

Anggota satu | Baqhiz faruq S_5026231212

Eksplorasi Dataset dan Analisis Data Awal(EDA)

#Dataset Lending Club Loan Data

**accepted_2007_to_2018Q4.csv
rejected_2007_to_2018Q4.csv**

accepted														
	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	hardship_payoff_balance_amount	hardship_last_payment_amount	disbursement_method
0	68407277	NaN	3600.0	3600.0	3600.0	36 months	13.99	123.03	C	C4	...	NaN	NaN	Cash
1	68355089	NaN	24700.0	24700.0	24700.0	36 months	11.99	820.28	C	C1	...	NaN	NaN	Cash
2	68341763	NaN	20000.0	20000.0	20000.0	60 months	10.78	432.66	B	B4	...	NaN	NaN	Cash
3	66310712	NaN	35000.0	35000.0	35000.0	60 months	14.85	829.90	C	C5	...	NaN	NaN	Cash
4	68476807	NaN	10400.0	10400.0	10400.0	60 months	22.45	289.91	F	F1	...	NaN	NaN	Cash
...
1089195	69463147	NaN	6000.0	6000.0	6000.0	36 months	16.59	212.70	D	D2	...	NaN	NaN	Cash
1089196	69465059	NaN	8000.0	8000.0	8000.0	36 months	5.32	240.92	A	A1	...	NaN	NaN	Cash
1089197	70160376	NaN	30000.0	30000.0	30000.0	60 months	17.27	749.94	D	D3	...	NaN	NaN	Cash
1089198	70140418	NaN	15000.0	15000.0	15000.0	60 months	13.44	344.69	C	C3	...	NaN	NaN	Cash
1089199	69586499	NaN	6000.0	6000.0	6000.0	36 months	12.88	201.82	C	C2	...	NaN	NaN	NaN

Data accepted ini berisi informasi pinjaman yang mencakup total 2.260.698 baris dan 151 kolom. Beberapa kolom kunci termasuk loan_amnt (jumlah pinjaman), int_rate (suku bunga), installment (angsuran), grade, dan sub_grade (peringkat kredit).

Terlihat banyak missing values pada kolom seperti member_id, hardship_payoff_balance_amount, dan hardship_last_payment_amount, yang mungkin memerlukan penanganan khusus dalam EDA. Beberapa kolom berbentuk kategorikal dengan format string seperti term (jangka waktu pinjaman) dan grade.

```
[7] rejected = pd.read_csv(path_rejected)
```

	Amount Requested	Application Date	Loan Title	Risk_Score	Debt-To-Income Ratio	Zip Code	State	Employment Length	Policy Code
0	1000.0	2007-05-26	Wedding Covered but No Honeymoon	693.0	10%	481xx	NM	4 years	0.0
1	1000.0	2007-05-26	Consolidating Debt	703.0	10%	010xx	MA	< 1 year	0.0
2	11000.0	2007-05-27	Want to consolidate my debt	715.0	10%	212xx	MD	1 year	0.0
3	6000.0	2007-05-27	waksman	698.0	38.64%	017xx	MA	< 1 year	0.0
4	1500.0	2007-05-27	mdrigo	509.0	9.43%	209xx	MD	< 1 year	0.0
...
27648736	10000.0	2016-12-31	Debt consolidation	590.0	41.26%	441xx	OH	< 1 year	0.0
27648737	10000.0	2016-12-31	moving	NaN	1.48%	207xx	MD	5 years	0.0
27648738	1200.0	2016-12-31	Other	686.0	10.26%	914xx	CA	< 1 year	0.0
27648739	25000.0	2016-12-31	debt_consolidation	NaN	17.71%	880xx	NM	< 1 year	0.0
27648740	15000.0	2016-12-31	Business	684.0	10.58%	113xx	NY	< 1 year	0.0

27648741 rows x 9 columns

Data rejected ini berisi 27.648.741 baris dan 9 kolom yang merepresentasikan aplikasi pinjaman yang ditolak. Beberapa kolom kunci mencakup Amount Requested (jumlah pinjaman yang diajukan), Risk_Score (skor risiko), dan Debt-To-Income Ratio (rasio utang terhadap pendapatan).

Terdapat missing values pada kolom Risk_Score dan Employment Length, yang menunjukkan bahwa beberapa pelamar mungkin tidak melengkapi informasi dengan lengkap. Kolom seperti Zip Code dan State mengandung data geografis yang dapat dianalisis untuk melihat pola penolakan berdasarkan lokasi.

```
[19] def struktur_dataset(df, name):
    print(f"\nStruktur Dataset - {name}")
    print(df.info())
    print("\nMissing Values:\n", df.isnull().sum())
    print("\nSample Data:\n", df.head())
```

```
[20] struktur_dataset(accepted, "Accepted Loans")
```

Struktur Dataset - Accepted Loans

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2260701 entries, 0 to 2260700
Columns: 151 entries, id to settlement_term
dtypes: float64(113), object(38)
```

```
memory usage: 2.5+ GB
```

```
None
```

```
Missing Values:
```

```
    id                      0  
member_id                2260701  
loan_amnt                 33  
funded_amnt                33  
funded_amnt_inv                33  
...  
settlement_status            2226455  
settlement_date              2226455  
settlement_amount            2226455  
settlement_percentage        2226455  
settlement_term              2226455  
Length: 151, dtype: int64
```

```
Sample Data:
```

```
      id  member_id  loan_amnt  funded_amnt  funded_amnt_inv  
term \\  
0   68407277       NaN     3600.0     3600.0     3600.0    36  
months  
1   68355089       NaN    24700.0    24700.0    24700.0    36  
months  
2   68341763       NaN   20000.0   20000.0   20000.0    60  
months  
3   66310712       NaN   35000.0   35000.0   35000.0    60  
months  
4   68476807       NaN   10400.0   10400.0   10400.0    60  
months  
  
      int_rate  installment  grade  sub_grade  ...  
hardship_payoff_balance_amount \\
```

0	13.99	123.03	C	C4	...
NaN					
1	11.99	820.28	C	C1	...
NaN					
2	10.78	432.66	B	B4	...
NaN					
3	14.85	829.90	C	C5	...
NaN					
4	22.45	289.91	F	F1	...
NaN					
hardship_last_payment_amount			disbursement_method		
debt_settlement_flag \					
0		Nan			Cash
N					
1		Nan			Cash
N					
2		Nan			Cash
N					
3		Nan			Cash
N					
4		Nan			Cash
N					
debt_settlement_flag_date settlement_status settlement_date \					
0		Nan			Nan
1		Nan			Nan
2		Nan			Nan
3		Nan			Nan
4		Nan			Nan
settlement_amount settlement_percentage settlement_term					
0		Nan			Nan
1		Nan			Nan

```
2           NaN        NaN        NaN
3           NaN        NaN        NaN
4           NaN        NaN        NaN
```

[5 rows x 151 columns]

Kode tersebut adalah fungsi Python Bernama struktur_dataset yang bertujuan untuk menganalisis struktur DataFrame. Fungsi ini mencetak ringkasan data menggunakan df.info() untuk melihat tipe data dan jumlah non-null values, menghitung jumlah missing values dengan df.isnull().sum(), serta menampilkan lima baris pertama data menggunakan df.head(). Saat dipanggil dengan struktur_dataset(accepted, "Accepted Loans"), fungsi ini akan menampilkan analisis dasar dari dataset "Accepted Loans".

Dataset ini berisi informasi mengenai pinjaman yang disetujui dengan total 2.260.701 entri dan 151 kolom. dataset ini menyajikan data yang sangat komprehensif mengenai berbagai aspek pinjaman, mulai dari informasi dasar seperti jumlah pinjaman hingga detail khusus terkait penyelesaian dan kondisi kesulitan pembayaran.

Dataset ini dikategorikan sebagai objek pandas DataFrame, yang merupakan struktur data dua dimensi dengan label di baris dan kolom. Terdapat 151 kolom yang terdiri dari:

- 113 kolom dengan tipe float64, yang umumnya berisi data numerik seperti jumlah pinjaman, tingkat bunga, dan persentase pembayaran.
- 38 kolom dengan tipe object, yang berisi data kategorikal dan teks seperti metode pembayaran dan status penyelesaian.

```
[ ] struktur_dataset(rejected, "Rejected Loans")
```

Struktur Dataset - Rejected Loans

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27648741 entries, 0 to 27648740
Data columns (total 9 columns):
 #   Column            Dtype
```

```
--- -----  
0   amount_requested      float64  
1   application_date      object  
2   loan_title            object  
3   risk_score             float64  
4   debt-to-income_ratio  object  
5   zip_code               object  
6   state                  object  
7   employment_length      object  
8   policy_code            float64
```

dtypes: float64(3), object(6)

memory usage: 1.9+ GB

None

Missing Values:

amount_requested	0
application_date	0
loan_title	1305
risk_score	18497630
debt-to-income_ratio	0
zip_code	293
state	22
employment_length	951355
policy_code	918

dtype: int64

Sample Data:

loan_title \	amount_requested	application_date	
0 Honeymoon	1000.0	2007-05-26	Wedding Covered but No

1	1000.0	2007-05-26	Consolidating
2	11000.0	2007-05-27	Want to consolidate my debt
3		6000.0	2007-05-27
4		1500.0	2007-05-27
waksman			
mdrigo			

```

risk_score debt-to-income_ratio zip_code state employment_length
\
0      693.0          10%    481xx   NM      4 years
1      703.0          10%    010xx   MA     < 1 year
2      715.0          10%    212xx   MD      1 year
3      698.0          38.64%  017xx   MA     < 1 year
4      509.0          9.43%   209xx   MD     < 1 year
policy_code
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0

```

Dengan menggunakan function struktur_data pada data rejected, kita dapatkan Dataset ini berisi data historis tentang aplikasi pinjaman yang ditolak dengan total 27.648.741 entri dan 9 kolom. Data ini menyimpan berbagai informasi penting yang mencakup jumlah yang diminta, skor risiko, tingkat utang terhadap pendapatan, dan beberapa atribut demografis peminjam. dataset ini memberikan wawasan yang sangat berguna untuk memahami karakteristik peminjam yang tidak memenuhi kriteria pinjaman, yang dapat digunakan dalam analisis risiko kredit atau peningkatan kebijakan pemberian pinjaman.

Dataset ini di klasifikasikan sebagai objek pandas DataFrame dengan 9 kolom, yang mencakup berbagai tipe data:d

- 3 kolom bertipe float64, yang umumnya berisi nilai numerik seperti jumlah yang diminta (Amount Requested) dan skor risiko (Risk_Score).
- 6 kolom bertipe object, yang merepresentasikan data kategorikal dan teks seperti tanggal aplikasi, judul pinjaman, dan lama masa kerja peminjam.

```

def analisis_deskriptif(df, name):
    print(f"\nAnalisis Deskriptif - {name}")
    print("\nStatistik Numerik:\n", df.describe().T)
    print("\nStatistik Kategorikal:\n", df.select_dtypes(include='object').describe().T)

analisis_deskriptif(accepted, "Accepted Loans")

```

Analisis Deskriptif - Accepted Loans

Statistik Numerik:

		count	mean	std
min \				
member_id		0.0	NaN	NaN
NaN				
loan_amnt	2260668.0	15046.931228	9190.245488	
500.00				
funded_amnt	2260668.0	15041.664057	9188.413022	
500.00				
funded_amnt_inv	2260668.0	15023.437745	9192.331679	
0.00				
int_rate	2260668.0	13.092829	4.832138	
5.31				
...
...				
hardship_payoff_balance_amount	10917.0	11636.883942	7625.988281	
55.73				
hardship_last_payment_amount	10917.0	193.994321	198.629496	
0.01				
settlement_amount	34246.0	5010.664267	3693.122590	
44.21				
settlement_percentage	34246.0	47.780365	7.311822	
0.20				
settlement_term	34246.0	13.191322	8.159980	
0.00				
	25%	50%	75%	max
member_id	NaN	NaN	NaN	NaN

loan_amnt	8000.00	12900.00	20000.0000	40000.00
funded_amnt	8000.00	12875.00	20000.0000	40000.00
funded_amnt_inv	8000.00	12800.00	20000.0000	40000.00
int_rate	9.49	12.62	15.9900	30.99
...
hardship_payoff_balance_amount	5627.00	10028.39	16151.8900	40306.41
hardship_last_payment_amount	44.44	133.16	284.1900	1407.86
settlement_amount	2208.00	4146.11	6850.1725	33601.00
settlement_percentage	45.00	45.00	50.0000	521.35
settlement_term	6.00	14.00	18.0000	181.00

[113 rows x 8 columns]

Statistik Kategorikal:

	count	unique	\
id	2260701	2260701	
term	2260668	2	
grade	2260668	7	
sub_grade	2260668	35	
emp_title	2093699	512694	
emp_length	2113761	11	
home_ownership	2260668	6	
verification_status	2260668	3	
issue_d	2260668	139	
loan_status	2260668	9	
pymnt_plan	2260668	2	
url	2260668	2260668	
desc	126065	124500	
purpose	2260668	14	
title	2237342	63154	
zip_code	2260667	956	
addr_state	2260668	51	
earliest_cr_line	2260639	754	
initial_list_status	2260668	2	
last_pymnt_d	2258241	136	

next_pymnt_d	915358	106
last_credit_pull_d	2260596	141
application_type	2260668	2
verification_status_joint	115730	3
sec_app_earliest_cr_line	108021	663
hardship_flag	2260668	2
hardship_type	10917	1
hardship_reason	10917	9
hardship_status	10917	3
hardship_start_date	10917	27
hardship_end_date	10917	28
payment_plan_start_date	10917	27
hardship_loan_status	10917	5
disbursement_method	2260668	2
debt_settlement_flag	2260668	2
debt_settlement_flag_date	34246	83
settlement_status	34246	3
settlement_date	34246	90
top \		
id		Total amount funded in policy code 2:
521953170		
term		36
months		
grade		
B		
sub_grade		
C1		
emp_title		
Teacher		
emp_length		10+
years		
home_ownership		
MORTGAGE		
verification_status		Source
Verified		
issue_d		

Mar-2016

loan_status Fully
Paid

pymnt_plan n

url <https://lendingclub.com/browse/loanDetail.acti...>

desc

purpose debt_consolidation

title consolidation Debt

zip_code 112xx

addr_state CA

earliest_cr_line Sep-2004

initial_list_status w

last_pymnt_d Mar-2019

next_pymnt_d Apr-2019

last_credit_pull_d Mar-2019

application_type Individual

verification_status_joint Verified Not

sec_app_earliest_cr_line Aug-2006

hardship_flag N

hardship_type DEFERRAL INTEREST ONLY-3 MONTHS

hardship_reason NATURAL_DISASTER

hardship_status COMPLETED

hardship_start_date Sep-2017

hardship_end_date	
Dec-2017	
payment_plan_start_date	
Sep-2017	
hardship_loan_status	Late (16-30 days)
disbursement_method	
Cash	
debt_settlement_flag	
N	
debt_settlement_flag_date	
Feb-2019	
settlement_status	
ACTIVE	
settlement_date	
Jan-2019	

	freq
id	1
term	1609754
grade	663557
sub_grade	145903
emp_title	38824
emp_length	748005
home_ownership	1111450
verification_status	886231
issue_d	61992
loan_status	1076751
pymnt_plan	2260048
url	1
desc	252
purpose	1277877
title	1153293
zip_code	23908
addr_state	314533
earliest_cr_line	15400
initial_list_status	1535467
last_pymnt_d	853003

next_pymnt_d	912221
last_credit_pull_d	1371381
application_type	2139958
verification_status_joint	57403
sec_app_earliest_cr_line	998
hardship_flag	2259836
hardship_type	10917
hardship_reason	2965
hardship_status	7819
hardship_start_date	2444
hardship_end_date	1756
payment_plan_start_date	1715
hardship_loan_status	4770
disbursement_method	2182546
debt_settlement_flag	2226422
debt_settlement_flag_date	2606
settlement_status	14704
settlement_date	1710

Fungsi analisis_deskriptif bertujuan memberikan ringkasan statistik dari dataset. Kode ini mencetak judul dengan nama dataset yang dianalisis. Bagian df.describe().T menampilkan statistik deskriptif (mean, median, std, dll.) untuk data numerik, sementara df.select_dtypes(include='object').describe().T menampilkan statistik kategorikal (seperti jumlah unik dan frekuensi terbanyak). Transpose (.T) digunakan agar hasil lebih mudah dibaca dalam format vertikal.

1. Statistik Numerik

Dataset ini terdiri dari 113 kolom dengan banyaknya data keuangan yang signifikan. Berikut adalah beberapa poin penting berdasarkan ringkasan statistik numerik:

Kolom Utama	Mean	Std Dev	Min	Median(50%)	Max
loan_amnt	\$15,046.93	\$9,190.25	\$500	\$12,900	\$40,000
funded_amnt	\$15,041.66	\$9,188.41	\$500	\$12,875	\$40,000
int_rate	13.09%	4.83%	5.31	12.62%	30.99%
hardship_payoff_balance_amount	\$11,636.88	\$7,625.98	\$55.73	\$10,028.39	\$40,306.41
settlement_amount	\$5,010.66	\$3,693.12	\$44.21	\$4,146.11	\$33,601
settlement_percentage	47.78%	7.31	0.20%	45.00%	521.35%

Jumlah Pinjaman: Median pinjaman yang disetujui sebesar \$12.9K, tetapi rata-rata 15K menunjukkan adanya beberapa pinjaman besar yang menaikkan rata-rata.

Tingkat Bunga: Median tingkat bunga 12.62%, dengan maksimum hingga 30.99%, menandakan beberapa peminjam berisiko tinggi.

Settlement Amount & Percentage:

- **Settlement Percentage** (persentase penyelesaian utang) cukup tinggi pada median 45%, tetapi terdapat beberapa kasus ekstrem dengan persentase lebih dari 500%, yang mungkin menunjukkan pelunasan yang tidak lazim atau restrukturisasi utang.

2. Statistik Kategorikal

Beberapa kolom kategorikal kunci menampilkan distribusi unik dan frekuensi tertinggi:

Kolom	Unique	Top Value	Freq
term	2	36 months	1.6M
grade	7	B	663K
home_ownership	6	MORTGAGE	1.1M
loan_status	9	Fully Paid	1.07M
purpose	14	debt_consolidation	1.27M
addr_state	51	CA (California)	314K
emp_length	11	10+years	784K
application_type	2	Individual	2.13M

Durasi Pinjaman (term): Mayoritas pinjaman memiliki jangka waktu 36 bulan, yang mencerminkan preferensi untuk jangka menengah.

Grade Kredit: Grade B dan C mendominasi, yang sesuai dengan profil peminjam berisiko menengah.

Kepemilikan Rumah: Mortgage adalah status paling umum, menunjukkan bahwa mayoritas peminjam memiliki kewajiban finansial lain.

Tujuan Pinjaman: Debt consolidation (konsolidasi utang) adalah alasan utama untuk pinjaman, dengan 1.27M kasus.

Lokasi: California (CA) adalah negara bagian dengan aplikasi terbanyak, mungkin karena ukuran populasi dan aktivitas ekonomi.

Lama Bekerja: Mayoritas peminjam memiliki pengalaman kerja 10+ tahun, menandakan stabilitas pekerjaan yang baik.

```
[111] analisis_deskriptif(rejected, "Rejected Loans")
```

Analisis Deskriptif - Rejected Loans

Statistik Numerik:

	count	mean	std	min	25%
\					
amount_requested	27648741.0	13133.240309	15009.636324	0.0	4800.0
risk_score	9151111.0	628.172090	89.936793	0.0	591.0
policy_code	27647823.0	0.006375	0.112737	0.0	0.0
	50%	75%	max		
amount_requested	10000.0	20000.0	1400000.0		
risk_score	637.0	675.0	990.0		
policy_code	0.0	0.0	2.0		

Statistik Kategorikal:

	count	unique	top	freq
application_date	27648741	4238	2018-12-04	42112
loan_title	27647436	73927	Debt consolidation	6418016
debt-to-income_ratio	27648741	126145		100%
zip_code	27648448	1001	112xx	267102
state	27648719	51	CA	3242169
employment_length	26697386	11	< 1 year	22994315

1. Statistik Numerik

Dataset ini mencakup atribut numerik utama yang memberikan gambaran jumlah pinjaman yang diminta dan risiko peminjam.

Kolom	Mean	Stv Dev	Min	Median (50%)	Max
Amount Requested	\$13,133.24	\$15,009.64	\$0.00	"\$"10,000.00	"\$"1,400,000.00
Risk_Score	628.17	89.93	0.00	637.00	990.00
Policy Code	0.0064	0.1127	0.00	0.00	2.00

Amount Requested:

- Median pinjaman sebesar \$10K, sementara rata-rata 13.1K, menunjukkan adanya beberapa permintaan pinjaman besar yang mendorong rata-rata naik.
- \$1.4M sebagai nilai maksimum mungkin merupakan outlier yang signifikan atau salah input.

Risk Score:

- Median 637 mendekati rata-rata 628, yang berarti distribusi risiko relatif normal.
- Skor 0 menunjukkan bahwa ada data yang mungkin kosong atau tidak valid.

Policy Code:

Median 0 menunjukkan sebagian besar data mengikuti kebijakan dasar, dengan beberapa kasus langka dengan kode 1 atau 2.

2. Statistik Kategorikal

Beberapa atribut kategorikal kunci menunjukkan distribusi unik yang relevan dengan profil peminjam dan alasan penolakan.

Kolom	Unique	Top Value	Freq
Application Date	4238	2018-12-04	42,112
Loan Title	72,927	Debt consolidation	6.41M
Debt-To-Income Ratio	126,145	100%	1.36M
Zip Code	1001	112xx	267,102
State	51	CA(California)	3.24M
Employment Length	11	< 1 year	22.99M

Loan Title:

Debt consolidation (Konsolidasi Utang) mendominasi dengan 6.41M aplikasi yang ditolak, menunjukkan tingginya permintaan untuk melunasi utang lain.

Debt-To-Income Ratio (DTI):

100% sebagai top value menunjukkan banyak peminjam yang seluruh pendapatannya sudah digunakan untuk membayar utang, menandakan risiko keuangan yang tinggi.

State:

California (CA) memiliki jumlah aplikasi tertinggi yang ditolak, dengan 3.24M kasus, konsisten dengan populasi dan aktivitas ekonomi yang besar.

Employment Length:

Sebagian besar peminjam memiliki pengalaman kerja < 1 tahun, menandakan stabilitas pekerjaan rendah sebagai salah satu alasan penolakan.

```
[112] missing_values = accepted.isnull().sum()
      missing_values = missing_values[missing_values > 0].sort_values(ascending=False)

      plt.figure(figsize=(14, 7))
      bars = plt.bar(missing_values.index, missing_values.values, color='skyblue')

      plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _: f'{int(x)}')))

      plt.title("Jumlah Missing Values per Kolom-Accepted Loans")
      plt.xlabel("Nama Kolom")
      plt.ylabel("Jumlah Missing Values")
      plt.xticks(rotation=75, ha='right', fontsize=8)
      plt.tight_layout()
      plt.show()
```

```
[ ] missing_values = accepted.isnull().sum()
missing_values = missing_values[missing_values > 0]
missing_values = missing_values.sort_values(ascending=False)

top_n = 50

if len(missing_values) > top_n:
    missing_values = missing_values.head(top_n)

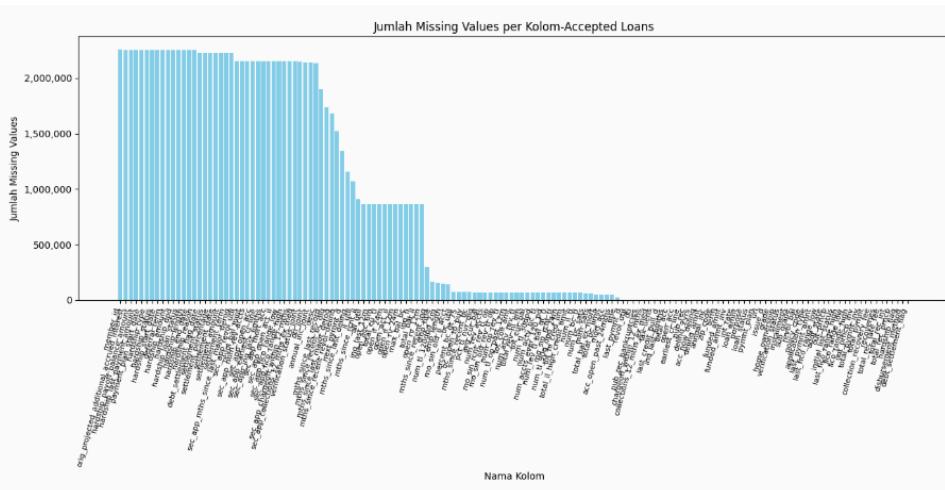
plt.figure(figsize=(14, 8))
bars = plt.bar(missing_values.index, missing_values.values, color='skyblue')

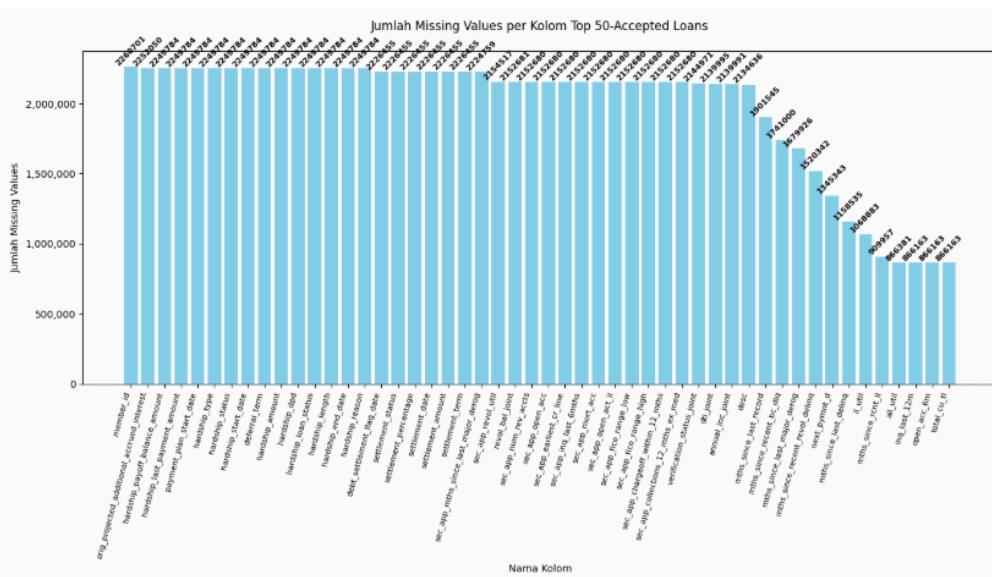
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2.0, yval + 0.5, int(yval),
             ha='center', va='bottom', fontsize=8, fontweight='bold', rotation=45)

plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _: f'{int(x)}')))

plt.title(f"Jumlah Missing Values per Kolom Top {top_n}-Accepted Loans\n")
plt.xlabel("Nama Kolom")
plt.ylabel("Jumlah Missing Values")

plt.xticks(rotation=75, ha='right', fontsize=8)
plt.tight_layout()
plt.show()
```





Kode di atas bertujuan untuk **menampilkan jumlah missing values (nilai yang hilang)** pada setiap kolom di DataFrame **accepted** dalam bentuk **bar chart**. Berikut penjelasannya:

1. **missing_values = accepted.isnull().sum()**
 - Mengecek jumlah nilai **Nan** (missing values) di setiap kolom DataFrame **accepted**.
2. **missing_values = missing_values[missing_values > 0].sort_values(ascending=False)**
 - Menyaring hanya kolom yang memiliki **missing values** (> 0).
 - Mengurutkan hasilnya dari jumlah terbesar ke terkecil.
3. **plt.figure(figsize=(14, 7))**
 - Mengatur ukuran plot menjadi **14x7** inci.
4. **bars = plt.bar(missing_values.index, missing_values.values, color='skyblue')**
 - Membuat **bar chart** dengan sumbu x adalah **nama kolom**, dan sumbu y adalah **jumlah missing values**.
 - Warna batang diatur menjadi **skyblue**.
5. **plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _: f'{int(x)}:,}))**

Menggunakan **fungsi formatter** untuk menampilkan angka dalam format **ribuan dengan koma** (contoh: 10,000).

6. **plt.title("Jumlah Missing Values per Kolom-Accepted Loans")**
 - Memberikan judul pada plot.
7. **plt.xlabel("Nama Kolom")** dan **plt.ylabel("Jumlah Missing Values")**
 - Memberi label pada sumbu x dan y.
8. **plt.xticks(rotation=75, ha='right', fontsize=8)**

- Memutar label pada sumbu x sebanyak **75 derajat** untuk menghindari tumpang tindih.
 - **ha='right'** memastikan teks rata kanan.
 - **fontsize=8** mengecilkan ukuran teks agar lebih muat.
9. **plt.tight_layout()**
- Memastikan semua elemen grafik tidak tumpang tindih.
10. **plt.show()**
- Menampilkan plot.

Visualisasi di atas menunjukkan distribusi missing values dalam dataset Accepted Loans (sample). Setiap kolom pada sumbu horizontal merepresentasikan atribut atau variabel dalam dataset, sementara setiap titik vertikal pada sumbu vertikal mewakili observasi (baris) individu. Warna hitam menandakan data yang tersedia (non-missing), sedangkan warna terang (beige) menunjukkan nilai yang hilang (missing values).

Terdapat **jumlah missing values yang signifikan** pada banyak kolom dalam dataset accepted. Beberapa kolom memiliki **lebih dari 2 juta nilai yang hilang**, sementara kolom lain memiliki **ratusan ribu hingga jutaan** missing values. Hal ini menunjukkan perlunya **tindakan data cleaning**, baik melalui **imputasi** atau **penghapusan kolom**, tergantung pada persentase dan pentingnya kolom tersebut dalam analisis atau model prediksi.

```

❷ accepted['loan_status'] = accepted['loan_status'].str.wrap(15)
status_counts = accepted['loan_status'].value_counts()
total = status_counts.sum()
percentages = (status_counts / total) * 100

plt.figure(figsize=(14, 7))
sns.barplot(x=status_counts.index, y=percentages, palette='viridis', hue=status_counts.index, legend=False)

for i, percent in enumerate(percentages):
    plt.text(i, percent + 0.5, f"{percent:.2f}%", ha='center', fontsize=10, fontweight='bold')

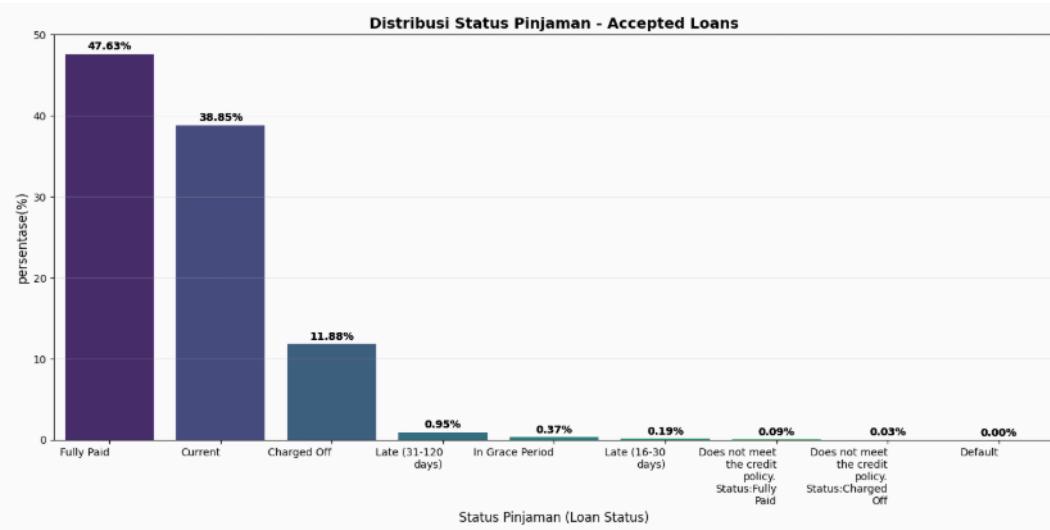
plt.title('Distribusi Status Pinjaman - Accepted Loans', fontsize=14, fontweight='bold')
plt.xlabel('Status Pinjaman (Loan Status)', fontsize=12)
plt.ylabel('persentase Peminjam', fontsize=12)

for i, percent in enumerate(percentages):
    plt.text(i, percent + 0.5, f"{percent:.2f}%", ha='center', fontsize=10, fontweight='bold')

plt.title('Distribusi Status Pinjaman - Accepted Loans', fontsize=14, fontweight='bold')
plt.xlabel('Status Pinjaman (Loan Status)', fontsize=12)
plt.ylabel('persentase(%)')
plt.xticks(ha='right')
plt.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.show()

```



Kode ini menganalisis dan memvisualisasikan **distribusi status pinjaman** dalam dataset **accepted**. Setiap kategori dihitung dan dikonversi menjadi **persentase**, lalu divisualisasikan menggunakan **barplot** dengan palet **viridis**. **Label persentase** ditambahkan di atas setiap batang untuk memperjelas proporsi masing-masing kategori. Grafik ini membantu mengidentifikasi **kategori dominan**, seperti **Fully Paid** atau **Charged Off**, serta mengevaluasi **kesehatan portofolio** dan **risiko kredit**.

