```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'german-bank-credit-data:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F4152127%2F7183093%2Fbundle%2Farchiv

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading german-bank-credit-data, 1412091 bytes compressed
[==================================================] 1412091 bytes downloaded
```

```
Downloaded and uncompressed: german-bank-credit-data
Data source import complete.
```

# German Bank Customer Segmentation

## Importing Libraries

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

warnings.filterwarnings('ignore')
```

```
german_bank = pd.read_csv('/kaggle/input/german-bank-credit-data/German Bank Segmentation./german_credit_data.csv')
german_bank.head()
```

| | Unnamed: 0 | Age | Sex | Job | Housing | Saving accounts | Checking account | Credit amount | Duration | Purpose |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 67 | male | 2 | own | NaN | little | 1169 | 6 | radio/TV |
| **1** | 1 | 22 | female | 2 | own | little | moderate | 5951 | 48 | radio/TV |
| **2** | 2 | 49 | male | 1 | own | little | NaN | 2096 | 12 | education |
| **3** | 3 | 45 | male | 2 | free | little | little | 7882 | 42 | furniture/equipment |
| **4** | 4 | 53 | male | 2 | free | little | little | 4870 | 24 | car |

Next steps: Generate code with german_bank View recommended plots

```
german_bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Unnamed: 0        1000 non-null   int64 
 1   Age               1000 non-null   int64 
 2   Sex               1000 non-null   object
 3   Job               1000 non-null   int64 
 4   Housing           1000 non-null   object
 5   Saving accounts   817 non-null    object
 6   Checking account  606 non-null    object
 7   Credit amount     1000 non-null   int64 
 8   Duration          1000 non-null   int64 
 9   Purpose           1000 non-null   object
dtypes: int64(5), object(5)
memory usage: 78.2+ KB
```

```
german_bank.describe()
```

| | Unnamed: 0 | Age | Job | Credit amount | Duration |
|---|---|---|---|---|---|
| **count** | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 |
| **mean** | 499.500000 | 35.546000 | 1.904000 | 3271.258000 | 20.903000 |
| **std** | 288.819436 | 11.375469 | 0.653614 | 2822.736876 | 12.058814 |
| **min** | 0.000000 | 19.000000 | 0.000000 | 250.000000 | 4.000000 |
| **25%** | 249.750000 | 27.000000 | 2.000000 | 1365.500000 | 12.000000 |
| **50%** | 499.500000 | 33.000000 | 2.000000 | 2319.500000 | 18.000000 |
| **75%** | 749.250000 | 42.000000 | 2.000000 | 3972.250000 | 24.000000 |
| **max** | 999.000000 | 75.000000 | 3.000000 | 18424.000000 | 72.000000 |

## Exploratory Data Analysis

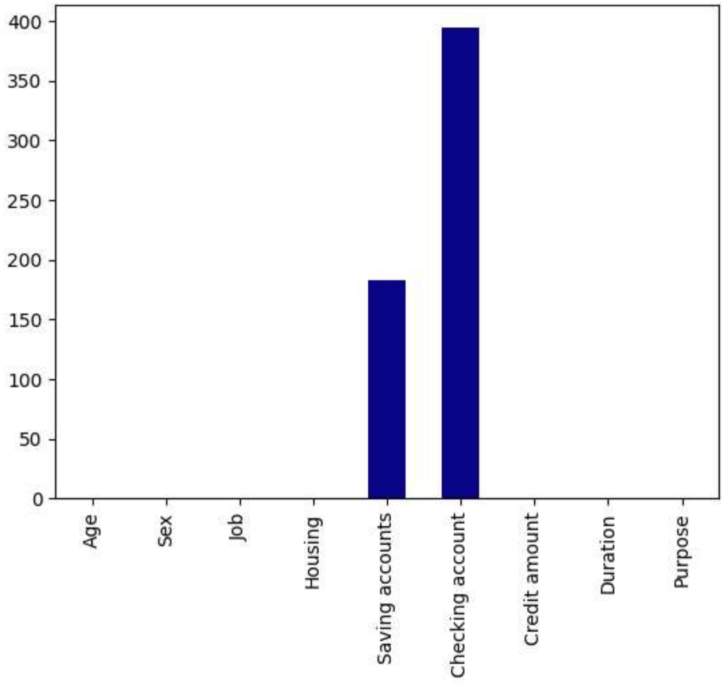
```
german_eda = german_bank.copy()
```

```
german_eda.drop('Unnamed: 0',axis=1,inplace=True)
```

## Null/Missing Values

