Department of Computer Science, University of York

DATA: Introduction to Data Science

Task 1: Domain Analysis (5 marks)

Given the business domain and the data overview presented (in the assessment paper), provide a brief description of

- the business problem and its significance to the relevant sector;
- the link between the business problem and the field of data science;
- the main areas of investigation; and
- potential ideas and solutions.

Word Limit: 300 words

Write your answer here (text cell(s) to be used, as appropriate)

The business problem within the financial sector revolves around optimizing customer service, risk assessment, and retention strategies. It's crucial due to the competitive landscape where personalized services and risk management are paramount. The link to data science is pivotal; leveraging client records, transaction histories, and demographic statistics allows for predictive modeling, risk profiling, and customer segmentation.

The main areas of investigation encompass customer behavior analysis, risk assessment, and retention strategy formulation. Understanding spending patterns, transaction frequencies, and account behaviors aids in predicting customer needs and identifying potential risks. Demographic data analysis helps in assessing regional financial stability and its impact on client behavior.

Potential solutions revolve around predictive modeling for risk assessment, segmentation to tailor services, and personalized retention strategies. Machine learning algorithms can predict potential risks based on transactional behavior, aiding in proactive risk management. Customer segmentation enables targeted offerings, enhancing satisfaction and retention. Additionally, leveraging demographic data allows for region-specific strategies, incentivizing loyalty and improving service in financially stable areas.

In essence, employing data science techniques offers insights into customer behavior, risk factors, and regional dynamics, enabling proactive decision-making. This aligns with the bank's goal of improving services and ensuring prudent risk management while fostering long-term customer relationships.

In [ ]:
```
### Write your answer here (code cell(s) to be used, as appropriate)
```

Task 2: Database Design (25 marks)

Having understood the business domain, present a conceptual design in the form of an entity-relationship (ER) model that would be helpful in creating a database for the bank.

The bank data currently exists in the form of a csv file called *BankRecords.csv*, provided on VLE (path given in page 5, assessment paper). This file has all the existing records. The table available in the csv file is unnormalised. The information about its different columns is given in Tables 1 and 2 (in the assessment paper).

Following the standard principles of database normalisation, normalise the given table (*BankRecords.csv*) to a database schema that has minimum redundancies. Then, using the designed schema, create an SQLite database.

Your answer should include the SQL statements needed to accomplish this step. Your submission should also include the created SQLite database file.

Your answer should clearly cover the following:

- Any assumptions you are making about the given scenario;
- The designated keys, existing relationships, and identified functional dependencies;
- The steps followed and justifications for the decisions made.

World Limit: 500 words. This limit applies only to the explanations. There is no limit on any associated code/SQL statements or figures.

Write your answer here (text cell(s) to be used, as appropriate)

According to the Top-Down approach of database normalisation process, the 1NF is already achieved as each cell of the database table contain only one value. Although it can be seen that there are lots of duplication of the same record throughout the columns and breaking down these records into their respective table results in the creation of tables residence, client, account, loan, order, transaction, lastly card.

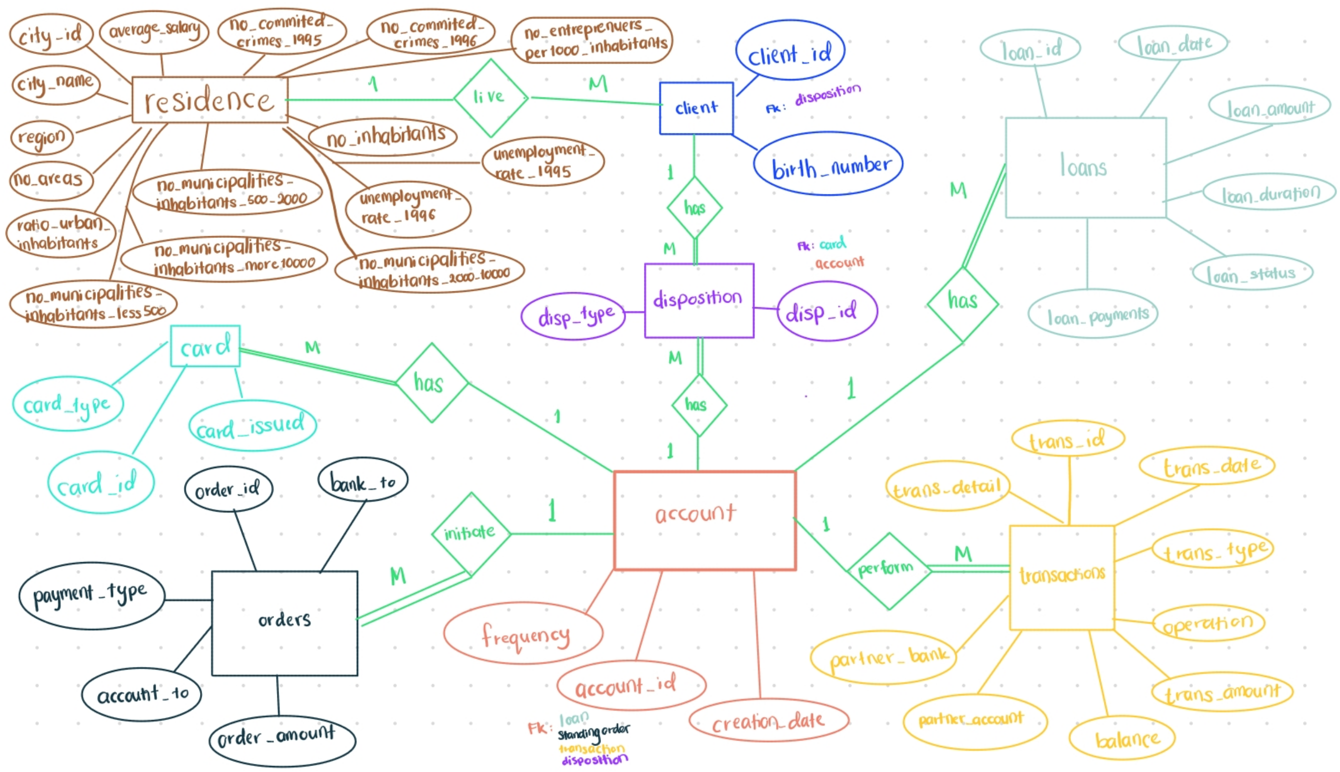
First and foremost, it can be seen that the records regarding the residence or demographic of the client is partially dependent on account_id and client_id, hence, the creation of tables residence, client, and account as shown in Code 1, 2, and 3. The residence table aids in the simplification and redundancy avoidance for the client table as a client can be from the same place. The attributes of the residence table have been discriptively renamed to avoid confusion; for instance, a1 to city_id which is the primary key that fully determines the table's other attributes.

It is stated that a client may have multiple accounts and an account may have an owner and/or a disponent owner which creates a many-to-many relationship. In this case, the relationship becomes complex and it is crucial to introduce a junction table to convert the existing relationship into one-to-many between the client and the account table. The table disponent is established to mitigate the issue by having a primary key disp_id and foreign keys consist of client_id, and account_id as the linker to the other tables. Another key attribute for this table is the disp_type which tells if a client is an owner or a disponent. Now all the tables has achieved 2NF and it is normalised as the primary keys disp_id, client_id, and account_id fully determine the tables' other attributes respectively which achieves full functional dependency.

Moving on to the non-primary key attributes for loan, order, transaction, and credit card, they are all partially dependent on account_id and their respective primary keys namely loan_id, order_id, transaction_id, and card_id. This therefore leads to the creation of dedicated tables for them such as loans table, orders table, transactions table, and card table as seen in Code 6, 7, 8, and 9 respectively. All of these tables each have a primary key which fully determines all the attributes in their respective table and achieve full functional dependency. On top of that, the tables loan, orders, and transactions have account_id as the foreign key to identify the account that facilitated in the financial operation. As for the card table, it has a disp_id as the foreign key to determine which account it belongs to and to which client.

Populating the completed and normalised database is shown in Code 10.

The entity-relationship model is as shown below.