Visualisasi di atas menampilkan distribusi status pinjaman dalam dataset Accepted Loans. Kategori Fully Paid dan Current mendominasi

data, dengan jumlah masing-masing melebihi ratusan ribu hingga jutaan kasus. Ini menunjukkan bahwa sebagian besar peminjam berhasil melunasi pinjaman mereka atau sedang aktif dalam proses pembayaran.

Charged Off (pinjaman yang dianggap tidak tertagih) memiliki jumlah yang jauh lebih kecil, tetapi tetap signifikan karena merepresentasikan kerugian bagi pemberi pinjaman. In Grace Period (masa tenggang) dan kategori keterlambatan seperti Late (31-120 days) dan Late (16-30 days) menunjukkan bahwa sebagian kecil peminjam mengalami kesulitan pembayaran, meski mungkin masih bisa pulih.

```
[ ] feature = 'int_rate'

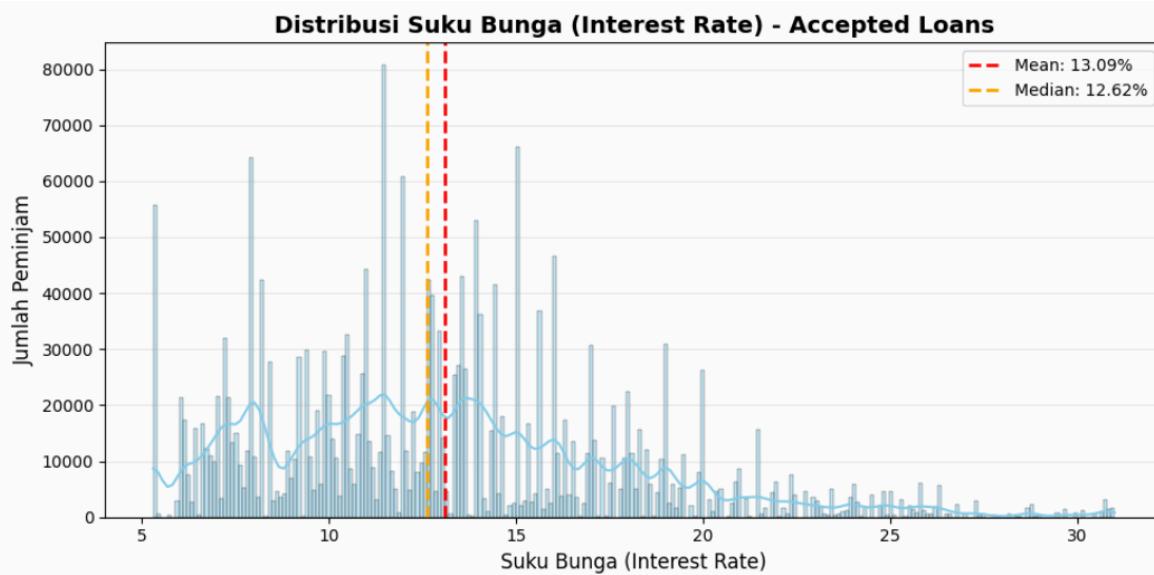
plt.figure(figsize=(10, 5))
sns.histplot(accepted[feature], kde=True, color='skyblue')

mean_val = accepted[feature].mean()
median_val = accepted[feature].median()

plt.axvline(mean_val, color='red', linestyle='--', linewidth=2, label=f'Mean: {mean_val:.2f}%')
plt.axvline(median_val, color='orange', linestyle='--', linewidth=2, label=f'Median: {median_val:.2f}%')

plt.title('Distribusi Suku Bunga (Interest Rate) - Accepted Loans', fontsize=14, fontweight='bold')
plt.xlabel('Suku Bunga (Interest Rate)', fontsize=12)
plt.ylabel('Jumlah Peminjam', fontsize=12)

plt.grid(axis='y', alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()
```



Kode ini bertujuan untuk memvisualisasikan **distribusi suku bunga (interest rate)** dalam dataset **accepted** menggunakan **histogram** dengan

Kernel Density Estimate (KDE) untuk menunjukkan pola sebaran data. Histogram dibuat menggunakan `sns.histplot()` dengan warna `skyblue`. Dua garis vertikal ditambahkan pada grafik untuk menandai `mean` (rata-rata) dan `median` (nilai tengah) dari suku bunga. Garis merah putus-putus mewakili `mean`, sementara garis oranye putus-putus mewakili `median`, dengan masing-masing label menampilkan nilai yang tepat. Grafik ini membantu mengidentifikasi `skewness` (kemiringan) data, apakah suku bunga cenderung `simetris`, `positif skewed` (lebih banyak nilai rendah), atau `negatif skewed` (lebih banyak nilai tinggi). Informasi ini penting untuk **analisis risiko, penetapan harga pinjaman, dan strategi keuangan**.

Visualisasi di atas menampilkan distribusi suku bunga (`int_rate`) pada pinjaman yang disetujui (Accepted Loans). Rentang suku bunga berkisar dari 5% hingga sekitar 30%, dengan konsentrasi terbesar berada pada kisaran 10% hingga 15%. Hal ini menunjukkan bahwa mayoritas pinjaman yang disetujui diberikan dengan suku bunga menengah, mencerminkan tingkat risiko yang dianggap wajar oleh pemberi pinjaman.

Terdapat pola penurunan jumlah pinjaman seiring dengan meningkatnya suku bunga. Suku bunga di atas 20% jauh lebih jarang, yang mungkin mengindikasikan tingginya risiko gagal bayar atau rendahnya minat peminjam pada tingkat bunga yang tinggi.

Plot kepadatan kernel (KDE) yang menyertainya menunjukkan puncak yang jelas pada kisaran 10% hingga 15%, menandakan bahwa kisaran ini merupakan titik paling umum dalam distribusi. Informasi ini berguna dalam mengevaluasi profil risiko dan menentukan strategi harga yang optimal bagi pemberi pinjaman.

```

feature = 'installment'

plt.figure(figsize=(10, 5))
sns.histplot(accepted[feature], kde=True, color='skyblue')

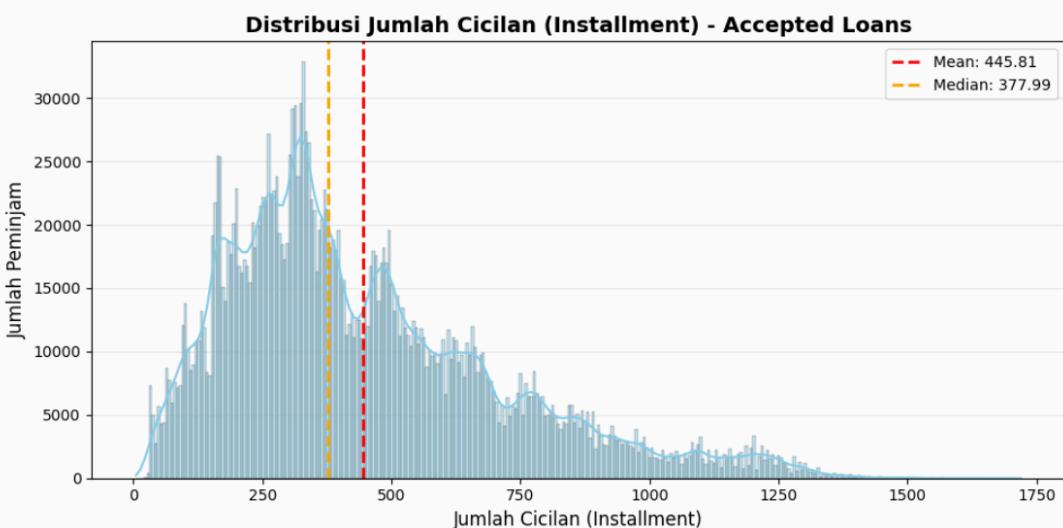
mean_val = accepted[feature].mean()
median_val = accepted[feature].median()

plt.axvline(mean_val, color='red', linestyle='--', linewidth=2, label=f'Mean: {mean_val:.2f}')
plt.axvline(median_val, color='orange', linestyle='--', linewidth=2, label=f'Median: {median_val:.2f}')

plt.title('Distribusi Jumlah Cicilan (Installment) - Accepted Loans', fontsize=14, fontweight='bold')
plt.xlabel('Jumlah Cicilan (Installment)', fontsize=12)
plt.ylabel('Jumlah Peminjam', fontsize=12)

plt.grid(axis='y', alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()

```



visualisasikan **distribusi jumlah cicilan (installment)** pada dataset **accepted** menggunakan histogram dengan **Kernel Density Estimate (KDE)**. Histogram dibuat dengan `sns.histplot()`, berwarna `skyblue`, dan ukuran grafik diatur menjadi **10x5**. Dua garis vertikal ditambahkan untuk menampilkan **mean** (rata-rata) dan **median** (nilai tengah) dari jumlah cicilan. Garis merah putus-putus menunjukkan **mean**, sementara garis oranye putus-putus menunjukkan **median**, masing-masing disertai label nilai yang ditampilkan hingga dua desimal.

Judul grafik memberikan konteks tentang analisis yang dilakukan, dengan label pada sumbu x menunjukkan **jumlah cicilan** dan sumbu y menunjukkan **jumlah peminjam**. Grid horizontal ditambahkan dengan transparansi **0.3** untuk membantu membaca nilai pada sumbu y. Akhirnya, `plt.tight_layout()` memastikan tata letak grafik lebih rapi, dan

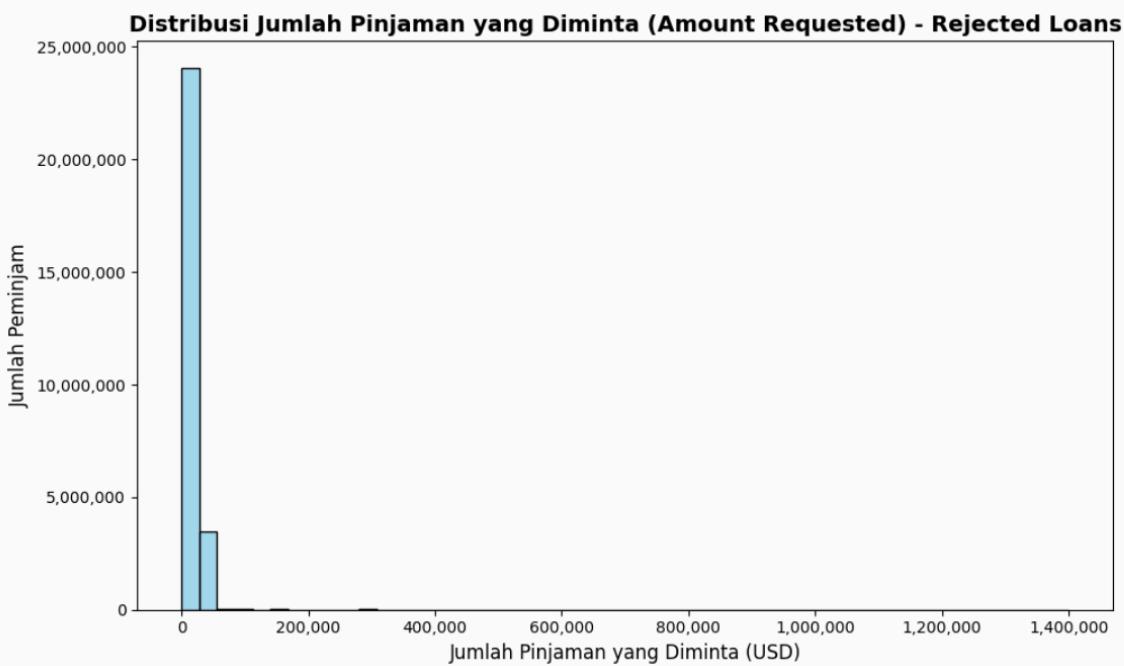
`plt.show()` menampilkan visualisasi akhir. Grafik ini membantu memahami bagaimana jumlah cicilan didistribusikan di antara para peminjam dan mengidentifikasi apakah distribusinya **normal**, **skewed**, atau memiliki anomali tertentu.

Grafik di atas menunjukkan distribusi jumlah cicilan (installment) pada pinjaman yang disetujui (Accepted Loans). Pola distribusi cenderung asimetris dengan puncak tertinggi berada pada kisaran 200 hingga 400. Hal ini menunjukkan bahwa mayoritas peminjam membayar cicilan bulanan dalam rentang tersebut.

Terdapat kecenderungan penurunan jumlah peminjam seiring dengan meningkatnya besaran cicilan, di mana hanya sedikit peminjam yang memiliki cicilan bulanan di atas 1000. Ini mungkin mengindikasikan bahwa pinjaman dengan cicilan tinggi diberikan kepada sejumlah kecil peminjam yang mungkin memiliki profil keuangan yang lebih kuat.

```
plt.figure(figsize=(10, 6))
sns.histplot(rejected['amount_requested'], bins=50, kde=False, color='skyblue')

plt.title('Distribusi Jumlah Pinjaman yang Diminta (Amount Requested) - Rejected Loans', fontsize=14, fontweight='bold')
plt.xlabel('Jumlah Pinjaman yang Diminta (USD)', fontsize=12)
plt.ylabel('Jumlah Peminjam', fontsize=12)
plt.gca().xaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _: f'{int(x)}'))
plt.gca().yaxis.set_major_formatter(mtick.FuncFormatter(lambda x, _: f'{int(x)}'))
plt.gca().yaxis.get_major_locator().set_params(integer=True)
plt.tight_layout()
plt.show()
```



Kode ini membuat histogram untuk memvisualisasikan distribusi **jumlah pinjaman yang diminta** dalam dataset **rejected**. Dengan **50 bin** dan warna **skyblue**, grafik menampilkan jumlah peminjam pada berbagai rentang pinjaman. Format angka pada sumbu **x** dan **y** dikonversi ke **ribuan** untuk meningkatkan keterbacaan, sementara **bilangan bulat** ditampilkan di sumbu **y**. Grafik ini membantu mengidentifikasi pola umum dan outlier dalam jumlah pinjaman yang diminta.

Grafik di atas menunjukkan distribusi jumlah pinjaman yang diminta (amount_requested) untuk aplikasi pinjaman yang ditolak (Rejected Loans). Pola distribusi ini memperlihatkan konsentrasi data yang sangat tinggi di dekat nilai pinjaman yang relatif kecil, dengan mayoritas permintaan terkumpul di sekitar angka rendah dalam skala ratusan hingga ribuan dolar. Hal ini dapat mengindikasikan bahwa sebagian besar pemohon yang ditolak meminta jumlah pinjaman kecil hingga menengah.

Selain itu, rentang sumbu **x** yang membentang hingga 1,4 juta menunjukkan bahwa terdapat beberapa aplikasi dengan jumlah pinjaman yang sangat besar. Namun, frekuensinya sangat rendah, terlihat dari distribusi yang nyaris mendekati nol di luar kisaran rendah. Ini mungkin mengindikasikan bahwa pengajuan dalam jumlah yang sangat tinggi cenderung tidak disetujui karena profil risiko atau kebijakan kredit yang ketat.

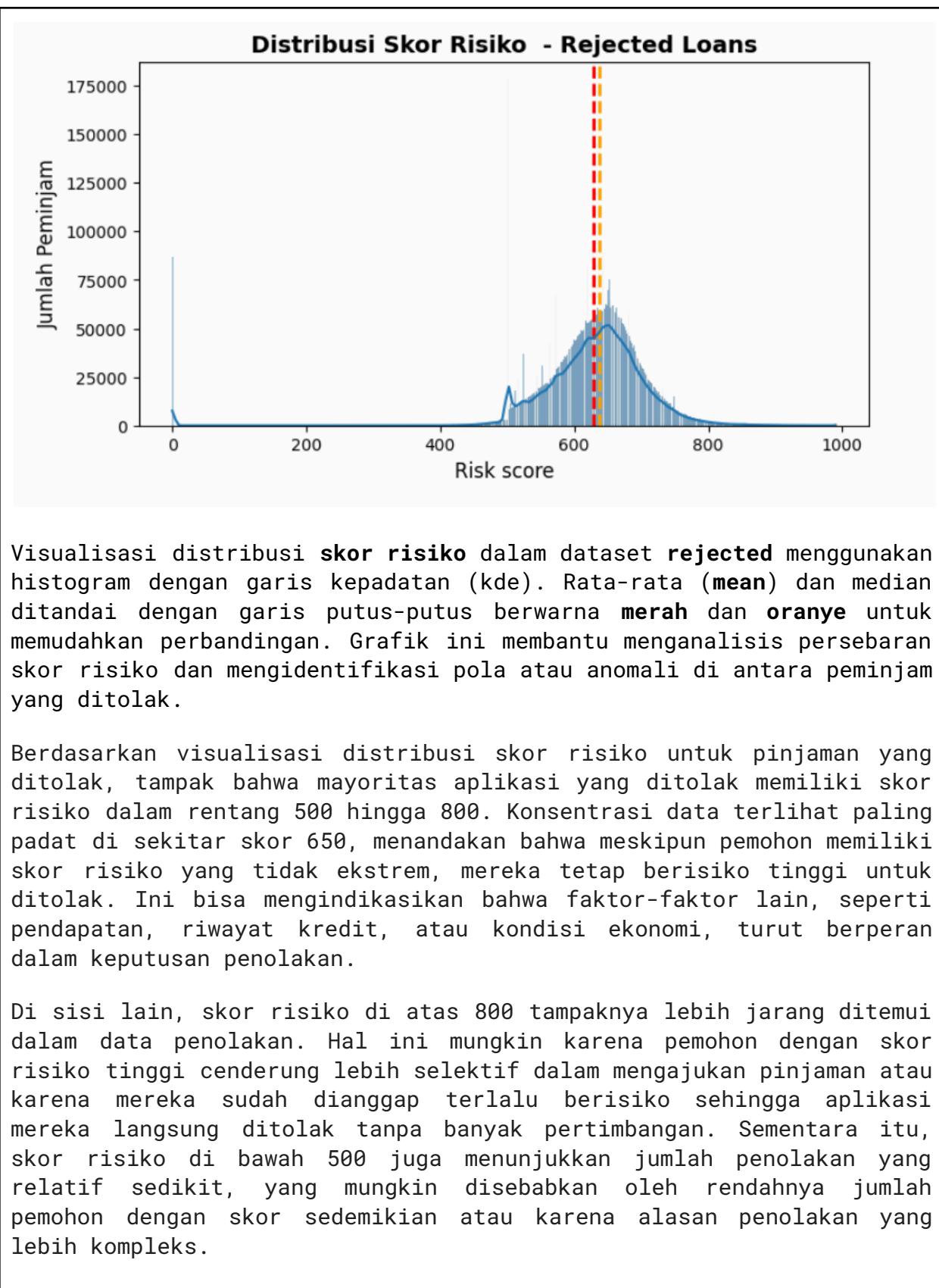
```
feature = 'risk_score'
plt.figure(figsize=(8, 4))

mean_val = rejected[feature].mean()
median_val = rejected[feature].median()

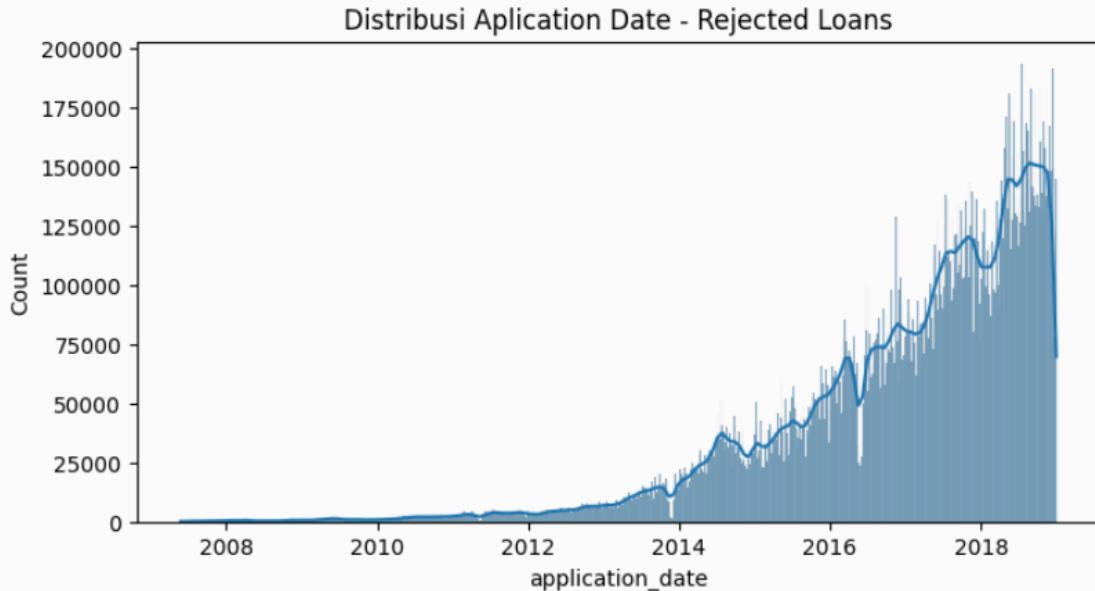
plt.axvline(mean_val, color='red', linestyle='--', linewidth=2, label=f'Mean: {mean_val:.2f}')
plt.axvline(median_val, color='orange', linestyle='--', linewidth=2, label=f'Median: {median_val:.2f}')

plt.title('Distribusi Skor Risiko - Rejected Loans', fontsize=14, fontweight='bold')
plt.xlabel('Risk score', fontsize=12)
plt.ylabel('Jumlah Peminjam', fontsize=12)

sns.histplot(rejected[feature], kde=True)
plt.show()
```



```
Rejected['application_date'] = pd.to_datetime(rejected['application_date'], errors='coerce')
feature = 'application_date'
plt.figure(figsize=(8, 4))
sns.histplot(rejected[feature], kde=True)
plt.title(f'Distribusi Application Date - Rejected Loans')
plt.show()
```



Kode ini mengonversi kolom `application_date` dalam dataset `rejected` menjadi format `datetime`, dengan parameter `errors='coerce'` untuk mengubah entri tidak valid menjadi `NaT`. Selanjutnya, kode membuat histogram dengan garis kepadatan (`kde`) untuk memvisualisasikan distribusi tanggal aplikasi dari pinjaman yang ditolak. Grafik ini membantu melihat pola atau tren pengajuan pinjaman dalam rentang waktu tertentu.

Berdasarkan grafik distribusi tanggal aplikasi untuk pinjaman yang ditolak, terlihat bahwa jumlah aplikasi pinjaman yang ditolak mengalami fluktuasi dari tahun 2008 hingga 2018. Pada tahun 2008, jumlah aplikasi yang ditolak relatif rendah, namun mulai meningkat secara signifikan pada tahun 2012 dan mencapai puncaknya sekitar tahun 2014. Setelah tahun 2014, terjadi penurunan yang cukup tajam dalam jumlah aplikasi yang ditolak, meskipun tetap ada fluktuasi hingga tahun 2018.

Berdasarkan grafik distribusi tanggal aplikasi untuk pinjaman yang ditolak, terlihat bahwa jumlah aplikasi pinjaman yang ditolak mengalami fluktuasi dari tahun 2008 hingga 2018. Pada tahun 2008, jumlah aplikasi yang ditolak relatif rendah, namun mulai meningkat

secara signifikan pada tahun 2012 dan mencapai puncaknya sekitar tahun 2014. Setelah tahun 2014, terjadi penurunan yang cukup tajam dalam jumlah aplikasi yang ditolak, meskipun tetap ada fluktuasi hingga tahun 2018.

#Dataset Home Credit Default Risk Data

application_train.csv

	train																	
	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	...	FLAG_DOCUMENT_18	FLAG_DOCUMENT_19	FLAG_DOCUMENT_20				
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597.5	24700.5	...	0.0	0.0	0.0				
1	100003	0	Cash loans	F	N	N	0	270000.0	1293502.5	35698.5	...	0.0	0.0	0.0				
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	135000.0	6750.0	...	0.0	0.0	0.0				
3	100006	0	Cash loans	F	N	Y	0	135000.0	312682.5	29686.5	...	0.0	0.0	0.0				
4	100007	0	Cash loans	M	N	Y	0	121500.0	513000.0	21865.5	...	0.0	0.0	0.0				
...				
163069	289052	0	Cash loans	M	Y	Y	0	184500.0	585000.0	35919.0	...	0.0	0.0	0.0				
163070	289053	0	Cash loans	F	Y	Y	1	90000.0	1506816.0	47443.5	...	0.0	0.0	0.0				
163071	289054	0	Cash loans	M	N	N	0	153000.0	144486.0	7695.0	...	0.0	0.0	0.0				
163072	289056	0	Cash loans	F	Y	Y	1	270000.0	1256400.0	36864.0	...	0.0	0.0	0.0				
163073	289058	0	Cash loans	F	N	Y	0	58500.0	239850.0	23494.5	...	NaN	NaN	NaN				

Ini adalah data awal dari application train, data ini memiliki 163074 rows x 122 columns

```
print('--- Struktur Dataset ---\n')
print(train.info())
print(train.describe().T)
```

--- Struktur Dataset ---

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 163074 entries, 0 to 163073
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(85), int64(21), object(16)
memory usage: 151.8+ MB
None
```

	count	mean	std
min \			
SK_ID_CURR	163074.0	194568.238119	54575.794931
100002.0			
TARGET	163074.0	0.081313	0.273316

0.0				
CNT_CHILDREN	163074.0	0.418982	0.723903	
0.0				
AMT_INCOME_TOTAL	163074.0	168941.723058	307062.842591	
25650.0				
AMT_CREDIT	163074.0	599165.706143	403211.090371	
45000.0				
...
...				
AMT_REQ_CREDIT_BUREAU_DAY	141203.0	0.006997	0.111043	
0.0				
AMT_REQ_CREDIT_BUREAU_WEEK	141203.0	0.033873	0.203282	
0.0				
AMT_REQ_CREDIT_BUREAU_MON	141203.0	0.266991	0.917754	
0.0				
AMT_REQ_CREDIT_BUREAU_QRT	141203.0	0.263918	0.609800	
0.0				
AMT_REQ_CREDIT_BUREAU_YEAR	141203.0	1.887913	1.864534	
0.0				
	25%	50%	75%	max
SK_ID_CURR	147212.25	194545.5	241801.75	289058.0
TARGET	0.00	0.0	0.00	1.0
CNT_CHILDREN	0.00	0.0	1.00	19.0
AMT_INCOME_TOTAL	112500.00	144000.0	202500.00	117000000.0
AMT_CREDIT	270000.00	513000.0	808650.00	4050000.0
...
AMT_REQ_CREDIT_BUREAU_DAY	0.00	0.0	0.00	9.0
AMT_REQ_CREDIT_BUREAU_WEEK	0.00	0.0	0.00	8.0
AMT_REQ_CREDIT_BUREAU_MON	0.00	0.0	0.00	24.0
AMT_REQ_CREDIT_BUREAU_QRT	0.00	0.0	0.00	8.0
AMT_REQ_CREDIT_BUREAU_YEAR	0.00	1.0	3.00	25.0
[106 rows x 8 columns]				
Dataset Home Credit Default Risk memiliki 307.511 entri dan 122 kolom dengan tipe data yang bervariasi, termasuk 65 kolom bertipe float64, 41 kolom bertipe int64, dan 16 kolom bertipe object. Memori yang				

digunakan sebesar 286,2+ MB.

Beberapa insight awal dari statistik deskriptif:

- **Kolom TARGET** menunjukkan bahwa 8,07% dari total entri menandakan kegagalan pembayaran.
- **CNT_CHILDREN** memiliki rata-rata 0,42 dengan nilai maksimum 19, menunjukkan beberapa debitur memiliki jumlah anak yang sangat tinggi.
- **AMT_INCOME_TOTAL** memiliki rata-rata Rp168.797.919, namun terdapat outlier dengan nilai maksimum Rp117.000.000.000.
- **AMT_CREDIT** memiliki distribusi yang lebar dengan median Rp513.531.000 dan maksimum Rp4.050.000.000.

Adanya perbedaan yang signifikan antara rata-rata dan nilai maksimum pada beberapa kolom menunjukkan kemungkinan keberadaan outlier yang perlu dianalisis lebih lanjut. Silakan lanjutkan dengan hasil kode berikutnya.

```
▶ data_types = train.dtypes.reset_index()
data_types.columns = ['Nama Kolom', 'Tipe Data']
print(data_types)

→
      Nama Kolom Tipe Data
0          SK_ID_CURR    int64
1            TARGET    int64
2        NAME_CONTRACT_TYPE    object
3          CODE_GENDER    object
4        FLAG_OWN_CAR    object
..           ...
117   AMT_REQ_CREDIT_BUREAU_DAY    float64
118   AMT_REQ_CREDIT_BUREAU_WEEK    float64
119   AMT_REQ_CREDIT_BUREAU_MON    float64
120   AMT_REQ_CREDIT_BUREAU_QRT    float64
121   AMT_REQ_CREDIT_BUREAU_YEAR    float64

[122 rows x 2 columns]
```

Kode ini untuk menampilkan **tipe data** dari setiap kolom dalam DataFrame `train` dengan format yang lebih terstruktur. Proses dimulai dengan **resetting index** pada `train.dtypes` agar hasilnya berubah menjadi DataFrame yang kemudian diberi dua kolom, yaitu '**Nama Kolom**' dan '**Tipe Data**'. Output yang dihasilkan memperlihatkan total 122 kolom, dengan tipe data seperti `int64`, `float64`, dan `object`. Contoh

kolom seperti `SK_ID_CURR` dan `TARGET` bertipe `int64`, `NAME_CONTRACT_TYPE` bertipe `object`, sementara `AMT_REQ_CREDIT_BUREAU_YEAR` bertipe `float64`. Proses ini penting dalam tahap **eksplorasi data** untuk memahami jenis data yang dimiliki sebelum melakukan **data preprocessing** seperti normalisasi, encoding, atau imputasi.

```
[ ] missing_values = train.isnull().sum().sort_values(ascending=False)
missing_percentage = (missing_values / len(train)) * 100
missing_df = pd.DataFrame({'Jumlah Missing': missing_values, 'Persentase': missing_percentage})
print(missing_df.head(10))
```

	Jumlah Missing	Persentase
COMMONAREA_AVG	114037	69.929603
COMMONAREA_MODE	114037	69.929603
COMMONAREA_MEDI	114037	69.929603
NONLIVINGAPARTMENTS_MEDI	113270	69.459264
NONLIVINGAPARTMENTS_MODE	113270	69.459264
NONLIVINGAPARTMENTS_AVG	113270	69.459264
LIVINGAPARTMENTS_AVG	111564	68.413113
LIVINGAPARTMENTS_MODE	111564	68.413113
LIVINGAPARTMENTS_MEDI	111564	68.413113
FONDKAPREMONT_MODE	111548	68.403302

Kode ini menganalisis **missing values** dalam DataFrame `train`. Pertama, variabel `missing_values` menghitung jumlah nilai yang hilang (`Nan`) di setiap kolom menggunakan `isnull().sum()`, kemudian diurutkan secara **descending** untuk menampilkan kolom dengan jumlah missing tertinggi di atas.

Kedua hasil tersebut digabungkan dalam DataFrame `missing_df` dengan dua kolom: '**Jumlah Missing**' dan '**Persentase**'. Baris `print(missing_df.head(10))` menampilkan **10 kolom teratas** dengan jumlah missing values tertinggi. Hasil menunjukkan bahwa kolom seperti `COMMONAREA_AVG` memiliki **114.037** nilai yang hilang, setara dengan **69,93%** dari total data. Analisis ini penting untuk menentukan apakah perlu dilakukan **imputasi**, **penghapusan kolom**, atau metode penanganan missing value lainnya.

```
[ ] print('--- Statistik Deskriptif Numerik ---\n')
print(train.describe().T)

→ --- Statistik Deskriptif Numerik ---

      count      mean       std      min  \
SK_ID_CURR    163074.0  194568.238119  54575.794931  100002.0
TARGET        163074.0     0.081313    0.273316     0.0
CNT_CHILDREN   163074.0     0.418982    0.723903     0.0
AMT_INCOME_TOTAL  163074.0  168941.723058  307062.842591  25650.0
AMT_CREDIT     163074.0   599165.706143  403211.090371  45000.0
...
AMT_REQ_CREDIT_BUREAU_DAY  141203.0     0.006997    0.111043     0.0
AMT_REQ_CREDIT_BUREAU_WEEK 141203.0     0.033873    0.203282     0.0
AMT_REQ_CREDIT_BUREAU_MON   141203.0     0.266991    0.917754     0.0
AMT_REQ_CREDIT_BUREAU_QRT   141203.0     0.263918    0.609800     0.0
AMT_REQ_CREDIT_BUREAU_YEAR  141203.0     1.887913    1.864534     0.0

      25%      50%      75%      max
SK_ID_CURR  147212.25  194545.5  241801.75  289058.0
TARGET      0.00      0.0      0.00      1.0
CNT_CHILDREN 0.00      0.0      1.00      19.0
AMT_INCOME_TOTAL 112500.00 144000.0 202500.00 117000000.0
AMT_CREDIT   270000.00 513000.0 808650.00 4050000.0
...
AMT_REQ_CREDIT_BUREAU_DAY  0.00      0.0      0.00      9.0
AMT_REQ_CREDIT_BUREAU_WEEK 0.00      0.0      0.00      8.0
AMT_REQ_CREDIT_BUREAU_MON   0.00      0.0      0.00      24.0
AMT_REQ_CREDIT_BUREAU_QRT   0.00      0.0      0.00      8.0
AMT_REQ_CREDIT_BUREAU_YEAR  0.00      1.0      3.00      25.0

[106 rows x 8 columns]
```

Kode ini menampilkan **statistik deskriptif** dari data numerik dalam DataFrame `train` menggunakan `train.describe().T`, yang kemudian di-print dengan judul "**Statistik Deskriptif Numerik**". Outputnya mencakup ringkasan statistik seperti **count** (jumlah data non-missing), **mean** (rata-rata), **std** (standar deviasi), **min** (nilai minimum), dan **max** (nilai maksimum), serta **percentiles** (25%, 50%/median, dan 75%) untuk setiap kolom numerik.

Misalnya, pada kolom **AMT_CREDIT**, terdapat **163.074** data dengan rata-rata kredit sebesar **599.165**, standar deviasi **403.211**, nilai minimum **45.000**, dan maksimum **4.050.000**. Statistik ini membantu memahami distribusi dan skala data, mendeteksi **outlier**, dan mengevaluasi variabilitas data, yang penting dalam proses **data preprocessing** dan **feature engineering**.

