Collect Dataset

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

```
# Unduh dan baca file Excel
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00350/default%20of%20credit%20card%20clients.xls"
df = pd.read_excel(url, header=1)  # header=1 karena baris ke-2 adalah nama kolom yang benar
```

```
# Hapus kolom 'ID'
df.drop(columns=["ID"], inplace=True)
```

```
# Simpan sebagai CSV
df.to_csv("credit_card_default_clean.csv", index=False)
print("CSV berhasil disimpan.")
```

```
CSV berhasil disimpan.
```

Exploratory Data Analysis

```
# Menampilkan beberapa baris pertama
df.head()
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AM
0	20000	2	2	1	24	2	2	-1	-1	-2	...	0	0	0	0	689	
1	120000	2	2	2	26	-1	2	0	0	0	...	3272	3455	3261	0	1000	10
2	90000	2	2	2	34	0	0	0	0	0	...	14331	14948	15549	1518	1500	10
3	50000	2	2	1	37	0	0	0	0	0	...	28314	28959	29547	2000	2019	12
4	50000	1	2	1	57	-1	0	-1	0	0	...	20940	19146	19131	2000	36681	100

5 rows × 24 columns

```
# Menampilkan informasi umum dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 24 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   LIMIT_BAL                   30000 non-null  int64
 1   SEX                         30000 non-null  int64
 2   EDUCATION                   30000 non-null  int64
 3   MARRIAGE                    30000 non-null  int64
 4   AGE                         30000 non-null  int64
 5   PAY_0                       30000 non-null  int64
 6   PAY_2                       30000 non-null  int64
 7   PAY_3                       30000 non-null  int64
 8   PAY_4                       30000 non-null  int64
 9   PAY_5                       30000 non-null  int64
 10  PAY_6                       30000 non-null  int64
 11  BILL_AMT1                   30000 non-null  int64
 12  BILL_AMT2                   30000 non-null  int64
 13  BILL_AMT3                   30000 non-null  int64
 14  BILL_AMT4                   30000 non-null  int64
 15  BILL_AMT5                   30000 non-null  int64
 16  BILL_AMT6                   30000 non-null  int64
 17  PAY_AMT1                    30000 non-null  int64
```

```
 18  PAY_AMT2                    30000 non-null  int64
 19  PAY_AMT3                    30000 non-null  int64
 20  PAY_AMT4                    30000 non-null  int64
 21  PAY_AMT5                    30000 non-null  int64
 22  PAY_AMT6                    30000 non-null  int64
 23  default payment next month  30000 non-null  int64
dtypes: int64(24)
memory usage: 5.5 MB
```

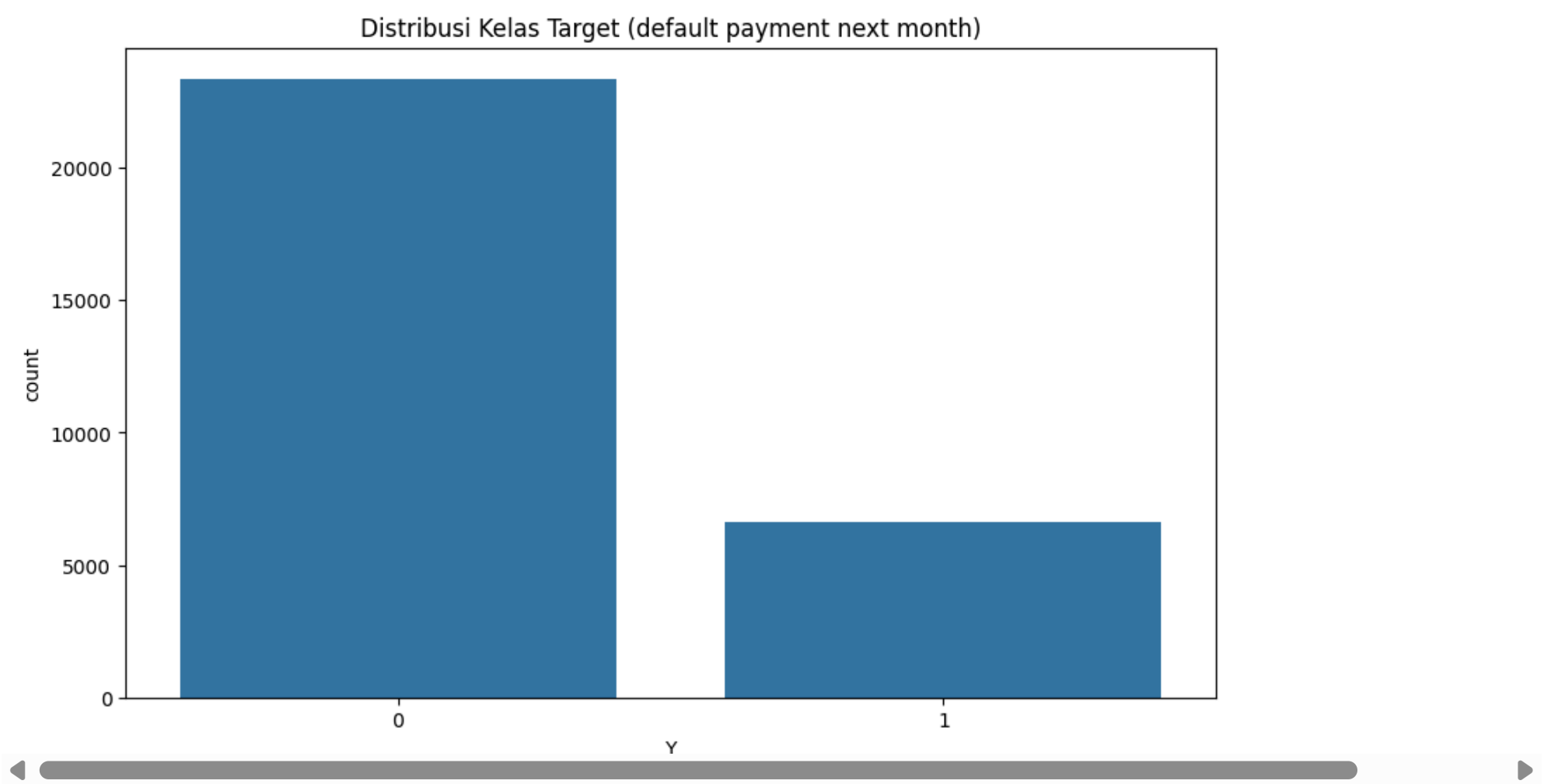
```
# Menampilkan deskripsi data
df.describe()
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.00000(
mean	167484.322667	1.603733	1.853133	1.551867	35.485500	-0.016700	-0.133767	-0.166200	-0.220667	-0.26620(
std	129747.661567	0.489129	0.790349	0.521970	9.217904	1.123802	1.197186	1.196868	1.169139	1.13318
min	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.00000(
25%	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.00000(
50%	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.000000	0.000000	0.000000	0.00000(
75%	240000.000000	2.000000	2.000000	2.000000	41.000000	0.000000	0.000000	0.000000	0.000000	0.00000(
max	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000	8.000000	8.000000	8.000000	8.00000(