```
null_values = german_eda.isna().sum()
```

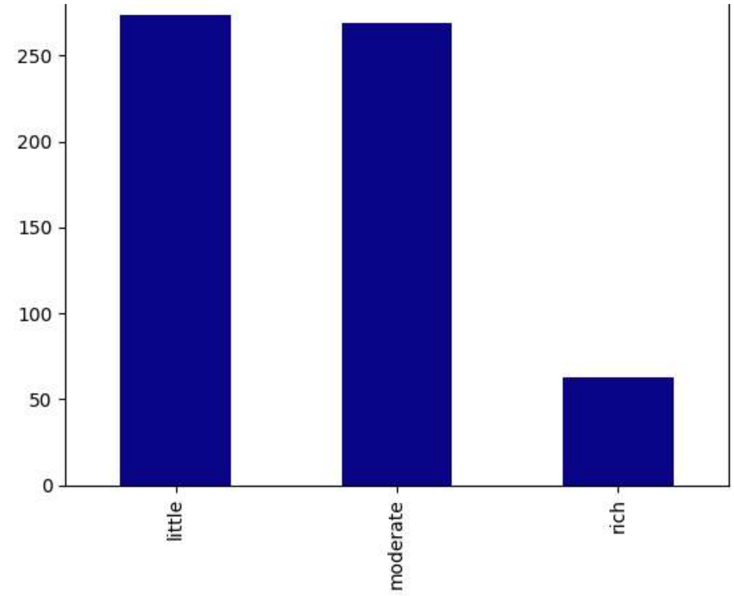
```
null_values.plot(kind='bar',cmap='plasma')
plt.show()
```

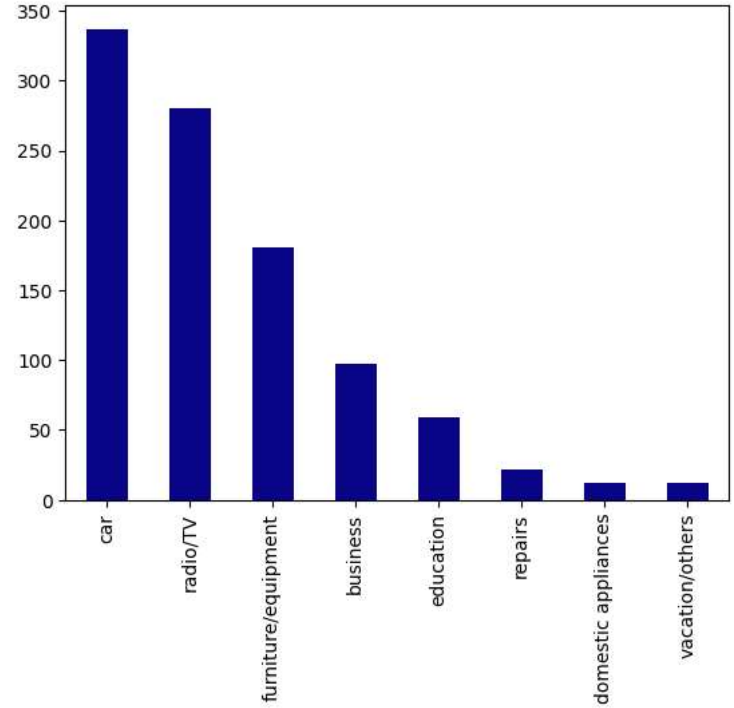


## Value Counts of Data

```
sex_counts = german_eda['Sex'].value_counts()
job_counts = german_eda['Job'].value_counts()
housing_counts = german_eda['Housing'].value_counts()
saving_counts = german_eda['Saving accounts'].value_counts()
checking_counts = german_eda['Checking account'].value_counts()
purpose_counts = german_eda['Purpose'].value_counts()
```

```
value_counts_data  = [sex_counts,job_counts,housing_counts,saving_counts,checking_counts,purpose_counts]
for count_data in value_counts_data:
    count_data.plot(kind='bar',cmap='plasma')
    plt.show()
```


250
200
150
100
50
0
little
moderate
rich


350
300
250
200
150
100
50
0
car
radio/TV
furniture/equipment
business
education
repairs
domestic appliances
vacation/others

## Replacing Null values in Savings Account and Checking Account

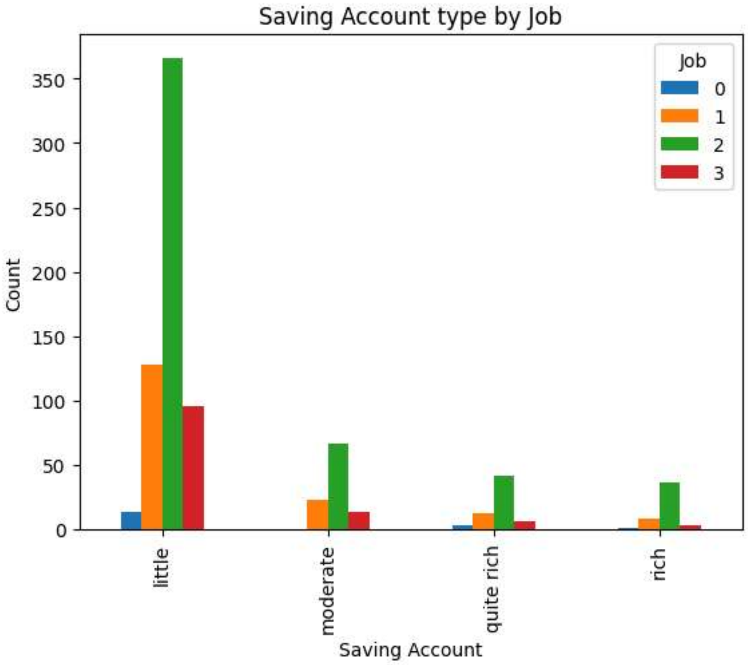
### Saving Accounts