```
[ ] print('--- Statistik Deskriptif Kategorikal ---\n')
categorical_cols = train.select_dtypes(include=['object']).columns
print(train[categorical_cols].describe().T)
```

--- Statistik Deskriptif Kategorikal ---

	count	unique	top	\
NAME_CONTRACT_TYPE	163074	2	Cash loans	
CODE_GENDER	163074	3	F	
FLAG_OWN_CAR	163074	2	N	
FLAG_OWN_REALTY	163074	2	Y	
NAME_TYPE_SUITE	162410	7	Unaccompanied	
NAME_INCOME_TYPE	163074	8	Working	
NAME_EDUCATION_TYPE	163074	5	Secondary / secondary special	
NAME_FAMILY_STATUS	163074	6	Married	
NAME_HOUSING_TYPE	163074	6	House / apartment	
OCCUPATION_TYPE	111895	18	Laborers	
WEEKDAY_APPR_PROCESS_START	163074	7	TUESDAY	
ORGANIZATION_TYPE	163074	58	Business Entity Type 3	
FONDKAPREMONT_MODE	51526	4	reg oper account	
HOUSETYPE_MODE	81085	3	block of flats	
WALLSMATERIAL_MODE	80050	7	Panel	
EMERGENCYSTATE_MODE	85636	2	No	
			freq	
NAME_CONTRACT_TYPE	147581			
CODE_GENDER	107327			
FLAG_OWN_CAR	107649			
FLAG_OWN_REALTY	113045			
NAME_TYPE_SUITE	131823			
NAME_INCOME_TYPE	84394			
NAME_EDUCATION_TYPE	115947			
NAME_FAMILY_STATUS	104460			
NAME_HOUSING_TYPE	144765			
OCCUPATION_TYPE	29348			
WEEKDAY_APPR_PROCESS_START	28461			
ORGANIZATION_TYPE	36272			
FONDKAPREMONT_MODE	39168			
HOUSETYPE_MODE	79685			
WALLSMATERIAL_MODE	34867			
EMERGENCYSTATE_MODE	84429			

Kode tersebut bertujuan untuk menampilkan **statistik deskriptif** dari kolom **kategorikal** dalam DataFrame `train`. Baris kode `train.select_dtypes(include=['object'])` berfungsi untuk memilih kolom dengan tipe data `object`, yang biasanya berisi data kategorikal. Kemudian, hasil statistiknya ditampilkan dengan `train[categorical_cols].describe().T`, yang memberikan ringkasan

berupa **count** (jumlah entri non-null), **unique** (jumlah kategori unik), **top** (kategori yang paling sering muncul), dan **freq** (frekuensi kemunculan kategori tersebut).

Dari output yang ditampilkan, kita dapat melihat bahwa beberapa kolom seperti **NAME_CONTRACT_TYPE** memiliki **2 kategori**, dengan kategori paling dominan adalah "**Cash loans**" yang muncul sebanyak **147.581 kali**. Sementara itu, kolom **ORGANIZATION_TYPE** memiliki jumlah kategori terbanyak, yaitu **58 kategori unik**, dengan kategori "**Business Entity Type 3**" sebagai yang paling sering muncul. Informasi ini sangat berguna untuk memahami distribusi data kategorikal dan mengidentifikasi kemungkinan ketidakseimbangan (imbalance) dalam kategori tertentu.

```
▶ missing_values = train.isnull().sum()
missing_percentage = (missing_values / len(train)) * 100

missing_df = pd.DataFrame({
    'Jumlah Missing': missing_values,
    'Persentase Missing': missing_percentage
}).sort_values(by='Jumlah Missing', ascending=False)

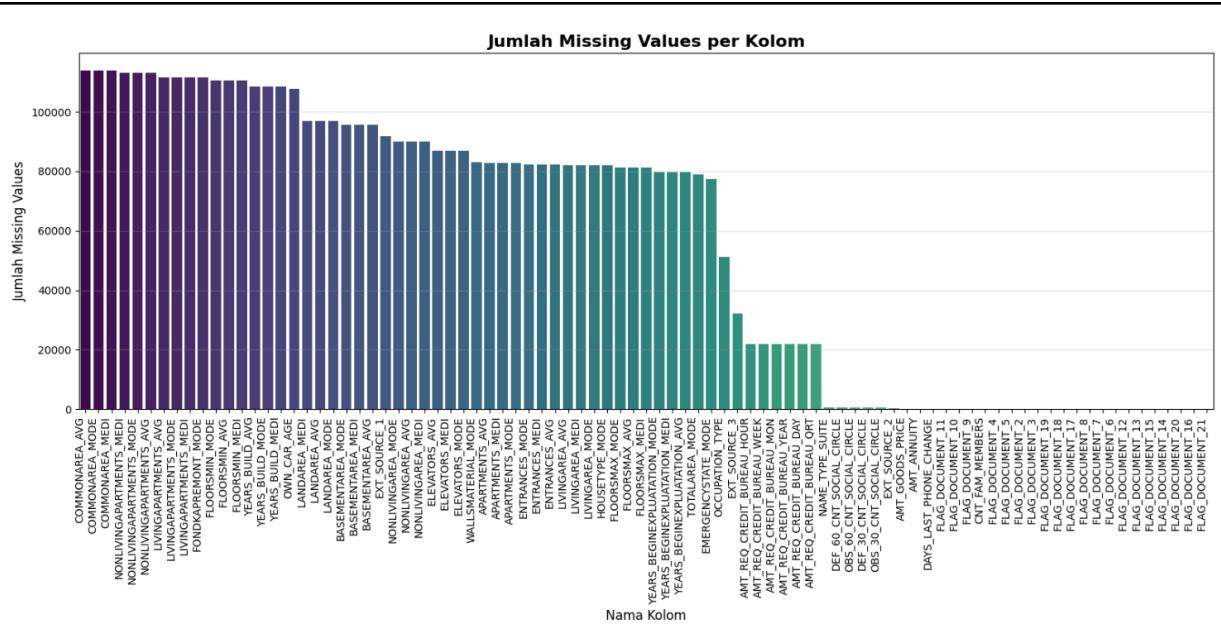
missing_df = missing_df[missing_df['Jumlah Missing'] > 0]

plt.figure(figsize=(16, 8))
sns.barplot(x=missing_df.index, y=missing_df['Jumlah Missing'], hue=missing_df.index, palette='viridis', legend=False)

plt.title('Jumlah Missing Values per Kolom', fontsize=16, fontweight='bold')
plt.xlabel('Nama Kolom', fontsize=12)
plt.ylabel('Jumlah Missing Values', fontsize=12)
plt.xticks(rotation=90, ha='right')

plt.grid(axis='y', alpha=0.3)

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

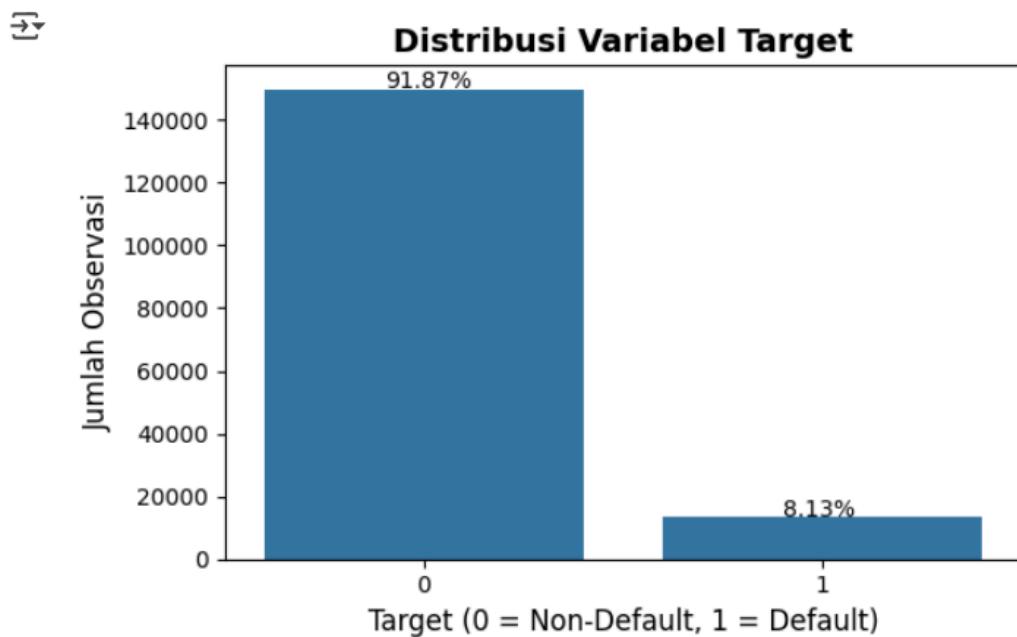


analisis data yang hilang (missing values) dalam dataset train. Pertama, jumlah nilai yang hilang dalam setiap kolom dihitung menggunakan `train.isnull().sum()` dan persentasenya diperoleh dengan membagi jumlah missing values dengan total data. Hasil ini kemudian disusun dalam bentuk DataFrame `missing_df`, diurutkan berdasarkan jumlah missing values secara menurun.

Selanjutnya, DataFrame difilter untuk hanya menyertakan kolom yang memiliki missing values lebih dari nol. Visualisasi dibuat menggunakan `seaborn.barplot` untuk menampilkan jumlah missing values per kolom dalam bentuk bar chart dengan palet warna viridis. Judul, label sumbu, rotasi label pada sumbu-x, dan grid ditambahkan untuk meningkatkan keterbacaan grafik.

Output grafik menunjukkan distribusi jumlah data yang hilang di setiap kolom, dengan urutan dari jumlah tertinggi hingga terendah. Grafik ini membantu dalam mengidentifikasi kolom mana yang memiliki masalah missing values yang signifikan, sehingga dapat menjadi acuan dalam proses pembersihan data.

```
[ ] plt.figure(figsize=(6,4))
sns.countplot(x='TARGET', data=train)
total = len(train)
for p in plt.gca().patches:
    percentage = f'{100 * p.get_height() / total:.2f}%'
    plt.gca().annotate(percentage, (p.get_x() + 0.3, p.get_height() + 500))
plt.title('Distribusi Variabel Target', fontsize=14, fontweight='bold')
plt.xlabel('Target (0 = Non-Default, 1 = Default)', fontsize=12)
plt.ylabel('Jumlah Observasi', fontsize=12)
plt.tight_layout()
plt.show()
```



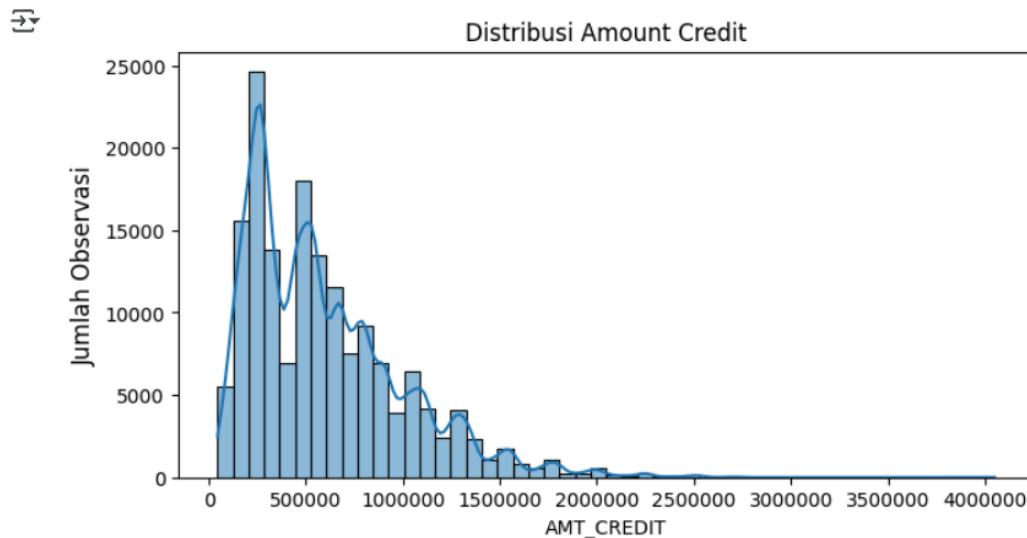
visualisasikan distribusi variabel target dalam dataset train menggunakan bar chart. sns.countplot digunakan untuk menghitung jumlah observasi dalam setiap kategori pada kolom TARGET, di mana 0 merepresentasikan "Non-Default" dan 1 merepresentasikan "Default". Total jumlah observasi dihitung dengan len(train).

Setiap bar dalam plot diberi anotasi berupa persentase yang dihitung dengan rumus $(\text{tinggi bar} / \text{total}) * 100$, diformat hingga dua desimal. Anotasi ditempatkan sedikit di atas masing-masing bar untuk meningkatkan keterbacaan. Judul grafik dan label sumbu juga disesuaikan untuk menjelaskan bahwa 0 berarti "Non-Default" dan 1 berarti "Default".

Output menunjukkan ketidakseimbangan kelas (class imbalance) yang

signifikan, dengan 91,87% data termasuk dalam kategori "Non-Default" dan hanya 8,13% dalam kategori "Default". Informasi ini sangat penting dalam pemodelan machine learning karena ketidakseimbangan data sering kali mempengaruhi kinerja model, khususnya dalam mendeteksi kasus minoritas.

```
plt.figure(figsize=(8, 4))
sns.histplot(train['AMT_CREDIT'], bins=50, kde=True)
plt.title('Distribusi Amount Credit')
plt.ylabel('Jumlah Observasi', fontsize=12)
plt.ticklabel_format(style='plain', axis='x')
plt.show()
```

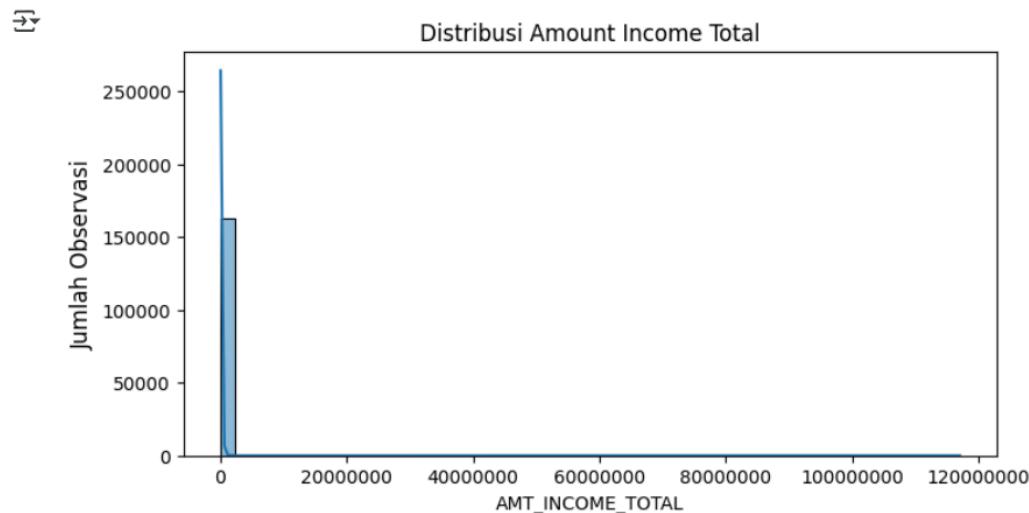


visualisasikan distribusi jumlah kredit (kolom AMT_CREDIT) dalam dataset train menggunakan histogram. Dengan parameter bins=50, data dibagi menjadi 50 interval, sedangkan kde=True menambahkan kurva distribusi kernel density estimation (KDE) untuk menunjukkan pola kepadatan data. Grafik ini membantu mengidentifikasi pola umum, outlier, dan kemiringan data.

Hasil plot menunjukkan bahwa sebagian besar nilai kredit terkonsentrasi di bawah 1.000.000, dengan jumlah observasi tertinggi di sekitar angka 500.000. Setelah itu, jumlah observasi menurun secara bertahap dengan beberapa outlier di kisaran angka yang lebih tinggi. Informasi ini bermanfaat untuk memahami skala dan distribusi nilai kredit, yang bisa menjadi penting dalam analisis risiko kredit

atau model prediksi.

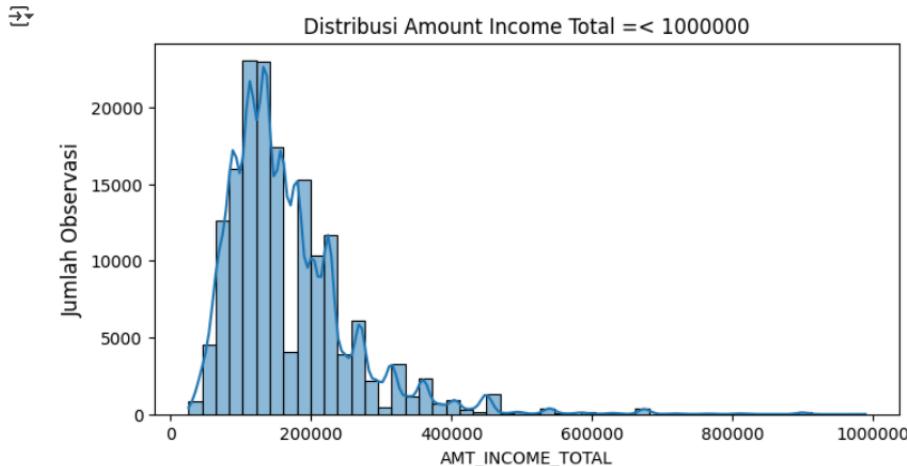
```
[ ] plt.figure(figsize=(8, 4))
sns.histplot(train['AMT_INCOME_TOTAL'], bins=50, kde=True)
plt.title('Distribusi Amount Income Total')
plt.ylabel('Jumlah Observasi', fontsize=12)
plt.ticklabel_format(style='plain', axis='x')
plt.show()
```



visualisasikan distribusi jumlah kredit (kolom AMT_CREDIT) dalam dataset train menggunakan histogram. Dengan parameter bins=50, data dibagi menjadi 50 interval, sedangkan kde=True menambahkan kurva distribusi kernel density estimation (KDE) untuk menunjukkan pola kepadatan data. Grafik ini membantu mengidentifikasi pola umum, outlier, dan kemiringan data.

Hasil plot menunjukkan bahwa sebagian besar nilai kredit terkonsentrasi di bawah 1.000.000, dengan jumlah observasi tertinggi di sekitar angka 500.000. Setelah itu, jumlah observasi menurun secara bertahap dengan beberapa outlier di kisaran angka yang lebih tinggi. Informasi ini bermanfaat untuk memahami skala dan distribusi nilai kredit, yang bisa menjadi penting dalam analisis risiko kredit atau model prediksi.

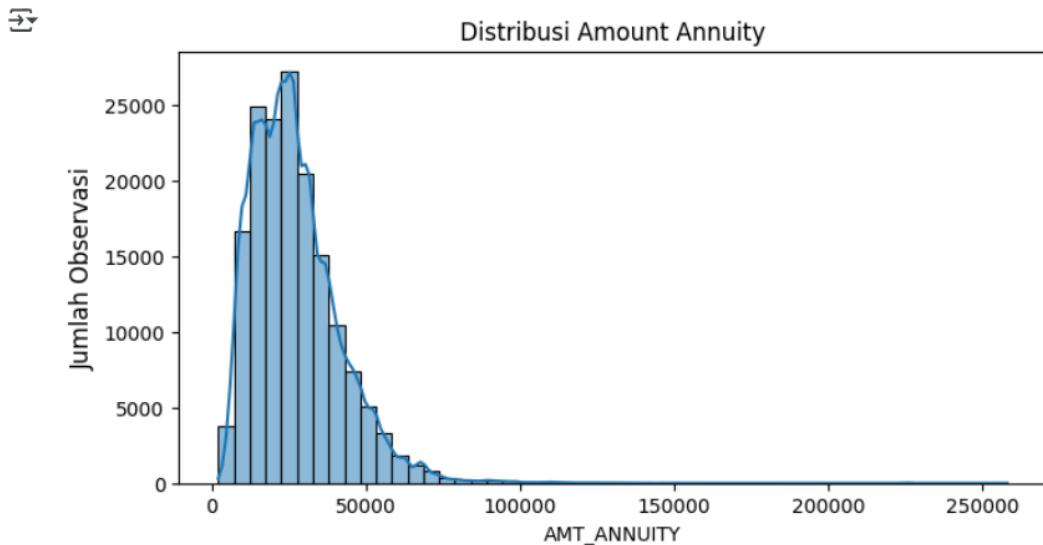
```
[ ] filtered_df = train[train['AMT_INCOME_TOTAL'] < 1000000]
plt.figure(figsize=(8, 4))
sns.histplot(filtered_df['AMT_INCOME_TOTAL'], bins=50, kde=True)
plt.title('Distribusi Amount Income Total <= 1000000')
plt.ylabel('Jumlah Observasi', fontsize=12)
plt.ticklabel_format(style='plain', axis='x')
plt.show()
```



Kode ini merupakan lanjutan dari visualisasi distribusi data pada dataset `train`, tetapi dengan fokus khusus pada kolom `AMT_INCOME_TOTAL` yang difilter hingga nilai kurang dari atau sama dengan 1.000.000. Tujuan dari filter ini adalah untuk menghilangkan outlier yang ekstrem agar distribusi utama terlihat lebih jelas.

Histogram yang dihasilkan menggunakan 50 bins, dengan garis KDE (`kde=True`) yang membantu memperlihatkan pola kepadatan data. Grafik menunjukkan bahwa sebagian besar penghasilan total berada di bawah 300.000, dengan puncak tertinggi di sekitar 200.000. Setelah itu, jumlah observasi menurun secara bertahap. Pola ini menandakan distribusi yang miring ke kanan, dengan sebagian besar data terkonsentrasi di rentang yang lebih rendah. Ini bisa menjadi indikasi bahwa sebagian besar individu dalam dataset memiliki pendapatan menengah ke bawah.

```
[ ] plt.figure(figsize=(8, 4))
sns.histplot(train['AMT_ANNUITY'], bins=50, kde=True)
plt.title('Distribusi Amount Annuity')
plt.ylabel('Jumlah Observasi', fontsize=12)
plt.ticklabel_format(style='plain', axis='x')
plt.show()
```



Kode ini bertujuan untuk memvisualisasikan distribusi kolom **AMT_ANNUITY** dari dataset **train** dalam bentuk histogram. Histogram menggunakan 50 bins dengan kernel density estimation (KDE) diaktifkan (**kde=True**) untuk menunjukkan pola kepadatan data secara lebih halus. Grafik ini menampilkan jumlah observasi di sumbu y dan nilai **AMT_ANNUITY** di sumbu x.

Berdasarkan output, distribusi cenderung miring ke kanan dengan mayoritas data terkonsentrasi di bawah 50.000. Puncak distribusi berada di sekitar 25.000 hingga 30.000, yang menunjukkan bahwa kebanyakan individu dalam dataset memiliki jumlah anuitas dalam rentang tersebut. Setelah itu, frekuensi observasi menurun seiring bertambahnya jumlah anuitas, dengan sedikit outlier pada nilai yang lebih tinggi. Visualisasi ini membantu dalam mengidentifikasi pola umum dan anomali dalam pembayaran anuitas.

```

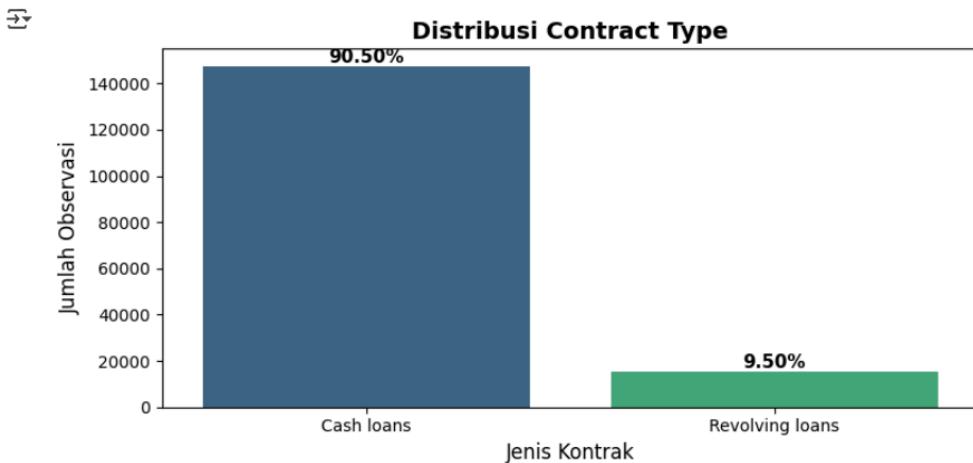
plt.figure(figsize=(8, 4))
sns.countplot(x='NAME_CONTRACT_TYPE', data=train, palette='viridis', hue = 'NAME_CONTRACT_TYPE', legend=False)

total = len(train)

for p in plt.gca().patches:
    height = p.get_height()
    percentage = f'{100 * height / total:.2f}%'
    plt.gca().annotate(percentage, (p.get_x() + p.get_width() / 2, height),
                       ha='center', va='bottom', fontsize=11, fontweight='bold')

plt.title('Distribusi Contract Type', fontsize=14, fontweight='bold')
plt.xlabel('Jenis Kontrak', fontsize=12)
plt.ylabel('Jumlah Observasi', fontsize=12)
plt.tight_layout()
plt.show()

```



Kode di atas memvisualisasi distribusi jenis kontrak dalam dataset train. Fungsi sns.countplot() digunakan untuk membuat diagram batang berdasarkan kolom NAME_CONTRACT_TYPE, dengan pewarnaan dari palet viridis dan tanpa menampilkan legenda. Setelah itu, total jumlah observasi dihitung, dan setiap batang dalam grafik diberikan anotasi berupa persentase dari total data. Anotasi ini diperoleh dengan membagi tinggi masing-masing batang dengan total jumlah observasi dan mengalikannya dengan 100. Selanjutnya, grafik diberi judul serta label pada sumbu-x dan sumbu-y agar lebih mudah dipahami. Fungsi plt.tight_layout() memastikan tampilan grafik lebih rapi sebelum akhirnya ditampilkan dengan plt.show().

Output yang dihasilkan adalah sebuah diagram batang yang menunjukkan distribusi dua jenis kontrak: "Cash loans" dan "Revolving loans". Batang untuk "Cash loans" jauh lebih tinggi dibanding "Revolving loans", menunjukkan bahwa mayoritas kontrak dalam dataset adalah pinjaman tunai. Tepat di atas setiap batang terdapat anotasi angka

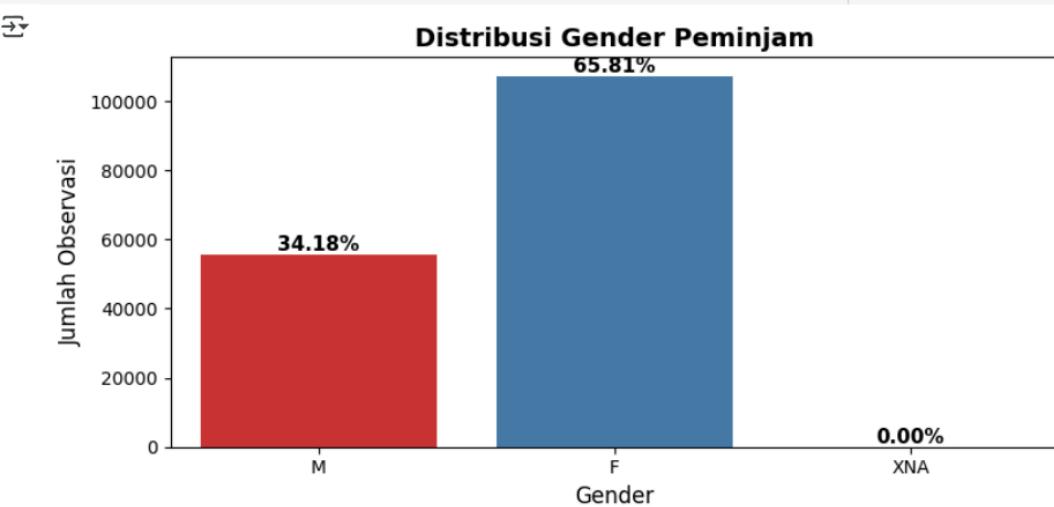
yang menunjukkan persentase masing-masing jenis kontrak dari total data, yaitu 90.50% untuk "Cash loans" dan 9.50% untuk "Revolving loans". Hal ini mengindikasikan bahwa dataset didominasi oleh satu jenis kontrak, yang bisa menjadi faktor penting dalam analisis lebih lanjut.

```
plt.figure(figsize=(8, 4))
sns.countplot(x='CODE_GENDER', data=train, palette='Set1', hue='CODE_GENDER', legend=False)

total = len(train)

for p in plt.gca().patches:
    height = p.get_height()
    percentage = f'{100 * height / total:.2f}%'
    plt.gca().annotate(percentage, (p.get_x() + p.get_width() / 2, height),
                       ha='center', va='bottom', fontsize=11, fontweight='bold')

plt.title('Distribusi Gender Peminjam', fontsize=14, fontweight='bold')
plt.xlabel('Gender', fontsize=12)
plt.ylabel('Jumlah Observasi', fontsize=12)
plt.tight_layout()
plt.show()
```



Kode ini menampilkan distribusi gender peminjam dalam dataset train. Dengan menggunakan sns.countplot(), kode ini memvisualisasikan jumlah observasi untuk setiap kategori dalam kolom CODE_GENDER, menggunakan palet warna Set1. Setelah itu, total jumlah observasi dihitung, dan setiap batang diberikan anotasi persentase berdasarkan proporsinya terhadap total data. Judul dan label sumbu ditambahkan agar grafik lebih mudah dipahami. Akhirnya, plt.tight_layout() memastikan grafik lebih rapi sebelum ditampilkan dengan plt.show().

Grafik yang dihasilkan menunjukkan distribusi peminjam berdasarkan gender, dengan tiga kategori: "M" (Male/Laki-laki), "F" (Female/Perempuan), dan "XNA". Dari hasil visualisasi, mayoritas peminjam adalah perempuan (65.81%), sedangkan laki-laki menyumbang 34.18% dari total data. Kategori "XNA", yang kemungkinan merupakan data yang tidak valid atau tidak teridentifikasi, memiliki persentase 0.00%, menunjukkan bahwa hampir tidak ada data dalam kategori ini. Grafik ini memberikan wawasan penting mengenai distribusi gender dalam dataset, yang bisa berguna untuk analisis lebih lanjut terkait pola pinjaman berdasarkan gender.

```

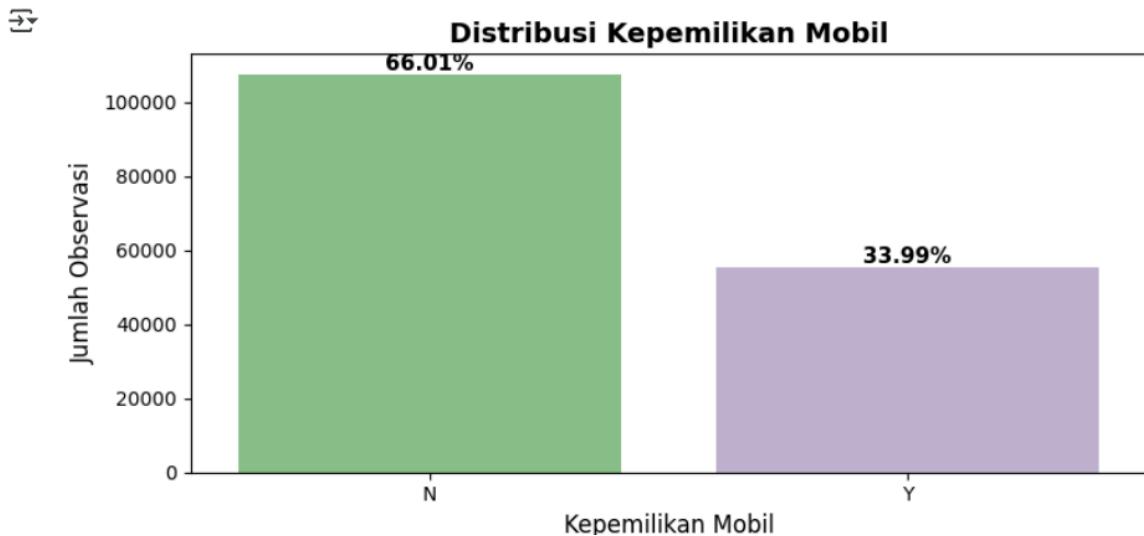
plt.figure(figsize=(8, 4))
sns.countplot(x='FLAG OWN CAR', data=train, palette='Accent', hue='FLAG OWN CAR', legend=False)

total = len(train)

for p in plt.gca().patches:
    height = p.get_height()
    percentage = f'{100 * height / total:.2f}%' 
    plt.gca().annotate(percentage, (p.get_x() + p.get_width() / 2, height),
                       ha='center', va='bottom', fontsize=11, fontweight='bold')

plt.title('Distribusi Kepemilikan Mobil', fontsize=14, fontweight='bold')
plt.xlabel('Kepemilikan Mobil', fontsize=12)
plt.ylabel('Jumlah Observasi', fontsize=12)
plt.tight_layout()
plt.show()

```



Kode ini bertujuan untuk membuat grafik batang yang menunjukkan distribusi kepemilikan mobil dalam dataset train. Dengan menggunakan Seaborn, fungsi sns.countplot() menampilkan jumlah observasi untuk setiap kategori dalam kolom FLAG OWN CAR, dengan pewarnaan dari palet Accent dan tanpa legenda. Setelah itu, total observasi dihitung, dan setiap batang diberikan anotasi berupa persentase berdasarkan proporsi terhadap total data. Grafik juga dilengkapi dengan judul, label pada sumbu-x dan sumbu-y, serta plt.tight_layout() untuk memastikan tampilan lebih rapi sebelum akhirnya ditampilkan menggunakan plt.show().