8 rows × 24 columns

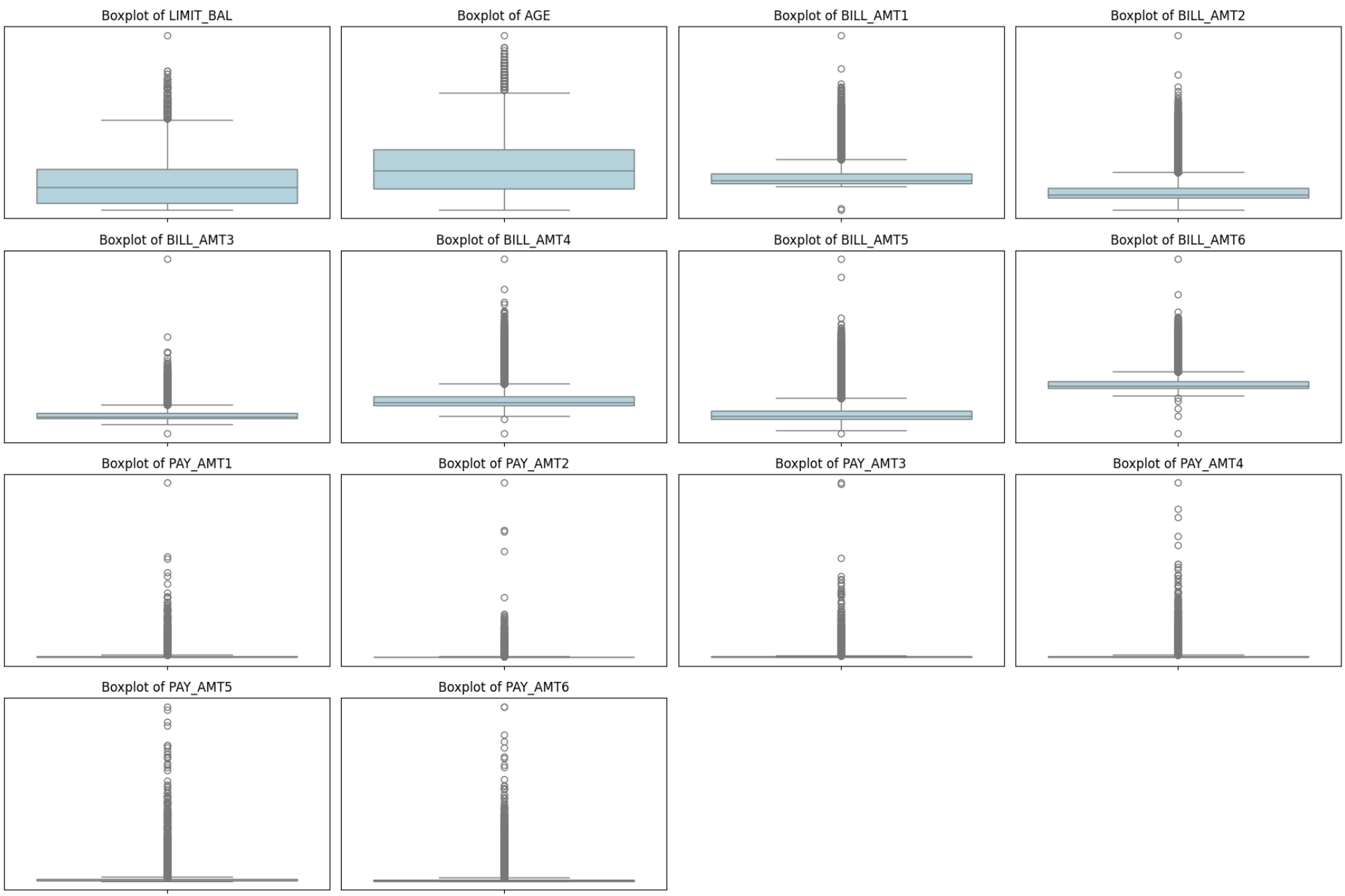
```
# Pastikan 'default payment next month' adalah kolom target, kita ubah namanya jadi 'Y' untuk konsistensi
df.rename(columns={'default payment next month': 'Y'}, inplace=True)
```

```
# Cek distribusi kelas
plt.figure(figsize=(10,6))
sns.countplot(data=df, x='Y', order=df['Y'].value_counts().index)
plt.title("Distribusi Kelas Target (default payment next month)")
plt.show()
```




```
# Visualisasi outlier

# Pastikan kolom numerik dalam tipe float
numerical_cols = [
    'LIMIT_BAL', 'AGE',
```

```
        'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3',
        'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6',
        'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3',
        'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6'
    ]
    # Pastikan kolom numerik bertipe float
    df[numerical_cols] = df[numerical_cols].apply(pd.to_numeric, errors='coerce')

    # Hitung dan simpan jumlah outlier
    outlier_counts = {}

    # Visualisasi boxplot dan hitung outlier berdasarkan IQR
    plt.figure(figsize=(18, 12))
    for i, col in enumerate(numerical_cols):
        plt.subplot(4, 4, i + 1)
        plt.yticks([])
        sns.boxplot(y=df[col], color='lightblue')
        plt.title(f'Boxplot of {col}')

        # Hitung IQR dan outlier
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]

        outlier_counts[col] = len(outliers)

    plt.tight_layout()
    plt.show()

    # Tampilkan jumlah outlier
    print("Jumlah outlier tiap kolom:")
    for col, count in outlier_counts.items():
        print(f"{col}: {count}")
```




```
Jumlah outlier tiap kolom:
LIMIT_BAL: 167
AGE: 272
BILL_AMT1: 2400
BILL_AMT2: 2395
```

```
BILL_AMT2: 2595
BILL_AMT3: 2469
BILL_AMT4: 2622
BILL_AMT5: 2725
BILL_AMT6: 2693
PAY_AMT1: 2745
PAY_AMT2: 2714
PAY_AMT3: 2598
PAY_AMT4: 2994
PAY_AMT5: 2945
PAY_AMT6: 2958
```

```
# Cek missing values
df.isnull().sum()
```