In [1]:

```
### Write your answer here (code cell(s) to be used, as appropriate)

# Code 1: residence table creation
CREATE TABLE residence (
        city_id INT NOT NULL PRIMARY KEY,
        city_name VARCHAR,
        region VARCHAR,
        no_inhabitants INT,
        no_areas INT,
        average_salary INT,
        ratio_urban_inhabitants DECIMAL,
        unemployment_rate_1995 DECIMAL,
        unemployment_rate_1996 DECIMAL,
        no_entrepreneurs_per1000_inhabitants INT,
        no_commited_crimes_1995 INT,
        no_commited_crimes_1996 INT,
        no_municipalities_inhabitations_less500 INT,
        no_municipalities_inhabitations_500_2000 INT,
        no_municipalities_inhabitations_2000_10000 INT,
        no_municipalities_inhabitations_more10000 INT
);

# Code 2: client table creation
CREATE TABLE client (
        client_id INT NOT NULL PRIMARY KEY,
        birth_number varchar,
        city_id INT,
        FOREIGN KEY (city_id) REFERENCES residence(city_id)
);

# Code 3: account table creation
CREATE TABLE account (
```

```
        account_id INT NOT NULL PRIMARY KEY,
        disp_id INT,
        creation_date varchar,
        frequency varchar
);

# Code 4: disposition table creation
CREATE TABLE disposition (
        disp_id INT NOT NULL PRIMARY KEY,
        client_id INT NOT NULL,
        account_id INT NOT NULL,
        disp_type VARCHAR,
        FOREIGN KEY (client_id) REFERENCES client(client_id),
        FOREIGN KEY (account_id) REFERENCES account(account_id)
);

# Code 6: transactions table creation
CREATE TABLE transactions (
        trans_id INT NOT NULL PRIMARY KEY,
        account_id INT,
        trans_detail VARCHAR,
        trans_date varchar,
        trans_type varchar,
        trans_amount DECIMAL,
        operation varchar,
        balance DECIMAL,
        partner_account INT,
        partner_bank VARCHAR(2),
        FOREIGN KEY (account_id) REFERENCES account(account_id)
);

# Code 7: orders table creation
CREATE TABLE orders (
        order_id INT NOT NULL PRIMARY KEY,
        account_id INT,
        bank_to varchar(2),
        account_to INT,
        order_amount DECIMAL,
        payment_type varchar,
        FOREIGN KEY (account_id) REFERENCES account(account_id)
);

# Code 8: loans table creation
CREATE TABLE loans (
        loan_id INT PRIMARY KEY,
        account_id INT,
        loan_date varchar,
        loan_amount DECIMAL,
        loan_duration INT,
        loan_status varchar(1),
        loan_payments DECIMAL,
        FOREIGN KEY (account_id) REFERENCES account(account_id)
);

# Code 9: card table creation
CREATE TABLE card (
        card_id INT NOT NULL PRIMARY KEY,
        disp_id INT,
        card_issued varchar,
        card_type varchar,
        FOREIGN KEY (disp_id) REFERENCES disposition(disp_id)
);


# Code 10: data insertion
INSERT INTO residence (city_id, city_name, region, no_inhabitants,
```

```
no_municipalities_inhabitations_less500,
no_municipalities_inhabitations_500_2000,
no_municipalities_inhabitations_2000_10000,
no_municipalities_inhabitations_more10000,
no_areas,
ratio_urban_inhabitants,
average_salary,
unemployment_rate_1995,
unemployment_rate_1996,
no_entrepreneurs_per1000_inhabitants,
no_commited_crimes_1995,
no_commited_crimes_1996)
SELECT DISTINCT field31, field32, field33, field34, field35, field36, field37, field38,
FROM BankRecords
WHERE field1 NOT IN (SELECT field1 FROM BankRecords LIMIT 1);


INSERT INTO client (client_id, birth_number, city_id)
SELECT DISTINCT field25, field30, field31
FROM BankRecords
WHERE field1 NOT IN (SELECT field1 FROM BankRecords LIMIT 1);


INSERT INTO account (account_id, frequency, creation_date)
SELECT DISTINCT field1, field2, field3
FROM BankRecords
WHERE field1 NOT IN (SELECT field1 FROM BankRecords LIMIT 1) AND field1 IS NOT NULL;


INSERT INTO disposition (disp_id, client_id, account_id, disp_type)
SELECT DISTINCT field24, field25, field1, field26
FROM BankRecords
WHERE field1 NOT IN (SELECT field1 FROM BankRecords LIMIT 1) AND field24 IS NOT NULL;


INSERT INTO loans (account_id, loan_id, loan_date, loan_amount, loan_duration, loan_paym
SELECT DISTINCT field1, field4, field5, field6, field7, field8, field9
FROM BankRecords
WHERE field1 NOT IN (SELECT field1 FROM BankRecords LIMIT 1) AND field4 IS NOT NULL;


INSERT INTO orders (account_id, order_id, bank_to, account_to, order_amount, payment_typ
SELECT DISTINCT field1, field10, field11, field12, field13, field14
FROM BankRecords
WHERE field1 NOT IN (SELECT field1 FROM BankRecords LIMIT 1) AND field10 IS NOT NULL;


INSERT INTO transactions (account_id, trans_id, trans_date, trans_type, operation, trans_
SELECT DISTINCT field1, field15, field16, field17, field18, field19, field20, field21, f
FROM BankRecords
WHERE field1 NOT IN (SELECT field1 FROM BankRecords LIMIT 1) AND field15 IS NOT NULL;


INSERT INTO card (disp_id, card_id, card_type, card_issued)
SELECT DISTINCT field24, field27, field28, field29
FROM BankRecords
WHERE field1 NOT IN (SELECT field1 FROM BankRecords LIMIT 1) AND field27 IS NOT NULL;
```

```
  Cell In[1], line 4
    CREATE TABLE residence (
           ^
SyntaxError: invalid syntax
```

Task 3: Research Design (25 Marks)

Using the database designed in Task 2, design and implement **five** potential modelling solutions to achieve the aim of the Data Intelligence team. You need to provide clear justifications about the techniques selected in the context of the 'problem in hand'. Your design must consist of a combination of inferential statistics, supervised learning algorithms, and unsupervised learning algorithms, and include **at least one** of those techniques. Finally, your modelling solutions should be of sufficient complexity, combining information from multiple tables from the database built in Task 2, as appropriate. Your answer should clearly show the queries made to the database. If amendments are made to the database, the commands should be clearly included in your answer.

Your answer should clearly cover the following:

- Any assumptions you are making about the given scenario;
- Any data processing and data integrity steps you would undertake to make the data fit for purpose;
- Which technique(s) you would apply for each solution and why;
- An evaluation of the techniques applied in terms of the accuracy of their results (or any other suitable evaluation measure);
- Algorithmic parameters should be adequately stated and discussed;
- A discussion of ethical considerations arising from the solutions selected.

World Limit: 500 words. This limit applies only to the explanations. There is no limit on any associated code or figures.

Write your answer here (text cell(s) to be used, as appropriate)

In regards to the first modelling solution, it is assumed that the data in the provided database is accurate and reliable. Another assumption that was made is that the selected features namely demographic and financial are assumed to be relevant for clustering clients. In terms of data processing and data integrity steps that were undertaken to make the data fit for the purpose include handling missing values in the selected features as well as dealing with duplicated records before applying the cluster algorithm. Not only that, data stardardization was also done by standardizing the numerical feature to ensure that all variables contribute equally to the clustering process. Moving on to the technique justification, K-means clustering was utilized because it is an unsupervised learning algorithm that can identify inherent patterns or groups within the data. Simplicity and efficiency are another good traits of K-means as it is computationally efficient and relatively simple to implement, making it suitable for an initial exploration of the dataset. Furthermore, the clusters can be evaluated based on the interpretability of the segments formed, for instance, "Do the clusters make sense in terms of financial behavior and demographic characteristics?". As for the algorithmic parameters, the number of clusters is a crucial parameter as the choice of k may involve experimentation or domain knowledge. Going into the ethical considerations, the number one thing is about the privacy concerns in which by ensuring that the clustering results do not reveal sensitive information about individual clients. The bank must adhere to privacy regulations.

In [21]:
```
### Write your answer here (code cell(s) to be used, as appropriate)
import pandas as pd
import numpy as np
import sqlite3

# load the data through the db file
fileDB  = "BankRecords_Normalised.db"
```

In [22]:
```python
#Connect to DB
def create_connection(path):
    connection = None
    try:
        connection = sqlite3.connect(path)
        print("Connection to SQLite DB successful")
    except IOError as e:
        print(f"The error '{e}' occurred")
    return connection

connection = create_connection(fileDB)
```

```
Connection to SQLite DB successful
```

Modelling Solution 1: Unsupervised Learning (K-means Clustering with PCA)

Assuming data reflects accurate customer behaviors, it is important to preprocess it to handle outliers, normalize scales, and encode categorical variables like loan status. K-means clustering was chosen for its simplicity and efficacy in segmentation. The scatter plot using PCA was instrumental in visualizing cluster separation and cohesion, serving as a qualitative evaluation of the K-means algorithm. However, PCA simplifies multi-dimensional data to two dimensions, which can obscure true distances and relationships. Parameters like cluster number were chosen based on the elbow plot. Ethically, care was taken to avoid biases in segmentation and to ensure customer data confidentiality.

In [23]:
```python
#Load data from DB using Pandas read_sql_query()
loadQuery  = """
SELECT
    t.account_id,
        AVG(trans_amount) AS avg_transaction_amount,
        SUM(CASE WHEN trans_type = 'Withdrawal' THEN trans_amount ELSE 0 END) AS total_w
        SUM(CASE WHEN trans_type = 'Credit' THEN trans_amount ELSE 0 END) AS total_credi
        l.loan_amount,
        l.loan_duration
FROM
        transactions t
JOIN
        loans l ON t.account_id = l.account_id
GROUP BY t.account_id;
"""

dfDB = pd.read_sql_query(sql=loadQuery, con=connection)
dfDB.head()
```

Out[23]:

	account_id	avg_transaction_amount	total_withdrawal	total_credit	loan_amount	loan_duration
0	2	6593.052929	1336983.8	1597053.5	80952	24
1	19	5199.722442	750127.3	793194.6	30276	12
2	25	10797.609854	1378415.0	1494372.1	30276	12
3	37	7293.491538	409469.7	497029.2	318480	60
4	38	4399.613077	225865.4	308265.3	110736	48

In [24]:
```python
# Check for duplicate records
duplicatesNum = dfDB.duplicated().sum()
print("There are %d duplicate records"% (duplicatesNum))
# df.duplicated()

# Check for Null value attributes
```