Grafik yang dihasilkan menunjukkan distribusi kepemilikan mobil dengan dua kategori: "N" (tidak memiliki mobil) dan "Y" (memiliki

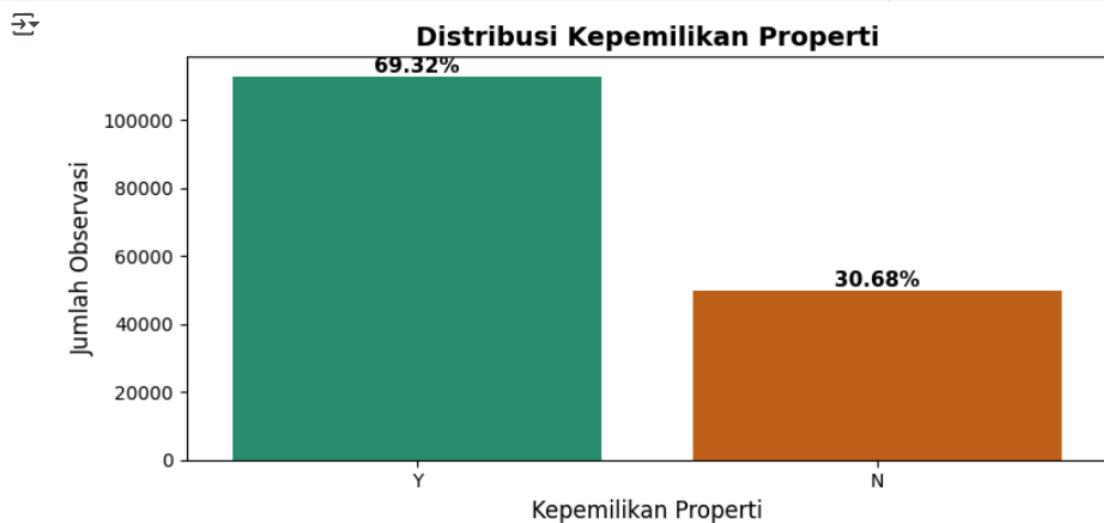
mobil). Dari hasil visualisasi, sebanyak 66.01% dari peminjam tidak memiliki mobil, sedangkan 33.99% memiliki mobil. Perbedaan yang cukup signifikan antara kedua kategori ini menunjukkan bahwa mayoritas peminjam dalam dataset tidak memiliki kendaraan pribadi, yang bisa menjadi faktor penting dalam analisis kredit atau profil nasabah.

```
plt.figure(figsize=(8, 4))
sns.countplot(x='FLAG_OWN_REALTY', data=train, palette='Dark2', hue='FLAG_OWN_REALTY', legend=False)

total = len(train)

for p in plt.gca().patches:
    height = p.get_height()
    percentage = f'{100 * height / total:.2f}%'
    plt.gca().annotate(percentage, (p.get_x() + p.get_width() / 2, height),
                       ha='center', va='bottom', fontsize=11, fontweight='bold')

plt.title('Distribusi Kepemilikan Properti', fontsize=14, fontweight='bold')
plt.xlabel('Kepemilikan Properti', fontsize=12)
plt.ylabel('Jumlah Observasi', fontsize=12)
plt.tight_layout()
plt.show()
```



Kode ini digunakan untuk membuat diagram batang yang menampilkan distribusi kepemilikan properti dalam dataset train. Dengan menggunakan Seaborn, fungsi sns.countplot() memvisualisasikan jumlah observasi untuk setiap kategori dalam kolom FLAG_OWN_REALTY, dengan palet warna Dark2 dan tanpa legenda. Total jumlah observasi dihitung, lalu setiap batang diberi anotasi persentase berdasarkan proporsinya terhadap total data. Grafik ini juga dilengkapi dengan judul, label

sumbu-x untuk kategori kepemilikan properti, serta label sumbu-y untuk jumlah observasi. Akhirnya, plt.tight_layout() digunakan untuk memastikan tata letak yang lebih rapi sebelum grafik ditampilkan dengan plt.show().

Dari hasil visualisasi, terlihat bahwa mayoritas peminjam dalam dataset memiliki properti (69.32%), sementara 30.68% tidak memiliki properti. Perbedaan ini menunjukkan bahwa lebih banyak peminjam yang memiliki aset properti dibandingkan yang tidak. Informasi ini bisa menjadi faktor penting dalam analisis risiko kredit atau kebijakan pemberian pinjaman, karena kepemilikan properti sering dikaitkan dengan stabilitas finansial yang lebih baik.

Anggota Dua | Kayla Putri Maharani_5026231158

Penanganan Data yang Hilang dan Pembersihan Data untuk kedua Dataset

Langkah :

- 1. Identifikasi dan Penghapusan Kolom dengan Banyak Missing Values**
 - Kolom dengan lebih dari 51% nilai hilang dihapus karena dianggap memiliki terlalu banyak data yang tidak lengkap, sehingga kurang bermanfaat untuk analisis.
 - Contoh pada dataset **accepted_2007_to_2018Q4**, kolom seperti **member_id**, **settlement_status**, **settlement_date**, dan beberapa kolom lain dihapus karena persentase missing values yang sangat tinggi (mendekati 100%).
 - Teknik ini membantu mengurangi noise dalam data serta meningkatkan efisiensi pemrosesan.
- 2. Imputasi Missing Values dengan Median atau Mode**
 - Untuk data numerik, nilai yang hilang diisi menggunakan **median**, karena median lebih tahan terhadap outlier dibandingkan mean.
 - Contoh pada dataset **application_train**, kolom seperti **OWN_CAR_AGE** dan **EXT_SOURCE_1** memiliki banyak nilai hilang, sehingga metode median digunakan untuk mengisi kekosongan tanpa menyebabkan bias yang signifikan.
 - Untuk data kategorikal, nilai yang hilang diisi menggunakan **mode** (nilai yang paling sering muncul), seperti dalam kolom **emp_title** pada dataset **accepted_2007_to_2018Q4**.
- 3. Penghapusan Baris dengan Missing Values Kecil (<10%)**
 - Jika suatu kolom memiliki nilai hilang kurang dari 10%, baris

yang mengandung missing values tersebut dihapus karena jumlahnya kecil dan dampaknya terhadap analisis minimal.

- Contoh pada dataset **rejected_2007_to_2018Q4**, kolom loan_title, zip_code, dan state hanya memiliki sedikit missing values sehingga baris yang mengandung nilai hilang tersebut dihapus.
- Ini bertujuan untuk menjaga kualitas data tanpa kehilangan informasi yang signifikan.

4. Identifikasi dan Penghapusan Kolom Redundan

- Beberapa kolom yang memiliki korelasi tinggi dalam missing values diidentifikasi dan hanya satu kolom yang dipertahankan.
- Contoh pada dataset **application_train**, ditemukan beberapa kolom yang selalu hilang bersama-sama (APARTMENTS_MEDI, BASEMENTAREA_AVG, dll.), sehingga hanya satu kolom yang dipertahankan dari setiap kelompok.
- Langkah ini mengurangi dimensi dataset tanpa kehilangan informasi penting.

5. Visualisasi dan Validasi Setelah Pembersihan

- Setelah proses pembersihan, dilakukan visualisasi missing values menggunakan **heatmap** untuk memastikan bahwa tidak ada lagi nilai yang hilang secara signifikan.
- Contoh pada dataset **installments_payments**, setelah dilakukan pembersihan, diperiksa kembali apakah terdapat missing values menggunakan teknik heatmap dan hasilnya menunjukkan bahwa data sudah bersih.

#Dataset Lending Club Loan Data

accepted_2007_to_2018Q4.csv

```

def missing_val(df):
    missing_count = df.isnull().sum()
    print("Missing values per column:")
    print(missing_count)

    # Calculate percentage of missing values
    missing_percentage = (df.isnull().mean() * 100).round(2)
    print("\nPercentage of missing values per column (%):")
    print(missing_percentage)

    # Create a summary DataFrame
    missing_summary = pd.DataFrame({
        'Missing_Count': missing_count,
        'Missing_Percentage (%)': missing_percentage
    })
    print("\nMissing Values Summary:")
    print(missing_summary)
    return missing_summary

```

Disini, dibuat rumus untuk menghitung missing value dari suatu dataset. Fungsi missing_val(df) digunakan untuk menganalisis data yang hilang (missing values) dalam DataFrame df. Pertama, fungsi ini menghitung jumlah nilai yang hilang di setiap kolom menggunakan df.isnull().sum(), lalu menampilkannya. Setelah itu, fungsi menghitung persentase nilai yang hilang per kolom dengan df.isnull().mean() * 100, membulatkannya hingga dua desimal, dan menampilkannya. Hasil analisis ini kemudian dikompilasi dalam sebuah DataFrame baru bernama missing_summary, yang berisi jumlah serta persentase nilai yang hilang untuk setiap kolom.

```
missing_summary = missing_val(df_accepted)
```

Missing values per column:	
id	0
member_id	2260701
loan_amnt	33
funded_amnt	33

```
funded_amnt_inv           33
                           ...
settlement_status          2226455
settlement_date            2226455
settlement_amount          2226455
settlement_percentage      2226455
settlement_term             2226455
Length: 151, dtype: int64
```

Percentage of missing values per column (%):

```
id                      0.00
member_id                100.00
loan_amnt                0.00
funded_amnt               0.00
funded_amnt_inv           0.00
                           ...
settlement_status          98.49
settlement_date            98.49
settlement_amount          98.49
settlement_percentage      98.49
settlement_term             98.49
Length: 151, dtype: float64
```

Missing Values Summary:

	Missing_Count	Missing_Percentage (%)
id	0	0.00
member_id	2260701	100.00
loan_amnt	33	0.00
funded_amnt	33	0.00
funded_amnt_inv	33	0.00
...
settlement_status	2226455	98.49
settlement_date	2226455	98.49
settlement_amount	2226455	98.49
settlement_percentage	2226455	98.49
settlement_term	2226455	98.49

[151 rows x 2 columns]

missing_summary = missing_val(accepted) berarti memanggil fungsi missing_val(accepted), yang akan menganalisis jumlah dan persentase nilai yang hilang dalam DataFrame accepted. Hasilnya, yang berupa

DataFrame ringkasan (missing_summary), akan disimpan dalam variabel missing_summary.

```
with pd.option_context('display.max_rows', 1000, 'display.width', 1000):
    print(missing_summary)
```



	Missing_Count	Missing_Percentage (%)
id	0	0.00
member_id	2260701	100.00
loan_amnt	33	0.00
funded_amnt	33	0.00
funded_amnt_inv	33	0.00
term	33	0.00
int_rate	33	0.00
installment	33	0.00
grade	33	0.00
sub_grade	33	0.00
emp_title	167002	7.39
emp_length	146940	6.50
home_ownership	33	0.00
annual_inc	37	0.00
verification_status	33	0.00
issue_d	33	0.00
loan_status	33	0.00
pymnt_plan	33	0.00

Menampilkan seluruh isi DataFrame missing_summary tanpa ada bagian yang terpotong. Dengan pd.option_context('display.max_rows', 1000, 'display.width', 1000), pandas sementara mengubah pengaturan agar bisa menampilkan hingga 1000 baris dan memperluas lebar tampilan agar tabel tidak terpotong. Setelah itu, print(missing_summary) digunakan untuk mencetak isi DataFrame yang berisi jumlah dan persentase nilai yang hilang di setiap kolom.

```
high_missing = missing_summary[missing_summary['Missing_Percentage (%)'] >= 51]
high_missing
```



Missing_Count Missing_Percentage (%)

member_id	2260701	100.00
desc	2134636	94.42
mths_since_last_delinq	1158535	51.25
mths_since_last_record	1901545	84.11
next_pymnt_d	1345343	59.51
mths_since_last_major_derog	1679926	74.31
annual_inc_joint	2139991	94.66
dti_joint	2139995	94.66
verification_status_joint	2144971	94.88
mths_since_recent_bc_dlq	1741000	77.01

Menyaring kolom yang memiliki persentase nilai hilang (missing values) lebih dari atau sama dengan 51%. Dengan missing_summary[missing_summary['Missing_Percentage (%)'] >= 51], DataFrame missing_summary hanya akan menampilkan baris yang memiliki persentase missing values minimal 51%. Hasilnya disimpan dalam variabel high_missing, yang kemudian ditampilkan untuk melihat daftar kolom dengan banyak data yang hilang.

```
filtered_summary = missing_summary[
    (missing_summary['Missing_Percentage (%)'] > 0) &
    (missing_summary['Missing_Percentage (%)'] <= 51)
]

#                                                 filtered_summary=
missing_summary[missing_summary['Missing_Percentage (%)'] <= 10]

filtered_summary
```

	Missing_Count	Missing_Percentage (%)
emp_title	167002	7.39
emp_length	146940	6.50
title	23359	1.03
dti	1744	0.08
revol_util	1835	0.08
last_pymnt_d	2460	0.11
collections_12_mths_ex_med	178	0.01
tot_coll_amt	70309	3.11
tot_cur_bal	70309	3.11

Selanjutnya, menyaring kolom dengan missing values dalam rentang tertentu. filtered_summary dibuat untuk menampilkan kolom dengan persentase nilai hilang lebih dari 0% hingga 51%. Alternatifnya, ada filter (dikomentari) untuk menampilkan kolom dengan maksimal 10% missing values. Hasilnya, filtered_summary menampilkan kolom yang masih bisa dipertimbangkan untuk diperbaiki daripada dihapus.

```
high_missing.info()
<class 'pandas.core.frame.DataFrame'>
Index: 44 entries, member_id to settlement_term
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   Missing_Count    44 non-null      int64  
 1   Missing_Percentage (%) 44 non-null      float64 
dtypes: float64(1), int64(1)
memory usage: 1.0+ KB
high_missing.info() menampilkan jumlah baris, kolom, tipe data, dan jumlah nilai non-null dalam DataFrame high_missing. Ini membantu mengevaluasi apakah kolom dengan missing values  $\geq 51\%$  masih bisa digunakan atau perlu dihapus.
```

```
# Get column names with  $\geq 51\%$  missing values
```

```

columns_to_drop = high_missing.index.tolist()

# Verify the list
print("Columns to drop:", columns_to_drop)
Columns to drop: ['member_id', 'desc', 'mths_since_last_delinq',
'mths_since_last_record', 'next_pymnt_d',
'mths_since_last_major_derog', 'annual_inc_joint', 'dti_joint',
'verification_status_joint', 'mths_since_recent_bc_dlq',
'mths_since_recent_revol_delinq', 'revol_bal_joint',
'sec_app_fico_range_low', 'sec_app_fico_range_high',
'sec_app_earliest_cr_line', 'sec_app_inq_last_6mths',
'sec_app_mort_acc', 'sec_app_open_acc', 'sec_app_revol_util',
'sec_app_open_act_il', 'sec_app_num_rev_accts',
'sec_app_chargeoff_within_12_mths',
'sec_app_collections_12_mths_ex_med',
'sec_app_mths_since_last_major_derog', 'hardship_type',
'hardship_reason', 'hardship_status', 'deferral_term',
'hardship_amount', 'hardship_start_date', 'hardship_end_date',
'payment_plan_start_date', 'hardship_length', 'hardship_dpd',
'hardship_loan_status', 'orig_projected_additional_accrued_interest',
'hardship_payoff_balance_amount', 'hardship_last_payment_amount',
'debt_settlement_flag_date', 'settlement_status', 'settlement_date',
'settlement_amount', 'settlement_percentage', 'settlement_term']

```

Setelah itu, ambil nama kolom yang memiliki $\geq 51\%$ missing values dan menyimpannya dalam `columns_to_drop`. Dengan `high_missing.index.tolist()`, daftar nama kolom diperoleh dari index DataFrame `high_missing`.

```

# Drop columns from the original DataFrame (df)
df_accepted_clean = accepted.drop(columns=columns_to_drop)

# Verify remaining columns
print("Remaining columns:", df_accepted_clean.columns.tolist())
Remaining columns: ['id', 'loan_amnt', 'funded_amnt',
'funded_amnt_inv', 'term', 'int_rate', 'installment', 'grade',
'sub_grade', 'emp_title', 'emp_length', 'home_ownership',
'annual_inc', 'verification_status', 'issue_d', 'loan_status',
'pymnt_plan', 'url', 'purpose', 'title', 'zip_code', 'addr_state',
'dti', 'delinq_2yrs', 'earliest_cr_line', 'fico_range_low']

```

```

'fico_range_high',      'inq_last_6mths',      'open_acc',      'pub_rec',
'revol_bal',           'revol_util',          'total_acc',     'initial_list_status',
'out_prncp',           'out_prncp_inv',       'total_pymnt',   'total_pymnt_inv',
'total_rec_prncp',    'total_rec_int',       'total_rec_late_fee',
'recoveries',          'collection_recovery_fee', 'last_pymnt_d',
'last_pymnt_amnt',    'last_credit_pull_d',    'last_fico_range_high',
'last_fico_range_low', 'collections_12_mths_ex_med', 'policy_code',
'application_type',   'acc_now_delinq',       'tot_coll_amt',  'tot_cur_bal',
'open_acc_6m',         'open_act_il',          'open_il_12m',   'open_il_24m',
'mths_since_rcnt_il', 'total_bal_il',         'il_util',      'open_rv_12m',
'open_rv_24m',         'max_bal_bc',          'all_util',     'total_rev_hi_lim',
'inq_fi',              'total_cu_tl',          'inq_last_12m',  'acc_open_past_24mths',
'avg_cur_bal',         'bc_open_to_buy',        'bc_util',
'chargeoff_within_12_mths', 'delinq_amnt',       'mo_sin_old_il_acct',
'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op', 'mo_sin_rcnt_tl',
'mort_acc',             'mths_since_recent_bc', 'mths_since_recent_inq',
'num_accts_ever_120_pd', 'num_actv_bc_tl',       'num_actv_rev_tl',
'num_bc_sats',          'num_bc_tl',            'num_il_tl',    'num_op_rev_tl',
'num_rev_accts',        'num_rev_tl_bal_gt_0',   'num_sats',
'num_tl_120dpd_2m',    'num_tl_30dpd',         'num_tl_90g_dpd_24m',
'num_tl_op_past_12m',  'pct_tl_nvr_dlq',       'percent_bc_gt_75',
'pub_rec_bankruptcies', 'tax_liens',            'tot_hi_cred_lim',
'total_bal_ex_mort',   'total_bc_limit',       'total_il_high_credit_limit',
'hardship_flag',        'disbursement_method', 'debt_settlement_flag']

```

Hapus kolom dengan $\geq 51\%$ missing values dari DataFrame accepted dan menyimpannya dalam df_accepted_clean. Dengan .drop(columns=columns_to_drop), hanya kolom yang masih relevan yang tersisa. Setelah itu, print(df_accepted_clean.columns.tolist()) digunakan untuk menampilkan daftar kolom yang masih ada dalam dataset setelah pembersihan.

```

# Create a DataFrame of column names and dtypes
dtype_df = pd.DataFrame({
    'Column': df_accepted_clean.columns,
    'Data Type': df_accepted_clean.dtypes.values
})
print(dtype_df)

          Column Data Type
0             id      int64
1      loan_amnt    float64

```

```
2           funded_amnt    float64
3           funded_amnt_inv  float64
4             term      object
..          ...
103          total_bc_limit  float64
104  total_il_high_credit_limit  float64
105          hardship_flag   object
106          disbursement_method  object
107          debt_settlement_flag  object
```

[108 rows x 2 columns]

Membuat DataFrame dtype_df yang berisi daftar nama kolom dan tipe datanya dari df_accepted_clean. Dengan df_accepted_clean.columns, semua nama kolom dikumpulkan, dan df_accepted_clean.dtypes.values mengambil tipe data masing-masing kolom. Kemudian, DataFrame ini dicetak untuk melihat struktur data yang tersisa setelah pembersihan.

```
# Temporarily set options for one output
with pd.option_context('display.max_rows', 1000, 'display.width',
1000):
    print(dtype_df)
```

```
→          Column Data Type
0            id      int64
1        loan_amnt  float64
2        funded_amnt  float64
3        funded_amnt_inv  float64
4          term      object
5        int_rate  float64
6      installment  float64
7          grade      object
8        sub_grade      object
9        emp_title      object
10       emp_length      object
11      home_ownership      object
12        annual_inc  float64
13  verification_status      object
14          issue_d      object
15        loan_status      object
```

Mengubah pengaturan tampilan agar dapat menampilkan hingga 1000 baris dan memperluas lebar tampilan. Setelah itu, print(dtype_df) digunakan untuk mencetak daftar kolom beserta tipe datanya, sehingga memudahkan pengecekan struktur dataset setelah proses pembersihan.

```
df_accepted_clean["title"]
```

```
        title
0    Debt consolidation
1        Business
2        NaN
3    Debt consolidation
4    Major purchase
...
25510   Debt consolidation
25511 Credit card refinancing
25512 Credit card refinancing
25513   Debt consolidation
25514        NaN
25515 rows × 1 columns

dtype: object

df_accepted_clean["title"] digunakan untuk mengakses kolom "title" dari DataFrame df_accepted_clean
```

```
# 1. Identify object columns
object_cols = df_accepted_clean.select_dtypes(include=[ 'object' ]).columns

# 2. Calculate missing stats
missing_stats = pd.DataFrame({
    'Missing_Count': df_accepted_clean[object_cols].isnull().sum(),
    'Missing_Percentage (%)': (df_accepted_clean[object_cols].isnull().mean() * 100).round(2)
})

# 3. Print sorted results
print("Missing values in object columns:")
print(missing_stats.sort_values('Missing_Percentage (%)', ascending=False))

# 4. Action: Fill missing values
# df[object_cols] = df[object_cols].fillna("Unknown")
```

	Missing_Count	Missing_Percentage (%)
emp_title	1527	5.98
emp_length	1519	5.95
title	133	0.52
last_pymnt_d	20	0.08
last_credit_pull_d	2	0.01
sub_grade	0	0.00
term	0	0.00
verification_status	0	0.00
home_ownership	0	0.00
grade	0	0.00
issue_d	0	0.00
url	1	0.00
purpose	1	0.00
pymnt_plan	1	0.00
loan_status	1	0.00
addr_state	1	0.00

Di sini dilakukan pengecekan terhadap data yang memiliki persentase missing values yang rendah untuk menilai dampaknya jika dihapus. Setelah dianalisis, ternyata penghapusan data tersebut hanya mempengaruhi sekitar 5% dari keseluruhan dataset. Dengan total data yang mencapai ratusan ribu hingga jutaan, dampak ini tergolong kecil, sehingga penghapusan data dengan missing values rendah dianggap tidak terlalu berisiko terhadap kualitas analisis secara keseluruhan.

```
# Get columns where missing percentage < 10%
low_missing_cols = missing_summary[missing_summary['Missing_Percentage (%)'] < 5].index.tolist()
print("Columns with <10% missing values:", low_missing_cols)
Columns with <10% missing values: ['id', 'loan_amnt', 'funded_amnt',
'funded_amnt_inv', 'term', 'int_rate', 'installment', 'grade',
'sub_grade', 'home_ownership', 'annual_inc', 'verification_status',
'issue_d', 'loan_status', 'pymnt_plan', 'url', 'purpose', 'title',
'zip_code', 'addr_state', 'dti', 'delinq_2yrs', 'earliest_cr_line',
'fico_range_low', 'fico_range_high', 'inq_last_6mths', 'open_acc',
'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
'initial_list_status', 'out_prncp', 'out_prncp_inv', 'total_pymnt',
'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
'last_fico_range_high', 'last_fico_range_low',
'collections_12_mths_ex_med', 'policy_code', 'application_type',
```

```
'acc_now_delinq', 'tot_coll_amt', 'tot_cur_bal', 'total_rev_hi_lim',
'acc_open_past_24mths', 'avg_cur_bal', 'bc_open_to_buy', 'bc_util',
'chargeoff_within_12_mths', 'delinq_amnt', 'mo_sin_old_il_acct',
'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op', 'mo_sin_rcnt_tl',
'mort_acc', 'mths_since_recent_bc', 'num_accts_ever_120_pd',
'num_actv_bc_tl', 'num_actv_rev_tl', 'num_bc_sats', 'num_bc_tl',
'num_il_tl', 'num_op_rev_tl', 'num_rev_accts', 'num_rev_tl_bal_gt_0',
'num_sats', 'num_tl_120dpd_2m', 'num_tl_30dpd', 'num_tl_90g_dpd_24m',
'num_tl_op_past_12m', 'pct_tl_nvr_dlq', 'percent_bc_gt_75',
'pub_rec_bankruptcies', 'tax_liens', 'tot_hi_cred_lim',
'total_bal_ex_mort', 'total_bc_limit', 'total_il_high_credit_limit',
'hardship_flag', 'disbursement_method', 'debt_settlement_flag']
```

Dengan menyaring kolom berdasarkan persentase nilai yang hilang, hanya kolom yang masih memiliki sebagian besar datanya yang akan diambil. Nama kolom yang memenuhi kriteria kemudian dikonversi menjadi daftar menggunakan .index.tolist(), sehingga dapat digunakan untuk analisis lebih lanjut atau pembersihan data. Terakhir, daftar kolom tersebut ditampilkan agar bisa diverifikasi. Jika ingin mencari kolom dengan kurang dari 10% missing values, cukup ubah angka 5 menjadi 10.

```
# Create a boolean mask for rows with missing values in ANY
low_missing column
rows_to_drop_mask = df_accepted_clean[low_missing_cols].isnull().any(axis=1)

# Count impacted rows
impacted_rows = rows_to_drop_mask.sum()
total_rows = len(df_accepted_clean)
print(f"Rows to drop: {impacted_rows} ({(impacted_rows / total_rows * 100):.2f}% of total)")

Rows to drop: 2078 (8.14% of total)
```

Hapus baris yang memiliki missing values dengan dampak rendah terhadap keseluruhan data.

```
df_accepted_clean = df_accepted_clean.dropna(subset=low_missing_cols)
print(f"New DataFrame shape: {df_accepted_clean.shape}")
```

```
New DataFrame shape: (23437, 108)
```

Buat dataframe baru, dan cetak ukuran dari dataframe baru tersebut.

```
missing_summary = missing_val(df_accepted_clean)
```

```
Missing values per column:  
id                      0  
loan_amnt                0  
funded_amnt              0  
funded_amnt_inv          0  
term                     0  
..  
total_bc_limit            0  
total_il_high_credit_limit 0  
hardship_flag              0  
disbursement_method        0  
debt_settlement_flag       0  
Length: 108, dtype: int64  
  
Percentage of missing values per column (%):  
id                      0.0  
loan_amnt                0.0  
funded_amnt              0.0  
funded_amnt_inv          0.0  
term                     0.0  
..  
total_bc_limit            0.0  
total_il_high_credit_limit 0.0  
hardship_flag              0.0  
disbursement_method        0.0  
debt_settlement_flag       0.0  
Length: 108, dtype: float64
```

Berisi jumlah nilai hilang dan persentasenya untuk setiap kolom. Data ini berguna untuk menganalisis tingkat missing values sebelum melakukan pembersihan atau imputasi data.

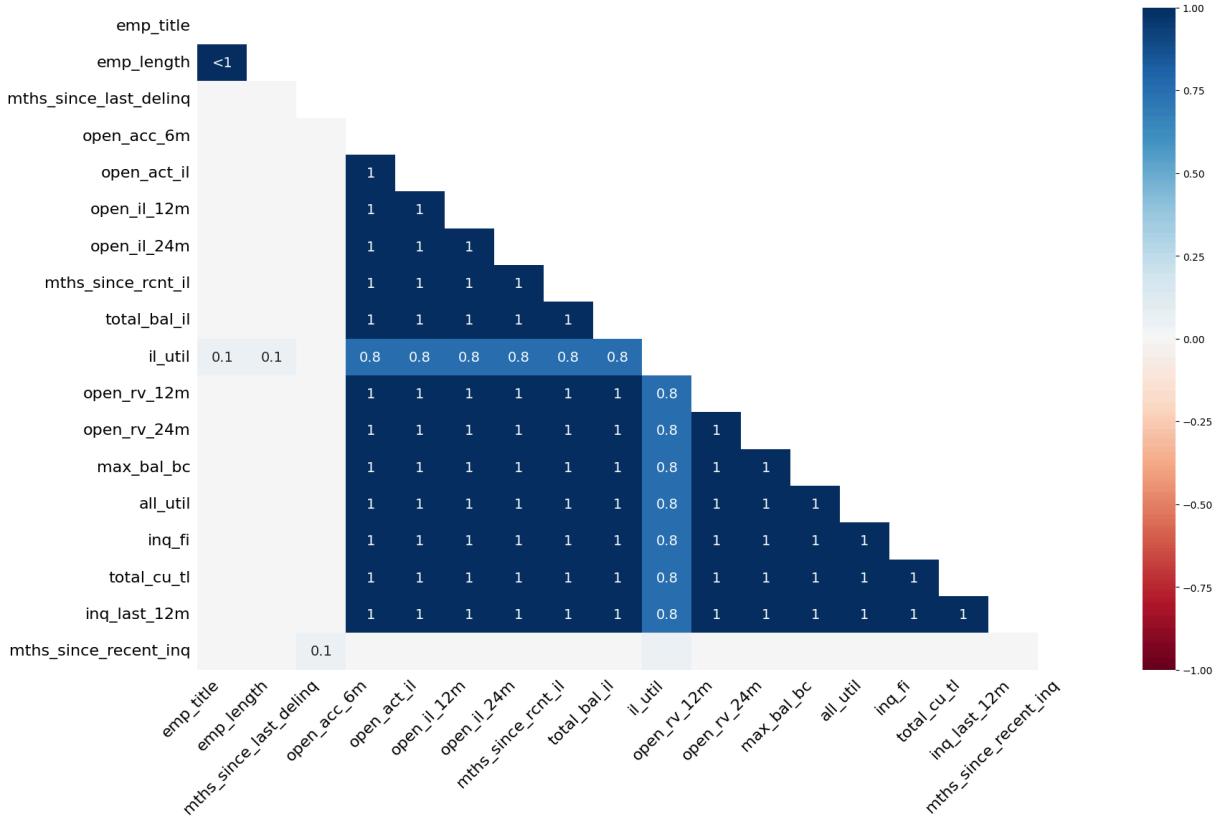
```
filtered_summary= missing_summary[missing_summary['Missing_Percentage  
(%)'] > 0]  
filtered_summary
```

	Missing_Count	Missing_Percentage (%)
emp_title	1333	5.69
emp_length	1329	5.67
mths_since_last_delinq	11083	47.29
open_acc_6m	3859	16.47
open_act_il	3859	16.47
open_il_12m	3859	16.47
open_il_24m	3859	16.47
mths_since_rcnt_il	3859	16.47
total_bal_il	3859	16.47
il_util	6037	25.76
open_rv_12m	3859	16.47
open_rv_24m	3859	16.47
max_bal_bc	3859	16.47
all_util	3859	16.47

Untuk kolom dengan missing values antara 5%-51%, terdapat banyak nilai yang identik. Oleh karena itu, perlu dilakukan evaluasi lebih lanjut untuk menentukan apakah seluruh kolom perlu dihapus atau cukup mempertahankan salah satu yang paling representatif.

```
# Visualize missing value patterns
```

```
msno.heatmap(df_accepted_clean)
```



Dapat dilihat bahwa banyak kolom memiliki nilai yang sama dengan kolom lainnya. Hal ini menunjukkan adanya dua kemungkinan korelasi terkait data yang hilang.

Sebagai contoh, jika suatu entri memiliki missing value di kolom `open_acc_6m`, maka kemungkinan besar kolom `inq_last_12m` juga memiliki missing value.

```
overlap = df_accepted_clean[['open_acc_6m',  
'inq_last_12m']].isnull().all(axis=1).sum()  
print(f"Rows where both are missing: {overlap}")  
Rows where both are missing: 3859
```

Selanjutnya, hitung jumlah baris di `df_accepted_clean` di mana kedua kolom "`open_acc_6m`" dan "`inq_last_12m`" sama-sama memiliki missing values. Dengan mengecek nilai `Nan` di kedua kolom lalu menjumlahkannya, diperoleh **3859** baris yang memenuhi kondisi ini, yang bisa menjadi pertimbangan dalam pembersihan data.