	0
LIMIT_BAL	0
SEX	0
EDUCATION	0
MARRIAGE	0
AGE	0
PAY_0	0
PAY_2	0
PAY_3	0
PAY_4	0
PAY_5	0
PAY_6	0
BILL_AMT1	0
BILL_AMT2	0
BILL_AMT3	0
BILL_AMT4	0
BILL_AMT5	0
BILL_AMT6	0
PAY_AMT1	0
PAY_AMT2	0
PAY_AMT3	0
PAY_AMT4	0
PAY_AMT5	0
PAY_AMT6	0
Y	0

```
# Cek duplikasi data
df.duplicated().sum()
```

```
np.int64(35)
```

```
# Tampilkan data duplikat
duplicates = df[df.duplicated()]
duplicates
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2
1980	150000	2	1	1	38	1	-2	-2	-2	-2	...	0	0	0	0	0
4585	150000	2	1	1	31	1	-2	-2	-2	-2	...	0	0	0	0	0
6022	210000	2	1	2	39	1	-2	-2	-2	-2	...	0	0	0	0	0
6466	210000	2	2	1	49	1	-2	-2	-2	-2	...	0	0	0	0	0
7319	500000	1	1	1	43	1	-2	-2	-2	-2	...	0	0	0	0	0
8320	360000	1	2	1	41	1	-2	-2	-2	-2	...	0	0	0	0	0
10250	50000	1	2	2	26	1	-2	-2	-2	-2	...	0	0	0	0	0
13106	360000	2	1	1	49	1	-2	-2	-2	-2	...	0	0	0	0	0
14294	20000	1	2	2	24	2	2	4	4	4	...	1650	1650	1650	0	0
15458	160000	1	2	2	28	-2	-2	-2	-2	-2	...	0	0	0	0	0
15617	200000	2	2	2	26	-2	-2	-2	-2	-2	...	0	0	0	0	0
15685	360000	1	1	2	29	1	-2	-2	-2	-2	...	0	0	0	0	0
17032	50000	2	1	2	23	1	-2	-2	-2	-2	...	0	0	0	0	0
19114	80000	2	2	1	31	-2	-2	-2	-2	-2	...	0	0	0	0	0
19487	180000	2	1	2	28	1	-2	-2	-2	-2	...	0	0	0	0	0
19604	110000	2	1	2	31	1	-2	-2	-2	-2	...	0	0	0	0	0
19897	100000	2	2	1	49	1	-2	-2	-2	-2	...	0	0	0	0	0
20875	230000	1	1	1	39	-1	-1	-1	-1	-1	...	660	660	660	660	660
21881	160000	2	3	2	26	-1	-1	-1	-1	-1	...	390	390	390	390	390
22162	360000	2	1	2	29	1	-2	-2	-2	-2	...	0	0	0	0	0
23877	200000	1	1	2	30	-2	-2	-2	-2	-2	...	0	0	0	0	0
24122	140000	1	1	2	29	1	-2	-2	-2	-2	...	0	0	0	0	0
25608	360000	2	1	1	41	-2	-2	-2	-2	-2	...	0	0	0	0	0
26249	90000	2	1	2	31	1	-2	-2	-2	-2	...	0	0	0	0	0

26805	300000	1	1	2	27	-2	-2	-2	-2	-2	...	0	0	0	0	0
27351	210000	1	2	1	39	-1	-1	-1	-1	-1	...	1443	1443	1443	1443	1443
27765	80000	2	2	2	25	-2	-2	-2	-2	-2	...	0	0	0	0	0
27928	150000	2	1	2	28	1	-2	-2	-2	-2	...	0	0	0	0	0
27966	360000	2	1	2	27	1	-2	-2	-2	-2	...	0	0	0	0	0
28228	200000	2	1	1	34	1	-2	-2	-2	-2	...	0	0	0	0	0
28779	200000	2	1	1	36	1	-2	-2	-2	-2	...	0	0	0	0	0
28983	80000	2	3	1	42	-2	-2	-2	-2	-2	...	0	0	0	0	0
29265	180000	1	2	1	26	-1	-1	-1	-1	-1	...	396	396	396	396	396
29823	220000	1	1	1	42	1	-2	-2	-2	-2	...	0	0	0	0	0
29909	360000	1	1	2	32	-2	-2	-2	-2	-2	...	0	0	0	0	0

35 rows × 24 columns

```
# Cek nilai unik / unique values
df.nunique()
```

	0
LIMIT_BAL	81
SEX	2
EDUCATION	7
MARRIAGE	4
AGE	56
PAY_0	11
PAY_2	11
PAY_3	11
PAY_4	11
PAY_5	10
PAY_6	10
BILL_AMT1	22723
BILL_AMT2	22346
BILL_AMT3	22026
BILL_AMT4	21548
BILL_AMT5	21010
BILL_AMT6	20604
PAY_AMT1	7943
PAY_AMT2	7899
PAY_AMT3	7518
PAY_AMT4	6937
PAY_AMT5	6897
PAY_AMT6	6939
Y	2

```
# Tampilkan nilai unik setiap fitur
for col in df.columns:
    unique_vals = df[col].unique()
    print(f"Kolom '{col}' memiliki {len(unique_vals)} nilai unik:")
    print(unique_vals)
    print("-" * 50)
```

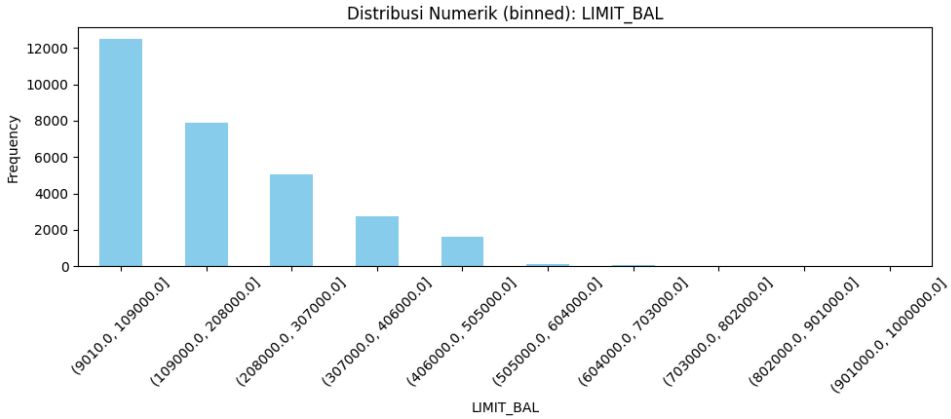
```
Kolom 'LIMIT_BAL' memiliki 81 nilai unik:
[  20000  120000   90000   50000  500000  100000  140000  200000  260000
  630000   70000  250000  320000  360000  180000  130000  450000   60000
  230000  160000  280000   10000   40000  210000  150000  380000  310000
  400000   80000  290000  340000  300000   30000  240000  470000  480000
  350000  330000  110000  420000  170000  370000  270000  220000  190000
  510000  460000  440000  410000  490000  390000  580000  600000  620000
  610000  700000  670000  680000  430000  550000  540000 1000000  530000
  710000  560000  520000  750000  640000   16000  570000  590000  660000
  720000  327680  740000  800000  760000  690000  650000  780000  730000]
--------------------------------------------------
Kolom 'SEX' memiliki 2 nilai unik:
[2 1]
--------------------------------------------------
Kolom 'EDUCATION' memiliki 7 nilai unik:
[2 1 3 5 4 6 0]
--------------------------------------------------
Kolom 'MARRIAGE' memiliki 4 nilai unik:
[1 2 3 0]
--------------------------------------------------
Kolom 'AGE' memiliki 56 nilai unik:
[24 26 34 37 57 29 23 28 35 51 41 30 49 39 40 27 47 33 32 54 58 22 25 31
 46 42 43 45 56 44 53 38 63 36 52 48 55 60 50 75 61 73 59 21 67 66 62 70
 72 64 65 71 69 68 79 74]
--------------------------------------------------
Kolom 'PAY_0' memiliki 11 nilai unik:
[ 2 -1  0 -2  1  3  4  8  7  5  6]
--------------------------------------------------
Kolom 'PAY_2' memiliki 11 nilai unik:
[ 2  0 -1 -2  3  5  7  4  1  6  8]
--------------------------------------------------
Kolom 'PAY_3' memiliki 11 nilai unik:
[-1  0  2 -2  3  4  6  7  1  5  8]
--------------------------------------------------
```

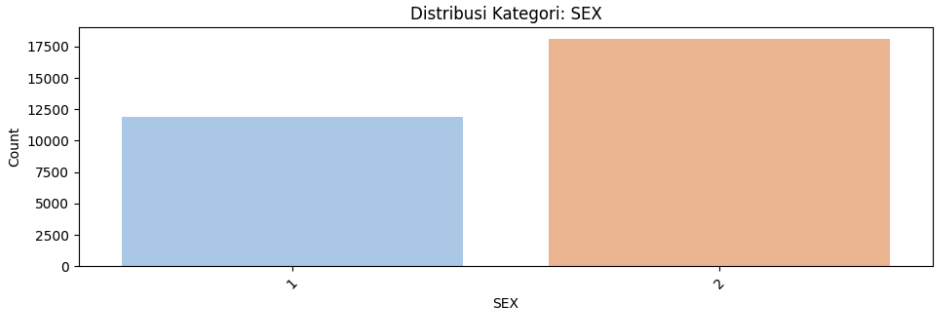
```
Kolom 'PAY_4' memiliki 11 nilai unik:
[-1  0 -2  2  3  4  5  7  6  1  8]
--------------------------------------------------
Kolom 'PAY_5' memiliki 10 nilai unik:
[-2  0 -1  2  3  5  4  7  8  6]
--------------------------------------------------
Kolom 'PAY_6' memiliki 10 nilai unik:
[-2  2  0 -1  3  6  4  7  8  5]
--------------------------------------------------
Kolom 'BILL_AMT1' memiliki 22723 nilai unik:
[ 3913  2682 29239 ...  1683 -1645 47929]
--------------------------------------------------
Kolom 'BILL_AMT2' memiliki 22346 nilai unik:
[ 3102  1725 14027 ...  3356 78379 48905]
--------------------------------------------------
Kolom 'BILL_AMT3' memiliki 22026 nilai unik:
[  689  2682 13559 ...  2758 76304 49764]
--------------------------------------------------
Kolom 'BILL_AMT4' memiliki 21548 nilai unik:
[    0  3272 14331 ... 20878 52774 36535]
--------------------------------------------------
Kolom 'BILL_AMT5' memiliki 21010 nilai unik:
[    0  3455 14948 ... 31237  5190 32428]
--------------------------------------------------
```

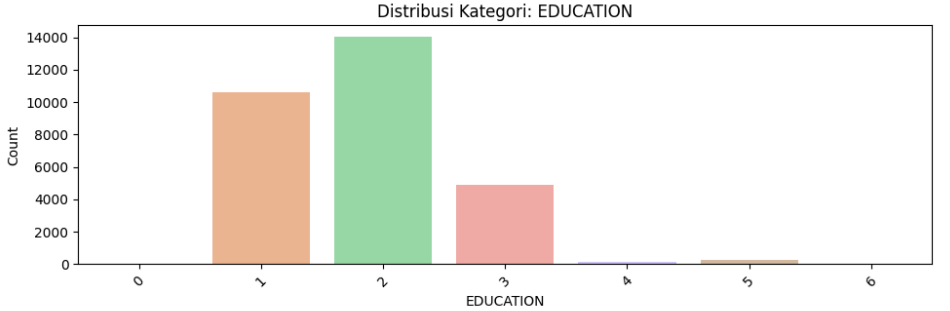
Visualization Features

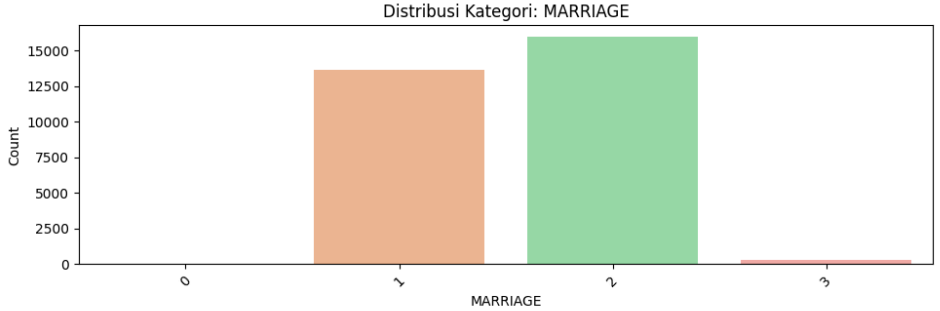
```
# Pilih 17 fitur (selain 'Y') — bisa dipilih manual atau otomatis
selected_features = df.drop(columns='Y').columns[:17]  # ambil 17 fitur pertama