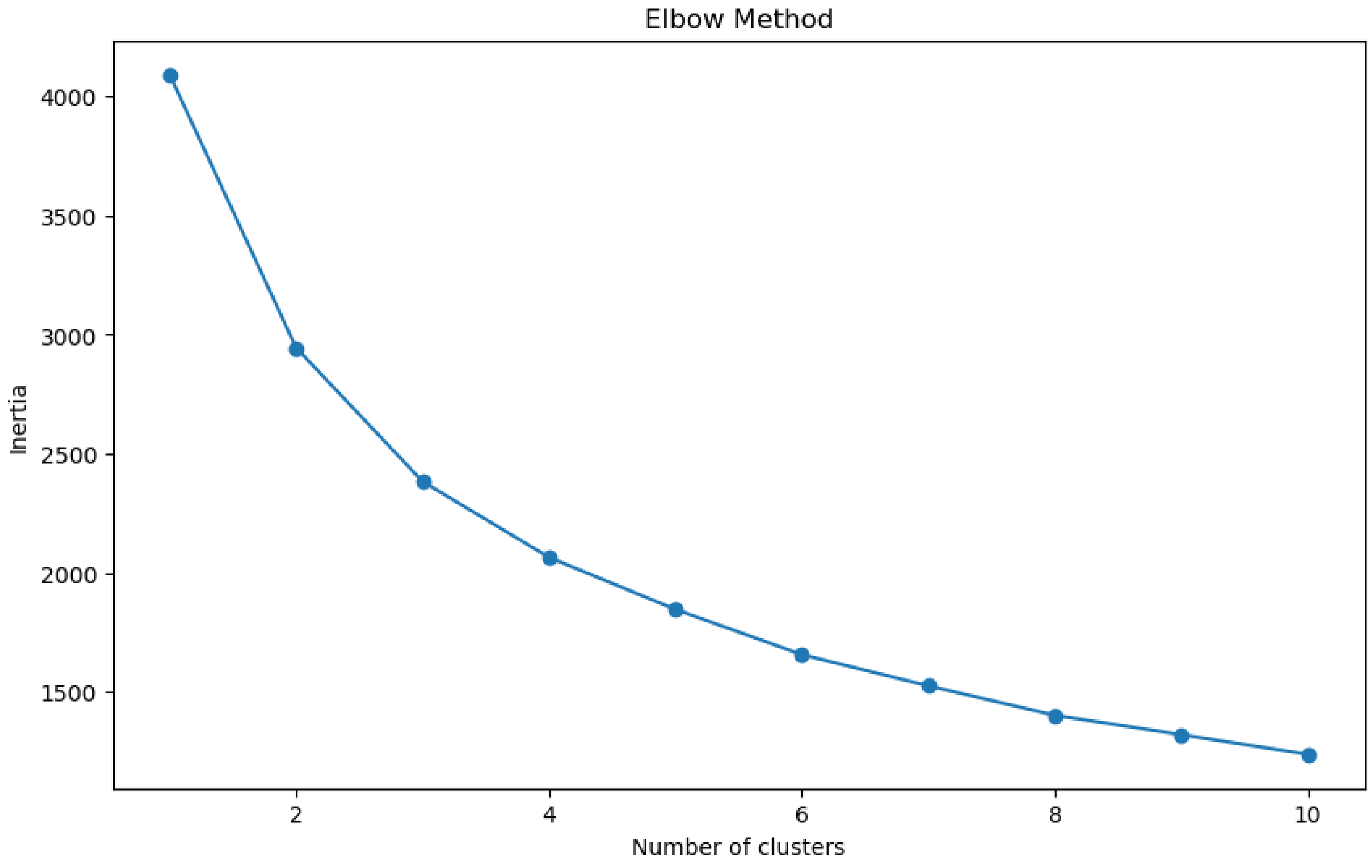
```
isNaNum = dfDB.isna().sum()
print(isNaNum)
```

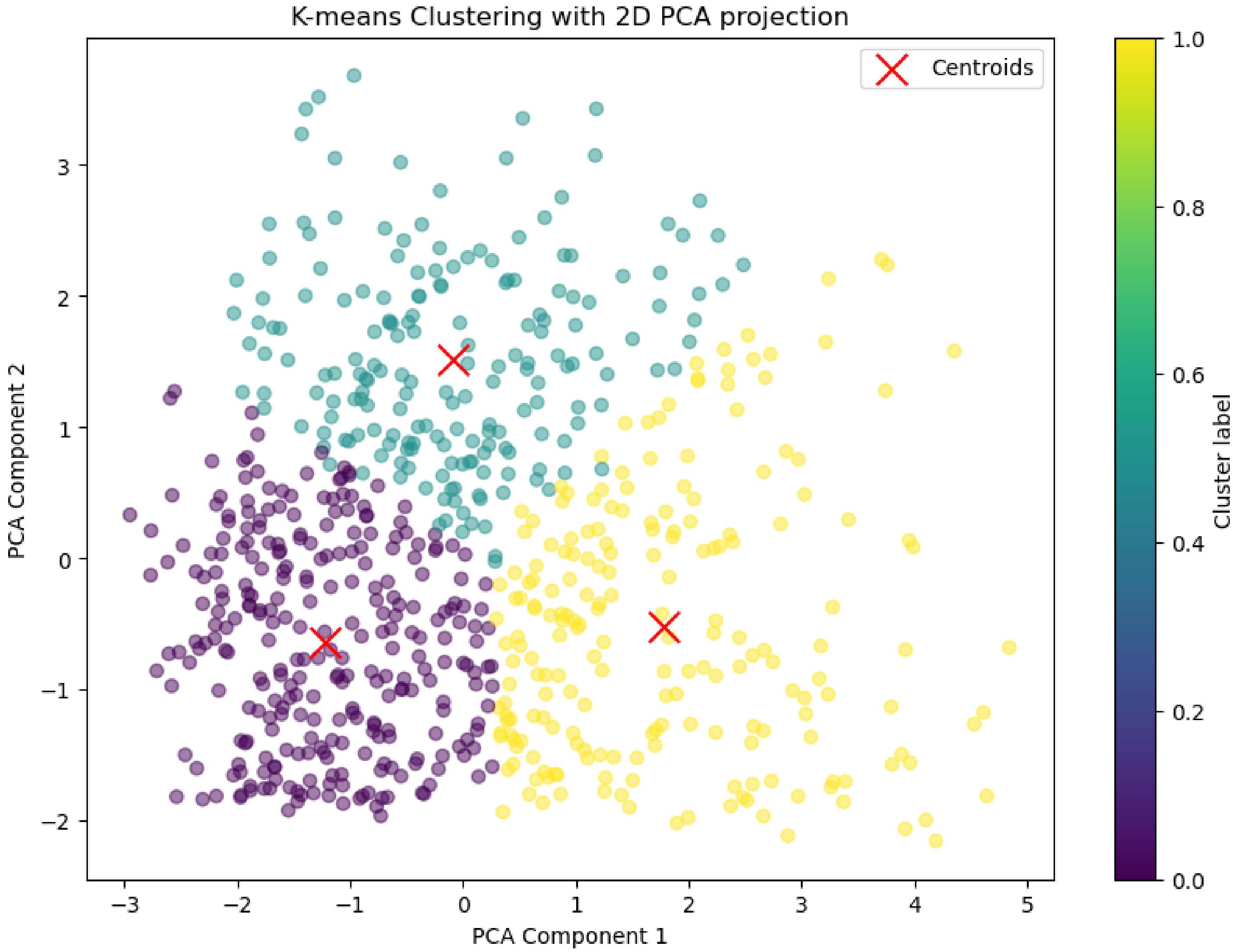
```
There are 0 duplicate records
account_id               0
avg_transaction_amount   0
total_withdrawal         0
total_credit             0
loan_amount              0
loan_duration            0
dtype: int64
```

In [25]:

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

# The attributes extracted are relevant because they provide a comprehensive view of cus
# Average transaction amount: Gives insight into spending habits
# Total withdrawals and credits: Reflect cash flow patterns
# Loan amount and duration: Indicate borrowing behavior and financial commitments
# Together these attributes paint a detailed picture of each customer’s financial profil
# which is critical for segmenting customers based on their banking activities, risk lev
# targeted product offerings. These financial indicators are core to understanding custo

# Standardize the features (important for K-means)
scaler = StandardScaler()
scaled_features = scaler.fit_transform(dfDB)

# Determine the optimal number of clusters using the Elbow Method
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

# Plotting the Elbow Method graph
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()

# Choose the appropriate number of clusters based on the elbow method
kmeans_optimal = KMeans(n_clusters=3, random_state=42)
kmeans_optimal.fit(scaled_features)

# Get the cluster labels and centroids
clusters = kmeans_optimal.labels_
centroids = kmeans_optimal.cluster_centers_

# Reduce the data to two components for visualization
pca = PCA(n_components=2)
reduced_features = pca.fit_transform(scaled_features)

# Project the centroids onto the same PCA space as the data
centroids_pca = pca.transform(centroids)

# Scatter plot of the two PCA components colored by cluster labels
plt.figure(figsize=(10, 7))
plt.scatter(reduced_features[:, 0], reduced_features[:, 1], c=clusters, cmap='viridis',
```

```
# Plot the centroids
plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1], marker='x', s=200, c='red', label=

# Labeling and visual layout
plt.title('K-means Clustering with 2D PCA projection')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.colorbar(label='Cluster label')
plt.legend()

plt.show()

# Keep in mind that PCA components are linear combinations of the original features,
# and they are chosen to represent as much of the data variation as possible.
# The axes of the plot ('PCA Component 1' and 'PCA Component 2') do not correspond to sp
# features but rather to these new composite features.
```





Modelling Solution 2: Supervised Learning (Decision Tree Classification)

As for the second modelling solution, in a banking scenario, predicting loan default is a common supervised learning problem where historical data can be used to train a model to make predictions on new clients. Decision Trees is chosen for its interpretability, which is crucial in a banking context where decisions need to be explained and justified which a well-performing model can assist in identifying high-risk clients. Algorithmic parameters include tree depth, feature selection criteria, and leaf node conditions. Proper tuning balances model complexity and predictive accuracy. Ensuring fairness and avoiding discriminatory features in model training are critical ethical considerations.

In [290…

```
#Load data from DB using Pandas read_sql_query()
loadQuery  = """
SELECT DISTINCT
    a.account_id,
    a.creation_date,
    a.frequency,
    r.average_salary,
    r.unemployment_rate_1995,
    r.no_commited_crimes_1995,
    l.loan_amount,
    l.loan_duration,
    l.loan_payments,
    l.loan_status
FROM
    account a
JOIN
    loans l ON a.account_id = l.account_id
```

```
JOIN
    disposition d ON a.account_id = d.account_id
JOIN
    client c ON d.client_id = c.client_id
JOIN
    residence r ON c.city_id = r.city_id
WHERE
    l.loan_date BETWEEN '950101' AND '951231';
"""

dfDB = pd.read_sql_query(sql=loadQuery, con=connection)
dfDB.head()
```

In [291…

```
# Check for duplicate records
duplicatesNum = dfDB.duplicated().sum()
# print(dfDB.describe)
print("There are %d duplicate records"% (duplicatesNum))
# df.duplicated()