Setelah menemukan korelasi antara kolom-kolom yang memiliki missing values, langkah selanjutnya adalah mengelompokkan kolom-kolom tersebut berdasarkan hubungan yang ditemukan. Setelah dikelompokkan, visualisasi dalam bentuk grafik akan dibuat untuk mempermudah analisis.

Sebagai contoh, jika kolom A memiliki korelasi dengan kolom B, dan kolom B juga memiliki korelasi dengan kolom C, maka ketiga kolom tersebut dapat dikelompokkan dalam satu grup. Hal ini menunjukkan adanya redundansi dalam data, sehingga dapat dipertimbangkan apakah semua kolom perlu dipertahankan atau cukup menyimpan salah satu yang paling representatif.

```
# Step 1: Compute missingness correlation  
missing_corr = df_accepted_clean.isnull().corr()  
  
# Step 2: Find all column pairs with correlation = 1  
perfect_corr_pairs = []  
cols = missing_corr.columns
```

```

for i in range(len(cols)):

    for j in range(i + 1, len(cols)):

        if missing_corr.iloc[i, j] == 1:

            perfect_corr_pairs.append((cols[i], cols[j]))

# Step 3: Group redundant columns

G = nx.Graph()

G.add_edges_from(perfect_corr_pairs)

redundant_groups = list(nx.connected_components(G))

# Step 4: Print results

print("Redundant column groups (drop all but one per group):")

for group in redundant_groups:

    print(f" - {list(group)}")

```

Redundant column groups (drop all but one per group):

- ['all_util', 'open_il_12m', 'inq_hi', 'open_rv_12m', 'max_bal_bc', 'mths_since_rcnt_il', 'open_rv_24m', 'open_act_il', 'inq_last_12m', 'total_cu_tl', 'open_il_24m', 'total_bal_il']

```

redundant_groups
[{'all_util',
 'inq_hi',
 'inq_last_12m',
 'max_bal_bc',
 'mths_since_rcnt_il',
 'open_act_il',
 'open_il_12m',
 'open_il_24m',
 'open_rv_12m',
 'open_rv_24m',
 'total_bal_il',
 'total_cu_tl'}]

```

Kolom-kolom yang teridentifikasi sebagai redundant akan dihapus untuk meminimalkan beban dataset. Dengan menghapus kolom yang memiliki korelasi tinggi dengan kolom lainnya, dataset menjadi lebih efisien tanpa kehilangan informasi yang signifikan.

Hal ini juga membantu dalam meningkatkan performa pemrosesan data serta mengurangi kompleksitas analisis selanjutnya.

```
columns_to_drop = []

for group in redundant_groups:
    group_list = list(group)
    keeper = group_list[0] # Keep the first column in the group
    columns_to_drop.extend(group_list[1:]) # Drop the rest

# Step 5: Drop redundant columns
df_accepted_clean = df_accepted_clean.drop(columns=columns_to_drop)

print(f"Dropped columns: {columns_to_drop}")
print(f"Remaining columns: {df_accepted_clean.columns.tolist()}")

Dropped columns: ['open_il_12m', 'inq_fi', 'open_rv_12m',
'max_bal_bc', 'mths_since_rcnt_il', 'open_rv_24m', 'open_act_il',
'inq_last_12m', 'total_cu_tl', 'open_il_24m', 'total_bal_il']
Remaining columns: ['id', 'loan_amnt', 'funded_amnt',
'funded_amnt_inv', 'term', 'int_rate', 'installment', 'grade',
'sub_grade', 'emp_title', 'emp_length', 'home_ownership',
'annual_inc', 'verification_status', 'issue_d', 'loan_status',
'pymnt_plan', 'url', 'purpose', 'title', 'zip_code', 'addr_state',
'dti', 'delinq_2yrs', 'earliest_cr_line', 'fico_range_low',
'fico_range_high', 'inq_last_6mths', 'mths_since_last_delinq',
'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
'initial_list_status', 'out_prncp', 'out_prncp_inv', 'total_pymnt',
'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
'last_fico_range_high', 'last_fico_range_low',
'collections_12_mths_ex_med', 'policy_code', 'application_type',
'acc_now_delinq', 'tot_coll_amt', 'tot_cur_bal', 'il_util',
'all_util', 'total_rev_hi_lim', 'acc_open_past_24mths',
'avg_cur_bal', 'bc_open_to_buy', 'bc_util',
```

```
'chargeoff_within_12_mths',      'delinq_amnt',      'mo_sin_old_il_acct',
'mo_sin_old_rev_tl_op',        'mo_sin_rcnt_rev_tl_op',   'mo_sin_rcnt_tl',
'mort_acc',          'mths_since_recent_bc',    'mths_since_recent_inq',
'num_accts_ever_120_pd',       'num_actv_bc_tl',     'num_actv_rev_tl',
'num_bc_sats',        'num_bc_tl',        'num_il_tl',        'num_op_rev_tl',
'num_rev_accts',        'num_rev_tl_bal_gt_0',    'num_sats',
'num_tl_120dpd_2m',        'num_tl_30dpd',      'num_tl_90g_dpd_24m',
'num_tl_op_past_12m',       'pct_tl_nvr_dlq',     'percent_bc_gt_75',
'pub_rec_bankruptcies',      'tax_liens',       'tot_hi_cred_lim',
'total_bal_ex_mort',        'total_bc_limit',   'total_il_high_credit_limit',
'hardship_flag',        'disbursement_method', 'debt_settlement_flag']
```

Proses ini bertujuan untuk menghapus kolom redundan dalam df_accepted_clean, di mana hanya satu kolom dari setiap kelompok yang dipertahankan, sementara sisanya dihapus. Daftar kolom yang akan dihapus dikumpulkan dalam columns_to_drop, lalu diterapkan pada DataFrame menggunakan .drop(columns=columns_to_drop). Setelahnya, daftar kolom yang dihapus serta kolom yang tersisa ditampilkan untuk memastikan pembersihan data telah dilakukan dengan benar.

```
missing_summary = missing_val(df_accepted_clean)
```

```
Missing values per column:
id                0
loan_amnt         0
funded_amnt       0
funded_amnt_inv   0
term               0
...
total_bc_limit    0
total_il_high_credit_limit 0
hardship_flag      0
disbursement_method 0
debt_settlement_flag 0
Length: 96, dtype: int64

Percentage of missing values per column (%):
id              0.0
loan_amnt        0.0
funded_amnt      0.0
funded_amnt_inv  0.0
term             0.0
...
total_bc_limit    0.0
total_il_high_credit_limit 0.0
hardship_flag      0.0
disbursement_method 0.0
debt_settlement_flag 0.0
Length: 96, dtype: float64
```

Hitung jumlah dan persentase missing values di setiap kolom dalam df_accepted_clean. Hasilnya disimpan dalam missing_summary, yang berisi ringkasan jumlah nilai yang hilang dan persentasenya untuk setiap kolom.

```
filtered_summary= missing_summary[missing_summary['Missing_Percentage
(%)'] > 0]
filtered_summary
```

	Missing_Count	Missing_Percentage (%)
emp_title	1333	5.69
emp_length	1329	5.67
mths_since_last_delinq	11083	47.29
il_util	6037	25.76
all_util	3859	16.47
mths_since_recent_inq	2459	10.49

Hasil filtering dari missing_summary, di mana hanya kolom dengan nilai yang hilang ($>0\%$) yang ditampilkan. Data ini mencakup jumlah dan persentase nilai yang hilang untuk setiap kolom, yang berguna untuk menganalisis bagian mana dari dataset yang perlu diproses lebih lanjut

```
missing_cols = df_accepted_clean.columns[df_accepted_clean.isnull().any()].tolist()
# Get data types for columns with missing values
print(df_accepted_clean[missing_cols].dtypes)
```

emp_title	object
emp_length	object
mths_since_last_delinq	float64
il_util	float64
all_util	float64
mths_since_recent_inq	float64
dtype:	object

Dari hasilnya, terdapat enam kolom dengan nilai yang hilang, di mana dua di antaranya bertipe object (emp_title dan emp_length), sementara empat lainnya bertipe float64.

```
df_accepted_clean.loc[df_accepted_clean['emp_length'].isnull(),
['emp_title', 'emp_length']]
```

	emp_title	emp_length
55	NaN	NaN
75	NaN	NaN
93	NaN	NaN
139	NaN	NaN
141	NaN	NaN
...
25414	NaN	NaN
25422	NaN	NaN
25457	NaN	NaN
25462	NaN	NaN
25471	Technician	NaN

1329 rows × 2 columns

Sisa kolom yang masih memiliki nilai kosong akan diproses melalui imputasi. Untuk kolom dengan tipe data kategorikal atau objek, nilai yang hilang akan diisi dengan kategori baru agar tidak ada informasi yang terbuang.

Sementara itu, untuk kolom dengan tipe data numerik, nilai yang hilang akan digantikan dengan median dari kolom tersebut guna menjaga distribusi data tetap stabil dan menghindari bias yang dapat terjadi jika menggunakan nilai rata-rata.

```
# Treat missingness as a separate category
```

```
df_accepted_clean['emp_title'].fillna('Unknown', inplace=True)
```

Kode ini menangani data hilang pada kolom 'emp_title' dengan menggantinya menjadi 'Unknown'. Pendekatan ini menjaga data tetap utuh tanpa menghapus baris, serta memungkinkan analisis lebih lanjut tanpa kehilangan informasi penting.

```
df_accepted_clean['mo_sin_old_il_acct'].fillna(df_accepted_clean['mo_sin_old_il_acct'].median(), inplace=True)
```

```
df_accepted_clean['mths_since_recent_inq'].fillna(df_accepted_clean['mths_since_recent_inq'].median(), inplace=True)
```

```
df_accepted_clean['num_tl_120dpd_2m'].fillna(df_accepted_clean['num_tl_120dpd_2m'].median(), inplace=True)
```

Menangani data hilang pada kolom 'mo_sin_old_il_acct', 'mths_since_recent_inq', dan 'num_tl_120dpd_2m' dengan menggantinya menggunakan nilai median masing-masing kolom.

```
df_accepted_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2137072 entries, 0 to 2260697
Data columns (total 96 columns):
 #   Column           Dtype  
 --- 
 0   id               object  
 1   loan_amnt        float64 
 2   funded_amnt      float64 
 3   funded_amnt_inv  float64 
 4   term              object  
 5   int_rate          float64 
 6   installment       float64 
 7   grade             object  
 8   sub_grade          object  
 9   emp_title          object  
 10  emp_length         object  
 11  home_ownership    object  
 12  annual_inc        float64
```

```
numeric_cols = df_accepted_clean.select_dtypes(include=[ 'number' ]).columns.tolist()
print("Numeric columns:", numeric_cols)

Numeric columns: ['loan_amnt', 'funded_amnt', 'funded_amnt_inv',
'int_rate', 'installment', 'annual_inc', 'dti', 'delinq_2yrs',
'fico_range_low', 'fico_range_high', 'inq_last_6mths', 'open_acc',
'pub_rec', 'revol_bal', 'revol_util', 'total_acc', 'out_prncp',
'out_prncp_inv', 'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp',
'total_rec_int', 'total_rec_late_fee', 'recoveries',
'collection_recovery_fee', 'last_pymnt_amnt', 'last_fico_range_high',
'last_fico_range_low', 'collections_12_mths_ex_med', 'policy_code',
'acc_now_delinq', 'tot_coll_amt', 'tot_cur_bal',
'mths_since_rcnt_il', 'il_util', 'open_rv_12m', 'total_rev_hi_lim',
'acc_open_past_24mths', 'avg_cur_bal', 'bc_open_to_buy', 'bc_util',
'chargeoff_within_12_mths', 'delinq_amnt', 'mo_sin_old_il_acct',
'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op', 'mo_sin_rcnt_tl',
'mort_acc', 'mths_since_recent_bc', 'mths_since_recent_inq',
'num_accts_ever_120_pd', 'num_actv_bc_tl', 'num_actv_rev_tl',
'num_bc_sats', 'num_bc_tl', 'num_il_tl', 'num_op_rev_tl',
```

```
'num_rev_accts',           'num_rev_tl_bal_gt_0',           'num_sats',
'num_tl_120dpd_2m',         'num_tl_30dpd',             'num_tl_90g_dpd_24m',
'num_tl_op_past_12m',       'pct_tl_nvr_dlq',          'percent_bc_gt_75',
'pub_rec_bankruptcies',    'tax_liens',              'tot_hi_cred_lim',
'total_bal_ex_mort',       'total_bc_limit',         'total_il_high_credit_limit']
```

Memilih semua kolom numerik dalam df_accepted_clean menggunakan select_dtypes(include=['number']) dan menyimpannya dalam daftar numeric_cols. Kemudian, daftar tersebut dicetak untuk menampilkan nama-nama kolom numerik dalam dataset.

```
missing_summary = missing_val(df_accepted_clean)
```

```
Missing values per column:  
id                      0  
loan_amnt                0  
funded_amnt               0  
funded_amnt_inv            0  
term                      0  
..  
total_bc_limit              0  
total_il_high_credit_limit 0  
hardship_flag                0  
disbursement_method            0  
debt_settlement_flag            0  
Length: 96, dtype: int64
```

```
Percentage of missing values per column (%):  
id                      0.0  
loan_amnt                0.0  
funded_amnt               0.0  
funded_amnt_inv            0.0  
term                      0.0
```

Menghitung jumlah dan persentase nilai yang hilang dalam setiap kolom pada dataset df_accepted_clean. Dari output tersebut, terlihat bahwa tidak ada nilai yang hilang dalam 96 kolom yang ditampilkan, karena semua nilai nol baik dalam jumlah absolut maupun persentase.

Dan setelah nya, dataset sudah bersih. Lakukan langkah yang sama untuk file csv lainnya.

```
rejected_2007_to_2018Q4.csv
```

```
rejected.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 461127 entries, 0 to 461126  
Data columns (total 9 columns):  
 #   Column           Non-Null Count  Dtype     
 ---  ----             
 0   amount_requested  461127 non-null  float64  
 1   application_date 461126 non-null  datetime64[ns]
```

```
2   loan_title          461113 non-null  object
3   risk_score           440940 non-null  float64
4   debt-to-income_ratio 461126 non-null  object
5   zip_code              461105 non-null  object
6   state                  461105 non-null  object
7   employment_length      456107 non-null  object
8   policy_code            461126 non-null  float64
dtypes: datetime64[ns](1), float64(3), object(5)
memory usage: 31.7+ MB
```

```
missing_summary = missing_val(rejected)
```

```
Missing values per column:
```

```
amount_requested          0
application_date           1
loan_title                 14
risk_score                  20187
debt-to-income_ratio        1
zip_code                     22
state                       22
employment_length           5020
policy_code                  1
dtype: int64
```

Hasil analisis nilai yang hilang dalam DataFrame rejected. Dari 9 kolom, risk_score memiliki jumlah nilai hilang terbesar, yaitu 20.187 atau sekitar 4,38%. employment_length juga memiliki 5.020 nilai hilang (1,09%). Kolom lain seperti loan_title, zip_code, dan state memiliki sedikit nilai yang hilang. Sementara itu, kolom amount_requested tidak memiliki nilai yang hilang sama sekali.

```
missing_summary[missing_summary['Missing_Count'] > 0]
```

	Missing_Count	Missing_Percentage (%)
loan_title	1305	0.00
risk_score	18497630	66.90
zip_code	293	0.00
state	22	0.00
employment_length	951355	3.44
policy_code	918	0.00

Menunjukkan jumlah dan persentase nilai yang hilang untuk beberapa kolom dalam missing_summary. Kolom risk_score memiliki jumlah nilai hilang terbesar, yaitu 18.497.630 atau sekitar 66,90%. employment_length juga memiliki banyak nilai hilang, yaitu 951.355 (3,44%). Kolom lain seperti loan_title, zip_code, state, dan policy_code memiliki nilai hilang dalam jumlah lebih kecil.

```
rejected['risk_score']
```

```
    risk_score
0      693.0
1      703.0
2      715.0
3      698.0
4      509.0
...
...
```

Untuk menampilkan kolom risk score

```
rejected['employment_length']
```

```
    employment_length
0          4 years
1        < 1 year
2          1 year
3        < 1 year
4        < 1 year
...
...
```

Untuk menampilkan kolom employment length

Risk Score dan Employment Length dapat diimputasi menggunakan nilai median atau mode karena kedua metode ini efektif dalam menangani missing values berdasarkan karakteristik distribusi data.

```
# First fill NA values in the column
```

```
df_rejected['Risk_Score'].fillna(df_rejected['Risk_Score'].median(),
inplace=True)
```

Pengisian nilai NA pada Risk_Score menggunakan median karena lebih robust terhadap outlier dibandingkan mean, terutama jika distribusi data tidak normal. Median dipilih agar nilai yang diimputasi lebih representatif tanpa terpengaruh oleh nilai ekstrem. Jika Risk_Score bersifat kategorikal atau memiliki nilai terbatas, mode bisa menjadi alternatif yang lebih sesuai.

```
missing_summary = missing_val(rejected)
```

```
Missing values per column:  
Amount Requested      0  
Application Date      0  
Loan Title            1305  
Risk_Score             0  
Debt-To-Income Ratio  0  
Zip Code               293  
State                  22  
Employment Length     0  
Policy Code            918  
dtype: int64
```

```
missing_summary[missing_summary['Missing_Count'] > 0]
```

	Missing_Count	Missing_Percentage (%)
Loan Title	1305	0.0
Zip Code	293	0.0
State	22	0.0
Policy Code	918	0.0

Missing values dengan persentase yang sangat rendah, yaitu di bawah 1% dari keseluruhan data, dapat dipertimbangkan untuk dihapus karena dampaknya terhadap analisis sangat minim, terutama mengingat jumlah data yang sangat besar.

```
# Get columns where missing percentage < 10%
low_missing_cols = missing_summary[missing_summary['Missing_Count'] > 0].index.tolist()
print("Columns with <10% missing values:", low_missing_cols)
Columns with <10% missing values: ['Loan Title', 'Zip Code', 'State',
'Policy Code']

# Create a boolean mask for rows with missing values in ANY
# low-missing column
rows_to_drop_mask = df_rejected[low_missing_cols].isnull().any(axis=1)

# Count impacted rows
impacted_rows = rows_to_drop_mask.sum()
total_rows = len(df_rejected)
```

```
print(f"Rows to drop: {impacted_rows} ({(impacted_rows / total_rows * 100):.2f}% of total)")

Rows to drop: 2516 (0.01% of total)
```

Baris dengan nilai yang hilang pada kolom dengan sedikit missing values diidentifikasi menggunakan mask boolean. Jumlah baris yang terdampak dihitung dan dibandingkan dengan total data, menunjukkan bahwa hanya 0,01% dari total baris yang akan dihapus, sehingga dampaknya terhadap keseluruhan dataset minimal.

```
df_rejected_clean = df_rejected.dropna(subset=low_missing_cols)

print(f"New DataFrame shape: {df_rejected_clean.shape}")

New DataFrame shape: (27646225, 9)
```

Baris dengan nilai yang hilang pada kolom dengan sedikit missing values dihapus untuk menjaga kualitas data. Setelah pembersihan, jumlah baris dalam `df_rejected_clean` menjadi 27.646.225, memastikan hanya data yang lebih lengkap dan andal yang dipertahankan.

```
missing_summary = missing_val(df_rejected_clean)
missing_summary[missing_summary['Missing_Count'] > 0]

Missing values per column:
Amount Requested      0
Application Date      0
Loan Title             0
Risk_Score              0
Debt-To-Income Ratio   0
Zip Code                0
State                   0
Employment Length       0
Policy Code              0
dtype: int64
```

Setelah proses imputasi dan penghapusan data yang memiliki nilai hilang, semua kolom dalam `df_rejected_clean` kini tidak memiliki nilai yang hilang sama sekali. Ini menandakan bahwa data sudah bersih dan siap untuk analisis lebih lanjut tanpa perlu menangani missing values lagi.

```
#Dataset Home Credit Default Risk Data
```

```
application_train.csv
```

```
# train.info()
```

```

# Create a DataFrame of column names and dtypes
dtype_df = pd.DataFrame({
    'Column': train.columns,
    'Data Type': train.dtypes.values
})
# print(dtype_df)

# Temporarily set options for one output
with pd.option_context('display.max_rows', 1000, 'display.width',
1000):
    print(dtype_df)

```

	Column	Data Type
0	SK_ID_CURR	int64
1	TARGET	int64
2	NAME_CONTRACT_TYPE	object
3	CODE_GENDER	object
4	FLAG_OWN_CAR	object
5	FLAG_OWN_REALTY	object
6	CNT_CHILDREN	int64
7	AMT_INCOME_TOTAL	float64
8	AMT_CREDIT	float64
9	AMT_ANNUITY	float64
10	AMT_GOODS_PRICE	float64

Datanya dioptimalkan dengan menurunkan format integer atau float secara otomatis sesuai kebutuhan, sehingga penggunaan memori lebih efisien tanpa mengubah isi data.

```

# Select only numeric columns (int64 and float64)
numeric_cols          =      train.select_dtypes(include=['int64',
'float64']).columns

# Downcast numeric columns
train[numeric_cols] = train[numeric_cols].apply(
    pd.to_numeric, downcast='integer'
)

# For float columns (separate step to avoid mixing types)

```

```
float_cols = train.select_dtypes(include=['float64']).columns
train[float_cols] = train[float_cols].apply(
    pd.to_numeric, downcast='float'
)
```

Pertama, kolom dengan tipe data `int64` dan `float64` dipilih. Kemudian, kolom integer dikonversi ke tipe data yang lebih kecil seperti `int32` atau `int16` sesuai kebutuhan. Setelah itu, kolom `float64` diproses secara terpisah untuk dikonversi ke tipe `float32` atau lebih kecil.

Lakukan langkah yang sama dengan dataset sebelumnya, yaitu, melakukan pengecekan missing values beserta persentasenya.

```
# df_application_train.info()

# Create a DataFrame of column names and dtypes
dtype_df = pd.DataFrame({
    'Column': df_application_train.columns,
    'Data Type': df_application_train.dtypes.values
})
# print(dtype_df)

# Temporarily set options for one output
with pd.option_context('display.max_rows', 1000, 'display.width',
1000):
    print(dtype_df)

      Column Data Type
0   SK_ID_CURR     int32
1       TARGET     int8
2 NAME_CONTRACT_TYPE    object
3   CODE_GENDER    object
4  FLAG_OWN_CAR    object
5  FLAG_OWN_REALTY    object
6   CNT_CHILDREN     int8
7 AMT_INCOME_TOTAL   float64
8      AMT_CREDIT    float32
9     AMT_ANNUITY    float32
10    AMT_GOODS_PRICE    float32

missing_summary = missing_val(train)
```

missing_summary

```
Missing values per column:  
SK_ID_CURR           0  
TARGET               0  
NAME_CONTRACT_TYPE  0  
CODE_GENDER          0  
FLAG_OWN_CAR         0  
...  
AMT_REQ_CREDIT_BUREAU_DAY 41519  
AMT_REQ_CREDIT_BUREAU_WEEK 41519  
AMT_REQ_CREDIT_BUREAU_MON 41519  
AMT_REQ_CREDIT_BUREAU_QRT 41519  
AMT_REQ_CREDIT_BUREAU_YEAR 41519  
Length: 122, dtype: int64
```

Ringkasan nilai yang hilang dalam dataset `df_application_train` menunjukkan bahwa beberapa kolom, seperti `SK_ID_CURR`, `TARGET`, `NAME_CONTRACT_TYPE`, `CODE_GENDER`, dan `FLAG_OWN_CAR`, tidak memiliki nilai yang hilang. Namun, terdapat kolom yang memiliki jumlah nilai hilang cukup signifikan, terutama yang berkaitan dengan `AMT_REQ_CREDIT_BUREAU` (DAY, WEEK, MON, QRT, YEAR), masing-masing dengan 41.519 nilai yang hilang. Informasi ini penting untuk menentukan strategi yang tepat dalam menangani data yang hilang, baik dengan imputasi maupun penghapusan, guna memastikan kualitas dataset tetap optimal untuk analisis lebih lanjut.

Melakukan pengecekan missing values yang melebihi 51%.

```
missing_summary[missing_summary['Missing_Percentage (%)'] >= 51]
```

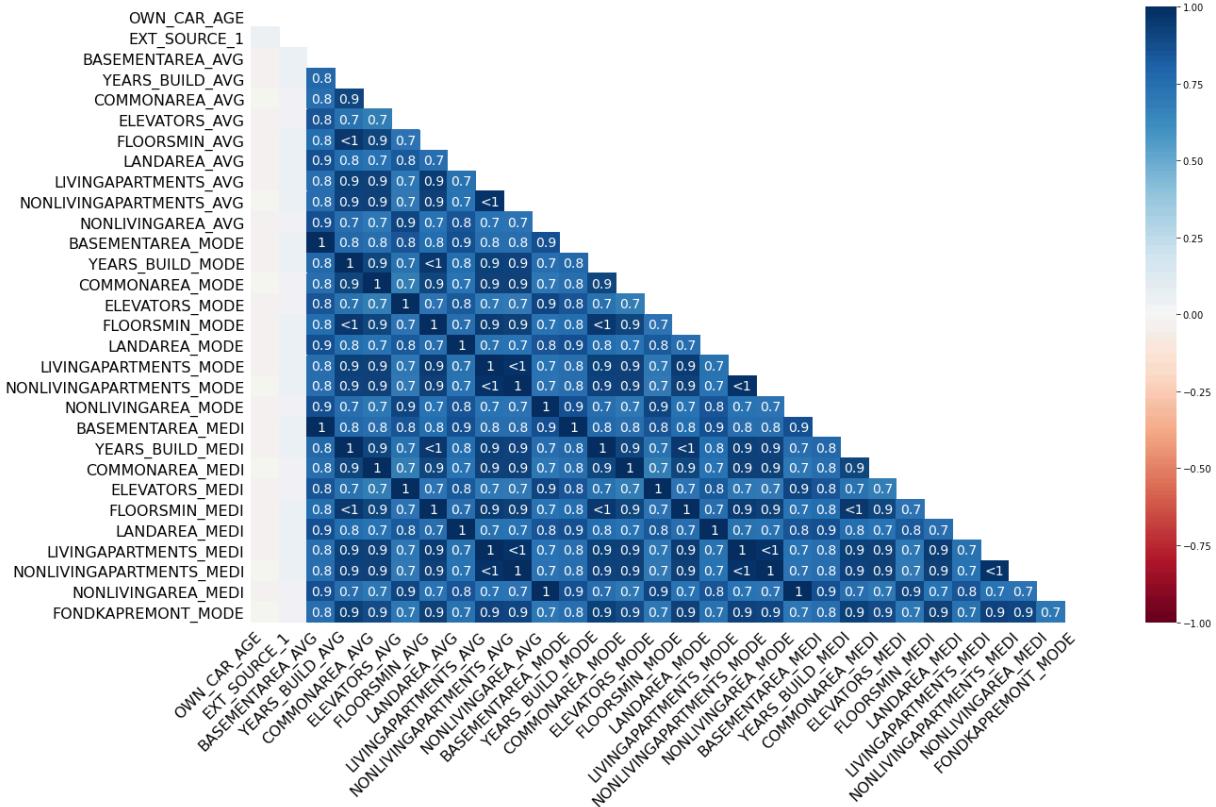
	Missing_Count	Missing_Percentage (%)
OWN_CAR_AGE	202929	65.99
EXT_SOURCE_1	173378	56.38
BASEMENTAREA_AVG	179943	58.52
YEARS_BUILD_AVG	204488	66.50
COMMONAREA_AVG	214865	69.87
ELEVATORS_AVG	163891	53.30
FLOORSMIN_AVG	208642	67.85
LANDAREA_AVG	182590	59.38

Beberapa kolom dalam dataset memiliki lebih dari 51% data yang hilang, seperti `OWN_CAR_AGE`, `EXT_SOURCE_1`, dan `COMMONAREA_AVG`. Karena jumlahnya cukup besar, perlu dipertimbangkan apakah kolom tersebut akan diimputasi atau dihapus, tergantung pada relevansinya dengan analisis.

```
# Get column names with ≥51% missing values  
columns_to_drop = missing_summary[missing_summary['Missing_Percentage (%)'] >= 51].index.tolist()
```

Mendapatkan daftar nama kolom yang memiliki **lebih dari atau sama dengan 51%** data yang hilang. Kolom-kolom ini kemudian bisa dipertimbangkan untuk dihapus agar dataset menjadi lebih bersih dan efisien.

```
# Visualize missing value patterns
msno.heatmap(df_application_train[columns_to_drop])
```



```
# Calculate pairwise missingness correlations
missing_corr = df_application_train.isnull().corr()

# Get column pairs where missingness correlation = 1
perfect_pairs = []
cols = missing_corr.columns

for i in range(len(cols)):
    for j in range(i + 1, len(cols)):      # Avoid duplicates and self-comparison
```

```

if missing_corr.iloc[i, j] == 1.0:
    perfect_pairs.append((cols[i], cols[j]))


# Print results
print("Columns always missing together (heatmap=1):")
for pair in perfect_pairs:
    print(f"- {pair[0]} ↔ {pair[1]}")

Columns always missing together (heatmap=1):
- APARTMENTS_AVG ↔ APARTMENTS_MODE
- APARTMENTS_AVG ↔ APARTMENTS_MEDI
- BASEMENTAREA_AVG ↔ BASEMENTAREA_MODE
- BASEMENTAREA_AVG ↔ BASEMENTAREA_MEDI
- YEARS_BEGINEXPLUATATION_AVG ↔ YEARS_BEGINEXPLUATATION_MODE
- YEARS_BEGINEXPLUATATION_AVG ↔ YEARS_BEGINEXPLUATATION_MEDI
- YEARS_BUILD_AVG ↔ YEARS_BUILD_MODE
- YEARS_BUILD_AVG ↔ YEARS_BUILD_MEDI
- COMMONAREA_AVG ↔ COMMONAREA_MODE
- COMMONAREA_AVG ↔ COMMONAREA_MEDI
- ELEVATORS_AVG ↔ ELEVATORS_MODE
- ELEVATORS_AVG ↔ ELEVATORS_MEDI
- ENTRANCES_AVG ↔ ENTRANCES_MODE
- ENTRANCES_AVG ↔ ENTRANCES_MEDI
- FLOORSMAX_AVG ↔ FLOORSMAX_MODE
- FLOORSMAX_AVG ↔ FLOORSMAX_MEDI
- FLOORSMIN_AVG ↔ FLOORSMIN_MODE
- FLOORSMIN_AVG ↔ FLOORSMIN_MEDI

```

Menemukan pasangan kolom dalam dataset yang selalu memiliki missing values bersamaan (korelasi = 1). Hasilnya membantu dalam menangani missing values dengan menghapus kolom yang redundant atau mengisi (imputasi) nilai yang hilang secara konsisten.

```

# Build a graph of perfect correlations
G = Graph()
G.add_edges_from(perfect_pairs)

# Find connected components (groups)
redundant_groups = list(nx.connected_components(G))
print("\nRedundant column groups (drop all but one per group):")
for group in redundant_groups:
    print(f"- {list(group)}")

```

```

Redundant column groups (drop all but one per group):
- ['APARTMENTS_MEDI', 'APARTMENTS_MODE', 'APARTMENTS_AVG']
- ['BASEMENTAREA_AVG', 'BASEMENTAREA_MODE', 'BASEMENTAREA_MEDI']
- ['YEARS_BEGINEXPLUATATION_AVG', 'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BEGINEXPLUATATION_MEDI']
- ['YEARS_BUILD_MODE', 'YEARS_BUILD_MEDI', 'YEARS_BUILD_AVG']
- ['COMMONAREA_MEDI', 'COMMONAREA_MODE', 'COMMONAREA_AVG']
- ['ELEVATORS_MODE', 'ELEVATORS_AVG', 'ELEVATORS_MEDI']
- ['ENTRANCES_MEDI', 'ENTRANCES_AVG', 'ENTRANCES_MODE']
- ['FLOORSMAX_AVG', 'FLOORSMAX_MEDI', 'FLOORSMAX_MODE']
- ['FLOORSMIN_MODE', 'FLOORSMIN_MEDI', 'FLOORSMIN_AVG']
- ['LANDAREA_MODE', 'LANDAREA_AVG', 'LANDAREA_MEDI']
- ['LIVINGAPARTMENTS_MEDI', 'LIVINGAPARTMENTS_MODE', 'LIVINGAPARTMENTS_AVG']
- ['LIVINGAREA_AVG', 'LIVINGAREA_MODE', 'LIVINGAREA_MEDI']

```

Berfungsi untuk mengelompokkan kolom-kolom dalam dataset yang selalu memiliki missing values bersamaan (korelasi = 1) menggunakan struktur graf. Dengan menambahkan hubungan antar kolom yang berkorelasi sempurna dalam graf, kode ini menemukan kelompok kolom yang saling terhubung.

```

# Keep the FIRST column in each group and drop the rest
columns_to_keep = [list(group)[0] for group in redundant_groups]
columns_to_drop = [
    col
    for group in redundant_groups
    for col in list(group)[1:] # Exclude the first column
]

print("Columns to KEEP (first in each group):", columns_to_keep)
print("Columns to DROP (redundant):", columns_to_drop)

Columns to KEEP (first in each group): ['APARTMENTS_MEDI',
'BASEMENTAREA_AVG', 'YEARS_BEGINEXPLUATATION_AVG',
'YEARS_BUILD_MODE', 'COMMONAREA_MEDI', 'ELEVATORS_MODE',
'ENTRANCES_MEDI', 'FLOORSMAX_AVG', 'FLOORSMIN_MODE', 'LANDAREA_MODE',
'LIVINGAPARTMENTS_MEDI', 'LIVINGAREA_AVG', 'NONLIVINGAPARTMENTS_AVG',
'NONLIVINGAREA_MEDI', 'OBS_30_CNT_SOCIAL_CIRCLE',
'AMT_REQ_CREDIT_BUREAU_QRT']
Columns to DROP (redundant): ['APARTMENTS_MODE', 'APARTMENTS_AVG',
'BASEMENTAREA_MODE', 'BASEMENTAREA_MEDI',
'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BEGINEXPLUATATION_MEDI',
'YEARS_BUILD_MEDI', 'YEARS_BUILD_AVG', 'COMMONAREA_MODE',
'COMMONAREA_AVG', 'ELEVATORS_AVG', 'ELEVATORS_MEDI', 'ENTRANCES_AVG',
'ENTRANCES_MODE', 'FLOORSMAX_MEDI', 'FLOORSMAX_MODE',
'FLOORSMIN_AVG', 'FLOORSMIN_MEDI', 'LANDAREA_AVG',
'LANDAREA_MODE', 'LIVINGAPARTMENTS_AVG', 'LIVINGAPARTMENTS_MEDI',
'LIVINGAREA_AVG', 'LIVINGAREA_MODE', 'LIVINGAREA_MEDI',
'NONLIVINGAPARTMENTS_AVG', 'NONLIVINGAREA_AVG',
'NONLIVINGAREA_MEDI', 'OBS_30_CNT_SOCIAL_CIRCLE',
'AMT_REQ_CREDIT_BUREAU_QRT']
```

```
'FLOORSMIN_MEDI', 'FLOORSMIN_AVG', 'LANDAREA_AVG', 'LANDAREA_MEDI',
'LIVINGAPARTMENTS_MODE', 'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_MODE',
'LIVINGAREA_MEDI', 'NONLIVINGAPARTMENTS_MODE',
'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_AVG',
'NONLIVINGAREA_MODE', 'OBS_60_CNT_SOCIAL_CIRCLE',
'DEF_30_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_YEAR',
'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_MON',
'AMT_REQ_CREDIT_BUREAU_DAY']
```

Berfungsi untuk mengurangi redundansi dalam dataset dengan memilih satu kolom dari setiap kelompok kolom yang selalu memiliki missing values bersamaan dan menghapus sisanya. Kolom pertama dari setiap kelompok dipilih untuk disimpan dalam columns_to_keep, sementara kolom lainnya dalam kelompok yang sama dimasukkan ke dalam columns_to_drop untuk dihapus.