```
german_eda['Saving accounts'].isna().value_counts()
```

```
     False    817
     True     183
     Name: Saving accounts, dtype: int64
```

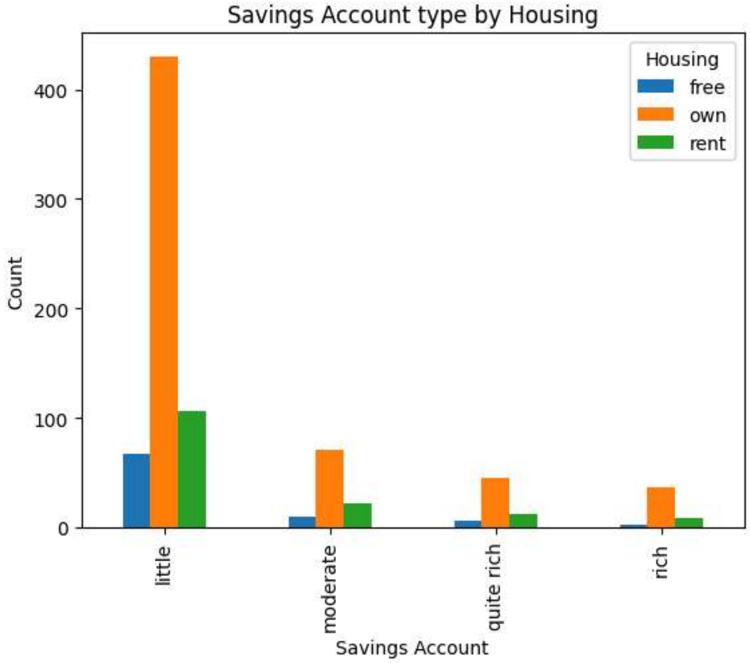
```
saving_job = german_eda.groupby(['Saving accounts','Job']).size().unstack()
```

```
saving_job.plot(kind='bar')
plt.title('Saving Account type by Job')
plt.xlabel('Saving Account')
plt.ylabel('Count')
plt.show()
```



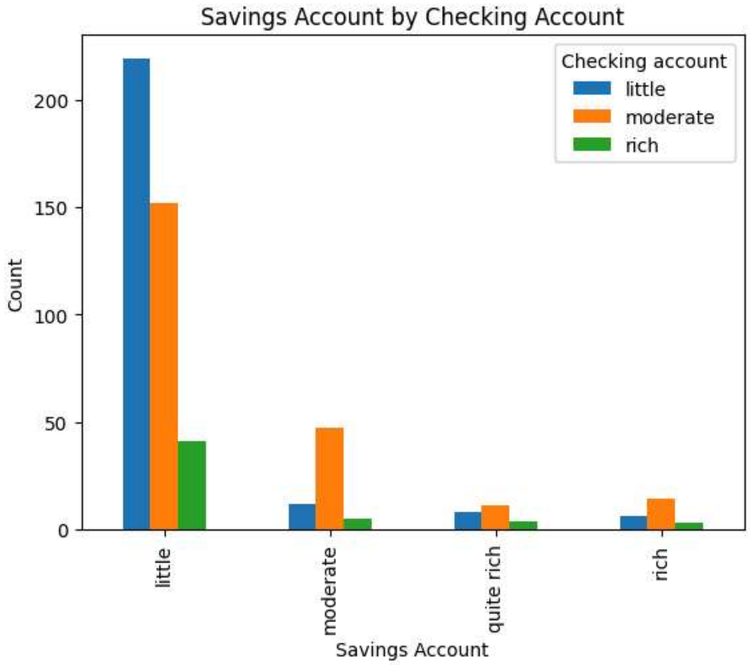
```
savings_house = german_eda.groupby(['Saving accounts','Housing']).size().unstack()
```

```
savings_house.plot(kind='bar')
plt.title('Savings Account type by Housing')
plt.xlabel('Savings Account')
plt.ylabel('Count')
plt.show()
```