# Check for Null value attributes
isNaNum = dfDB.isna().sum()
print(isNaNum)
```

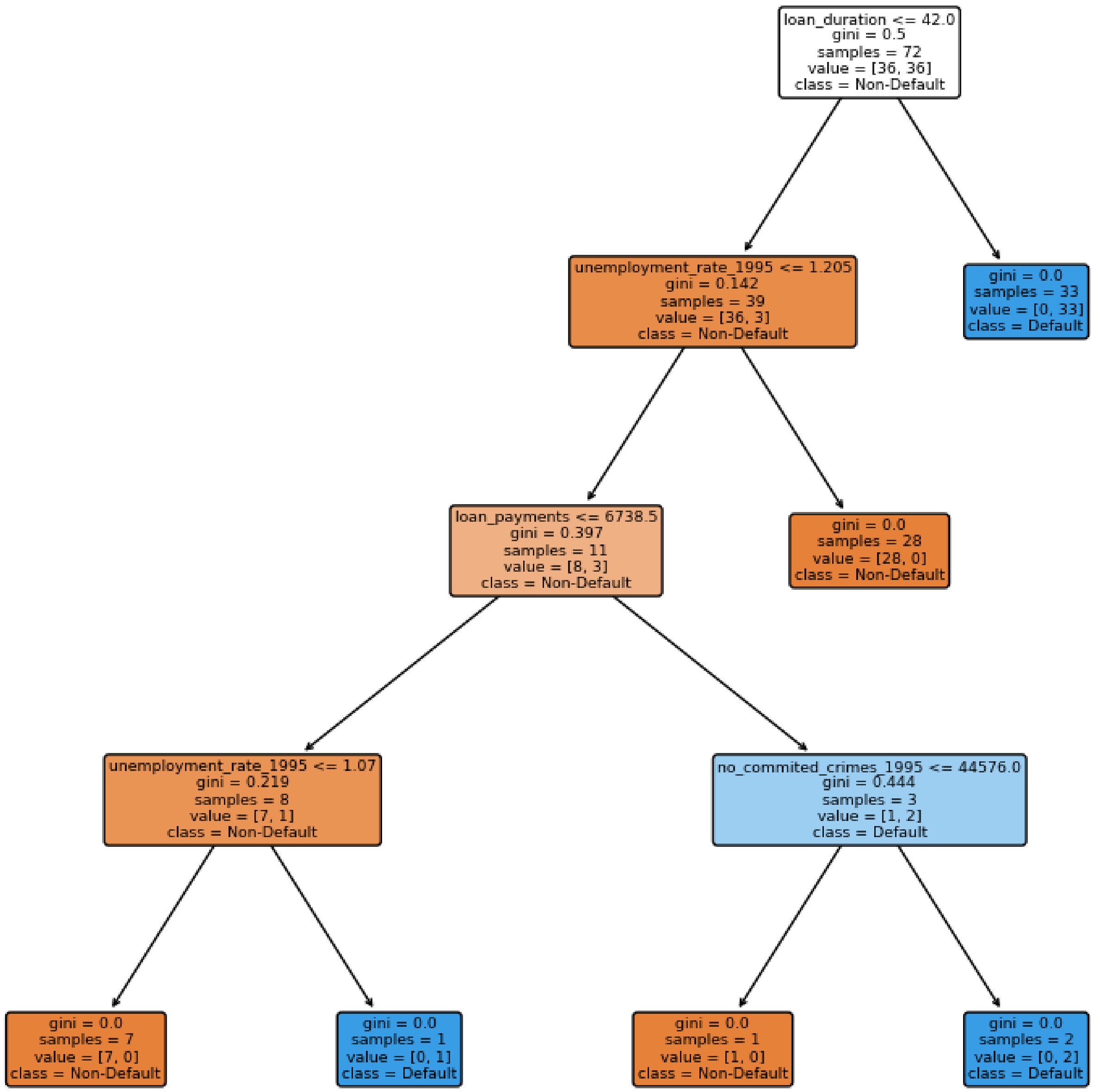
```
There are 0 duplicate records
account_id                 0
creation_date              0
frequency                  0
average_salary             0
unemployment_rate_1995     0
no_commited_crimes_1995    0
loan_amount                0
loan_duration              0
loan_payments              0
loan_status                0
dtype: int64
```

In [295…

```
# Using a classification algorithm to predict whether a client is likely to default on a

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt_solution2

# Selecting relevant features and target variable
# The features selected for this model are relevant because they directly relate to a cl
# Average Salary: Higher salaries typically indicate a greater ability to make loan paym
# Unemployment Rate: Reflects the job market stability. Higher unemployment rates can in
# Number of Committed Crimes: This can be an indirect indicator of socio-economic stabil
# Loan Amount: Larger loans represent a higher financial burden on the borrower, potenti
# Loan Duration: Longer loan terms can increase uncertainty about a borrower's future ab
# Loan Payments: The size of loan payments relative to the borrower's income can affect
# Each feature offers insight into the risk profile of a borrower, and when combined, th
features = ['average_salary', 'unemployment_rate_1995', 'no_commited_crimes_1995', 'loan
target = 'loan_status'

# Extracting selected features and target variable
X = dfDB[features]
y = dfDB[target]

# Converting 'loan_status' to binary (0 for non-default, 1 for default)
y = y.apply(lambda x: 0 if x == 'A' else 1)

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

```
# Building and training the Decision Tree model
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train, y_train)

# Predicting on the test set
y_pred = dt_classifier.predict(X_test)

# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Displaying classification report for more detailed evaluation
classification_report_solution2 = classification_report(y_test, y_pred)
print(classification_report_solution2)

# Visualizing the Decision Tree
plt_solution2.figure(figsize=(10, 10))
plot_tree(dt_classifier, filled=True, feature_names=features, class_names=['Non-Default'
plt_solution2.show()
```

```
Accuracy: 0.89
              precision    recall  f1-score   support

           0       0.82      1.00      0.90         9
           1       1.00      0.78      0.88         9

    accuracy                           0.89        18
   macro avg       0.91      0.89      0.89        18
weighted avg       0.91      0.89      0.89        18
```



Modelling Solution 3: Inferential Statistics

Moving on to the third modelling solution, Inferential Statistics is employed to assess whether observed differences in transaction amounts between clients with and without loan defaults are statistically significant, providing insights into financial behavior associated with loan risk. The t-test helps determine if the observed differences are likely due to chance, contributing to robust decision-making. Its accuracy is contingent on assumptions like normality and independence. In terms of algorithmic parameters, critical parameters include sample sizes, and degrees of freedom. Proper calculation and understanding of t and p values are crucial for valid statistical inferences. As for the ethical consideration, ensuring data privacy and avoiding biases in sample selection are paramount. Transparent communication about statistical interpretations and limitations is necessary for ethical decision-making.

In [12]:
```
#Load data from DB using Pandas read_sql_query()
loadQuery  = """
SELECT DISTINCT
    a.account_id,
    t.trans_date,
```

```
    t.trans_amount,
    l.loan_status
FROM
    transactions t
JOIN
    account a ON t.account_id = a.account_id
JOIN
    loans l ON a.account_id = l.account_id;
"""

dfDB = pd.read_sql_query(sql=loadQuery, con=connection)
dfDB.head()
```

Out[12]:

	account_id	trans_date	trans_amount	loan_status
0	2	930226	1100.0	A
1	2	930312	20236.0	A
2	2	930328	3700.0	A
3	2	930331	13.5	A
4	2	930412	20236.0	A

In [13]:

```
# Check for duplicate records
duplicatesNum = dfDB.duplicated().sum()
print("There are %d duplicate records"% (duplicatesNum))
# duplicates = dfDB[(dfDB.duplicated())]
# duplicates

# Check for Null value attributes
isNaNum = dfDB.isna().sum()
print(isNaNum)

# Drop irrelevant data
dfDB = dfDB.drop(columns=['account_id', 'trans_date'])
```

```
There are 0 duplicate records
account_id      0
trans_date      0
trans_amount    0
loan_status     0
dtype: int64
```

In [14]:

```
# Performing a hypothesis test to determine if there is a significant difference in the
# clients who default on loans (loan_status = 1) and those who do not (loan_status = 0).
# This could provide insights into whether certain financial behaviors are associated wi

# Null hypothesis and alternative hypothesis
# H0: There is no significant difference in the average transaction amounts between clie
# H1: There is a significant difference in the average transaction amounts between clien

from scipy.stats import ttest_ind, t
import numpy as np

# Converting 'loan_status' to binary (0 for non-default, 1 for default)
dfDB['loan_status'] = dfDB['loan_status'].apply(lambda x: 0 if x == 'A' else 1)

# Selecting relevant data for the test
# The data selected for this model, encompassing transaction amounts, and loan status ar
# Transaction Amounts: They provide direct insight into the spending and saving behavior
# Loan Status: This is a crucial variable for the bank, as it directly indicates whether
# By comparing transaction behaviors of clients with different loan statuses, the bank c
default_transactions = dfDB[dfDB['loan_status'] == 1]['trans_amount']
non_default_transactions = dfDB[dfDB['loan_status'] == 0]['trans_amount']
```

```
# Sample sizes
sample_size1 = len(default_transactions)
sample_size2 = len(non_default_transactions)

# Standard deviation for each group
# s1 = np.std(default_transactions)
# s2 = np.std(non_default_transactions)

# Degrees of freedom
dof = sample_size1 + sample_size2 - 2

# Performing a two-tailed critical t-value test at a 95% confidence level
alpha = 0.05
t_crit = t.ppf(alpha/2, dof)


# Performing a two-sample independent t-test
t_stat, p_value = ttest_ind(default_transactions, non_default_transactions, equal_var=Fa

# Displaying the values
print(f'Degrees of Freedom: {dof}')
print(f'Two-tailed Critical t-value: {t_crit}')
print(f'T-statistic: {t_stat:.2f}')
print(f'P-value: {p_value:.4f}')
```

```
Degrees of Freedom: 190851
Two-tailed Critical t-value: -1.95997641458244
T-statistic: -0.37
P-value: 0.7139
```

With significance level alpha=0.05, dof=190851 (two-tailed): t_crit = ±1.960

The t statistic result is 0.37 < 1.960

- We cannot reject H0
- There is statistically no significant difference in the average transaction amounts between clients who default on loans and those who do not.