# Ukuran plot
plt.figure(figsize=(20, 40))

# Loop semua fitur untuk divisualisasikan
for i, col in enumerate(selected_features):
    plt.subplot(9, 2, i+1)  # 9 baris x 2 kolom (cukup untuk 17 fitur)

    # Cek tipe fitur: numerik atau kategorikal
    if df[col].nunique() <= 10:
        # Fitur kategorikal: plot jumlah masing-masing kategori
        sns.countplot(data=df, x=col, hue=col, palette="pastel", legend=False)
        plt.title(f'Distribusi Kategori: {col}')
```

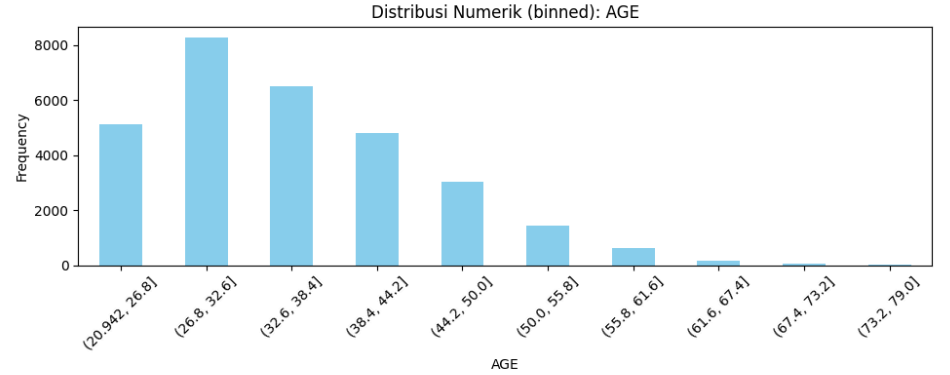
```
            plt.xlabel(col)
            plt.ylabel('Count')
        else:
            # Fitur numerik: bin-kan dan plot distribusi
            binned = pd.cut(df[col], bins=10)
            binned_counts = binned.value_counts().sort_index()
            binned_counts.plot(kind='bar', color='skyblue')
            plt.title(f'Distribusi Numerik (binned): {col}')
            plt.xlabel(col)
            plt.ylabel('Frequency')

        plt.xticks(rotation=45)