```
saving_credit = german_eda.groupby(['Saving accounts','Checking account']).size().unstack()
```

```
saving_credit.plot(kind='bar')
plt.title('Savings Account by Checking Account')
plt.xlabel('Savings Account')
plt.ylabel('Count')
plt.show()
```

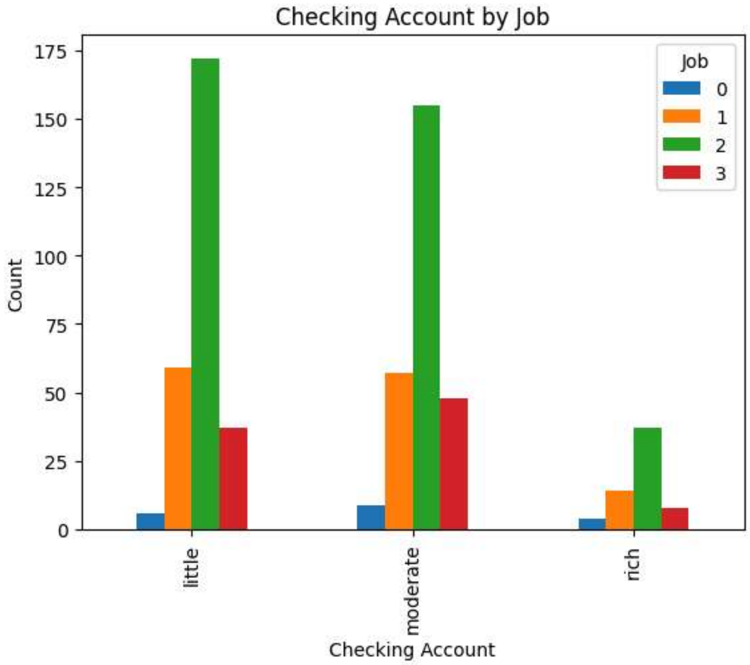


```
german_eda['Saving accounts'].fillna('little',inplace=True)
```

## Checking account

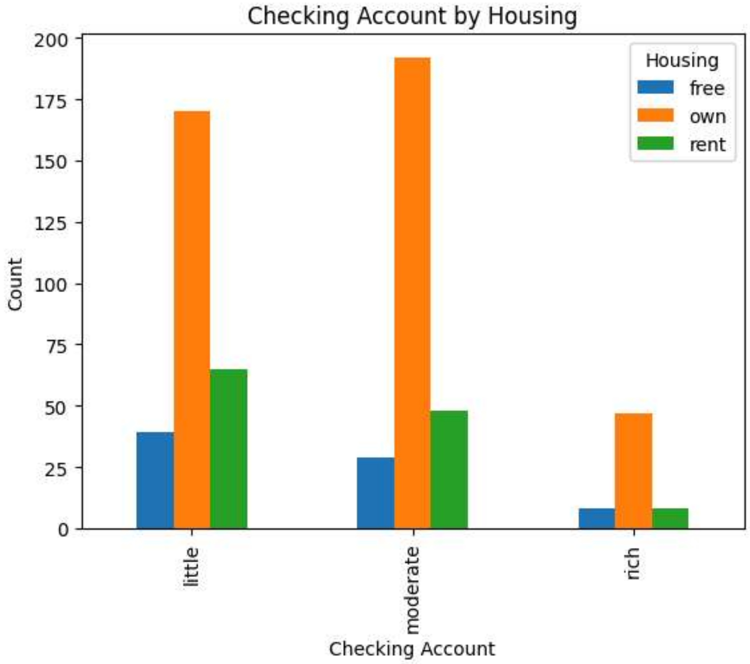
```
checking_job = german_eda.groupby(['Checking account','Job']).size().unstack()
```

```
checking_job.plot(kind='bar')
plt.title('Checking Account by Job')
plt.xlabel('Checking Account')
plt.ylabel('Count')
plt.show()
```



```
checking_house = german_eda.groupby(['Checking account','Housing']).size().unstack()
```

```
checking_house.plot(kind='bar')
plt.title('Checking Account by Housing')
plt.xlabel('Checking Account')
plt.ylabel('Count')
plt.show()
```