In [15]:
```
# Drawing conclusions

# Comparison between absolute t value and critical t value
if np.abs(t_stat) > np.abs(t_crit):
    conclusion_t_value = f"Reject the null hypothesis. (|t| = {np.abs(t_stat):.4f} > Cri
else:
    conclusion_t_value = f"Fail to reject the null hypothesis. (|t| = {np.abs(t_stat):.4
print(conclusion_t_value)


# Comparison between p value and alpha significance level
alpha = 0.05
if p_value < alpha:
    conclusion_p_value = f"Reject the null hypothesis. (p-value = {p_value:.4f} < alpha
else:
    conclusion_p_value = f"Fail to reject the null hypothesis. (p-value = {p_value:.4f}
print(conclusion_p_value)
```

```
Fail to reject the null hypothesis. (|t| = 0.3666 ≤ Critical t = 1.9600)
Fail to reject the null hypothesis. (p-value = 0.7139 ≥ alpha = 0.05)
```

Modelling Solution 4: Supervised Learning (Decision Tree Classification)

On to the fourth modelling solution, Decision Trees is employed for classifying clients' eligible card type based on transaction history and loan information. The approach is interpretable, aiding in customer segmentation in terms of loyalty. It accommodates non-linear relationships, capturing nuanced patterns in financial behavior. Accuracy is used for model evaluation, gauging the correct predictions. However, given imbalanced classes, precision, recall, and F1-score is also considered for a more comprehensive assessment. As for the parameters, the default parameters is used yet adjusting max_depth to control model complexity so that a balance is struck to prevent overfitting while maintaining interpretability. Privacy concerns arise as model insights could reveal sensitive financial behaviors. Strict adherence to privacy regulations is crucial to prevent unintended disclosure of personal information.

In [219…

```
#Load data from DB using Pandas read_sql_query()
# loadQuery  = "SELECT t.trans_amount, t.trans_date, t.balance, d.client_id, cc.card_typ
loadQuery = """
SELECT
    a.account_id,
    COUNT(t.trans_id) AS total_trans,
    AVG(t.trans_amount) AS avg_trans_amount,
    AVG(t.balance) AS avg_balance,
    l.loan_amount,
    l.loan_payments,
    cc.card_type
    FROM transactions t
JOIN
    disposition d ON t.account_id = d.account_id
JOIN
    card cc ON d.disp_id = cc.disp_id
JOIN
    account a ON d.account_id = a.account_id
JOIN
    loans l ON d.account_id = l.account_id
GROUP
    BY a.account_id;
"""

dfDB = pd.read_sql_query(sql=loadQuery, con=connection)
dfDB.head()
```

Out[219]:

```
<bound method DataFrame.info of      account_id  total_trans  avg_trans_amount   avg_bal
ance  loan_amount  \
0            97          274       4486.337226  39814.610949       102876
1           105           59       8368.811864  30200.596610       352704
2           110          209       6839.223445  45365.278469       162576
3           132          198      12807.854040  52603.140404        88440
4           226          129       7233.487597  49698.993798       109344
..          ...          ...               ...           ...          ...
165       11079          246      16430.450407  73271.688211        98304
166       11138          325      16527.956923  73259.423692        89880
167       11141          152      11159.265132  61940.017763        44940
168       11186          339      12455.629499  67157.030973       392460
169       11359          378       7799.157143  36105.771958        54024

     loan_payments card_type
0             8573   classic
1             7348   classic
2             4516   classic
3             7370   classic
4             9112   classic
..             ...       ...
165           8192   classic
166           3745   classic
167           3745   classic
168           6541    junior
```

```
169            4502   classic

[170 rows x 7 columns]>
```

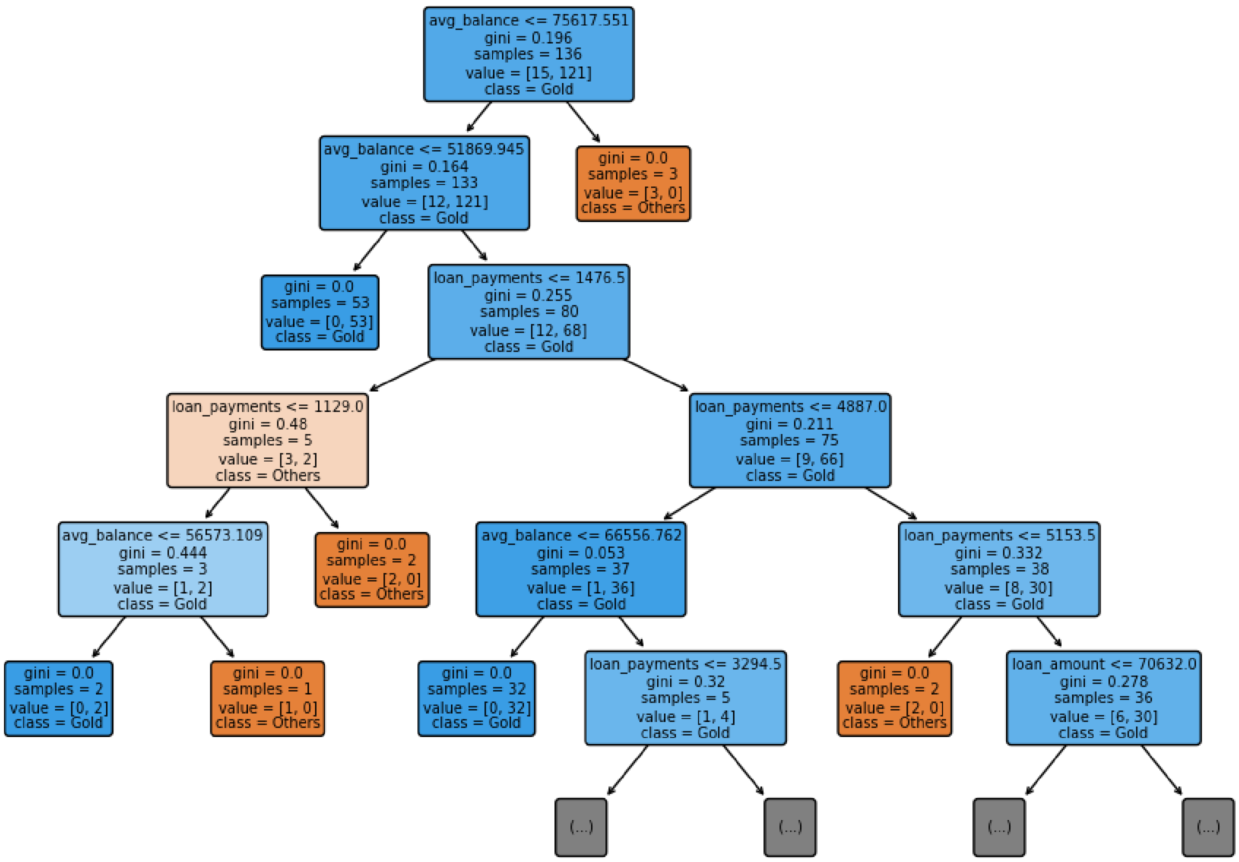
In [32]:
```
# Check for duplicate records
duplicatesNum = dfDB.duplicated().sum()
print("There are %d duplicate records"% (duplicatesNum))
# df.duplicated()

# Check for Null value attributes
isNaNum = dfDB.isna().sum()
print(isNaNum)
```

```
There are 0 duplicate records
account_id          0
total_trans         0
avg_trans_amount    0
avg_balance         0
loan_amount         0
loan_payments       0
card_type           0
dtype: int64
```

In [40]:
```
# Classify clients based on their transaction history and loan information to determine

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report, precision_score, reca
import pandas as pd
import matplotlib.pyplot as plt_solution4

# Selecting relevant features and target variable
# The features selectedare relevant because they provide a comprehensive view of a clien
# Total Transactions: Indicates client engagement with the bank. More transactions might
# Average Transaction Amount: Reflects the client's spending power and transaction habit
# Average Balance: Suggests the financial health and stability of the client. Higher bal
# Loan Amount: Larger loans might correlate with higher credit needs or financial trust
# Loan Payments: Regular and timely payments can indicate financial reliability and disc
# These features collectively help assess the creditworthiness and risk associated with
features = ['total_trans', 'avg_trans_amount', 'avg_balance', 'loan_amount', 'loan_payme
target = 'card_type'

# Extracting selected features and target variable
X = dfDB[features]
y = dfDB[target]

# Converting 'card_type' to binary (0 for gold, 1 for others)
y = y.apply(lambda x: 0 if x == 'gold' else 1)

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# Creating and training the Decision Tree model
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Making predictions on the test set
predictions = model.predict(X_test)

# Evaluating the model
accuracy = accuracy_score(y_test, predictions)
print(f'Accuracy: {accuracy:.2f}')

# Displaying classification report for more detailed evaluation
classification_report_solution4 = classification_report(y_test, predictions)
print('Classification Report:')
```

```
print(classification_report_solution4)

# Visualizing the Decision Tree
plt_solution4.figure(figsize=(12, 8))
plot_tree(model, max_depth=5, filled=True, feature_names=features, class_names=['Others'
plt_solution4.show()
```

```
Accuracy: 0.88
Classification Report:
              precision    recall  f1-score   support

           0       0.20      1.00      0.33         1
           1       1.00      0.88      0.94        33

    accuracy                           0.88        34
   macro avg       0.60      0.94      0.63        34
weighted avg       0.98      0.88      0.92        34
```



Modelling Solution 5: Linear Regression

For the last modelling solution, Linear Regression was chosen for its suitability in predicting continuous target variables like loan amounts based on various features. With the inclusion of R-squared value, it can capture the variance in loan amounts from customer transactions, and an MSE value can indicate potential prediction errors. Parameters like the 80-20 train-test split and feature selection are critical for model performance. Ethically, ensuring the model does not reinforce biases or unfair lending practices is paramount, necessitating regular audits for fairness and transparency in its predictive outcomes.

