

The dashboard allows filtering by customer segment, product type, and financial category. Trend charts visualize performance over time, while bar and pie charts show segment contribution.

Key insights derived from the dashboard:
- Identification of high-value customer segments
- Loan and deposit distribution trends
- Revenue contribution by category
- Performance comparison across customer types

The dashboard supports strategic decision-making by providing clear visual summaries and drill-down capability.

5. Conclusion

This report demonstrates the complete analytics workflow from raw data processing and EDA to interactive dashboard insights. The findings provide actionable business intelligence for improving banking performance.