    plt.tight_layout()
    plt.show()
```

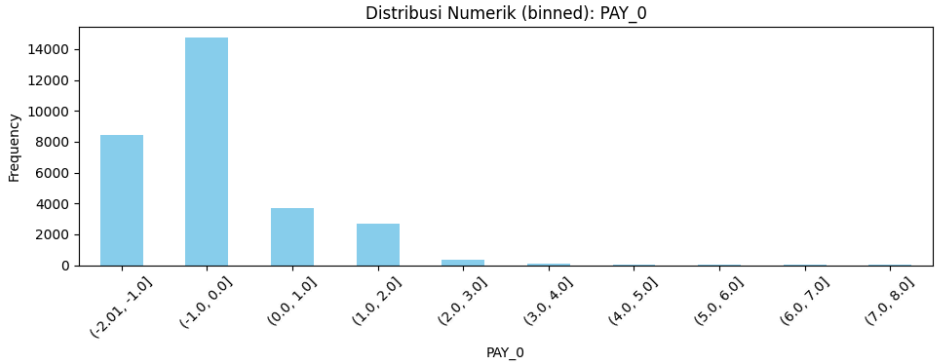

Distribusi Numerik (binned): LIMIT_BAL
Frequency
LIMIT_BAL
(9010.0, 109000.0]
(109000.0, 208000.0]
(208000.0, 307000.0]
(307000.0, 406000.0]
(406000.0, 505000.0]
(505000.0, 604000.0]
(604000.0, 703000.0]
(703000.0, 802000.0]
(802000.0, 901000.0]
(901000.0, 1000000.0]

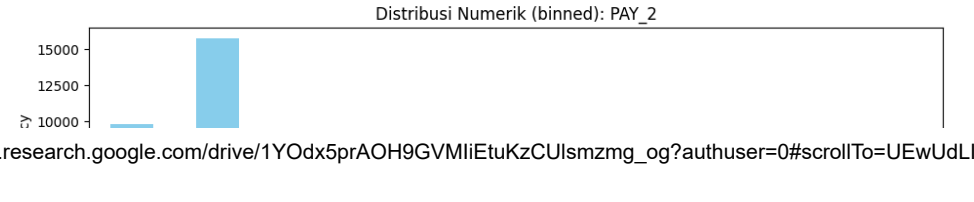

Distribusi Kategori: SEX
Count
SEX
1
2

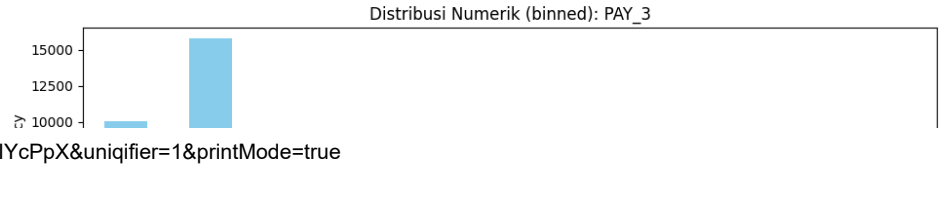

Distribusi Kategori: EDUCATION
Count
EDUCATION
0
1
2
3
4
5
6

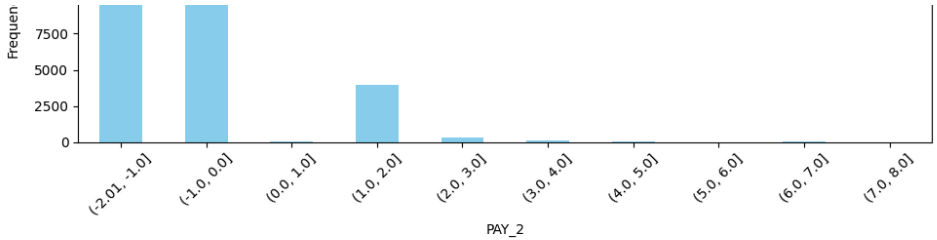

Distribusi Kategori: MARRIAGE
Count
MARRIAGE
0
1
2
3

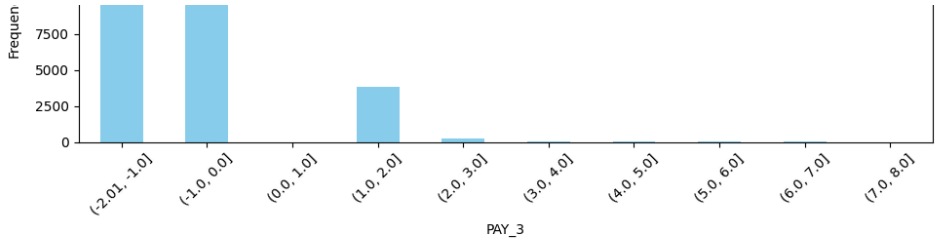

Distribusi Numerik (binned): AGE
Frequency
AGE
(20.942, 26.8]
(26.8, 32.6]
(32.6, 38.4]
(38.4, 44.2]
(44.2, 50.0]
(50.0, 55.8]
(55.8, 61.6]
(61.6, 67.4]
(67.4, 73.2]
(73.2, 79.0]

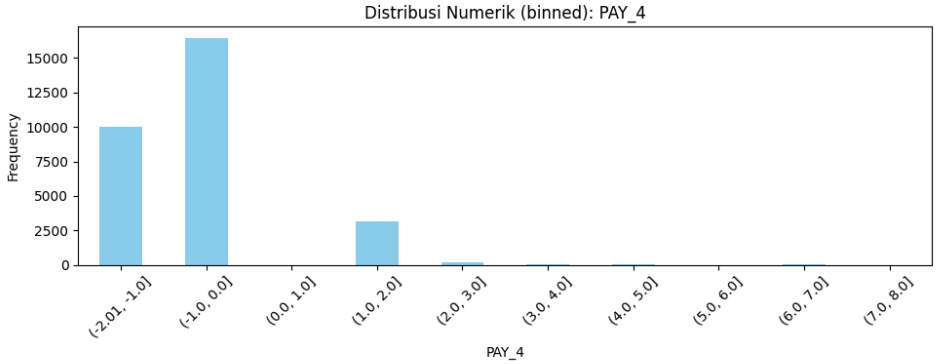

Distribusi Numerik (binned): PAY_0
Frequency
PAY_0
(-2.01, -1.0]
(-1.0, 0.0]
(0.0, 1.0]
(1.0, 2.0]
(2.0, 3.0]
(3.0, 4.0]
(4.0, 5.0]
(5.0, 6.0]
(6.0, 7.0]
(7.0, 8.0]

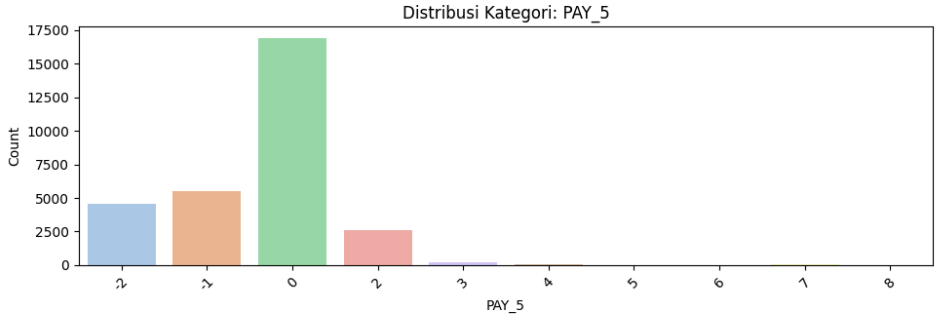

Distribusi Numerik (binned): PAY_2

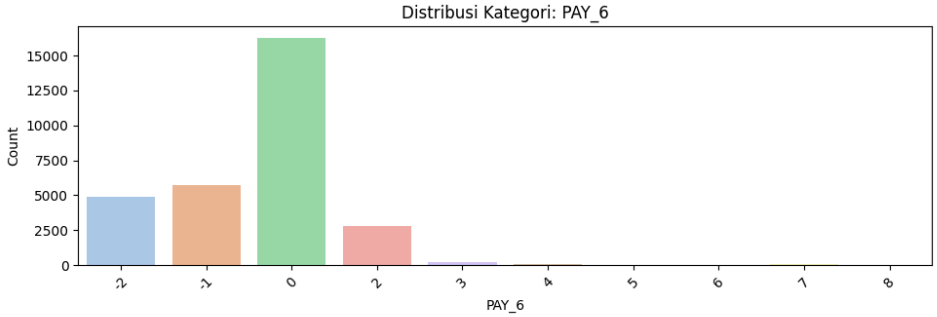

Distribusi Numerik (binned): PAY_3

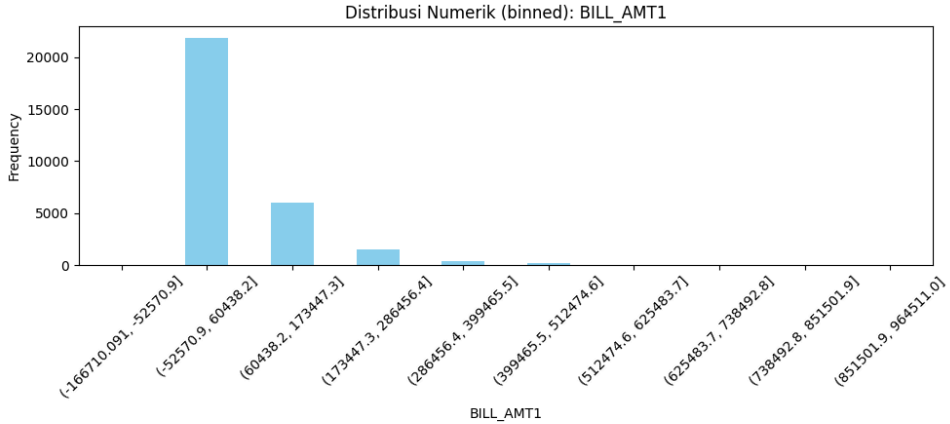

Frequen
7500
5000
2500
0
(-2.01, -1.0]
(-1.0, 0.0]
(0.0, 1.0]
(1.0, 2.0]
(2.0, 3.0]
(3.0, 4.0]
(4.0, 5.0]
(5.0, 6.0]
(6.0, 7.0]
(7.0, 8.0]
PAY_2

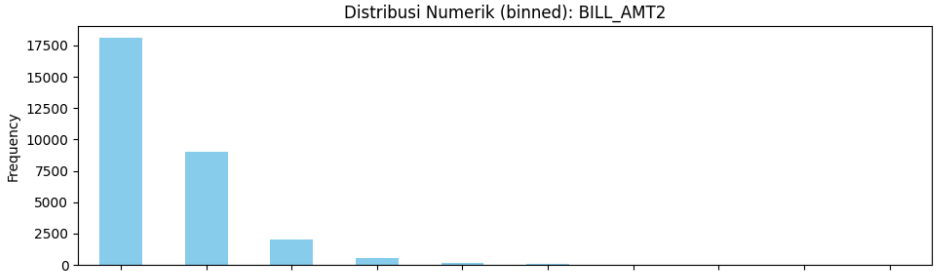

Frequen
7500
5000
2500
0
(-2.01, -1.0]
(-1.0, 0.0]
(0.0, 1.0]
(1.0, 2.0]
(2.0, 3.0]
(3.0, 4.0]
(4.0, 5.0]
(5.0, 6.0]
(6.0, 7.0]
(7.0, 8.0]
PAY_3

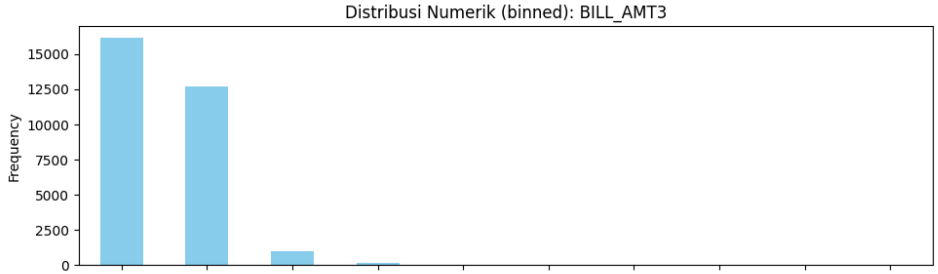

Distribusi Numerik (binned): PAY_4
Frequency
15000
12500
10000
7500
5000
2500
0
(-2.01, -1.0]
(-1.0, 0.0]
(0.0, 1.0]
(1.0, 2.0]
(2.0, 3.0]
(3.0, 4.0]
(4.0, 5.0]
(5.0, 6.0]
(6.0, 7.0]
(7.0, 8.0]
PAY_4


Distribusi Kategori: PAY_5
Count
17500
15000
12500
10000
7500
5000
2500
0
-2
-1
0
2
3
4
5
6
7
8
PAY_5


Distribusi Kategori: PAY_6
Count
15000
12500
10000
7500
5000
2500
0
-2
-1
0
2
3
4
5
6
7
8
PAY_6

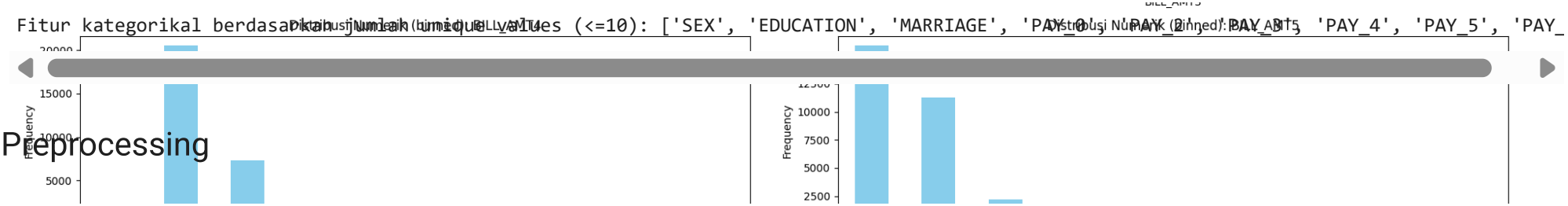

Distribusi Numerik (binned): BILL_AMT1
Frequency
20000
15000
10000
5000
0
(-166710.091, -52570.9]
(-52570.9, 60438.2]
(60438.2, 173447.3]
(173447.3, 286456.4]
(286456.4, 399465.5]
(399465.5, 512474.6]
(512474.6, 625483.7]
(625483.7, 738492.8]
(738492.8, 851501.9]
(851501.9, 964511.0]
BILL_AMT1


Distribusi Numerik (binned): BILL_AMT2
Frequency
17500
15000
12500
10000
7500
5000
2500
0


Distribusi Numerik (binned): BILL_AMT3
Frequency
15000
12500
10000
7500
5000
2500
0