```
german_eda.drop('Checking account',axis=1,inplace=True)
```

```
german_eda.isna().sum()
```

```
Age                 0
Sex                 0
Job                 0
Housing             0
```

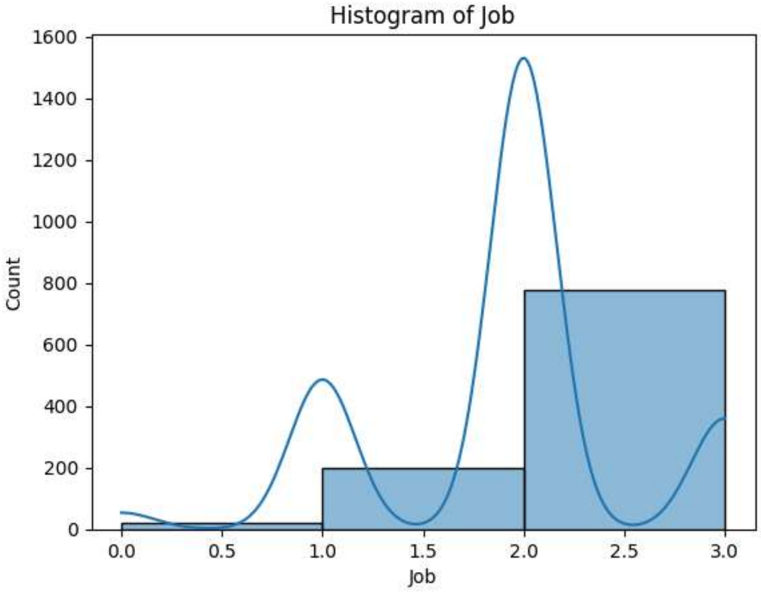
```
Saving accounts    0
Credit amount      0
Duration           0
Purpose            0
dtype: int64
```

## Data Distribution

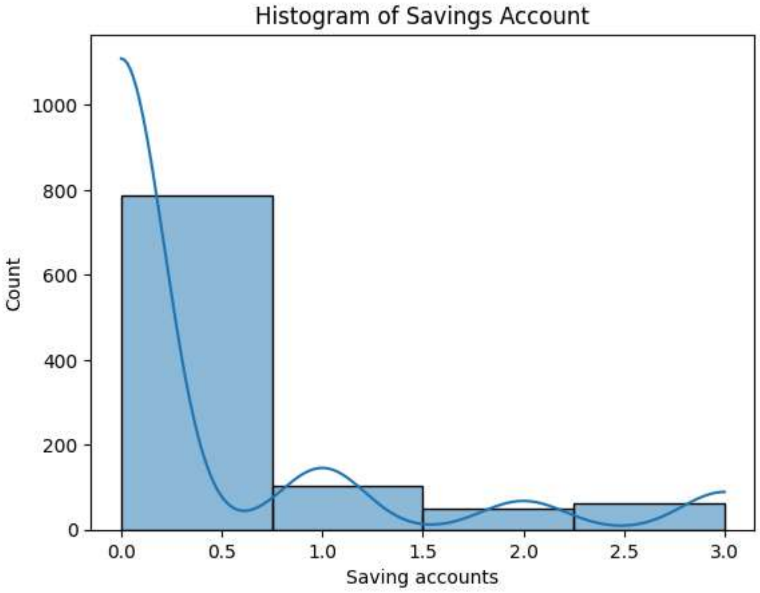
```
german_eda['Saving accounts'].replace({'little':0,'moderate':1,'rich':2,'quite rich':3},inplace=True)
```

```
sns.histplot(data=german_eda,x='Age',kde=True)
plt.title('Histogram of Age')
plt.show()
```

```
sns.histplot(data=german_eda,x='Job',bins=3,kde=True)
plt.title('Histogram of Job')
plt.show()
```

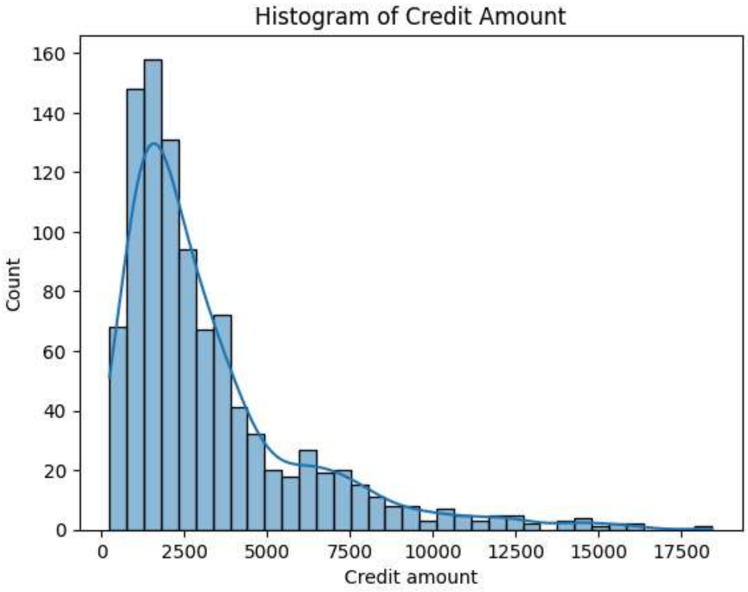


```
sns.histplot(data=german_eda,x='Saving accounts',bins=4,kde=True)
plt.title('Histogram of Savings Account')
plt.show()
```