In [215…

```
#Load data from DB using Pandas read_sql_query()
loadQuery  = """
SELECT DISTINCT
    a.account_id,
    t.trans_amount,
```

```
        t.trans_date,
    CASE WHEN l.loan_amount IS NOT NULL THEN 1 ELSE 0 END AS has_loan,
    l.loan_amount,
    l.loan_payments,
    l.loan_duration
FROM
    transactions t
JOIN
    account a ON t.account_id = a.account_id
JOIN
    disposition d ON a.account_id = d.account_id
JOIN
    client c ON d.client_id = c.client_id
LEFT JOIN
    loans l ON a.account_id = l.account_id
WHERE
        l.loan_date BETWEEN '960101' AND '961231'
    AND t.trans_type = 'Withdrawal'
        AND t.trans_date BETWEEN '960101' AND '961231';
"""

dfDB = pd.read_sql_query(sql=loadQuery, con=connection)
dfDB.head()
```

Out[215]:

```
<bound method DataFrame.info of       account_id  trans_amount trans_date  has_loan  loa
n_amount  \
0             19        4800.0     960110         1        30276
1             19        2100.0     960112         1        30276
2             19       16300.0     960115         1        30276
3             19       14200.0     960130         1        30276
4             19          14.6     960131         1        30276
...          ...           ...        ...       ...          ...
5440       11362         330.0     961207         1       129408
5441       11362       10400.0     961207         1       129408
5442       11362          56.0     961208         1       129408
5443       11362        4780.0     961210         1       129408
5444       11362          14.6     961231         1       129408

      loan_payments  loan_duration
0              2523             12
1              2523             12
2              2523             12
3              2523             12
4              2523             12
...             ...            ...
5440           5392             24
5441           5392             24
5442           5392             24
5443           5392             24
5444           5392             24

[5445 rows x 7 columns]>
```

In [216…

```
# Check for duplicate records
duplicatesNum = dfDB.duplicated().sum()
print("There are %d duplicate records"% (duplicatesNum))
# duplicates = dfDB[(dfDB.duplicated())]
# duplicates

# Check for Null value attributes
print(dfDB.isna().sum())
print(dfDB.head())

# Handling missing values
dfDB.fillna(0, inplace=True)  # Replace NaN values with 0 for simplicity
print(dfDB.isna().sum())
```

```
There are 0 duplicate records
account_id        0
trans_amount      0
trans_date        0
has_loan          0
loan_amount       0
loan_payments     0
loan_duration     0
dtype: int64
   account_id  trans_amount trans_date  has_loan  loan_amount  loan_payments  \
0          19        4800.0     960110         1        30276           2523
1          19        2100.0     960112         1        30276           2523
2          19       16300.0     960115         1        30276           2523
3          19       14200.0     960130         1        30276           2523
4          19          14.6     960131         1        30276           2523

   loan_duration
0             12
1             12
2             12
3             12
4             12
account_id        0
trans_amount      0
trans_date        0
has_loan          0
loan_amount       0
loan_payments     0
loan_duration     0
dtype: int64
```

In [217…

```
# Convert 'trans_date' to datetime format
dfDB['trans_date'] = pd.to_datetime(dfDB['trans_date'], format='%y%m%d')

# Extract month and year from 'trans_date'
dfDB['month_year'] = dfDB['trans_date'].dt.to_period('M')

# Calculate the average transaction per month
average_transactions_per_month = dfDB.groupby(['account_id', 'month_year'])['trans_amoun
average_transactions_per_month.rename(columns={'trans_amount': 'avg_trans_amount'}, inpl

# Merge the original DataFrame with the calculated averages
newDF = pd.merge(dfDB, average_transactions_per_month, on=['account_id', 'month_year'],

# Remove unwanted data
newDF = newDF.drop(['trans_amount', 'trans_date'], axis=1).groupby(['account_id', 'month

print(newDF)
```

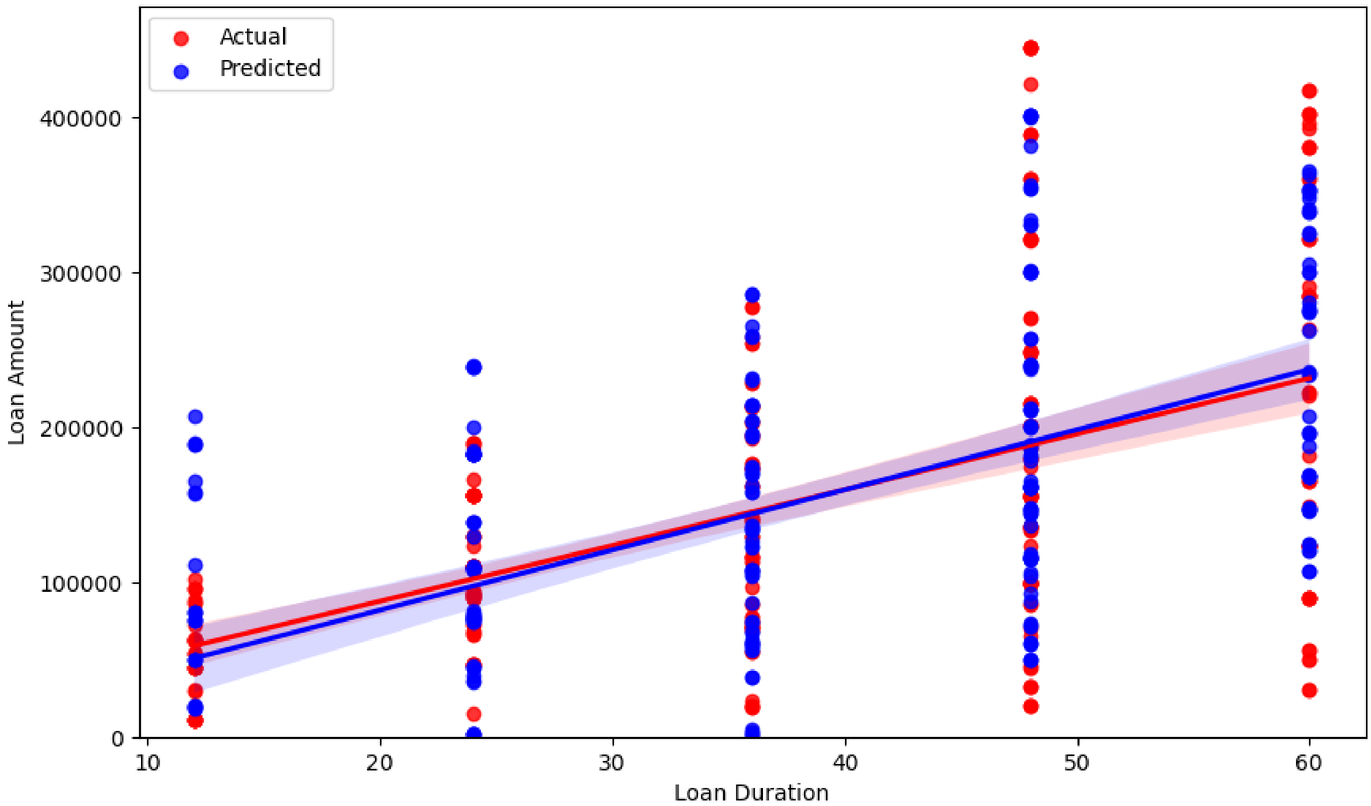
```
      account_id month_year  avg_trans_amount  has_loan  loan_amount  \
0             19    1996-01       7482.920000         1        30276
1             19    1996-02        907.300000         1        30276
2             19    1996-03       5328.650000         1        30276
3             19    1996-04       4554.125000         1        30276
4             19    1996-05       3678.650000         1        30276
...          ...        ...               ...       ...          ...
1267       11362    1996-08       2256.200000         1       129408
1268       11362    1996-09       2158.514286         1       129408
1269       11362    1996-10       1611.200000         1       129408
1270       11362    1996-11       1929.942857         1       129408
1271       11362    1996-12       2758.700000         1       129408

      loan_payments  loan_duration
0              2523             12
1              2523             12
```

```
2               2523            12
3               2523            12
4               2523            12
...              ...           ...
1267            5392            24
1268            5392            24
1269            5392            24
1270            5392            24
1271            5392            24

[1272 rows x 7 columns]
```

In [218…

```python
# A scenario where you want to predict a client's loan amount based on their monthly tra

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt_solution5
import seaborn as sns

# Selecting features and target variable
# The features selected for the model are financially significant:
# Loan Duration: It indicates the term of the loan, which is directly related to the ris
# Average Transaction Amount: This reflects the customer's financial activity and turnov
# Loan Payments: The regularity and amount of loan payments can provide insights into a
# These features are indicative of a customer's financial behavior, creditworthiness, an
features = ['loan_duration', 'avg_trans_amount', 'loan_payments']
target = 'loan_amount'

X = newDF[features]
y = newDF[target]

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# Creating and training the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Making predictions on the test set
predictions = model.predict(X_test)

# Evaluating the model
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

print(f'Mean Squared Error: {mse:.2f}')
print(f'R-squared: {r2:.2f}')

# Visualizing the results using Seaborn regplot
plt_solution5.figure(figsize=(10, 6))
sns.regplot(x=X_test['loan_duration'], y=y_test, color='red', label='Actual')
sns.regplot(x=X_test['loan_duration'], y=predictions, color='blue', label='Predicted')

plt_solution5.xlabel('Loan Duration')
plt_solution5.ylim(0)
plt_solution5.ylabel('Loan Amount')
plt_solution5.legend()
plt_solution5.show()
```

```
Mean Squared Error: 1150273629.55
R-squared: 0.90
```



Task 4: Experimental Results and Analysis (25 Marks)

Given the **five** modelling solutions implemented above, analyse, discuss and present your findings to the key stakeholders of the bank.