```
# Berdasarkan jumlah nilai unik (misalnya <=15)
categorical_by_unique_values = [col for col in df.columns if df[col].nunique() <= 15 and col != 'Y']
print("Fitur kategorikal berdasarkan jumlah unique values (<=10):", categorical_by_unique_values)
```

```
Fitur kategorikal berdasarkan jumlah unique values (<=10): ['SEX', 'EDUCATION', 'MARRIAGE', 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_
```



Preprocessing

```
# Import Library
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
```

Handling Missing Values

```
# Ganti nilai kosong dalam bentuk string dengan NaN
df.replace([' ', '', 'NA', 'NaN', 'NULL', 'null'], np.nan, inplace=True)
```

```
# Ganti semua nilai NaN dengan mean masing-masing kolom
df = df.fillna(df.mean(numeric_only=True))
```

```
for col in df.columns:
    unique_vals = df[col].unique()
    print(f"Kolom '{col}' memiliki {len(unique_vals)} nilai unik:")
    print(unique_vals)
    print("-" * 50)
```

```
Kolom 'LIMIT_BAL' memiliki 81 nilai unik:
[  20000  120000   90000   50000  500000  100000  140000  200000  260000
  630000   70000  250000  320000  360000  180000  130000  450000   60000
  230000  160000  280000   10000   40000  210000  150000  380000  310000
  400000   80000  290000  340000  300000   30000  240000  470000  480000
```