```
sns.histplot(data=german_eda,x='Credit amount',kde=True)
```

```
plt.title('Histogram of Credit Amount')
plt.show()
```



## Preprocessing Data

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OneHotEncoder,StandardScaler
from sklearn.compose import ColumnTransformer

num_attribs = ['Age','Job','Saving accounts','Credit amount','Duration']
cat_attribs = ['Sex','Housing']
num_pipeline = make_pipeline(StandardScaler())
cat_pipeline = make_pipeline(OneHotEncoder())
preprocessing = ColumnTransformer([
    ('num',num_pipeline,num_attribs),('cat',cat_pipeline,cat_attribs)
])
```

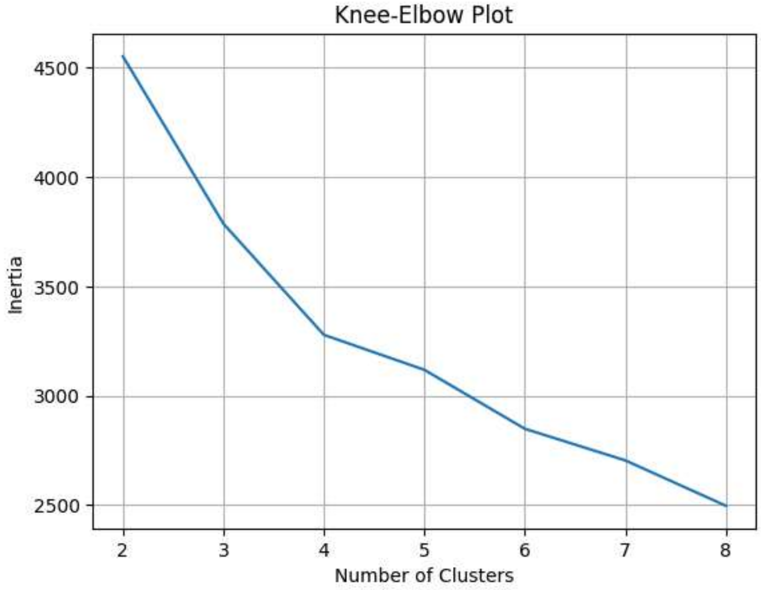
## PCA + KMeans

```
from sklearn.decomposition import PCA

pca = make_pipeline(preprocessing,PCA(n_components=.95))
```

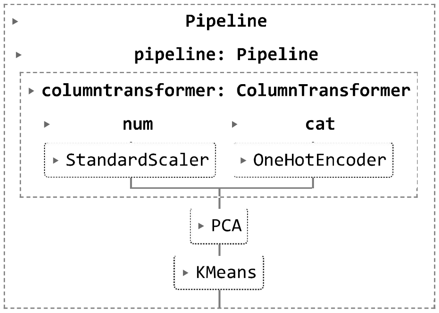
```
german_eda.drop('Purpose',axis=1,inplace=True)
german = german_eda.copy()
```

```
from sklearn.cluster import KMeans

inertias = []
n_clusters = [2,3,4,5,6,7,8]

for cluster in n_clusters:
    kmeans_german = make_pipeline(pca,KMeans(n_clusters=cluster,n_init='auto',random_state=42))
    kmeans_german.fit(german)
    inertias.append(kmeans_german['kmeans'].inertia_)
```

```
plt.plot(n_clusters,inertias)
plt.title('Knee-Elbow Plot')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.grid()
plt.show()
```



```
kmeans_final = make_pipeline(pca,KMeans(n_clusters=4,n_init='auto',random_state=42))
kmeans_final.fit(german)
```


Pipeline

pipeline: Pipeline

columntransformer: ColumnTransformer

num

cat

StandardScaler

OneHotEncoder

PCA

KMeans

```
german['cluster_label'] = kmeans_final['kmeans'].labels_
```

```
transformed_german = kmeans_final.transform(german)
```

```
transformed_columns = [f'transformed_feature_{i}' for i in range(transformed_german.shape[1])]
transformed_df = pd.DataFrame(transformed_german, columns=transformed_columns)
```

```
transformed_df.head()
```

| | transformed_feature_0 | transformed_feature_1 | transformed_feature_2 | transformed_feature_3 |
|---|---|---|---|---|
| **0** | 3.510828 | 4.571742 | 1.816986 | 4.199212 |
| **1** | 3.153190 | 1.970039 | 4.037108 | 4.501855 |
| **2** | 2.298796 | 3.793420 | 1.269146 | 3.529889 |
| **3** | 3.446843 | 1.343676 | 3.090733 | 4.383188 |
| **4** | 2.658024 | 2.413510 | 1.488324 | 3.704256 |