Your answer should clearly cover the following:

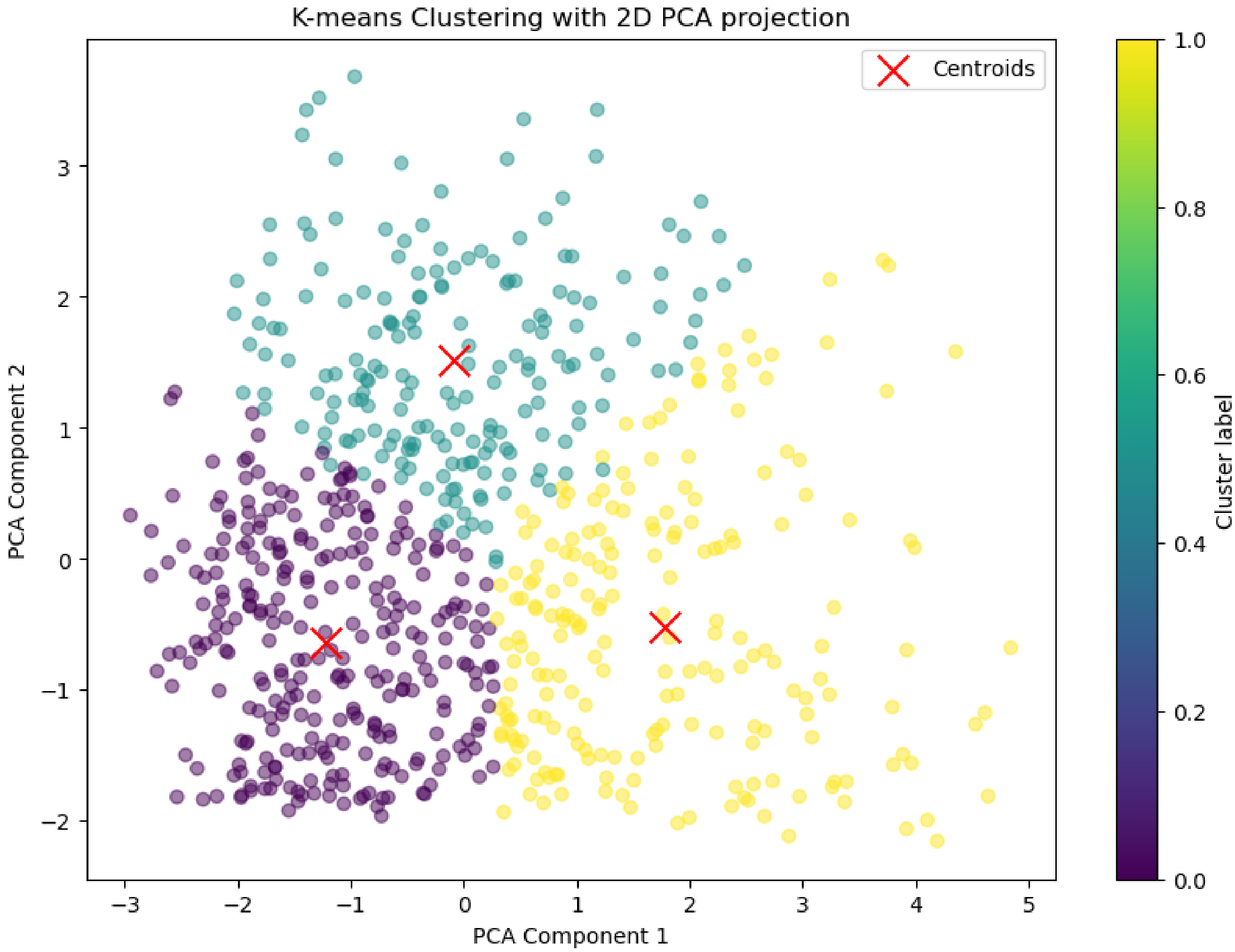
- Present your findings in a clear and concise manner;
- Discuss your results in the context of the selected solution;
- Discuss how these results can help the bank in performing customer risk assessment and establishing customer retention strategies;
- Present the limitations (if any) of your solutions in a clear and concise manner.

World Limit: 500 words. This limit applies only to the explanations. There is no limit on any associated code or figures.

Write your answer here (text cell(s) to be used, as appropriate)

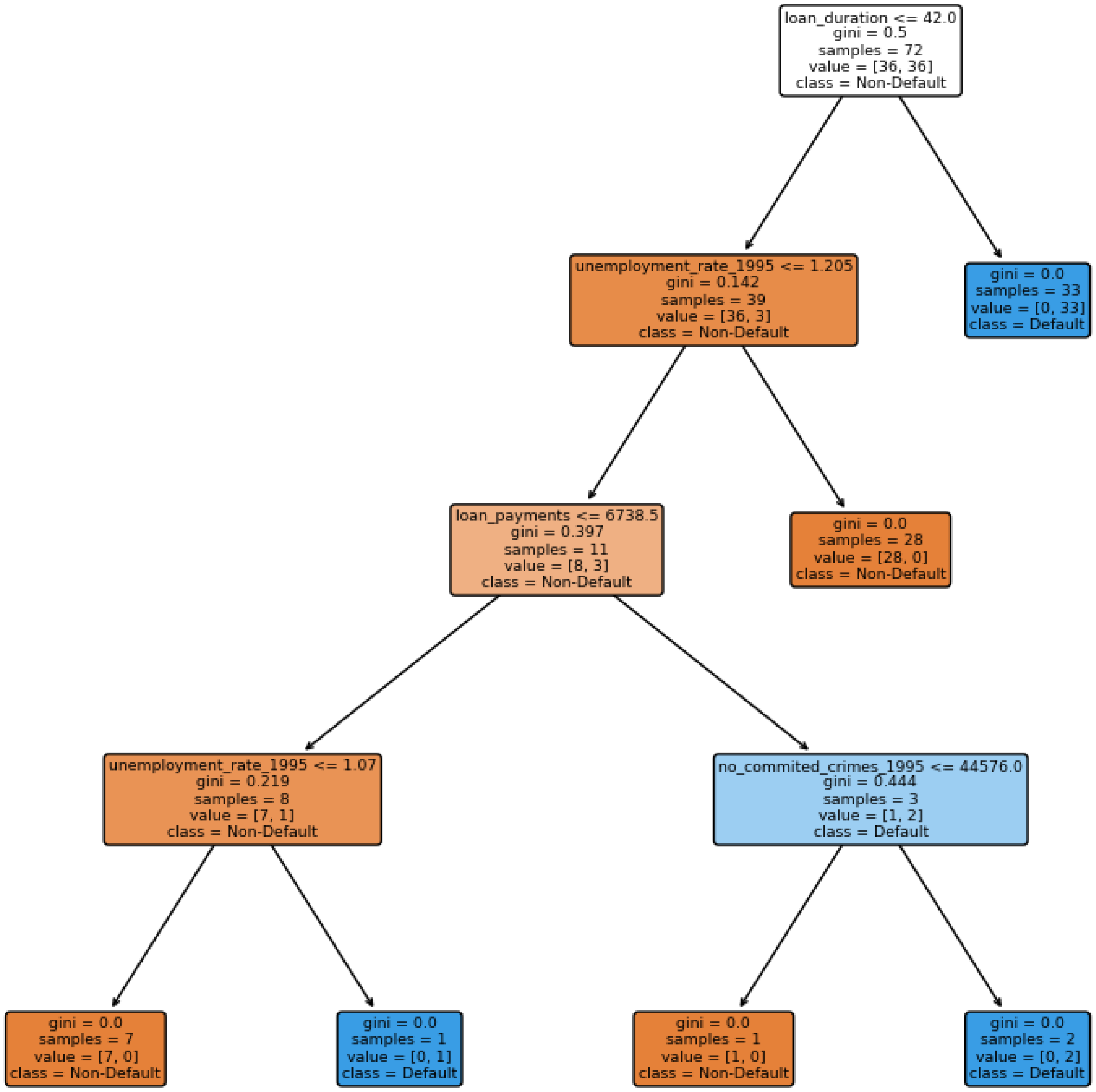
Modelling Solution 1: Unsupervised Learning (K-mean Clustering wtih PCA)

The K-means clustering analysis segmented the bank's customers into three groups, each with distinct financial behaviors suggesting varying levels of risk and opportunity for tailored banking services. Cluster 0 (Purple) indicates low-risk customers ideal for loyalty programs, while Cluster 1 (Yellow) may require closer risk monitoring and Cluster 2 (Cyan) needs personalized services for retention. The model's limitations include potential information loss from dimensionality reduction and assumptions of cluster shapes. These insights are pivotal for developing targeted risk management and customer retention strategies but should be applied flexibly to accommodate individual customer profiles.



Modelling Solution 2: Supervised Learning (Decision Tree Classification)

According to the Decision Tree model, it accurately predicts loan defaults with an 89% success rate, offering a clear view of potential risks in lending. Precision for default predictions is perfect, meaning no false alarms on defaults, while recall shows some defaults may go unnoticed. This model helps prioritize risk assessment efforts, though it is based on older data (1995) and may not reflect current trends. Limitations include possible overfitting and not accounting for all influencing factors. Despite these, the model is a valuable tool for enhancing the bank's decision-making process regarding loan approvals.