```
  350000  330000  110000  420000  170000  370000  270000  220000  190000
  510000  460000  440000  410000  490000  390000  580000  600000  620000
  610000  700000  670000  680000  430000  550000  540000 1000000  530000
  710000  560000  520000  750000  640000   16000  570000  590000  660000
  720000  327680  740000  800000  760000  690000  650000  780000  730000]
--------------------------------------------------
Kolom 'SEX' memiliki 2 nilai unik:
[2 1]
--------------------------------------------------
Kolom 'EDUCATION' memiliki 7 nilai unik:
[2 1 3 5 4 6 0]
--------------------------------------------------
Kolom 'MARRIAGE' memiliki 4 nilai unik:
[1 2 3 0]
--------------------------------------------------
Kolom 'AGE' memiliki 56 nilai unik:
[24 26 34 37 57 29 23 28 35 51 41 30 49 39 40 27 47 33 32 54 58 22 25 31
 46 42 43 45 56 44 53 38 63 36 52 48 55 60 50 75 61 73 59 21 67 66 62 70
 72 64 65 71 69 68 79 74]
--------------------------------------------------
Kolom 'PAY_0' memiliki 11 nilai unik:
[ 2 -1  0 -2  1  3  4  8  7  5  6]
--------------------------------------------------
Kolom 'PAY_2' memiliki 11 nilai unik:
[ 2  0 -1 -2  3  5  7  4  1  6  8]
--------------------------------------------------
Kolom 'PAY_3' memiliki 11 nilai unik:
[-1  0  2 -2  3  4  6  7  1  5  8]
--------------------------------------------------
Kolom 'PAY_4' memiliki 11 nilai unik:
[-1  0 -2  2  3  4  5  7  6  1  8]
--------------------------------------------------
Kolom 'PAY_5' memiliki 10 nilai unik:
[-2  0 -1  2  3  5  4  7  8  6]
--------------------------------------------------
Kolom 'PAY_6' memiliki 10 nilai unik:
[-2  2  0 -1  3  6  4  7  8  5]
--------------------------------------------------
Kolom 'BILL_AMT1' memiliki 22723 nilai unik:
[ 3913  2682 29239 ...  1683 -1645 47929]
--------------------------------------------------
Kolom 'BILL_AMT2' memiliki 22346 nilai unik:
[ 3102  1725 14027 ...  3356 78379 48905]
--------------------------------------------------
Kolom 'BILL_AMT3' memiliki 22026 nilai unik:
```

```
[  689  2682 13559 ...  2758 76304 49764]
--------------------------------------------------
Kolom 'BILL_AMT4' memiliki 21548 nilai unik:
[    0  3272 14331 ... 20878 52774 36535]
--------------------------------------------------
Kolom 'BILL_AMT5' memiliki 21010 nilai unik:
[    0  3455 14948 ... 21237  5190 32428]
```

Handling Data Duplicates

```
# Atasi duplikat data
df.drop_duplicates(inplace=True)
```

```
# Cek duplikasi data
df.duplicated().sum()
```

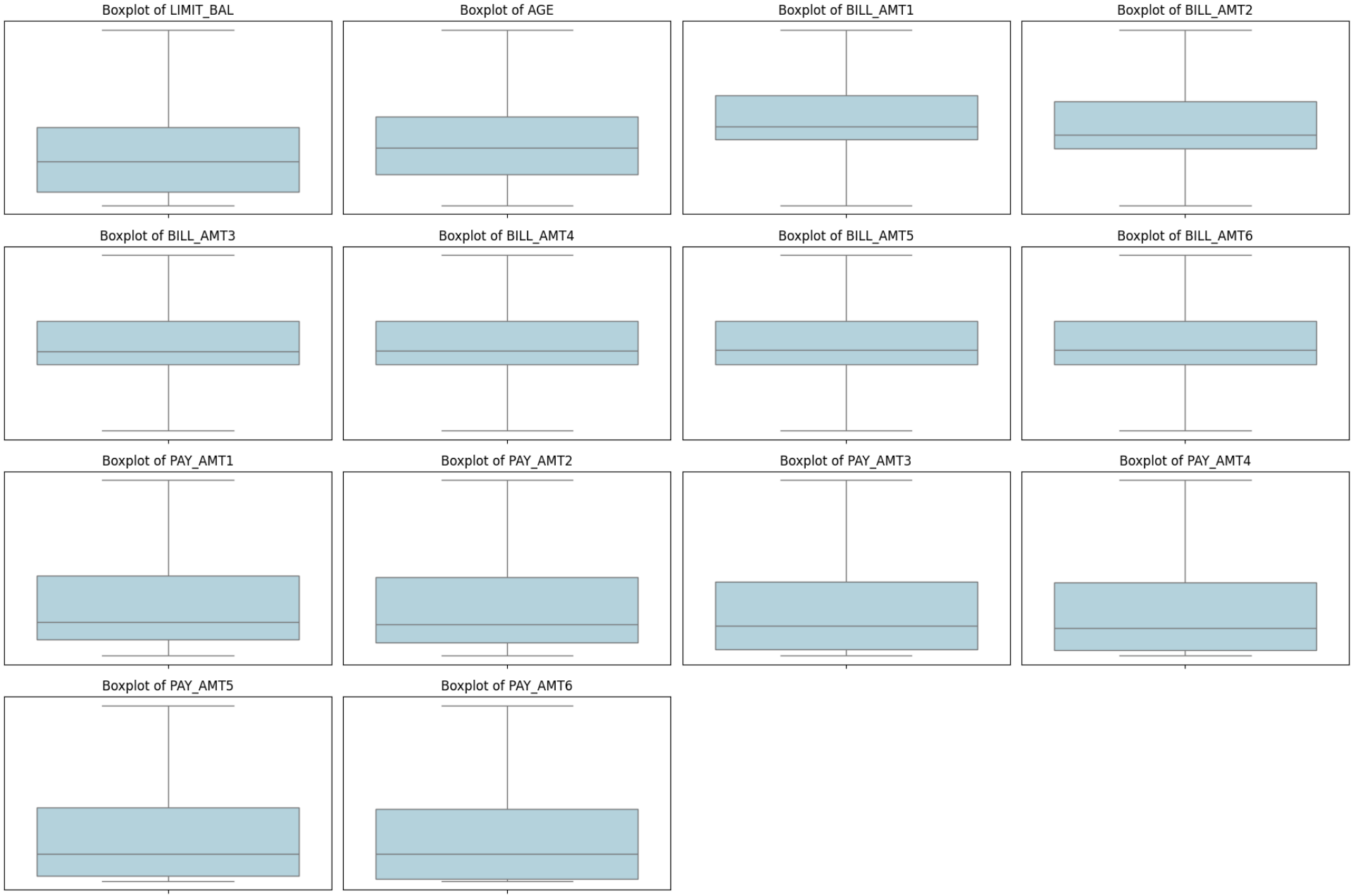
```
np.int64(0)
```

Handling Outliers

```
# Atasi outlier

for col in numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Terapkan capping langsung di df
    df[col] = df[col].clip(lower=lower_bound, upper=upper_bound)
```

```
# Visualisasi outlier

# Pastikan kolom numerik dalam tipe float
numerical_cols = [
    'LIMIT_BAL', 'AGE',
```

```
        'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3',
        'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6',
        'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3',
        'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6'
    ]
    # Pastikan kolom numerik bertipe float
    df[numerical_cols] = df[numerical_cols].apply(pd.to_numeric, errors='coerce')

    # Hitung dan simpan jumlah outlier
    outlier_counts = {}

    # Visualisasi boxplot dan hitung outlier berdasarkan IQR
    plt.figure(figsize=(18, 12))
    for i, col in enumerate(numerical_cols):
        plt.subplot(4, 4, i + 1)
        plt.yticks([])
        sns.boxplot(y=df[col], color='lightblue')
        plt.title(f'Boxplot of {col}')

        # Hitung IQR dan outlier
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]

        outlier_counts[col] = len(outliers)

    plt.tight_layout()
    plt.show()

    # Tampilkan jumlah outlier
    print("Jumlah outlier tiap kolom:")
    for col, count in outlier_counts.items():
        print(f"{col}: {count}")
```