Next steps: Generate code with transformed_df View recommended plots

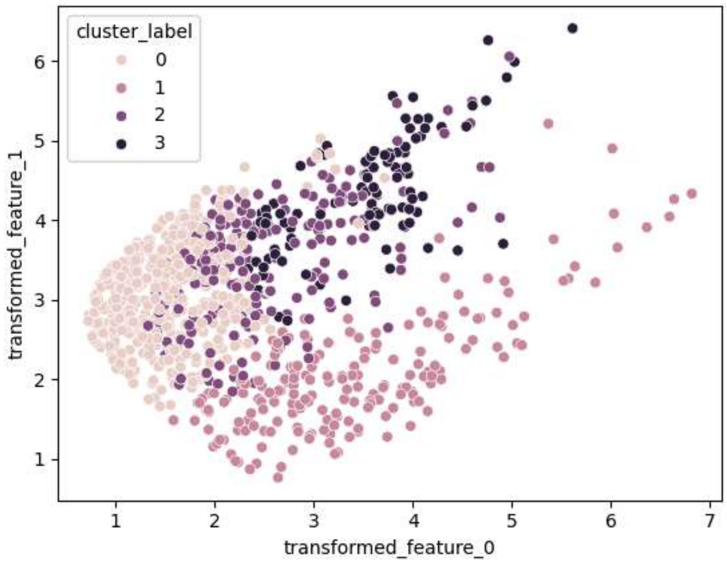
```
transformed_df['cluster_label'] = kmeans_final['kmeans'].labels_
```

```
transformed_df.head()
```

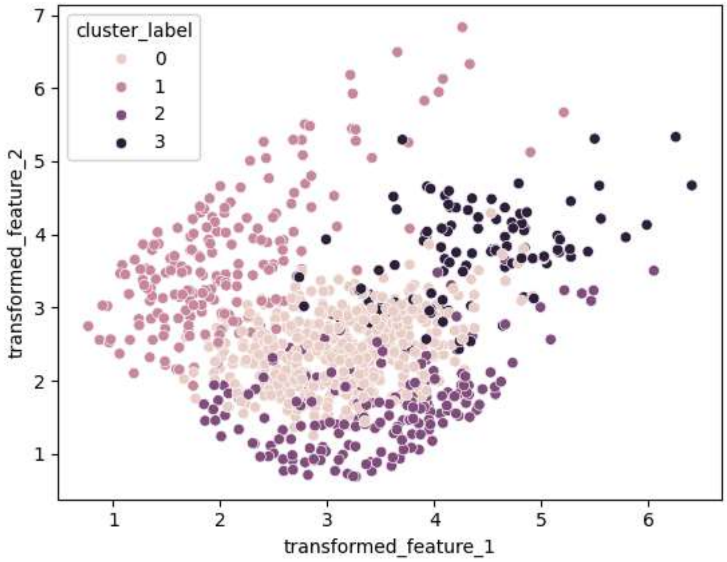
| | transformed_feature_0 | transformed_feature_1 | transformed_feature_2 | transformed_feature_3 | cluster_label |
|---|---|---|---|---|---|
| 0 | 3.510828 | 4.571742 | 1.816986 | 4.199212 | 2 |
| 1 | 3.153190 | 1.970039 | 4.037108 | 4.501855 | 1 |
| 2 | 2.298796 | 3.793420 | 1.269146 | 3.529889 | 2 |
| 3 | 3.446843 | 1.343676 | 3.090733 | 4.383188 | 1 |
| 4 | 2.658024 | 2.413510 | 1.488324 | 3.704256 | 2 |

Next steps: Generate code with `transformed_df` View recommended plots

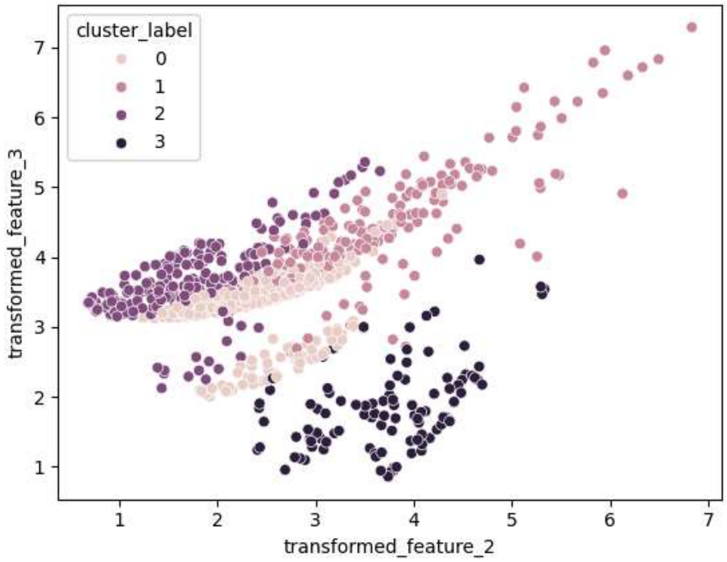
```
sns.scatterplot(data=transformed_df,x='transformed_feature_0',y='transformed_feature_1',hue='cluster_label')
plt.show()
```



```
sns.scatterplot(data=transformed_df,x='transformed_feature_1',y='transformed_feature_2',hue='cluster_label')
plt.show()
```



```
sns.scatterplot(data=transformed_df,x='transformed_feature_2',y='transformed_feature_3',hue='cluster_label')
plt.show()
```



```
german.head()
```