Modelling Solution 3: Inferential Statistics

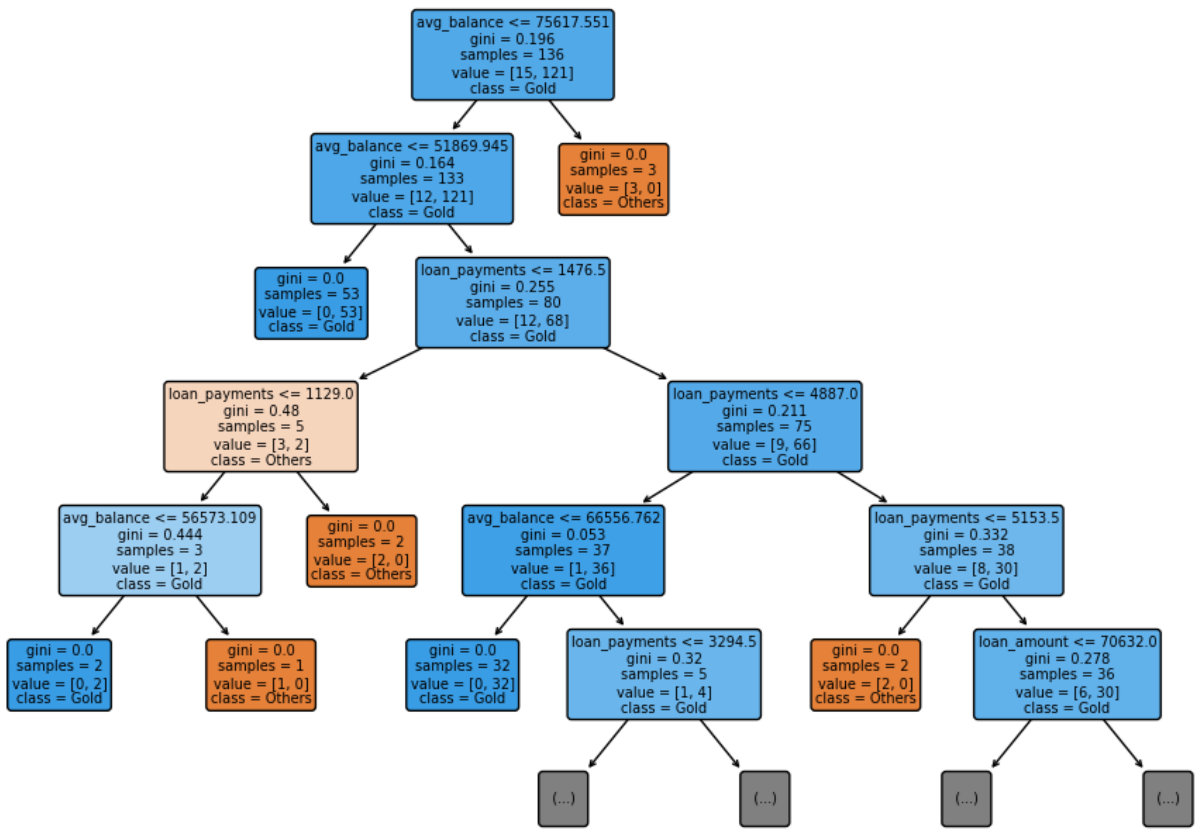
The analysis using a two-sample t-test revealed no significant difference in average transaction amounts between clients who default on loans (loan_status = 1) and those who don't (loan_status = 0), with a t-statistic of -0.37 and a high p-value of 0.7139. This suggests that transaction amounts alone may not be a reliable indicator for predicting loan defaults. For the bank, this means transaction behavior should be combined with other factors for more effective risk assessment. However, the study's limitation lies in its focus on only transaction amounts, potentially overlooking other relevant financial behaviors or customer attributes.

Null and alternative hypothesis are as follows, H0: There is no significant difference in the average transaction amounts between clients who default on loans and those who do not, while H1: There is a significant difference in the average transaction amounts between clients who default on loans and those who do not.

With significance level alpha=0.05, dof=190851 (two-tailed): t_crit = ±1.960, the t statistic result is 0.37 < 1.960 which says that we cannot reject H0 and therefore, there is statistically no significant difference in the average transaction amounts between clients who default on loans and those who do not.

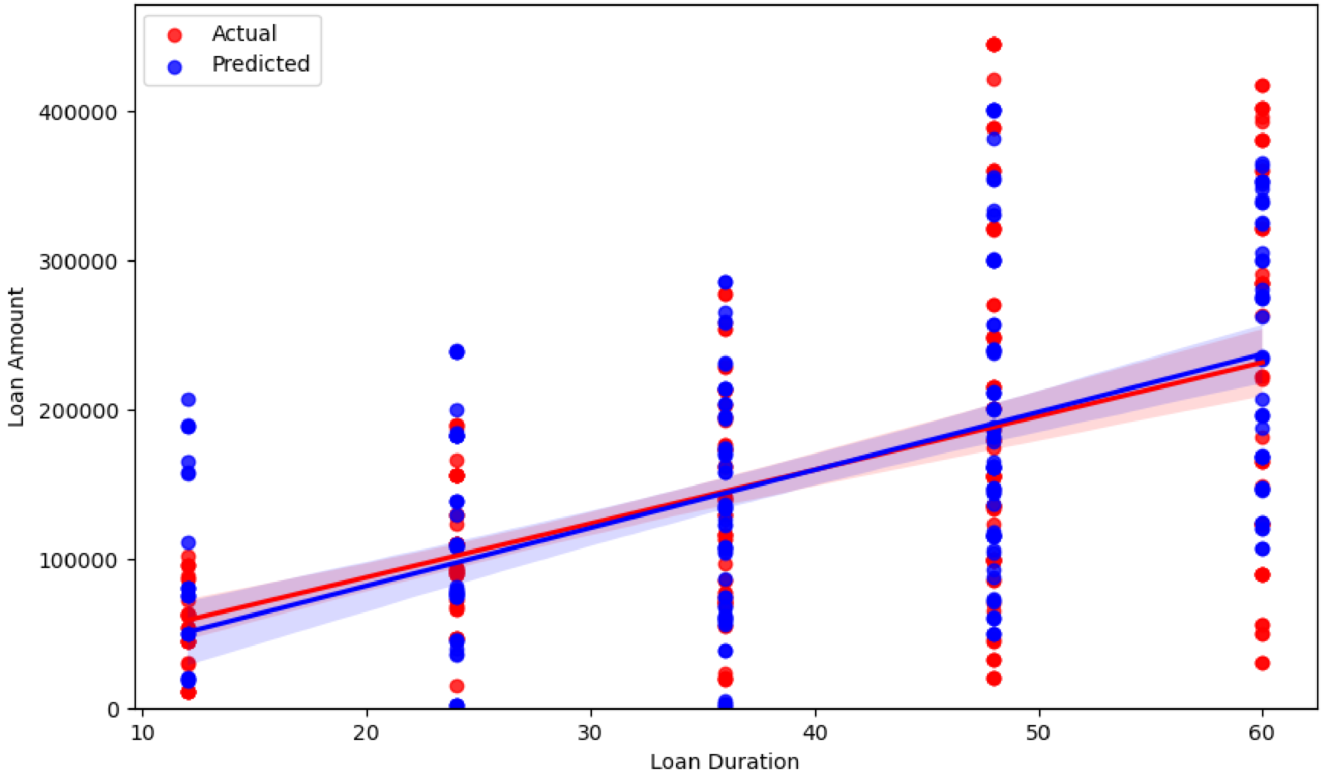
Modelling Solution 4: Supervised Learning (Decision Tree Classification)

The Decision Tree model classifies clients for credit card type with 88% accuracy, significantly identifying 'Others' with perfect precision. It indicates that 'average balance' and 'loan payments' are pivotal for classification. For customer retention scheme to reward high-value and loyal customers, the model suggests clients with lower loan payments and higher balances may be eligible for 'Gold' cards. However, the model's predictive power for 'Gold' is limited by the imbalanced data, raising concerns about its reliability for this class. Future models should address this imbalance to enhance predictive confidence.



Modelling Solution 5: Linear Regression

The linear regression model demonstrates a strong ability to predict loan amounts based on customer transaction data, with an R-squared value of 0.90, indicating a high level of variance explained by the model. This is useful for assessing customer risk by estimating potential loan sizes more accurately. However, the high MSE of 1,150,273,629.55 points to discrepancies in individual predictions, particularly for larger loans. This suggests while the model is generally reliable, it may require refinement for precision in high-value loan assessments, ensuring the bank can mitigate risk effectively.



In [26]:
```
### Write your answer here (code cell(s) to be used, as appropriate)

# More on Modelling Solution 1
# Cluster 0 (Purple): Characterized by a dense grouping, suggesting homogeneity in behav
# This could represent a segment with consistent spending and saving patterns, possibly

# Cluster 1 (Yellow): Moderate variability and positioned towards higher values on PCA C
# This may correlate with customers having higher loan amounts or more extended loan dur
# implying a segment potentially more profitable but possibly at higher risk.

# Cluster 2 (Cyan): Balanced features with slight overlap with Cluster 1, potentially in
# group or customers with a mix of behaviors from the other two segments.

# Centroids are marked by red 'X's, showing the average location of accounts within each
```

Task 5: Conclusion (10 Marks)

Given the insights derived from Tasks 1-4, provide a conclusion that clearly covers the following:

- A summary of the main points;
- A discussion of the significance of your results;
- Any recommendation(s) resulting from your analysis;
- Any overall ethical considerations arising from the data analysis of this business domain.

World Limit: 300 words.

Write your answer here (text cell(s) to be used, as appropriate)

In [ ]:
```
### Write your answer here (code cell(s) to be used, as appropriate)
```

Overall Academic Quality (10 Marks)

10 marks are allocated for the clarity and cohesiveness of your answers (both text and code) across all tasks with appropriate, relevant and effective analysis and presentation of the results.

Deliverables

You should submit the following to the submission point on the teaching portal:

1. the SQLite database produced in Task 2;
2. the completed Jupyter notebook (both .ipynb and HTML files) that also includes the SQL statements (Task 2), the research design and its implementation (Task 3), and the analysis and presentation of your results (Task 4);
3. any figures or diagrams that are included in your answers in the Jupyter notebook.

For each task where text is required, we have provided guidelines above on the suggested word counts. Exceeding the word count will result in any work beyond the word count being disregarded when assessing.