Pada setiap graph, hanya kolom pertama yang dipertahankan, sementara kolom lainnya dihapus karena bersifat redundant.

```
df_application_train_clean =  
df_application_train.drop(columns=columns_to_drop)  
print(f"New shape: {df_application_train_clean.shape} (dropped  
{len(columns_to_drop)} columns)")  
New shape: (307511, 86) (dropped 36 columns)
```

```
missing_summary = missing_val(df_application_train_clean)  
missing_summary[missing_summary['Missing_Percentage (%)'] >=  
51].index.tolist()  
  
Missing values per column:  
SK_ID_CURR 0  
TARGET 0  
NAME_CONTRACT_TYPE 0  
CODE_GENDER 0  
FLAG_OWN_CAR 0  
...  
FLAG_DOCUMENT_18 0  
FLAG_DOCUMENT_19 0  
FLAG_DOCUMENT_20 0  
FLAG_DOCUMENT_21 0  
AMT_REQ_CREDIT_BUREAU_QRT 41519  
Length: 86, dtype: int64
```

Analisis jumlah missing values dalam dataset dan mengidentifikasi kolom yang memiliki persentase nilai hilang lebih dari atau sama dengan 51%.

Sisa missing values yang mencapai atau melebihi 51% akan langsung dicoba diimputasi menggunakan median atau mode.

```
columns_high_missing =  
missing_summary[missing_summary['Missing_Percentage (%)'] >=  
51].index.tolist()  
  
columns_high_missing  
[ 'OWN_CAR_AGE',  
  'EXT_SOURCE_1',  
  'BASEMENTAREA_AVG',  
  'NONLIVINGAPARTMENTS_AVG',  
  'YEARS_BUILD_MODE',  
  'ELEVATORS_MODE',  
  'FLOORSMIN_MODE',  
  'LANDAREA_MODE',  
  'COMMONAREA_MEDI',  
  'LIVINGAPARTMENTS_MEDI',  
  'NONLIVINGAREA_MEDI',  
  'FONDKAPREMONT_MODE']  
identifikasi kolom dalam dataset yang memiliki persentase nilai hilang lebih dari atau sama dengan 51%.
```

```
def impute_missing(df_, columns_to_impute):  
    # Separate numerical and categorical columns  
    num_cols =  
    df_[columns_to_impute].select_dtypes(include=['number']).columns  
    cat_cols =  
    df_[columns_to_impute].select_dtypes(include=['object',  
                                             'category']).columns  
  
    # Impute numerical (median) and categorical (mode)  
    df_[num_cols] = df_[num_cols].fillna(df_[num_cols].median())  
    for col in cat_cols:  
        df_[col] = df_[col].fillna(df_[col].mode()[0])
```

```
return df_application_train_clean

# Usage
df_application_train_clean =  
impute_missing(df_application_train_clean,  
columns_to_impute=columns_high_missing)
```

Menangani nilai yang hilang dalam dataset dengan dua pendekatan berbeda tergantung pada jenis datanya. Kolom numerik diisi dengan nilai median, sementara kolom kategorikal diisi dengan modus (nilai yang paling sering muncul). Proses ini memastikan bahwa data tetap lengkap dan tidak ada nilai yang hilang, sehingga dapat digunakan untuk analisis atau model machine learning tanpa kehilangan informasi penting.

```
missing_summary = missing_val(df_application_train_clean)  
missing_summary[missing_summary['Missing_Percentage (%)'] > 0]  
  
Missing values per column:  
SK_ID_CURR           0  
TARGET               0  
NAME_CONTRACT_TYPE  0  
CODE_GENDER          0  
FLAG_OWN_CAR         0  
...  
FLAG_DOCUMENT_18     0  
FLAG_DOCUMENT_19     0  
FLAG_DOCUMENT_20     0  
FLAG_DOCUMENT_21     0  
AMT_REQ_CREDIT_BUREAU_QRT 41519  
Length: 86, dtype: int64
```

Analisis jumlah data yang hilang dalam setiap kolom dari dataset df_application_train_clean. Fungsi missing_val() kemungkinan besar menghitung persentase nilai yang hilang di setiap kolom. Kemudian, kode tersebut memfilter hanya kolom yang memiliki persentase nilai hilang lebih dari 0%, sehingga hanya menampilkan kolom yang memiliki data yang hilang.

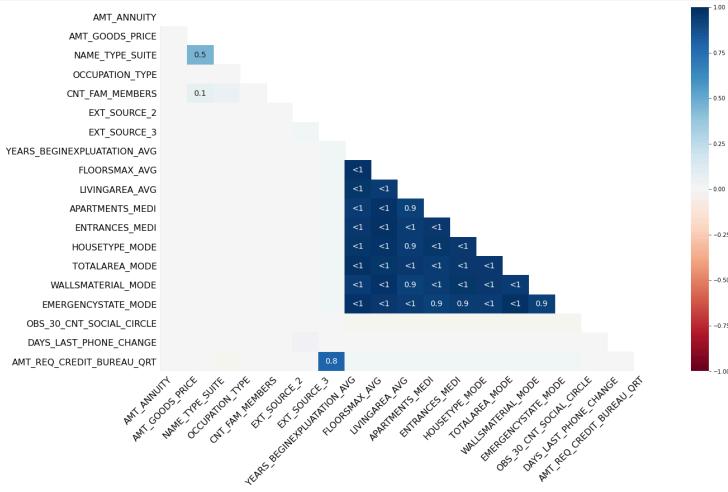
Missing values lainnya juga diperiksa korelasinya seperti yang telah dilakukan sebelumnya.

```
rest_missing = missing_summary[missing_summary['Missing_Count'] >  
0].index.tolist()
```

membuat daftar kolom yang memiliki nilai hilang dalam dataset. Kolom dengan jumlah nilai hilang lebih dari nol difilter, lalu namanya dikonversi menjadi daftar dan disimpan dalam `rest_missing` untuk keperluan imputasi atau penghapusan.

```
# Visualize missing value patterns
```

```
msno.heatmap(df_application_train[rest_missing])
```



Karena tidak ada yang bernilai 1, maka kolom tersebut tetap dipertahankan dan langsung dilakukan imputasi.

```
df_application_train_clean =  
impute_missing(df_application_train_clean,  
columns_to_impute=rest_missing)
```

```
missing_summary = missing_val(df_application_train_clean)  
missing_summary[missing_summary['Missing_Percentage (%)'] > 0]
```

```
Missing values per column:  
SK_ID_CURR          0  
TARGET              0  
NAME_CONTRACT_TYPE 0  
CODE_GENDER         0  
FLAG_OWN_CAR        0  
                         ..  
FLAG_DOCUMENT_18    0  
FLAG_DOCUMENT_19    0  
FLAG_DOCUMENT_20    0  
FLAG_DOCUMENT_21    0  
AMT_REQ_CREDIT_BUREAU_QRT 0  
Length: 86, dtype: int64
```

Setelah proses imputasi, semua kolom dalam dataset tidak lagi memiliki nilai yang hilang. Hal ini terlihat dari hasil missing_summary, di mana semua kolom memiliki jumlah dan persentase nilai hilang sebesar 0%.

installments_payments.csv

```
# Select only numeric columns (int64 and float64)  
numeric_cols = df_installments_payments.select_dtypes(include=['int64',  
'float64']).columns  
  
# Downcast numeric columns  
df_installments_payments[numeric_cols] = df_installments_payments[numeric_cols].apply(  
    pd.to_numeric, downcast='integer'  
)  
  
# For float columns (separate step to avoid mixing types)  
float_cols = df_installments_payments.select_dtypes(include=['float64']).columns  
df_installments_payments[float_cols] = df_installments_payments[float_cols].apply(  
    pd.to_numeric, downcast='float'
```

```
)
```

mengoptimalkan penggunaan memori dalam dataset dengan melakukan downcast pada kolom numerik.

```
df_installments_payments.info()  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 13605401 entries, 0 to 13605400  
Data columns (total 8 columns):  
 #   Column           Dtype     
---  --  
 0   SK_ID_PREV      int32  
 1   SK_ID_CURR      int32  
 2   NUM_INSTALMENT_VERSION  int16  
 3   NUM_INSTALMENT_NUMBER  int16  
 4   DAYS_INSTALMENT    int16  
 5   DAYS_ENTRY_PAYMENT float32  
 6   AMT_INSTALMENT     float64  
 7   AMT_PAYMENT        float64  
dtypes: float32(1), float64(2), int16(3), int32(2)  
memory usage: 441.2 MB
```

```
missing_summary = missing_val(df_installments_payments)  
missing_summary[missing_summary['Missing_Percentage (%)'] > 0]  
Cek missing values
```

```
df_installments_payments[df_installments_payments['DAYS_ENTRY_PAYMENT'] .isnull()]
```

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	
3764207	1531600	103793		1	7
3764208	1947105	159974		1	24
3764209	1843773	167270		1	22
3764210	1691592	192536		1	5
3764211	1531299	157088		0	11

Menunjukkan 2.905 baris dengan nilai hilang di kolom 'DAYS_ENTRY_PAYMENT'.

```
# Get columns where missing percentage < 10%
```

```
low_missing_cols = missing_summary[missing_summary['Missing_Count'] > 0].index.tolist()
```

```
print("Columns with >0 missing values:", low_missing_cols)
```

```
Columns with >0 missing values: ['AMT_INSTALMENT', 'AMT_PAYMENT']
```

hasilnya di dapat bahwa missing val nya tidak signifikan

```
# Create a boolean mask for rows with missing values in ANY  
low-missing column
```

```
rows_to_drop_mask = df_installments_payments[low_missing_cols].isnull().any(axis=1)
```

```
# Count impacted rows
```

```
impacted_rows = rows_to_drop_mask.sum()
```

```
total_rows = len(df_installments_payments)
```

```
print(f"Rows to drop: {impacted_rows} ({(impacted_rows / total_rows * 100):.2f}% of total)")
```

```
Rows to drop: 2905 (0.02% of total)
```

Drop baris

```
df_installments_payments_clean = df_installments_payments.dropna(subset=low_missing_cols)
```

```
print(f"New DataFrame shape: {df_installments_payments_clean.shape}")
```

```
New DataFrame shape: (13602496, 8)
```

Buat dataframe baru untuk data yang sudah bersih

Anggota Tiga | Nicholas Evan Sitanggang_5026231146

Transformasi dan Rekayasa Fitur untuk Kedua Dataset

#Dataset Lending Club Loan Data

accepted_2007_to_2018Q4.csv

rejected_2007_to_2018Q4.csv

```
import pandas as pd
# import category_encoders as ce
from sklearn.preprocessing import LabelEncoder
```

Kode di atas dimulai dengan mengimpor pustaka pandas (`import pandas as pd`), yang digunakan untuk manipulasi dan analisis data dalam bentuk tabel (DataFrame dan Series). Kemudian, kode mengimpor LabelEncoder dari `sklearn.preprocessing`, yang digunakan untuk mengonversi variabel kategori menjadi angka secara ordinal.

```
def downcasting(df):
    # Select only numeric columns (int64 and float64)
    numeric_cols      = df.select_dtypes(include=['int64',
    'float64']).columns

    # Downcast numeric columns
    df[numeric_cols] = df[numeric_cols].apply(
        pd.to_numeric, downcast='integer'
    )

    # For float columns (separate step to avoid mixing types)
    float_cols = df.select_dtypes(include=['float64']).columns
    df[float_cols] = df[float_cols].apply(
        pd.to_numeric, downcast='float'
    )

    return df
```

Fungsi `downcasting(df)` bertujuan untuk mengurangi penggunaan memori dataset dengan mengonversi tipe data numerik ke format yang lebih kecil tanpa kehilangan informasi.

Pertama, fungsi memilih kolom yang bertipe int64 dan float64

```
menggunakan df.select_dtypes(include=['int64', 'float64']).columns. Kemudian, pada langkah pertama downcasting, semua kolom numerik dikonversi ke tipe data integer dengan pd.to_numeric(downcast='integer'). Setelah itu, fungsi kembali memilih kolom bertipe float64 dan mengonversinya menjadi tipe data float dengan pd.to_numeric(downcast='float'). Pemisahan langkah untuk integer dan float ini dilakukan agar tidak terjadi kesalahan pengubahan tipe data. Akhirnya, dataset yang telah dioptimalkan dikembalikan.
```

```
df_accepted_clean = downcasting(df_accepted_clean)
df_rejected_clean = downcasting(df_rejected_clean)
```

```
ipython-input-117-b59a6271e4a9>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df[numeric_cols] = df[numeric_cols].apply(
ipython-input-117-b59a6271e4a9>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df[float_cols] = df[float_cols].apply(
```

Kode ini menerapkan fungsi `downcasting(df)` pada dua DataFrame, yaitu `df_accepted_clean` dan `df_rejected_clean`.

```
df_accepted_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 23437 entries, 0 to 25513
Data columns (total 96 columns):
 #   Column           Non-Null Count Dtype
 ---  ----
 0   id              23437 non-null  int32
 1   loan_amnt        23437 non-null  int32
 2   funded_amnt      23437 non-null  int32
 3   funded_amnt_inv  23437 non-null  int32
```

4	int_rate	23437	non-null	float32
5	installment	23437	non-null	float32
6	annual_inc	23437	non-null	float64
7	issue_d	23437	non-null	datetime64[ns]
8	dti	23437	non-null	float32
9	delinq_2yrs	23437	non-null	int8
10	earliest_cr_line	23437	non-null	datetime64[ns]
11	fico_range_low	23437	non-null	int16
12	fico_range_high	23437	non-null	int16
13	inq_last_6mths	23437	non-null	int8
14	mths_since_last_delinq	12354	non-null	float32
15	open_acc	23437	non-null	int8
16	pub_rec	23437	non-null	int8
17	revol_bal	23437	non-null	int32
18	revol_util	23437	non-null	float32
19	total_acc	23437	non-null	int8
20	out_prncp	23437	non-null	float64
21	out_prncp_inv	23437	non-null	float64
22	total_pymnt	23437	non-null	float64
23	total_pymnt_inv	23437	non-null	float64
24	total_rec_prncp	23437	non-null	float64
25	total_rec_int	23437	non-null	float64
26	total_rec_late_fee	23437	non-null	float32
27	recoveries	23437	non-null	float64
28	collection_recovery_fee	23437	non-null	float32
29	last_pymnt_d	23437	non-null	datetime64[ns]
30	last_pymnt_amnt	23437	non-null	float64
31	last_credit_pull_d	23437	non-null	datetime64[ns]
32	last_fico_range_high	23437	non-null	int16
33	last_fico_range_low	23437	non-null	int16
34	collections_12_mths_ex_med	23437	non-null	int8
35	policy_code	23437	non-null	int8

36	acc_now_delinq	23437	non-null	int8
37	tot_coll_amt	23437	non-null	int32
38	tot_cur_bal	23437	non-null	int32
39	il_util	17400	non-null	float32
40	all_util	19578	non-null	float32
41	total_rev_hi_lim	23437	non-null	int32
42	acc_open_past_24mths	23437	non-null	int8
43	avg_cur_bal	23437	non-null	int32
44	bc_open_to_buy	23437	non-null	int32
45	bc_util	23437	non-null	float32
46	chargeoff_within_12_mths	23437	non-null	int8
47	delinq_amnt	23437	non-null	int32
48	mo_sin_old_il_acct	23437	non-null	int16
49	mo_sin_old_rev_tl_op	23437	non-null	int16
50	mo_sin_rcnt_rev_tl_op	23437	non-null	int16
51	mo_sin_rcnt_tl	23437	non-null	int16
52	mort_acc	23437	non-null	int8
53	mths_since_recent_bc	23437	non-null	int16
54	mths_since_recent_inq	23437	non-null	int8
55	num_accts_ever_120_pd	23437	non-null	int8
56	num_actv_bc_tl	23437	non-null	int8
57	num_actv_rev_tl	23437	non-null	int8
58	num_bc_sats	23437	non-null	int8
59	num_bc_tl	23437	non-null	int8
60	num_il_tl	23437	non-null	int8
61	num_op_rev_tl	23437	non-null	int8
62	num_rev_accts	23437	non-null	int8
63	num_rev_tl_bal_gt_0	23437	non-null	int8
64	num_sats	23437	non-null	int8
65	num_tl_120dpd_2m	23437	non-null	int8
66	num_tl_30dpd	23437	non-null	int8
67	num_tl_90g_dpd_24m	23437	non-null	int8

```
68 num_tl_op_past_12m           23437 non-null int8
69 pct_tl_nvr_dlq              23437 non-null float32
70 percent_bc_gt_75             23437 non-null float32
71 pub_rec_bankruptcies         23437 non-null int8
72 tax_liens                    23437 non-null int8
73 tot_hi_cred_lim              23437 non-null int32
74 total_bal_ex_mort             23437 non-null int32
75 total_bc_limit                23437 non-null int32
76 total_il_high_credit_limit    23437 non-null int32
77 zip_3digits                  23437 non-null int16
78 term_encoded                  23437 non-null int8
79 grade_encoded                 23437 non-null int8
80 sub_grade_encoded              23437 non-null int8
81 emp_title_encoded              23437 non-null int16
82 emp_length_encoded             23437 non-null int8
83 home_ownership_encoded        23437 non-null int8
84 verification_status_encoded   23437 non-null int8
85 loan_status_encoded            23437 non-null int8
86 pymnt_plan_encoded             23437 non-null int8
87 url_encoded                   23437 non-null int16
88 purpose_encoded                23437 non-null int8
89 title_encoded                  23437 non-null int8
90 addr_state_encoded              23437 non-null int8
91 initial_list_status_encoded   23437 non-null int8
92 application_type_encoded      23437 non-null int8
93 hardship_flag_encoded          23437 non-null int8
94 disbursement_method_encoded   23437 non-null int8
95 debt_settlement_flag_encoded  23437 non-null int8
dtypes: datetime64[ns](4),   float32(12),   float64(9),   int16(12),
int32(15),   int8(44)
memory usage: 6.4 MB
```

Dataset ini terdiri dari 2.137.072 entri dengan 96 kolom. Setelah dilakukan downcasting, terdapat perubahan tipe data pada beberapa kolom numerik, yang bertujuan untuk mengoptimalkan penggunaan memori.

Perubahan yang Terjadi:

- **Kolom Integer** (`int64 → int32`) = `loan_amnt` dan `funded_amnt` berhasil dikonversi menjadi `int32`, sehingga mengurangi konsumsi memori.
- **Kolom Float** (`float64 → float32`) = Kolom seperti `int_rate`, `installment`, dan `dti` telah berubah menjadi `float32`, yang membantu menghemat penggunaan memori sambil mempertahankan presisi angka desimal.

Namun, `funded_amnt_inv` dan `annual_inc` masih bertipe `float64`, yang berarti downcasting mungkin belum diterapkan ke semua kolom atau beberapa nilai memiliki presisi yang lebih tinggi dari `float32`.

- **Kolom Kategorikal (object)** = Sebagian besar kolom masih bertipe `object`, yang menunjukkan bahwa mereka berisi data teks atau kategori.

```
# Convert "Dec-2015" to datetime (defaults to day=1)
df_accepted_clean['issue_d'] = pd.to_datetime(df_accepted_clean['issue_d'], format='%b-%Y')
df_accepted_clean['last_pymnt_d'] = pd.to_datetime(df_accepted_clean['last_pymnt_d'], format='%b-%Y')
df_accepted_clean['last_credit_pull_d'] = pd.to_datetime(df_accepted_clean['last_credit_pull_d'],
format='%b-%Y')
df_accepted_clean['earliest_cr_line'] = pd.to_datetime(df_accepted_clean['earliest_cr_line'], format='%b-%Y')
```

Mengubah beberapa kolom tanggal dalam dataset `df_accepted_clean` dari format teks (`object`) menjadi format `datetime64`. Format yang digunakan adalah '`%b-%Y`', yang berarti:

- `%b` → Tiga huruf pertama dari nama bulan (contoh: Jan, Feb, Dec).

- %Y → Tahun dalam format empat digit (contoh: 2015, 2020).

Tanpa hari spesifik → Secara default, Pandas akan mengasumsikan hari sebagai 1 jika tidak disediakan.

```
# Check if 'zip_code' or a similar column exists
print(df_accepted_clean.columns)

Index(['id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'int_rate',
       'installment', 'annual_inc', 'issue_d', 'dti', 'delinq_2yrs',
       'earliest_cr_line', 'fico_range_low', 'fico_range_high',
       'inq_last_6mths', 'mths_since_last_delinq', 'open_acc', 'pub_rec',
       'revol_bal', 'revol_util', 'total_acc', 'out_prncp', 'out_prncp_inv',
       'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
       'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
       'last_fico_range_high', 'last_fico_range_low',
       'collections_12_mths_ex_med', 'policy_code', 'acc_now_delinq',
       'tot_coll_amt', 'tot_cur_bal', 'il_util', 'all_util',
       'total_rev_hi_lim', 'acc_open_past_24mths', 'avg_cur_bal',
       'bc_open_to_buy', 'bc_util', 'chargeoff_within_12_mths', 'delinq_amnt',
       'mo_sin_old_il_acct', 'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op',
       'mo_sin_rcnt_tl', 'mort_acc', 'mths_since_recent_bc',
       'mths_since_recent_inq', 'num_accts_ever_120_pd', 'num_actv_bc_tl',
       'num_actv_rev_tl', 'num_bc_sats', 'num_bc_tl', 'num_il_tl',
       'num_op_rev_tl', 'num_rev_accts', 'num_rev_tl_bal_gt_0', 'num_sats',
       'num_tl_120dpd_2m', 'num_tl_30dpd', 'num_tl_90g_dpd_24m',
       'num_tl_op_past_12m', 'pct_tl_nvr_dlq', 'percent_bc_gt_75',
       'pub_rec_bankruptcies', 'tax_liens', 'tot_hi_cred_lim',
       'total_bal_ex_mort', 'total_bc_limit', 'total_il_high_credit_limit',
       'zip_3digits', 'term_encoded', 'grade_encoded', 'sub_grade_encoded',
       'emp_title_encoded', 'emp_length_encoded', 'home_ownership_encoded',
       'verification_status_encoded', 'loan_status_encoded',
       'pymnt_plan_encoded', 'url_encoded', 'purpose_encoded', 'title_encoded',
       'addr_state_encoded', 'initial_list_status_encoded',
       'application_type_encoded', 'hardship_flag_encoded',
       'disbursement_method_encoded', 'debt_settlement_flag_encoded'],
      dtype='object')
```

Melakukan pengecekan terhadap nama-nama kolom dalam dataset df_accepted_clean khususnya untuk memastikan apakah kolom 'zip_code' atau yang serupa ada dalam data. Hasilnya menunjukkan bahwa kolom tersebut memang ada, bersama dengan banyak kolom lainnya yang mencakup berbagai aspek informasi. Beberapa kolom berkaitan dengan identitas dasar, seperti 'id', 'addr_state', dan 'zip_code', sementara yang lain terkait dengan informasi pinjaman, seperti 'loan_amnt', 'funded_amnt', dan 'int_rate'. Ada juga kolom yang menggambarkan

```
status kredit dan pembayaran, seperti 'loan_status', 'total_pymnt', dan  
'total_rec_prncp', serta fitur yang berkaitan dengan riwayat kredit, misalnya  
'fico_range_high', 'fico_range_low', dan 'open_acc'.
```

```
df_accepted_clean['zip_3digits'] =  
pd.to_numeric(df_accepted_clean['zip_code'].str[:3],  
errors='coerce').fillna(0).astype(int)
```

Menambahkan fitur untuk mengubah Kolom 'zip_code' menjadi kolom baru bernama 'zip_3digits', yang hanya mengambil tiga digit pertama dari setiap kode pos. Proses ini dilakukan dengan cara mengonversi nilai dalam kolom 'zip_code' menjadi string, mengambil tiga karakter pertama, lalu mengubahnya kembali menjadi tipe numerik. Jika terjadi kesalahan atau terdapat nilai yang tidak bisa dikonversi, nilai tersebut akan diisi dengan 0 menggunakan `fillna(0)`, kemudian diubah ke tipe data integer.

```
df_accepted_clean.drop(columns=['zip_code'], inplace=True)
```

Kolom 'zip_code' dihapus dari dataset `df_accepted_clean` karena telah diubah menjadi kolom 'zip_3digits'. Penghapusan ini dilakukan menggunakan metode `.drop(columns=['zip_code'], inplace=True)`, yang berarti perubahan dilakukan langsung pada dataframe tanpa perlu menyimpannya kembali ke variabel baru.

Langkah ini bertujuan untuk mengurangi redundansi dalam data, mengingat tiga digit pertama dari kode pos sudah cukup untuk analisis berbasis wilayah.

```
df_accepted_categorical =  
df_accepted_clean.select_dtypes(include=['object', 'category'])
```

Dataset `df_accepted_clean` difilter untuk memilih hanya kolom-kolom yang bertipe data kategori, baik yang berupa object maupun category. Hasilnya disimpan dalam dataframe baru bernama `df_accepted_categorical`.

Langkah ini bertujuan untuk mengelompokkan data kategorikal secara terpisah agar lebih mudah dianalisis atau diolah lebih lanjut, misalnya dalam proses encoding

(seperti Label Encoding atau One-Hot Encoding) sebelum digunakan dalam model machine learning.

```
cat_cols = df_accepted_categorical.columns

for col in cat_cols:
    num_categories = df_accepted_categorical[col].nunique()
    print(f'{col}: {num_categories} categories')

'term': 2 categories
'grade': 7 categories
'sub_grade': 35 categories
'emp_title': 11392 categories
'emp_length': 11 categories
'home_ownership': 3 categories
'verification_status': 3 categories
'loan_status': 6 categories
'pymnt_plan': 2 categories
'url': 23437 categories
'purpose': 12 categories
'title': 12 categories
'addr_state': 49 categories
'initial_list_status': 2 categories
'application_type': 2 categories
'hardship_flag': 2 categories
'disbursement_method': 1 categories
'debt_settlement_flag': 2 categories
```

menghitung dan mencetak jumlah kategori unik dalam setiap kolom kategorikal dari DataFrame `df_accepted_categorical`.

Penjelasan per baris:

1. `cat_cols = df_accepted_categorical.columns`

- Mengambil semua nama kolom dalam DataFrame `df_accepted_categorical`, yang diasumsikan hanya berisi variabel kategorikal.
2. `for col in cat_cols:`
- Melakukan iterasi (loop) untuk setiap kolom dalam daftar kolom kategorikal.
3. `num_categories = df_accepted_categorical[col].nunique()`
- Menghitung jumlah nilai unik (kategori) dalam kolom tersebut menggunakan `.nunique()`.
4. `print(f'{col}': {num_categories} categories")`
- Mencetak nama kolom dan jumlah kategorinya dalam format yang mudah dibaca.

Hasil dari proses pengecekan jumlah kategori unik dalam setiap kolom kategorikal dari dataset `df_accepted_clean`.

Beberapa hal yang dapat disimpulkan dari hasil ini:

1. Kolom dengan jumlah kategori yang sangat besar
 - `emp_title` memiliki 465,745 kategori, menunjukkan bahwa hampir setiap entri memiliki nilai unik, sehingga mungkin tidak terlalu berguna untuk analisis kategorikal.
 - `title` memiliki 38,721 kategori, yang juga cukup tinggi dan mungkin mengandung data yang sangat spesifik atau bisa jadi mirip dengan kolom lain.
 - `id` dan `url` memiliki jumlah kategori yang sama dengan jumlah baris (2,137,072 kategori), yang menunjukkan bahwa ini adalah identifier unik dan

mungkin tidak diperlukan untuk analisis lebih lanjut.

2. Kolom dengan jumlah kategori yang wajar untuk encoding
 - grade (7 kategori), sub_grade (35 kategori), loan_status (7 kategori), dan home_ownership (6 kategori) memiliki jumlah kategori yang masih dapat diolah dengan teknik seperti Label Encoding.
3. Kolom biner atau hampir biner
 - pymnt_plan, initial_list_status, application_type, hardship_flag, disbursement_method, dan debt_settlement_flag hanya memiliki 2 kategori, sehingga bisa dengan mudah diubah menjadi variabel biner (0 dan 1).

```
df_accepted_clean[['issue_d', 'last_pymnt_d', 'last_credit_pull_d',  
'earliest_cr_line']]
```

	issue_d	last_pymnt_d	last_credit_pull_d	earliest_cr_line
0	2015-12-01	2019-01-01	2019-03-01	2003-08-01
1	2015-12-01	2016-06-01	2019-03-01	1999-12-01
3	2015-12-01	2019-02-01	2019-03-01	2008-09-01
4	2015-12-01	2016-07-01	2018-03-01	1998-06-01
5	2015-12-01	2017-05-01	2017-05-01	1987-10-01
...
25509	2015-12-01	2016-07-01	2017-06-01	2004-04-01
25510	2015-12-01	2017-01-01	2017-07-01	2001-11-01
25511	2015-12-01	2018-11-01	2019-03-01	1991-08-01
25512	2015-12-01	2018-06-01	2019-03-01	1988-07-01
25513	2015-12-01	2016-11-01	2017-12-01	2004-02-01

23437 rows × 4 columns

Kode ini digunakan untuk menampilkan subset dari DataFrame `df_accepted_clean`, yang berisi empat kolom terkait tanggal, yaitu `issue_d`, `last_pymnt_d`, `last_credit_pull_d`, dan `earliest_cr_line`.