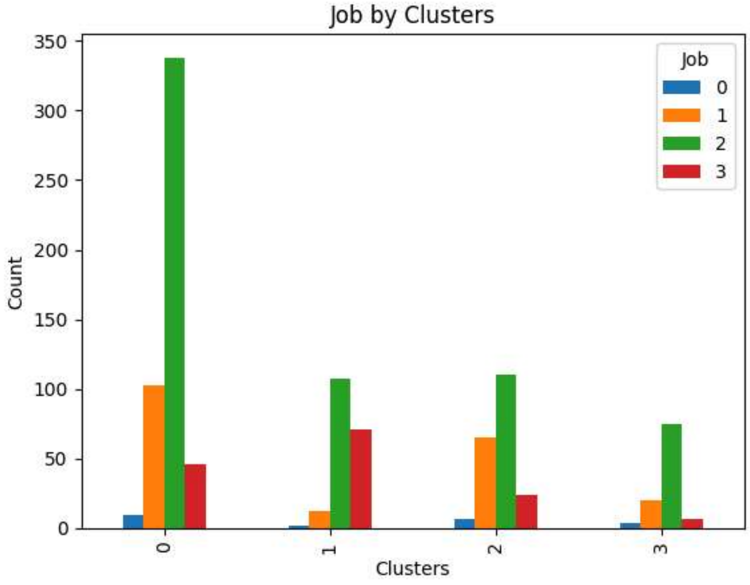
|   | Age | Sex | Job | Housing | Saving accounts | Credit amount | Duration | cluster_label |
|---|---|---|---|---|---|---|---|---|
| **0** | 67 | male | 2 | own | 0 | 1169 | 6 | 2 |
| **1** | 22 | female | 2 | own | 0 | 5951 | 48 | 1 |
| **2** | 49 | male | 1 | own | 0 | 2096 | 12 | 2 |
| **3** | 45 | male | 2 | free | 0 | 7882 | 42 | 1 |
| **4** | 53 | male | 2 | free | 0 | 4870 | 24 | 2 |

Next steps: Generate code with german View recommended plots

## Cluster Analysis

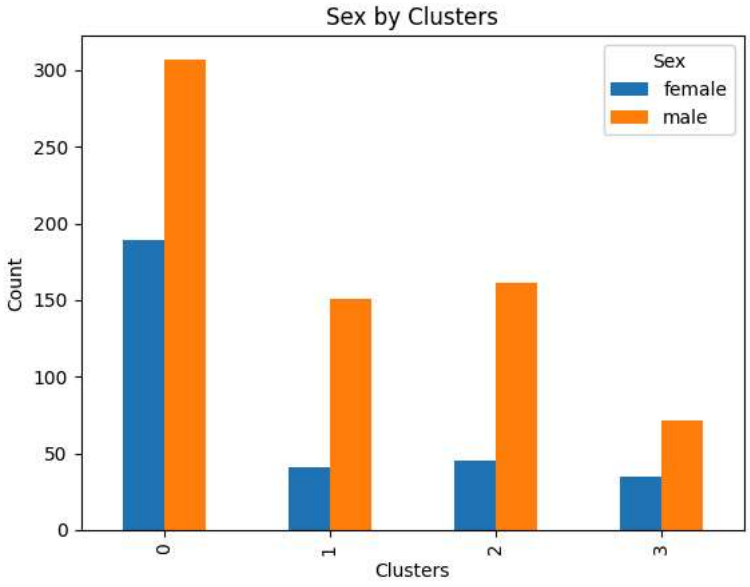
```
cluster_job = german.groupby(['cluster_label','Job']).size().unstack()
```

```
cluster_job.plot(kind='bar')
plt.title('Job by Clusters')
plt.xlabel('Clusters')
plt.ylabel('Count')
plt.show()
```



```
cluster_gender = german.groupby(['cluster_label','Sex']).size().unstack()
```

```
cluster_gender.plot(kind='bar')
plt.title('Sex by Clusters')
plt.xlabel('Clusters')
plt.ylabel('Count')
plt.show()
```



```
sns.boxplot(data=german,x='cluster_label',y='Age')
plt.title('Box Plot of Clusters by Age')
plt.show()
```