```
Jumlah outlier tiap kolom:
LIMIT_BAL: 0
AGE: 0
BILL_AMT1: 0
BILL_AMT2: 0
```

```
BILL_AMT2: 0
BILL_AMT3: 0
BILL_AMT4: 0
BILL_AMT5: 0
BILL_AMT6: 0
PAY_AMT1: 0
PAY_AMT2: 0
PAY_AMT3: 0
```

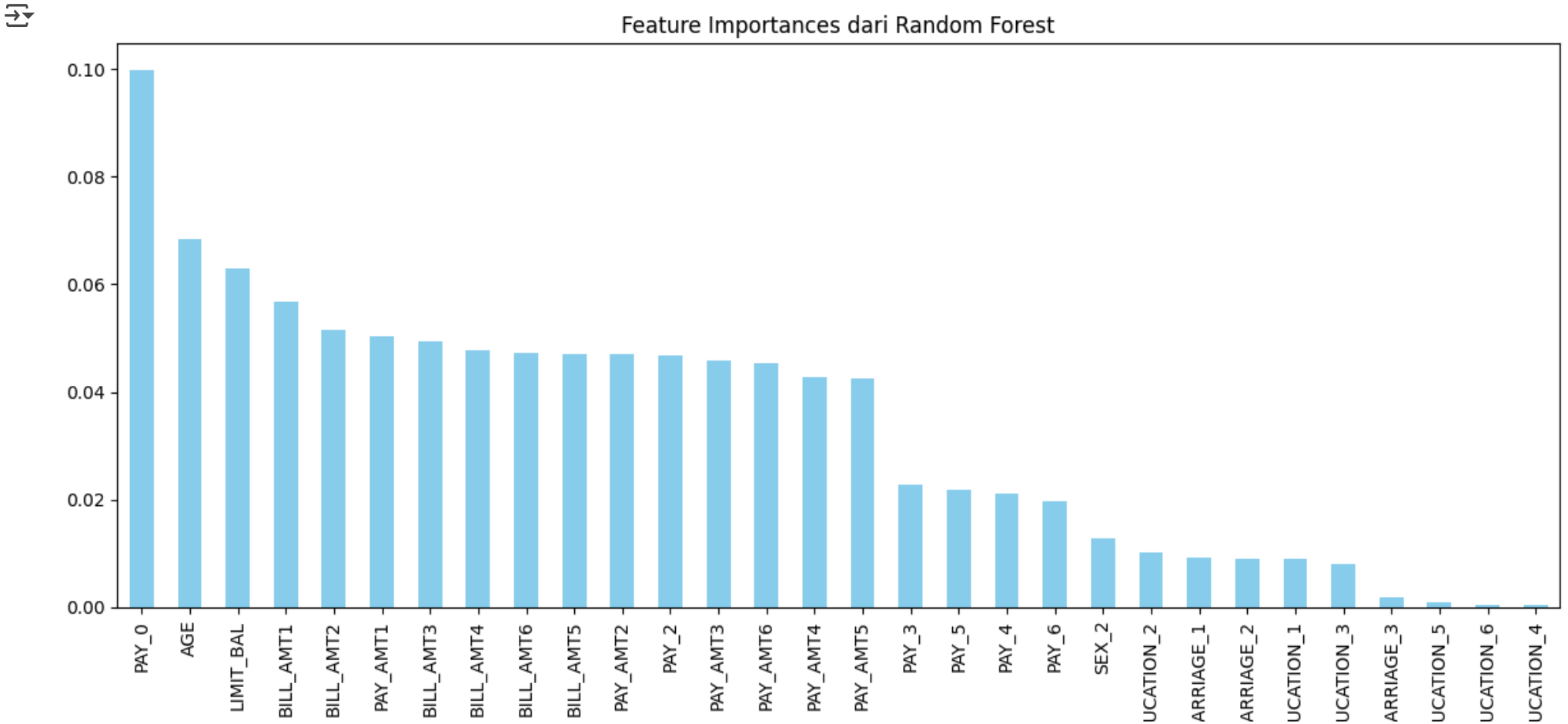
Convert Data to Ordinal - Nominal

```
# Fitur ordinal: LabelEncoder
ordinal_cols = ['PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']
le = LabelEncoder()

for col in ordinal_cols:
    df[col] = le.fit_transform(df[col])

# Fitur nominal: One Hot Encoding
nominal_cols = ['SEX', 'EDUCATION', 'MARRIAGE']
df = pd.get_dummies(df, columns=nominal_cols, drop_first=True)
```

Check Correlations

```
# Pisahkan fitur dan target
X = df.drop(columns=['Y'])
y = df['Y']

# Random Forest training
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X, y)

# Hitung feature importance
importances = pd.Series(rf.feature_importances_, index=X.columns).sort_values(ascending=False)

# Visualisasi Feature Importance
plt.figure(figsize=(12, 6))
importances.plot(kind='bar', color='skyblue')
plt.title('Feature Importances dari Random Forest')
plt.tight_layout()
plt.show()

# Threshold dan identifikasi grup fitur
```

```
threshold = 0.02
prefix_importance = defaultdict(list)

# Gabungkan OHE berdasarkan prefix-nya
for col in importances.index:
    prefix = col.split('_')[0]  # Ambil awalan kolom
    prefix_importance[prefix].append(importances[col])

# Identifikasi grup fitur OHE dengan importance rendah seluruhnya
drop_groups = [prefix for prefix, vals in prefix_importance.items() if all(val < threshold for val in vals)]

print("\nFitur kategorikal atau one-hot yang akan dihapus seluruhnya:", drop_groups)

# Drop kolom dari grup yang harus dihapus
cols_to_drop = [col for col in df.columns if any(col.startswith(prefix) for prefix in drop_groups)]
df.drop(columns=cols_to_drop, inplace=True)

print(f"\nTotal fitur yang dihapus: {len(cols_to_drop)}")
```




```
# Cek data akhir
df.head()
```

```
Total fitur yang dihapus: 10
```

	LIMIT_BAL	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PA
0	20000	24.0	4	4	1	1	0	0	3913.0	3102.0	...	0.0	0	0	0	689	
1	120000	26.0	1	4	2	2	2	3	2682.0	1725.0	...	3272.0	3455	3261	0	1000	
2	90000	34.0	2	2	2	2	2	2	29239.0	14027.0	...	14331.0	14948	15549	1518	1500	
3	50000	37.0	2	2	2	2	2	2	46990.0	48233.0	...	28314.0	28959	29547	2000	2019	
4	50000	57.0	1	2	1	2	2	2	8617.0	5670.0	...	20940.0	19146	19131	2000	11225	

5 rows × 21 columns

Balancing Target SMOTE

```
# Pisahkan fitur dan target
X = df.drop(columns=['Y'])
y = df['Y']

# SMOTE hanya untuk data numerik - tidak perlu encode target biner
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Visualisasi distribusi sebelum dan sesudah SMOTE
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Sebelum SMOTE
sns.countplot(x=y, ax=axes[0], hue=y, legend=False, palette="pastel")
axes[0].set_title('Distribusi Sebelum SMOTE')
axes[0].set_xlabel('Kelas')
axes[0].set_ylabel('Jumlah')

# Sesudah SMOTE
sns.countplot(x=y_resampled, ax=axes[1], hue=y_resampled, legend=False, palette="pastel"
axes[1].set_title('Distribusi Sesudah SMOTE')
axes[1].set_xlabel('Kelas')
axes[1].set_ylabel('Jumlah')
```