Kolom `issue_d` merepresentasikan tanggal penerbitan pinjaman, sedangkan `last_pymnt_d` menunjukkan tanggal pembayaran terakhir yang dilakukan oleh peminjam. Kolom `last_credit_pull_d` mencatat kapan terakhir kali data kredit peminjam diperbarui oleh perusahaan, dan `earliest_cr_line` menunjukkan kapan peminjam pertama kali membuka kredit. Dengan memilih kolom ini menggunakan double brackets, kode ini memastikan bahwa hasilnya tetap berupa DataFrame. Tujuan utama dari kode ini adalah untuk memverifikasi apakah konversi tanggal telah berhasil, memeriksa adanya nilai yang hilang atau tidak sesuai, serta mendukung eksplorasi lebih lanjut seperti analisis berbasis waktu atau transformasi tambahan.

```
# Find columns with exactly 2 unique values
cat1_col      = [col      for      col      in      cat_cols      if
df_accepted_categorical[col].nunique() < 2137072]

print("Categorical columns:", cat1_col)

Categorical columns: ['term', 'grade', 'sub_grade', 'emp_title',
'emp_length', 'home_ownership', 'verification_status', 'loan_status',
'pymnt_plan', 'url', 'purpose', 'title', 'addr_state',
'initial_list_status', 'application_type', 'hardship_flag',
'disbursement_method', 'debt_settlement_flag']

Kode ini digunakan untuk memilih kolom kategorikal dalam
df_accepted_categorical yang memiliki jumlah kategori unik kurang
dari 2.137.072, kemudian mencetak daftar kolom yang memenuhi syarat.
Proses ini dilakukan menggunakan list comprehension, di mana setiap
kolom dalam cat_cols diperiksa jumlah kategori uniknya menggunakan
df_accepted_categorical[col].nunique(). Jika jumlahnya kurang dari
batas yang telah ditentukan, maka kolom tersebut dimasukkan ke dalam
daftar cat1_col.
```

```
Setelah daftar kolom yang memenuhi kriteria dibuat, kode  
print("Categorical columns:", cat1_col) digunakan untuk mencetak  
hasilnya. Ini membantu dalam memahami struktur dataset dengan  
menunjukkan kolom mana yang memiliki jumlah kategori yang lebih  
sedikit dari batas yang telah ditetapkan.
```

Tujuan dari kode ini adalah untuk memfilter kolom kategorikal sehingga hanya mencakup kolom dengan jumlah kategori yang tidak terlalu besar, sehingga lebih mudah dikelola dalam analisis dan pemrosesan data. Selain itu, ini membantu dalam proses encoding, karena kolom dengan jumlah kategori yang sangat besar dapat menyebabkan masalah dalam algoritma machine learning.

```
for col in cat1_col:  
    print(f"\nColumn: {col}")  
    print(df_accepted_categorical[col].value_counts())
```

```
Column: term
term
36 months    16098
60 months    7339
Name: count, dtype: int64
```

```
Column: grade
grade
B      7094
C      6853
A      4114
D      3196
E      1636
F      450
G      94
Name: count, dtype: int64
```

```
Column: sub_grade
sub_grade
C1     1604
B4     1559
B3     1514
B5     1425
C4     1425
C2     1403
C3     1378
B2     1348
B1     1248
A5     1166
C5     1043
A1     942
D1     873
A4     786
D2     679
A2     630
D3     610
```

```
A3      590  
D4      564  
D5      470  
E1      414  
E2      382  
E3      347  
E4      276  
E5      217  
F1      143  
F2      105  
F3      99  
F4      56  
F5      47  
G1      31  
G2      29  
G3      16  
G4      10  
G5      8
```

```
Name: count, dtype: int64
```

```
Column: emp_title
```

```
emp_title  
Unknown          1333  
Teacher          479  
Manager          398  
Owner            232  
RN               182
```

```
...
```

```
Network Systems Engineer      1  
Aviation Security Specialist 1  
DE Underwriter                1  
School psychologist            1  
Peace officer                 1
```

```
Name: count, Length: 11392, dtype: int64
```

```
Column: emp_length
```

```
emp_length
10+ years      7730
< 1 year       2173
2 years        2100
3 years        1907
1 year         1511
5 years        1459
4 years        1361
8 years        1222
6 years         946
9 years         865
7 years         834
Name: count, dtype: int64
```

```
Column: home_ownership
home_ownership
MORTGAGE      11466
RENT          9222
OWN           2749
Name: count, dtype: int64
```

```
Column: verification_status
verification_status
Source Verified    9659
Not Verified      7051
Verified          6727
Name: count, dtype: int64
```

```
Column: loan_status
loan_status
Fully Paid        16508
Charged Off       4204
Current           2526
Late (31-120\ndays)   124
In Grace Period     61
Late (16-30\ndays)    14
Name: count, dtype: int64
```

```
Column: pymnt_plan
pymnt_plan
n    23434
y      3
Name: count, dtype: int64

Column: url
url
https://lendingclub.com/browse/loanDetail.action?loan_id=66474152      1
https://lendingclub.com/browse/loanDetail.action?loan_id=68407277      1
https://lendingclub.com/browse/loanDetail.action?loan_id=68355089      1
https://lendingclub.com/browse/loanDetail.action?loan_id=66310712      1
https://lendingclub.com/browse/loanDetail.action?loan_id=68476807      1
..
https://lendingclub.com/browse/loanDetail.action?loan_id=68506798      1
https://lendingclub.com/browse/loanDetail.action?loan_id=68566886      1
https://lendingclub.com/browse/loanDetail.action?loan_id=68009401      1
https://lendingclub.com/browse/loanDetail.action?loan_id=68476702      1
https://lendingclub.com/browse/loanDetail.action?loan_id=68436666      1
Name: count, Length: 23437, dtype: int64

Column: purpose
purpose
debt_consolidation    13284
credit_card            5865
home_improvement       1323
other                  1304
major_purchase          517
medical                 274
small_business           251
car                      212
vacation                 151
moving                   144
house                     97
renewable_energy          15
Name: count, dtype: int64
```

```
Column: title
title
Debt consolidation      13282
Credit card refinancing 5867
Home improvement        1323
Other                   1305
Major purchase          516
Medical expenses        274
Business                251
Car financing           212
Vacation                152
Moving and relocation   143
Home buying             97
Green loan              15
Name: count, dtype: int64
```

```
Column: addr_state
addr_state
CA      3190
TX      2029
NY      1795
FL      1642
IL      872
GA      857
NJ      836
PA      803
OH      797
VA      707
NC      668
MI      625
MD      569
AZ      537
MA      533
CO      508
WA      485
MN      428
```

```
f      3506
Name: count, dtype: int64

Column: application_type
application_type
Individual    23256
Joint App      181
Name: count, dtype: int64

Column: hardship_flag
hardship_flag
N      23434
Y      3
Name: count, dtype: int64

Column: disbursement_method
disbursement_method
Cash    23437
Name: count, dtype: int64

Column: debt_settlement_flag
debt_settlement_flag
N      22732
Y      705
Name: count, dtype: int64
```

TN	412
MO	375
IN	366
CT	336
NV	314
WI	304
AL	301
SC	298
LA	297
OR	276
KY	237
KS	218
OK	210
AR	189
UT	132
NM	130
NH	122
MS	122
HI	109
NE	101
ME	101
RI	94
WV	81
ND	66
MT	63
DE	61
DC	55
WY	51
AK	50
SD	44
VT	41

Name: count, dtype: int64

Column: initial_list_status

initial_list_status
w 19931

Perulangan (for col in cat1_col) digunakan untuk mengiterasi setiap

```
kolom dalam daftar cat1_col. Dalam setiap iterasi, kode  
print(f"\nColumn: {col}") mencetak nama kolom saat ini, disertai  
dengan pemisah baris baru (\n) agar output lebih rapi. Selanjutnya,  
df_accepted_categorical[col].value_counts() digunakan untuk  
menghitung frekuensi kemunculan setiap kategori dalam kolom tersebut,  
lalu hasilnya dicetak ke layar.
```

```
def labelEncoding(df, var):  
    le = LabelEncoder()  
    df[var+'_encoded'] = le.fit_transform(df[var])  
  
    # To see the mapping  
    print("Maps of "+var)  
    print(dict(zip(le.classes_, le.transform(le.classes_))))  
  
    return df
```

Fungsi labelEncoding(df, var) melakukan Label Encoding pada kolom kategorikal menggunakan LabelEncoder dari sklearn.preprocessing. Fungsi ini mengubah nilai kategorikal menjadi numerik dan menyimpannya dalam kolom baru {var}_encoded. Selain itu, fungsi mencetak peta nilai kategori asli ke angka untuk referensi. Ini berguna dalam machine learning agar algoritma dapat bekerja dengan data numerik. Hasilnya adalah DataFrame yang diperbarui dengan kolom tambahan berisi nilai yang telah dienkode.

```
df_accepted_clean
```

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	...	pub_rec_bankruptcies	tax_liens	tot_hi_cred_lim	...
0	68407277	3600	3600	3600.0	36 months	13.990000	123.029999	C	C4	leadman	...	0	0	178050	...
1	68355089	24700	24700	24700.0	36 months	11.990000	820.280029	C	C1	Engineer	...	0	0	314017	...
3	66310712	35000	35000	35000.0	60 months	14.850000	829.900024	C	C5	Information Systems Officer	...	0	0	381215	...
4	68476807	10400	10400	10400.0	60 months	22.450001	289.910004	F	F1	Contract Specialist	...	0	0	439570	...
5	68426831	11950	11950	11950.0	36 months	13.440000	405.179993	C	C3	Veterinary Technician	...	0	0	16900	...
...
2260691	89996426	32000	32000	32000.0	60 months	14.490000	752.739990	C	C4	Sales Manager	...	0	0	524379	...
2260692	90006534	16000	16000	16000.0	60 months	12.790000	362.339996	C	C1	Manager	...	3	0	87473	...
2260693	89955820	24000	24000	24000.0	60 months	10.490000	515.739990	B	B3	Current Operations Officer	...	0	2	128902	...
2260695	88977788	24000	24000	24000.0	60 months	10.490000	515.739990	B	B3	Database Administrator	...	0	1	227883	...
2260697	88224441	24000	24000	24000.0	60 months	14.490000	564.559998	C	C4	Program Manager	...	1	0	84664	...

2137172 rows × 96 columns

Setelah menerapkan Label Encoding, variabel kategorikal dalam dataset Lending Club Loan telah dikonversi menjadi format numerik. Contohnya, kolom grade dan sub_grade yang sebelumnya berupa huruf kini direpresentasikan sebagai angka yang sesuai dengan kategori aslinya. Selain itu, kolom seperti emp_title, yang berisi berbagai jenis pekerjaan, kini memiliki nilai numerik unik untuk setiap posisi pekerjaan. Transformasi ini bertujuan untuk memungkinkan model machine learning memahami variabel kategorikal tanpa kehilangan informasi, serta meningkatkan efisiensi pemrosesan data dalam tahap analisis dan prediksi selanjutnya.

```
df_accepted_fe = df_accepted_clean
Pembuatan label baru untuk feature engineering
```

```
for col in cat1_col:
    print(col)
    df_accepted_fe = labelEncoding(df_accepted_fe, col)
```

```

term
Maps of term
{'36 months': 0, '60 months': 1}
grade
Maps of grade
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6}
sub_grade
Maps of sub_grade
{'A1': 0, 'A2': 1, 'A3': 2, 'A4': 3, 'A5': 4, 'B1': 5, 'B2': 6, 'B3': 7, 'B4': 8, 'B5': 9, 'C1': 10, 'C2': 11, 'C3': 12, 'C4': 13, 'C5': 14, 'D1': 15, 'D2': 16, 'D3': 17, 'D4': 18,
emp_title
Maps of emp_title
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)

Maps of emp_length
{'1 year': 0, '10+ years': 2, '2 years': 3, '3 years': 4, '4 years': 5, '5 years': 6, '6 years': 7, '7 years': 8, '8 years': 9, '9 years': 10, '< 1 year': 11, 'Unknown': 11}
home_ownership
Maps of home_ownership
{'ANY': 0, 'MORTGAGE': 1, 'NONE': 2, 'OTHER': 3, 'OWN': 4, 'RENT': 5}
verification_status
Maps of verification_status
{'Not Verified': 0, 'Source Verified': 1, 'Verified': 2}
loan_status
Maps of loan_status
{'Charged Off': 0, 'Current': 1, 'Default': 2, 'Fully Paid': 3, 'In Grace Period': 4, 'Late (16-30 days)': 5, 'Late (31-120 days)': 6}
 pymt_plan
Maps of pymt_plan
{'n': 0, 'y': 1}
purpose

Maps of purpose
{'car': 0, 'credit_card': 1, 'debt_consolidation': 2, 'educational': 3, 'home_improvement': 4, 'house': 5, 'major_purchase': 6, 'medical': 7, 'moving': 8, 'other': 9, 'renewable_electricity': 10}
title
Maps of title
{'A debt consolidation': 0, 'A lending hand': 1, 'Consolidation': 2, 'Debt Consolidation': 3, 'LOAN': 4, 'Personal loan': 5, 'Three year debit free': 6, 'debit consolidation': 7}
addr_state
Maps of addr_state
{'AK': 0, 'AL': 1, 'AR': 2, 'AZ': 3, 'CA': 4, 'CO': 5, 'CT': 6, 'DC': 7, 'DE': 8, 'FL': 9, 'GA': 10, 'HI': 11, 'IA': 12, 'ID': 13, 'IL': 14, 'IN': 15, 'KS': 16, 'KY': 17, 'LA': 18, 'MA': 19, 'MD': 20, 'ME': 21, 'MI': 22, 'MN': 23, 'MO': 24, 'NC': 25, 'ND': 26, 'NE': 27, 'NH': 28, 'NJ': 29, 'NM': 30, 'NV': 31, 'NY': 32, 'PA': 33, 'RI': 34, 'SD': 35, 'TN': 36, 'TX': 37, 'UT': 38, 'VA': 39, 'VT': 40, 'WA': 41, 'WI': 42, 'WV': 43, 'WY': 44}
initial_list_status
Maps of initial_list_status
{'f': 0, 'w': 1}
application_type
Maps of application_type
{'Individual': 0, 'Joint App': 1}
hardship_flag
Maps of hardship_flag
{'N': 0, 'Y': 1}
disbursement_method
Maps of disbursement_method
{'Cash': 0, 'DirectPay': 1}
debt_settlement_flag
Maps of debt_settlement_flag
{'N': 0, 'Y': 1}

```

Menerapkan Label Encoding pada setiap kolom dalam daftar `cat1_col`, yang berisi kolom-kolom kategorikal yang telah difilter sebelumnya.

Pertama, perulangan (`for col in cat1_col`) digunakan untuk mengiterasi setiap kolom dalam daftar `cat1_col`. Dalam setiap iterasi, kode `print(col)` mencetak nama kolom yang sedang diproses untuk memberikan indikasi progres. Selanjutnya, fungsi `labelEncoding(df_accepted_fe, col)` digunakan untuk menerapkan Label Encoding pada kolom tersebut, dan hasilnya diperbarui kembali ke dalam `df_accepted_fe`.

```
# Alternative syntax (same result)
df_accepted_fe.drop(columns=cat1_col, inplace=True)
```

```
df_accepted_fe.drop(columns=['url'], inplace=True)
```

Menghapus kolom "url" dari dataframe df_accepted_fe.

```
df_accepted_fe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 23437 entries, 0 to 25513
Data columns (total 96 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               23437 non-null   int32  
 1   loan_amnt        23437 non-null   int32  
 2   funded_amnt      23437 non-null   int32  
 3   funded_amnt_inv  23437 non-null   int32  
 4   int_rate          23437 non-null   float32 
 5   installment       23437 non-null   float32 
 6   annual_inc        23437 non-null   float64 
 7   issue_d           23437 non-null   datetime64[ns]
 8   dti               23437 non-null   float32 
 9   delinq_2yrs       23437 non-null   int8   
 10  earliest_cr_line 23437 non-null   datetime64[ns]
 11  fico_range_low   23437 non-null   int16  
 12  fico_range_high  23437 non-null   int16  
 13  inq_last_6mths   23437 non-null   int8   
 14  mths_since_last_delinq 12354 non-null   float32 
 15  open_acc          23437 non-null   int8   
 16  pub_rec           23437 non-null   int8   
 17  revol_bal         23437 non-null   int32  
 18  revol_util        23437 non-null   float32 
 19  total_acc          23437 non-null   int8   
 20  out_prncp         23437 non-null   float64 
 21  out_prncp_inv    23437 non-null   float64 
 22  total_pymnt       23437 non-null   float64 
 23  total_pymnt_inv  23437 non-null   float64 
 24  total_rec_prncp  23437 non-null   float64 
 25  total_rec_int     23437 non-null   float64 
 26  total_rec_late_fee 23437 non-null   float32
```

27	recoveries		23437	non-null	float64
28	collection_recovery_fee		23437	non-null	float32
29	last_pymnt_d		23437	non-null	datetime64[ns]
30	last_pymnt_amnt		23437	non-null	float64
31	last_credit_pull_d		23437	non-null	datetime64[ns]
32	last_fico_range_high		23437	non-null	int16
33	last_fico_range_low		23437	non-null	int16
34	collections_12_mths_ex_med		23437	non-null	int8
35	policy_code		23437	non-null	int8
36	acc_now_delinq		23437	non-null	int8
37	tot_coll_amt		23437	non-null	int32
38	tot_cur_bal		23437	non-null	int32
39	il_util		17400	non-null	float32
40	all_util		19578	non-null	float32
41	total_rev_hi_lim		23437	non-null	int32
42	acc_open_past_24mths		23437	non-null	int8
43	avg_cur_bal		23437	non-null	int32
44	bc_open_to_buy		23437	non-null	int32
45	bc_util		23437	non-null	float32
46	chargeoff_within_12_mths		23437	non-null	int8
47	delinq_amnt		23437	non-null	int32
48	mo_sin_old_il_acct		23437	non-null	int16
49	mo_sin_old_rev_tl_op		23437	non-null	int16
50	mo_sin_rcnt_rev_tl_op		23437	non-null	int16
51	mo_sin_rcnt_tl		23437	non-null	int16
52	mort_acc		23437	non-null	int8
53	mths_since_recent_bc		23437	non-null	int16
54	mths_since_recent_inq		23437	non-null	int8
55	num_accts_ever_120_pd		23437	non-null	int8
56	num_actv_bc_tl		23437	non-null	int8
57	num_actv_rev_tl		23437	non-null	int8
58	num_bc_sats		23437	non-null	int8

```
59 num_bc_t1           23437 non-null int8
60 num_il_t1           23437 non-null int8
61 num_op_rev_tl       23437 non-null int8
62 num_rev_accts      23437 non-null int8
63 num_rev_tl_bal_gt_0 23437 non-null int8
64 num_sats            23437 non-null int8
65 num_tl_120dpd_2m    23437 non-null int8
66 num_tl_30dpd        23437 non-null int8
67 num_tl_90g_dpd_24m 23437 non-null int8
68 num_tl_op_past_12m 23437 non-null int8
69 pct_tl_nvr_dlq     23437 non-null float32
70 percent_bc_gt_75    23437 non-null float32
71 pub_rec_bankruptcies 23437 non-null int8
72 tax_liens           23437 non-null int8
73 tot_hi_cred_lim    23437 non-null int32
74 total_bal_ex_mort   23437 non-null int32
75 total_bc_limit      23437 non-null int32
76 total_il_high_credit_limit 23437 non-null int32
77 zip_3digits         23437 non-null int16
78 term_encoded         23437 non-null int8
79 grade_encoded        23437 non-null int8
80 sub_grade_encoded   23437 non-null int8
81 emp_title_encoded   23437 non-null int16
82 emp_length_encoded  23437 non-null int8
83 home_ownership_encoded 23437 non-null int8
84 verification_status_encoded 23437 non-null int8
85 loan_status_encoded  23437 non-null int8
86 pymnt_plan_encoded  23437 non-null int8
87 url_encoded          23437 non-null int16
88 purpose_encoded      23437 non-null int8
89 title_encoded        23437 non-null int8
90 addr_state_encoded  23437 non-null int8
91 initial_list_status_encoded 23437 non-null int8
92 application_type_encoded 23437 non-null int8
93 hardship_flag_encoded 23437 non-null int8
94 disbursement_method_encoded 23437 non-null int8
95 debt_settlement_flag_encoded 23437 non-null int8
dtypes: datetime64[ns](4), float32(12), float64(9), int16(12), int32(15), int8(44)
memory usage: 6.4 MB
```

Menampilkan informasi mengenai struktur dataframe df_accepted_fe. Output yang dihasilkan mencakup jumlah total baris dan kolom, nama kolom beserta tipe datanya, serta jumlah nilai yang tidak kosong di setiap kolom. Selain itu, informasi mengenai tipe data dalam dataframe juga ditampilkan, seperti jumlah kolom yang memiliki tipe data numerik (int64 dan float64).

```
df_rejected_clean
```

	amount_requested	application_date	loan_title	risk_score	debt-to-income_ratio	zip_code	state	employment_length	policy_code
0	1000.0	2007-05-26	Wedding Covered but No Honeymoon	693	10%	481xx	NM	4 years	0
1	1000.0	2007-05-26	Consolidating Debt	703	10%	010xx	MA	< 1 year	0
2	11000.0	2007-05-27	Want to consolidate my debt	715	10%	212xx	MD	1 year	0
3	6000.0	2007-05-27	waksman	698	38.64%	017xx	MA	< 1 year	0
4	1500.0	2007-05-27	mdrigo	509	9.43%	209xx	MD	< 1 year	0
...
461121	35000.0	2012-03-13	debt_consolidation	586	42.22%	103xx	NY	< 1 year	0
461122	15000.0	2012-03-13	credit_card	576	14.77%	914xx	CA	< 1 year	0
461123	1000.0	2012-03-13	Vacation	798	0%	956xx	CA	Unknown	0
461124	20000.0	2012-03-13	major_purchase	724	17.1%	108xx	NY	< 1 year	0
461125	15000.0	2012-03-13	car	604	17.76%	658xx	MO	< 1 year	0

461092 rows × 9 columns

Menampilkan data dari df_rejected_clean

df_rejected_clean.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 461092 entries, 0 to 461125
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   amount_requested    461092 non-null   float64
 1   application_date    461092 non-null   datetime64[ns]
 2   loan_title          461092 non-null   object 
 3   risk_score          461092 non-null   int16  
 4   debt-to-income_ratio 461092 non-null   object 
 5   zip_code            461092 non-null   object 
 6   state               461092 non-null   object 
 7   employment_length    461092 non-null   object 
 8   policy_code          461092 non-null   int8  
dtypes: datetime64[ns](1), float64(1), int16(1), int8(1), object(5)
memory usage: 29.5+ MB
```

Menampilkan informasi dari df_rejected_clean beserta tipe data dari masing masing coloum

```
df_rejected_clean['zip_3digits'] = pd.to_numeric(df_rejected_clean['zip_code'].str[:3], errors='coerce').fillna(0).astype(int)
df_rejected_clean.drop(columns=['zip_code'], inplace=True)
```

```
<ipython-input-145-3aad4436b691>:1: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
df_rejected_clean['zip_3digits'] = pd.to_numeric(df_rejected_clean['zip_code'].str[:3], errors='coerce').fillna(0).astype(int)  
<ipython-input-145-3aad4436b691>:2: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
df_rejected_clean.drop(columns=['zip_code'], inplace=True)
```

Mengolah data pada dataframe df_rejected_clean dengan mengekstrak tiga digit pertama dari kolom zip_code dan menyimpannya dalam kolom baru bernama zip_3digits. Proses ini dilakukan dengan mengambil substring dari tiga karakter pertama dalam zip_code, kemudian mengonversinya menjadi nilai numerik menggunakan pd.to_numeric(), dengan opsi errors='coerce' untuk menggantikan nilai yang tidak dapat dikonversi menjadi NaN. Selanjutnya, nilai NaN yang muncul akan diisi dengan angka 0, lalu dikonversi menjadi tipe data integer. Setelah itu, kolom asli zip_code dihapus menggunakan drop(columns=['zip_code'], inplace=True), karena informasi yang dibutuhkan telah disimpan dalam kolom baru. Hasil akhirnya, dataframe hanya menyimpan tiga digit pertama dari kode pos dalam bentuk numerik.

```
# Convert a single column  
df_rejected_clean['application_date'] =  
pd.to_datetime(df_rejected_clean['application_date'])  
  
<ipython-input-147-b03c67e42fb4>:2: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
df_rejected_clean['application_date'] = pd.to_datetime(df_rejected_clean['application_date'])
```

Mengonversi kolom application_date dalam dataframe df_rejected_clean menjadi format datetime menggunakan fungsi pd.to_datetime(). Dengan konversi ini, nilai dalam kolom application_date yang sebelumnya mungkin berbentuk string atau format lain akan diubah menjadi tipe data datetime, memungkinkan manipulasi yang lebih mudah seperti perhitungan selisih tanggal, ekstraksi informasi seperti tahun atau bulan, serta penyortiran berdasarkan waktu.

```
df_rejected_clean['debt-to-income_ratio'] =  
df_rejected_clean['debt-to-income_ratio'].str.replace('%',  
'').astype(float) / 100
```

```
<ipython-input-149-997f54ea69e9>:1: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
df_rejected_clean['debt-to-income_ratio'] = df_rejected_clean['debt-to-income_ratio'].str.replace('%', '').astype(float) / 100
```

Mengonversi nilai dalam kolom debt-to-income_ratio dari format string yang mengandung simbol persen (%) menjadi nilai numerik dalam bentuk desimal.

```
df_rejected_clean.select_dtypes(include=['object', 'category'])
```

	loan_title	state	employment_length
0	Wedding Covered but No Honeymoon	NM	4 years
1	Consolidating Debt	MA	< 1 year
2	Want to consolidate my debt	MD	1 year
3	waksman	MA	< 1 year
4	mdrigo	MD	< 1 year
...
461121	debt_consolidation	NY	< 1 year
461122	credit_card	CA	< 1 year
461123	Vacation	CA	Unknown
461124	major_purchase	NY	< 1 year
461125	car	MO	< 1 year

461092 rows × 3 columns

Mengekstrak dan menganalisis kolom kategorikal dari DataFrame df_rejected_clean.

Pertama, df_rejected_categorical = df_rejected_clean.select_dtypes(include=['object', 'category']) digunakan untuk memilih hanya kolom dengan tipe data object atau category, yang biasanya berisi data kategorikal. Hasilnya disimpan

```
dalam DataFrame df_rejected_categorical.
```

Selanjutnya, cat_cols_rejected = df_rejected_categorical.columns mengambil daftar nama kolom kategorikal dari DataFrame tersebut. Perulangan (for col in cat_cols_rejected) kemudian digunakan untuk mengiterasi setiap kolom dalam daftar ini.

Di dalam perulangan, df_rejected_categorical[col].nunique() menghitung jumlah kategori unik dalam setiap kolom, lalu hasilnya dicetak dalam format '<nama kolom>': <jumlah kategori> categories.

Tujuan dari kode ini adalah untuk memahami distribusi kategori dalam dataset pinjaman yang ditolak (df_rejected_clean). Ini membantu dalam mengidentifikasi jumlah kategori unik di setiap kolom, yang penting untuk menentukan strategi encoding yang sesuai sebelum pemrosesan lebih lanjut dalam analisis data atau machine learning.

```
# Method 1: Using select_dtypes()
df_rejected_categorical = df_rejected_clean.select_dtypes(include=['object', 'category'])

cat_cols_rejected = df_rejected_categorical.columns

for col in cat_cols_rejected:
    num_categories = df_rejected_categorical[col].nunique()
    print(f'{col}: {num_categories} categories')

'loan_title': 55457 categories
'state': 51 categories
'employment_length': 12 categories
```

```
for col in cat_cols_rejected:
    print(f"\nColumn: {col}")
    print(df_rejected_categorical[col].value_counts())

Column: loan_title
loan_title
debt_consolidation           129088
other                          53957
car                            29961
credit_card                     25147
small_business                  23600
...
Independent Undergrad Student In Need      1
pay off accounts                   1
getting my life back together       1
Emergency Loan to cover college tuition   1
Tax on laflin                      1
Name: count, Length: 55457, dtype: int64

Column: state
state
CA      61275
TX      37604
NY      36207
FL      35812
PA      21236
IL      19689
GA      18024
OH      17906
NJ      17126
VA      14741
MI      12539
MD      10872
MA      10810
NC      10761
AZ      9934
MO      9392
```

CO	8504
WA	8485
AL	8181
SC	7687
CT	7382
WI	6979
LA	6810
KY	6427
MN	6390
AR	5515
NV	5474
OK	5073
OR	4354
KS	3488
WV	3234
UT	3110
NM	2706
NH	2617
HI	2392
RI	2225
DE	1671
AK	1209
MT	1207
DC	1162
VT	1001
WY	975
SD	955
MS	523
IN	520
TN	406
NE	157
IA	144
ME	101
ID	68
ND	32

Name: count, dtype: int64

Column: employment_length

```
Column: employment_length  
employment_length  
< 1 year      317432  
10+ years     29213  
2 years       21722  
1 year        21632  
3 years       17006  
4 years       13061  
5 years       11270  
6 years       8461  
7 years       6032  
8 years       5685  
Unknown        5019  
9 years       4559  
Name: count, dtype: int64
```

Menampilkan distribusi nilai unik dalam setiap kolom kategorikal yang terdapat dalam df_rejected_categorical.

Pertama, perulangan for col in cat_cols_rejected digunakan untuk mengiterasi setiap kolom dalam daftar cat_cols_rejected, yang berisi nama-nama kolom kategorikal dari dataset pinjaman yang ditolak (df_rejected_clean). Pada setiap iterasi, print(f"\nColumn: {col}") mencetak nama kolom yang sedang diproses, disertai dengan pemisah baris baru (\n) agar output lebih rapi dan mudah dibaca.

Selanjutnya, df_rejected_categorical[col].value_counts() menghitung jumlah kemunculan (frekuensi) setiap kategori dalam kolom tersebut, lalu hasilnya ditampilkan.

Tujuan dari kode ini adalah untuk memahami distribusi data dalam variabel kategorikal, mengidentifikasi kategori yang paling umum atau jarang muncul, serta mendeteksi kemungkinan ketidakseimbangan dalam dataset. Informasi ini sangat berguna sebelum melakukan encoding atau

```
analisis lebih lanjut dalam pemodelan machine learning.
```

```
for col in cat_cols_rejected:  
    print(col)  
    df_rejected = labelEncoding(df_rejected_clean, col)  
  
loan_title  
<ipython-input-110-68c33afe4a2b>:3: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
df[var+'_encoded'] = le.fit_transform(df[var])  
Maps of loan title  
{' ': np.int64(0), '      to pay off settlement': np.int64(1), ' time to fix up the house': np.int64(2), ' Borrower added on 04/16/10 > Requesting $8000 - $10,000. Higher amount pr  
state  
Maps of state  
{'AK': np.int64(0), 'AL': np.int64(1), 'AR': np.int64(2), 'AZ': np.int64(3), 'CA': np.int64(4), 'CO': np.int64(5), 'CT': np.int64(6), 'DC': np.int64(7), 'DE': np.int64(8), 'FL': np.int  
employment_length  
Maps of employment_length  
{'1 year': np.int64(0), '10+ years': np.int64(1), '2 years': np.int64(2), '3 years': np.int64(3), '4 years': np.int64(4), '5 years': np.int64(5), '6 years': np.int64(6), '7 years': np  
<ipython-input-110-68c33afe4a2b>:3: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
df[var+'_encoded'] = le.fit_transform(df[var])  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
df[var+'_encoded'] = le.fit_transform(df[var])
```

Menerapkan Label Encoding pada setiap kolom kategorikal dalam df_rejected_clean.

Pertama, perulangan for col in cat_cols_rejected digunakan untuk mengiterasi setiap kolom dalam daftar cat_cols_rejected, yang berisi nama-nama kolom kategorikal dalam dataset pinjaman yang ditolak. Pada setiap iterasi, print(col) mencetak nama kolom yang sedang diproses untuk memantau progres encoding.

Selanjutnya, df_rejected = labelEncoding(df_rejected_clean, col) menerapkan fungsi labelEncoding() pada kolom tersebut dan memperbarui hasilnya ke dalam df_rejected

```
df_rejected_clean.drop(columns=cat_cols_rejected, inplace=True)
```

```
<ipython-input-155-e4f32cb1736c>:1: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy  
df_rejected_clean.drop(columns=cat_cols_rejected, inplace=True)
```

Menghapus semua kolom kategorikal dalam cat_cols_rejected dari DataFrame df_rejected_clean.

Fungsi drop(columns=cat_cols_rejected, inplace=True) secara langsung menghapus kolom-kolom dalam daftar cat_cols_rejected, yang sebelumnya telah diidentifikasi

sebagai kolom kategorikal dalam df_rejected_clean. Parameter inplace=True memastikan bahwa perubahan dilakukan langsung pada DataFrame tanpa perlu menyimpannya kembali dalam variabel baru.

df_rejected_clean

	amount_requested	application_date	risk_score	debt-to-income_ratio	policy_code	zip_3digits	loan_title_encoded	state_encoded	employment_length_encoded	grid	info	edit
0	1000.0	2007-05-26	693	0.1000	0	481	38631	32	4			
1	1000.0	2007-05-26	703	0.1000	0	10	7297	19	10			
2	11000.0	2007-05-27	715	0.1000	0	212	38400	20	0			
3	6000.0	2007-05-27	698	0.3864	0	17	54985	19	10			
4	1500.0	2007-05-27	509	0.0943	0	209	48140	20	10			
...			
461121	35000.0	2012-03-13	586	0.4222	0	103	42861	34	10			
461122	15000.0	2012-03-13	576	0.1477	0	914	42364	4	10			
461123	1000.0	2012-03-13	798	0.0000	0	956	37973	4	11			
461124	20000.0	2012-03-13	724	0.1710	0	108	47892	34	10			
461125	15000.0	2012-03-13	604	0.1776	0	658	40994	24	10			

461092 rows x 9 columns

Menampilkan data setelah diproses, tampak bahwa tidak perlu di normalisasi numerik lagi, tidak ada tipe data yang bertipe kategori

```
df_accepted_fe.to_csv(' [FeatureEngineered]accepted_2007_to_2018Q4.csv', index=False)
df_rejected_clean.to_csv(' [FeatureEngineered]rejected_2007_to_2018Q4.csv', index=False)
```

Tahap Transformasi dan Rekayasa Fitur telah selesai, dan hasilnya disimpan dalam dua file CSV:

- [FeatureEngineered]accepted_2007_to_2018Q4.csv – Berisi data pemohon yang diterima dengan fitur yang telah direkayasa.
- [FeatureEngineered]rejected_2007_to_2018Q4.csv – Berisi data pemohon yang ditolak dengan fitur yang telah dibersihkan dan direkayasa.

#Dataset Home Credit Default Risk Data

```
application_train.csv
application_test.csv
installments_payments.csv
```

```
import pandas as pd
# import category_encoders as ce
from sklearn.preprocessing import LabelEncoder
```

Mengimpor pustaka pandas (`import pandas as pd`), yang digunakan untuk manipulasi dan analisis data dalam bentuk tabel (`DataFrame` dan `Series`). Selanjutnya, terdapat baris yang dikomentari (`# import category_encoders as ce`), yang menunjukkan bahwa pustaka `category_encoders` awalnya mungkin digunakan untuk mengubah variabel kategori menjadi numerik, tetapi saat ini tidak diaktifkan. Kemudian, kode mengimpor `LabelEncoder` dari `sklearn.preprocessing`, yang digunakan untuk mengonversi variabel kategori menjadi angka secara ordinal.

```
def downcasting(df):
    # Select only numeric columns (int64 and float64)
    numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns

    # Downcast numeric columns
    df[numeric_cols] = df[numeric_cols].apply(
        pd.to_numeric, downcast='integer'
    )

    # For float columns (separate step to avoid mixing types)
    float_cols = df.select_dtypes(include=['float64']).columns
    df[float_cols] = df[float_cols].apply(
        pd.to_numeric, downcast='float'
    )

    return df
```

Fungsi `downcasting(df)` bertujuan untuk mengurangi penggunaan memori

dataset dengan mengonversi tipe data numerik ke format yang lebih kecil tanpa kehilangan informasi.

Pertama, fungsi memilih kolom yang bertipe int64 dan float64 menggunakan `df.select_dtypes(include=['int64', 'float64']).columns`. Kemudian, pada langkah pertama downcasting, semua kolom numerik dikonversi ke tipe data integer dengan `pd.to_numeric(downcast='integer')`. Setelah itu, fungsi kembali memilih kolom bertipe float64 dan mengonversinya menjadi tipe data float dengan `pd.to_numeric(downcast='float')`. Pemisahan langkah untuk integer dan float ini dilakukan agar tidak terjadi kesalahan pengubahan tipe data. Akhirnya, dataset yang telah dioptimalkan dikembalikan.

Teknik ini berguna untuk menghemat memori, terutama saat bekerja dengan dataset besar, tanpa mengubah nilai data yang tersimpan.

```
df_application_train_clean = downcasting(df_application_train_clean)
df_installments_payments_clean
= downcasting(df_installments_payments_clean)
```

```
<ipython-input-54-b59a6271e4a9>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df[numeric_cols] = df[numeric_cols].apply(
<ipython-input-54-b59a6271e4a9>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df[float_cols] = df[float_cols].apply(
```

Fungsi `downcasting` akan mengonversi tipe data numerik dalam kedua DataFrame tersebut ke format yang lebih kecil untuk menghemat penggunaan memori. Kolom dengan tipe int64 akan dikonversi ke tipe integer dengan ukuran lebih kecil (int32, int16, atau int8 sesuai dengan nilai yang tersimpan), sedangkan kolom dengan tipe float64 akan dikonversi ke tipe float yang lebih efisien (float32 atau

```
float16).
```

Hasilnya, kedua dataset (`df_application_train_clean` dan `df_installments_payment_clean`) akan memiliki ukuran memori yang lebih kecil tanpa mengubah isi datanya, sehingga analisis dan pemrosesan data dapat berjalan lebih cepat dan efisien.

`df_application_train_clean`

SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	...	NONLIVINGAPARTMENTS_MEDI	FONDK
0	100002	1	Cash loans	M	N	Y	0	202500.0	406597.5	24700.5	...	0.0000
1	100003	0	Cash loans	F	N	N	0	270000.0	1293502.5	35698.5	...	0.0039
2	100004	0	Revolving loans	M	Y	Y	0	67500.0	135000.0	6750.0	...	0.0000
3	100006	0	Cash loans	F	N	Y	0	185000.0	312682.5	28686.5	...	0.0000
4	100007	0	Cash loans	M	N	Y	0	121500.0	513000.0	21865.5	...	0.0000
...
163069	289052	0	Cash loans	M	Y	Y	0	184500.0	585000.0	35919.0	...	0.0000
163070	289053	0	Cash loans	F	Y	Y	1	90000.0	1506816.0	47443.5	...	0.0000
163071	289054	0	Cash loans	M	N	N	0	153000.0	144486.0	7695.0	...	0.0000
163072	289056	0	Cash loans	F	Y	Y	1	270000.0	1256400.0	36864.0	...	0.0116
163073	289058	0	Cash loans	F	N	Y	0	58500.0	239850.0	23494.5	...	0.0000

Kode `df_application_train_clean` digunakan untuk menampilkan isi dari DataFrame `df_application_train_clean`

```
# Method 1: Using select_dtypes()
df_application_train_categorical = df_application_train_clean.select_dtypes(include=[ 'object',
'category'])
```

Dengan menggunakan metode `select_dtypes(include=['object', 'category'])`, hanya kolom yang memiliki tipe data object (string) atau category yang dipilih dan disimpan dalam DataFrame baru `df_application_train_categorical`. Hal ini memungkinkan pemisahan variabel kategorikal dari variabel numerik untuk dianalisis atau diproses lebih lanjut.

Tujuan utama dari kode ini adalah untuk mempermudah eksplorasi dan transformasi data kategorikal, seperti encoding atau analisis distribusi.

```
cat_cols_application_train = df_application_train_categorical.columns

for col in cat_cols_application_train:
    num_categories = df_application_train_categorical[col].nunique()
    print(f'{col}: {num_categories} categories')

'NAME_CONTRACT_TYPE': 2 categories
'CODE_GENDER': 3 categories
'FLAG_OWN_CAR': 2 categories
'FLAG_OWN_REALTY': 2 categories
'NAME_TYPE_SUITE': 7 categories
'NAME_INCOME_TYPE': 8 categories
'NAME_EDUCATION_TYPE': 5 categories
'NAME_FAMILY_STATUS': 6 categories
'NAME_HOUSING_TYPE': 6 categories
'OCCUPATION_TYPE': 18 categories
'WEEKDAY_APPR_PROCESS_START': 7 categories
'ORGANIZATION_TYPE': 58 categories
'FONDKAPREMONT_MODE': 4 categories
'HOUSETYPE_MODE': 3 categories
'WALLSMATERIAL_MODE': 7 categories
'EMERGENCYSTATE_MODE': 2 categories
```

cat_cols_application_train = df_application_train_categorical.columns mengambil daftar semua kolom kategorikal dalam DataFrame. Kemudian, perulangan `for col in cat_cols_application_train` digunakan untuk mengiterasi setiap kolom dalam daftar tersebut.

Di dalam perulangan, `df_application_train_categorical[col].nunique()` menghitung jumlah kategori unik dalam kolom tersebut. Hasilnya kemudian dicetak dalam format '`<nama kolom>: <jumlah kategori> categories`', sehingga memudahkan untuk memahami kompleksitas data kategorikal dalam dataset.

```
def labelEncoding(df, col):
    # Create a LabelEncoder instance
    le = LabelEncoder()

    # Fit and transform the column
    df[col] = le.fit_transform(df[col])

    return df
```

Fungsi menerima dua parameter: `df`, yaitu DataFrame yang ingin diproses, dan `col`, yaitu nama kolom yang akan dienkode. Di dalam fungsi, objek `LabelEncoder` dari scikit-learn dibuat dengan `le = LabelEncoder()`. Label encoder ini akan digunakan untuk mengonversi nilai kategorikal menjadi bentuk numerik.

Selanjutnya, kode `df[col] = le.fit_transform(df[col])` melakukan dua proses sekaligus:

1. `fit()` – Mempelajari nilai unik dalam kolom yang diberikan.
2. `transform()` – Mengonversi setiap nilai kategori menjadi angka berdasarkan pemetaan yang telah dipelajari.

Sebagai contoh, jika sebuah kolom memiliki kategori `['Low', 'Medium', 'High']`, setelah Label Encoding akan menjadi `[1, 2, 0]`, tergantung urutan yang dipelajari oleh encoder.

Terakhir, fungsi mengembalikan DataFrame yang telah dimodifikasi.

```
for col in cat_cols_application_train:
    print(col)
    df_application_train_clean=
```

```
labelEncoding(df_application_train_clean, col)
df_application_train_clean.drop(columns=[col], inplace=True)

NAME_CONTRACT_TYPE
CODE_GENDER
FLAG_OWN_CAR
FLAG_OWN_REALTY
NAME_TYPE_SUITE
NAME_INCOME_TYPE
NAME_EDUCATION_TYPE
NAME_FAMILY_STATUS
NAME_HOUSING_TYPE
OCCUPATION_TYPE
WEEKDAY_APPR_PROCESS_START
ORGANIZATION_TYPE
FONDKAPREMONT_MODE
HOUSETYPE_MODE
WALLSMATERIAL_MODE
EMERGENCYSTATE_MODE
```

Menerapkan Label Encoding pada setiap kolom kategorikal dalam `df_application_train_clean`, kemudian menghapus kolom asli setelah encoding selesai.

Pertama, perulangan `for col in cat_cols_application_train` digunakan untuk mengiterasi setiap kolom kategorikal dalam dataset. Pada setiap iterasi, `print(col)` mencetak nama kolom yang sedang diproses, membantu dalam memantau progres encoding.

Selanjutnya, `df_application_train_clean = labelEncoding(df_application_train_clean, col)` menerapkan fungsi Label Encoding pada kolom yang sedang diproses. Fungsi `labelEncoding()` mengonversi nilai kategorikal dalam kolom menjadi angka menggunakan `LabelEncoder` dari scikit-learn.

Setelah encoding selesai,

```
df_application_train_clean.drop(columns=[col], inplace=True)  
menghapus kolom asli yang berisi nilai kategorikal, sehingga hanya  
nilai numerik hasil encoding yang tersisa dalam DataFrame.
```

```
df_application_train_clean.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 163074 entries, 0 to 163073
Data columns (total 51 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   SK_ID_CURR       163074 non-null   int32  
 1   TARGET          163074 non-null   int8   
 2   CNT_CHILDREN    163074 non-null   int8   
 3   AMT_INCOME_TOTAL 163074 non-null   float64 
 4   AMT_CREDIT       163074 non-null   float32 
 5   AMT_ANNUITY      163074 non-null   float32 
 6   AMT_GOODS_PRICE  163074 non-null   float32 
 7   REGION_POPULATION_RELATIVE 163074 non-null   float32 
 8   DAYS_BIRTH       163074 non-null   int16  
 9   DAYS_EMPLOYED    163074 non-null   int32  
 10  DAYS_REGISTRATION 163074 non-null   int16  
 11  DAYS_ID_PUBLISH 163074 non-null   int16  
 12  OWN_CAR_AGE     163074 non-null   float32 
 13  FLAG_MOBIL      163074 non-null   int8   
 14  FLAG_EMP_PHONE  163074 non-null   int8   
 15  FLAG_WORK_PHONE 163074 non-null   int8   
 16  FLAG_CONT_MOBILE 163074 non-null   int8   
 17  FLAG_PHONE      163074 non-null   int8   
 18  FLAG_EMAIL      163074 non-null   int8   
 19  CNT_FAM_MEMBERS 163074 non-null   float32 
 20  REGION_RATING_CLIENT 163074 non-null   int8   
 21  REGION_RATING_CLIENT_W_CITY 163074 non-null   int8  
 22  HOUR_APPR_PROCESS_START 163074 non-null   int8  
 23  REG_REGION_NOT_LIVE_REGION 163074 non-null   int8  
 24  REG_REGION_NOT_WORK_REGION 163074 non-null   int8  
 25  LIVE_REGION_NOT_WORK_REGION 163074 non-null   int8  
 26  REG_CITY_NOT_LIVE_CITY 163074 non-null   int8  
 27  REG_CITY_NOT_WORK_CITY 163074 non-null   int8  
 28  LIVE_CITY_NOT_WORK_CITY 163074 non-null   int8  
 29  EXT_SOURCE_1     163074 non-null   float32 
 30  EXT_SOURCE_2     163074 non-null   float32 
 31  EXT_SOURCE_3     163074 non-null   float32 

```

```

32 BASEMENTAREA_AVG           163074 non-null float32
33 COMMONAREA_AVG             163074 non-null float32
34 ENTRANCES_AVG              163074 non-null float32
35 FLOORSMIN_AVG              163074 non-null float32
36 LANDAREA_AVG                163074 non-null float32
37 LIVINGAREA_AVG              163074 non-null float32
38 NONLIVINGAREA_AVG            163074 non-null float32
39 ELEVATORS_MODE               163074 non-null float32
40 FLOORSMAX_MODE              163074 non-null float32
41 APARTMENTS_MEDI              163074 non-null float32
42 YEARS_BEGINEXPLUATATION_MEDI 163074 non-null float32
43 YEARS_BUILD_MEDI              163074 non-null float32
44 LIVINGAPARTMENTS_MEDI          163074 non-null float32
45 NONLIVINGAPARTMENTS_MEDI        163074 non-null float32
46 TOTALAREA_MODE                163074 non-null float32
47 DEF_60_CNT_SOCIAL_CIRCLE       163074 non-null float32
48 DAYS_LAST_PHONE_CHANGE         163074 non-null float32
49 FLAG_DOCUMENT_14                163074 non-null float32
50 AMT_REQ_CREDIT_BUREAU_DAY       163074 non-null float32
dtypes: float32(28), float64(1), int16(3), int32(2), int8(17)
memory usage: 23.5 MB

```

df_installments_payments_clean

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	DAYSTINSTALMENT	DAYSENTRY_PAYMENT	AMTINSTALMENT	AMTPAYMENT
0	1054186	161674		1	6	-1180	-1187	6948.360
1	1330831	151639		0	34	-2156	-2156	1716.525
2	2085231	193053		2	1	-63	-63	25425.000
3	2452527	199697		1	3	-2418	-2426	24350.130
4	2714724	167756		1	2	-1383	-1366	2165.040
...
2229301	1054930	126482		9	11	-341	-399	2540.070
2229302	2387650	168074		1	7	-2171	-2187	23250.195
2229303	1880340	143047		1	19	-463	-466	19025.100
2229304	2225749	192652		1	3	-569	-608	5033.250
2229305	1054695	112235		0	107	-1082	-1080	3375.000

2229306 rows × 8 columns

Kode `df_installments_payments_clean` digunakan untuk menampilkan isi dari DataFrame `df_installments_payments_clean`.

`df_installments_payments_clean.info()`

```
<class 'pandas.core.frame.DataFrame'>
Index: 2229306 entries, 0 to 2229305
Data columns (total 8 columns):
 #   Column           Dtype  
 --- 
 0   SK_ID_PREV        int32  
 1   SK_ID_CURR        int32  
 2   NUM_INSTALMENT_VERSION  int8   
 3   NUM_INSTALMENT_NUMBER  int16  
 4   DAYS_INSTALMENT     int16  
 5   DAYS_ENTRY_PAYMENT  int16  
 6   AMT_INSTALMENT      float64 
 7   AMT_PAYMENT         float64 
dtypes: float64(2), int16(3), int32(2), int8(1)
memory usage: 82.9 MB
```

Kode `df_installments_payments_clean.info()` digunakan untuk menampilkan ringkasan struktur DataFrame `df_installments_payments_clean`. Dan semua data sudah pada format numerikal.

```
df_application_train_clean.to_csv(' [FeatureEngineered]application_train.csv', index=False)
df_installments_payments_clean.to_csv(' [FeatureEngineered]installments_payments.csv', index=False)
```

Tahap Transformasi dan Rekayasa Fitur telah selesai, dan hasilnya disimpan dalam dua file CSV:

- `[FeatureEngineered]application_train.csv` – Berisi data pemohon yang diterima dengan fitur yang telah direkayasa.
- `[FeatureEngineered]installments_payments.csv` – Berisi data pemohon yang ditolak dengan fitur yang telah dibersihkan dan direkayasa.

Anggota Empat | Tahiyah Mufhimah_5026231170

Analisis Ketidakseimbangan Kelas dan Perbandingan Metode Prapemrosesan untuk Kedua Dataset

Langkah:

1. Identifikasi Variabel Target dan Distribusi Kelas
2. Pengukuran Skewness dan Kurtosis
3. Penanganan Ketidakseimbangan Kelas dengan menerapkan beberapa metode untuk menyeimbangkan data
4. Evaluasi Performa Model Setelah Prapemrosesan
5. Perbandingan Kesiapan Dataset untuk Prediksi

Dibuat sebuah fungsi bernama **downcasting(df)** yang bertujuan untuk mengurangi penggunaan memori dalam DataFrame dengan mengonversi tipe data numerik ke tipe yang lebih kecil (downcasting).

```
def downcasting(df):  
    # Select only numeric columns (int64 and float64)  
    numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns  
  
    # Downcast numeric columns  
    df[numeric_cols] = df[numeric_cols].apply(  
        pd.to_numeric, downcast='integer'  
    )  
  
    # For float columns (separate step to avoid mixing types)  
    float_cols = df.select_dtypes(include=['float64']).columns  
    df[float_cols] = df[float_cols].apply(  
        pd.to_numeric, downcast='float'  
    )  
  
    return df
```

Fungsi ini kemudian diterapkan pada beberapa DataFrame, seperti `df_accepted_clean` dan `df_rejected_clean`, untuk mengoptimalkan ukuran

data yang digunakan dalam analisis. Proses downcasting ini berguna untuk menghemat penggunaan memori tanpa mengubah nilai asli dalam dataset.

```
df_accepted = downcasting(df_accepted_clean)
df_rejected = downcasting(df_rejected_clean)
```

#Dataset Lending Club Loan Data

`accepted_2007_to_2018Q4.csv`

`df_accepted`

	<code>id</code>	<code>loan_amnt</code>	<code>funded_amnt</code>	<code>funded_amnt_inv</code>	<code>term</code>	<code>int_rate</code>	<code>installment</code>	<code>grade</code>	<code>sub_grade</code>	<code>emp_title</code>	...	<code>percent_bc_gt_75</code>	<code>pub_rec_bankruptcies</code>	<code>tax_liens</code>	<code>tot_hi_cred_lim</code>
0	68407277	3600	3600	3600.0	36 months	13.990000	123.029999	C	C4	leadman	...	0.000000	0	0	178050
1	68355089	24700	24700	24700.0	36 months	11.990000	820.280029	C	C1	Engineer	...	7.700000	0	0	314017
2	66310712	35000	35000	35000.0	60 months	14.850000	829.900024	C	C5	Information Systems Officer	...	0.000000	0	0	381215
3	68476807	10400	10400	10400.0	60 months	22.450001	289.910004	F	F1	Contract Specialist	...	60.000000	0	0	439570
4	68426831	11950	11950	11950.0	36 months	13.440000	405.179993	C	C3	Veterinary Technician	...	100.000000	0	0	16900
...
17067	89996426	32000	32000	32000.0	60 months	14.490000	752.739990	C	C4	Sales Manager	...	0.000000	0	0	524379
17068	90006534	16000	16000	16000.0	60 months	12.790000	362.339996	C	C1	Manager	...	50.000000	3	0	87473
17069	89955820	24000	24000	24000.0	60 months	10.490000	515.739990	B	B3	Current Operations Officer	...	0.000000	0	2	128902
17070	88977788	24000	24000	24000.0	60 months	10.490000	515.739990	B	B3	Database Administrator	...	42.900002	0	1	227883
37071	88224441	24000	24000	24000.0	60 months	14.490000	564.559998	C	C4	Program Manager	...	40.000000	1	0	84664

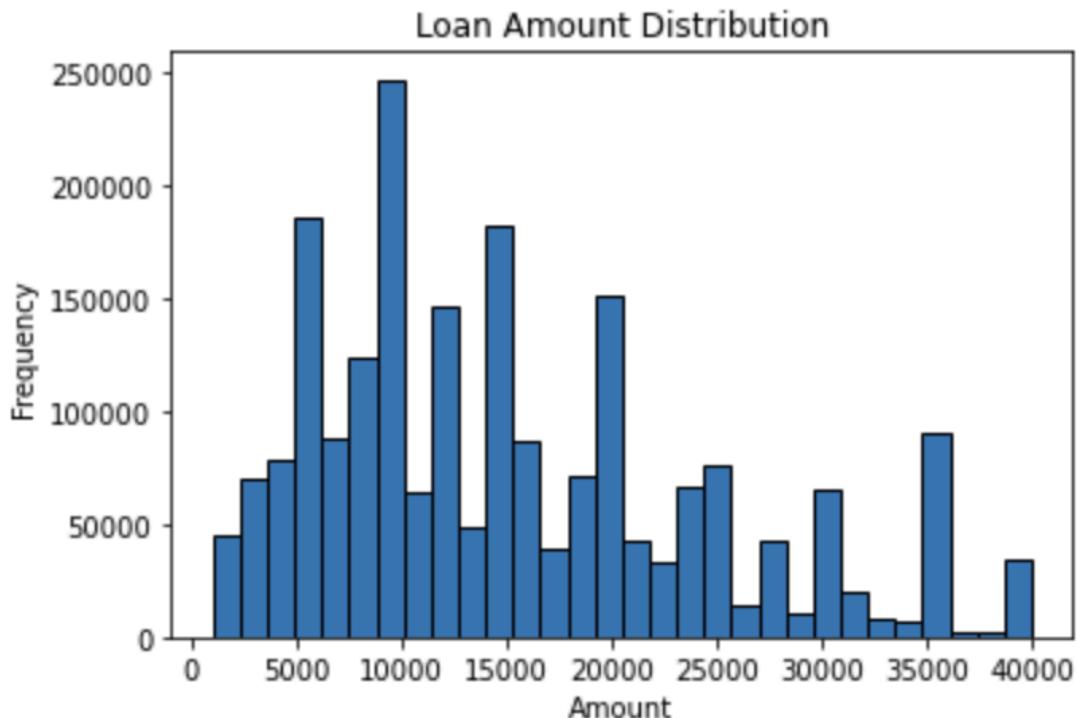
`df_accepted.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2137072 entries, 0 to 2137071
Data columns (total 96 columns):
 #   Column           Dtype  
 --- 
 0   id               int32  
 1   loan_amnt        int32  
 2   funded_amnt      int32  
 3   funded_amnt_inv float64 
 4   term             object  
 5   int_rate          float32 
 6   installment       float32 
 7   grade            object  
 8   sub_grade         object  
 9   emp_title         object  
 10  emp_length       object  
 11  home_ownership   object  
 12  annual_inc       float64 
 13  verification_status object  
 14  issue_d           object  
 15  loan_status       object  
 16  pymnt_plan        object  
 17  url              object  
 18  purpose           object  
 19  title             object  
 ...
 94  disbursement_method object  
 95  debt_settlement_flag object  
dtypes: float32(9), float64(10), int16(14), int32(14), int8(26), object(23)
memory usage: 835.6+ MB
```

```
plt.hist(df_accepted['loan_amnt'], bins=30, edgecolor='black')
plt.title("Loan Amount Distribution")
plt.xlabel("Amount")
plt.ylabel("Frequency")
plt.show()
```

dibuat visualisasi untuk melihat distribusi jumlah pinjaman dalam dataset **df_accepted**. Histogram digunakan untuk menampilkan frekuensi dari berbagai jumlah pinjaman yang diajukan, dengan membagi rentang nilai pinjaman menjadi 30 kelompok. Sumbu X merepresentasikan jumlah pinjaman, sementara sumbu Y menunjukkan frekuensi kemunculan setiap rentang nilai. Selain itu, histogram diberikan garis tepi hitam agar

lebih mudah dibaca, serta ditambahkan judul dan label sumbu agar lebih informatif.



Dari hasil visualisasi, terlihat bahwa jumlah pinjaman yang paling sering diajukan berada pada kisaran 5.000 hingga 15.000, dengan puncak tertinggi di sekitar 10.000. Frekuensi pengajuan pinjaman menurun seiring dengan meningkatnya jumlah pinjaman, terutama di atas 35.000. Pola ini menunjukkan bahwa mayoritas peminjam cenderung mengajukan pinjaman dalam jumlah sedang, bukan dalam jumlah yang terlalu kecil atau sangat besar.

```
# Print basic statistics
print("Statistical Summary:")
print(df_accepted['loan_amnt'].describe())

# Additional metrics
print("\nSkewness:", round(df_accepted['loan_amnt'].skew(), 2))      #
Measure of asymmetry
print("Kurtosis:", round(df_accepted['loan_amnt'].kurt(), 2))       #
Measure of tail heaviness
```

Kode ini digunakan untuk menghitung ringkasan statistik dasar dari variabel **loan_amnt** dalam dataset **df_accepted**. Fungsi **.describe()**

memberikan informasi seperti jumlah data (count), nilai rata-rata (mean), standar deviasi (std), nilai minimum dan maksimum, serta persentil (25%, 50%, 75%). Selain itu, kode ini juga menghitung nilai skewness dan kurtosis untuk mengevaluasi distribusi data.

Statistical Summary:

```
count      2.137072e+06
mean       1.518515e+04
std        9.212660e+03
min        1.000000e+03
25%        8.000000e+03
50%        1.300000e+04
75%        2.000000e+04
max        4.000000e+04
```

Name: loan_amnt, dtype: float64

Skewness: 0.77

Kurtosis: -0.14

Dari hasil yang ditampilkan, nilai skewness sebesar 0.77 menunjukkan bahwa distribusi jumlah pinjaman sedikit condong ke kanan, tetapi tidak terlalu ekstrem. Ini berarti ada beberapa nilai pinjaman yang lebih besar dari rata-rata, namun distribusi masih relatif simetris. Sedangkan nilai kurtosis sebesar -0.14 mengindikasikan bahwa distribusi memiliki puncak yang hampir normal, tanpa ekor yang terlalu berat.

Skewness and kurtosis are important tools for analyzing the shape of data distributions, giving us a clearer picture of how data behaves.

- **Skewness** tells us if the data leans to one side:
 - Positive skew means the tail is longer on the right.
 - Negative skew means the tail is longer on the left.
 - Skewness between -0.5 and 0.5 indicates a nearly symmetrical distribution.
- **Kurtosis** describes the “tails” and “peaks” of the distribution:
 - High kurtosis (leptokurtic) means heavy tails with more outliers.
 - Low kurtosis (platykurtic) means light tails with fewer outliers.
 - Near-zero excess kurtosis (mesokurtic) is close to a normal distribution.

Together, these measures help identify patterns, outliers, and whether the data is suitable for statistical models. Skewed data, for example, might require transformation to resemble a normal distribution for better model performance, especially in regression.

By using skewness and kurtosis, you can better understand your data and make informed decisions in your analysis.

Karena skewness dan kurtosis berada dalam rentang yang wajar dan tidak menunjukkan ketidakseimbangan yang ekstrem, maka tidak diperlukan teknik penyeimbangan data seperti transformasi log atau normalisasi lebih lanjut terhadap variabel **loan_amnt**.

Cek variable grade yang potensi imbalance

```
# For classification target (e.g., 'label')
class_counts = df_accepted['grade_encoded'].value_counts()
print("Class Distribution:")
print(class_counts)

# Normalized percentages
class_percent = df_accepted['grade_encoded'].value_counts(normalize=True) * 100
print("\nClass Percentages:")
print(class_percent.round(2))
```

Kode ini digunakan untuk mengevaluasi distribusi kelas pada variabel **grade_encoded** dalam dataset **df_accepted**, yang berpotensi mengalami ketidakseimbangan (imbalance). Pertama, kode menghitung jumlah masing-masing kelas menggunakan **.value_counts()**, kemudian menampilkan

distribusi kelas dalam bentuk angka absolut. Setelah itu, kode menghitung persentase setiap kelas dengan membagi jumlah masing-masing kelas dengan total keseluruhan, lalu mengalikannya dengan 100 untuk mendapatkan hasil dalam bentuk persen.

```
Class Distribution:  
grade  
B    627382  
C    619191  
A    409965  
D    305173  
E    126146  
F    38246  
G    10969  
Name: count, dtype: int64  
  
Class Percentages:  
grade  
B    29.36  
C    28.97  
A    19.18  
D    14.28  
E    5.90  
F    1.79  
G    0.51  
Name: proportion, dtype: float64
```

Dari hasil yang ditampilkan, terlihat bahwa distribusi kelas tidak merata. Kelas B dan C memiliki jumlah observasi terbesar dengan masing-masing sekitar 29.36% dan 28.97%, sedangkan kelas G memiliki jumlah yang sangat kecil, hanya 0.51%. Ketidakseimbangan ini dapat mempengaruhi performa model dalam tugas klasifikasi. Oleh karena itu, teknik penanganan ketidakseimbangan data seperti oversampling pada kelas minoritas atau undersampling pada kelas mayoritas mungkin diperlukan untuk memastikan bahwa model tidak bias terhadap kelas dengan jumlah data yang lebih besar.

```
majority_count = class_counts.max()  
minority_count = class_counts.min()  
imbalance_ratio = majority_count / minority_count
```

```
print(f"\nImbalance Ratio: {imbalance_ratio:.1f}:1")
```

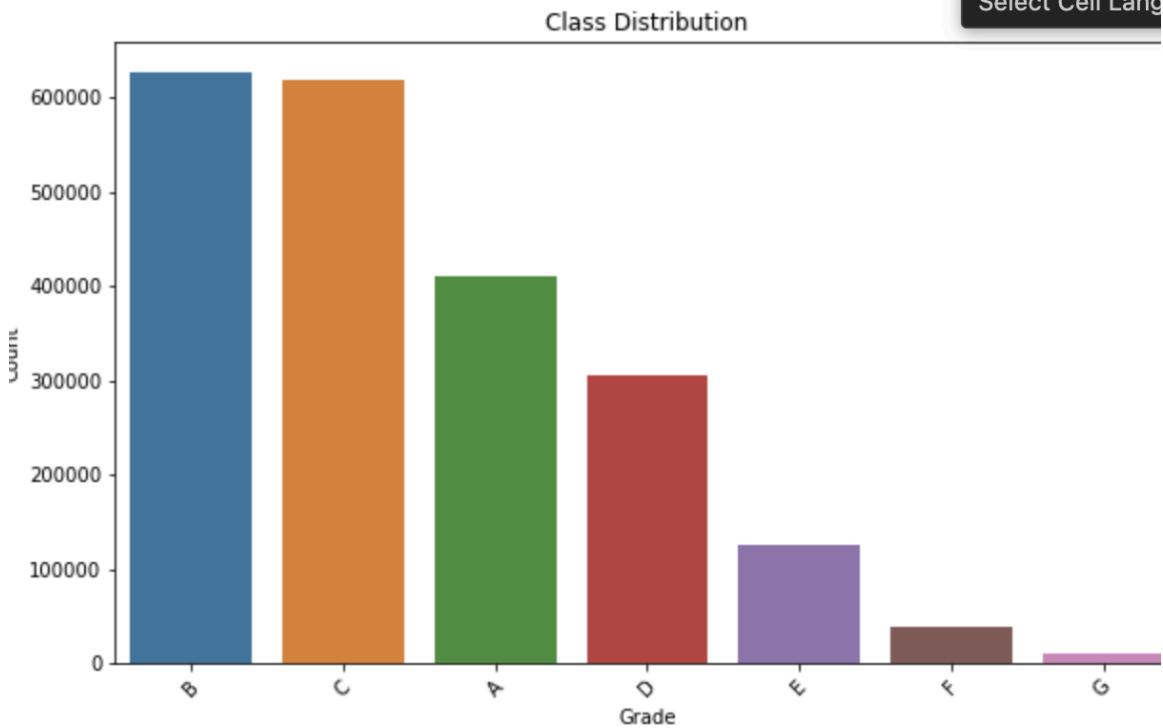
Kode ini digunakan untuk menghitung rasio ketidakseimbangan (imbalance ratio) pada variabel **grade_encoded**. Pertama, jumlah data pada kelas mayoritas dihitung dengan **class_counts.max()**, sedangkan jumlah data pada kelas minoritas dihitung dengan **class_counts.min()**. Kemudian, rasio ketidakseimbangan dihitung dengan membagi jumlah kelas mayoritas dengan kelas minoritas.

Imbalance Ratio: 57.2:1

Hasil yang diperoleh menunjukkan bahwa rasio ketidakseimbangan adalah **57.2:1**, yang berarti jumlah data di kelas mayoritas 57.2 kali lebih banyak dibandingkan dengan kelas minoritas. Dengan rasio sebesar ini, **dataset sangat tidak seimbang**.

```
# Plot class frequencies
plt.figure(figsize=(10, 6))
sns.countplot(x='grade_encoded', data=small_df,
               order=small_df['grade_encoded'].value_counts().index)
plt.title('Class Distribution')
plt.xlabel('Grade')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

Kode ini digunakan untuk mengevaluasi distribusi kelas pada variabel **grade_encoded** dalam dataset **df_accepted**, yang berpotensi mengalami ketidakseimbangan (imbalance). Kode ini menggunakan **sns.countplot()** untuk memvisualisasikan jumlah observasi di setiap kategori **grade_encoded**, diurutkan berdasarkan jumlah kemunculan dari yang paling banyak hingga yang paling sedikit.



Dari hasil visualisasi, terlihat bahwa terdapat ketidakseimbangan kelas (**class imbalance**) dalam variabel target grade. Beberapa kelas, seperti B dan C, memiliki jumlah sampel yang jauh lebih banyak dibandingkan kelas lainnya, sementara kelas seperti F dan G memiliki jumlah sampel yang sangat sedikit. Ketidakseimbangan ini dapat menyebabkan model cenderung bias terhadap kelas mayoritas, sehingga kurang akurat dalam memprediksi kelas dengan jumlah sampel lebih kecil.

Untuk menangani masalah ini dan memastikan setiap kelas dianggap sama pentingnya, **dapat digunakan metode class weights**.

mengapa pakai class weighting?

Class weighting memungkinkan permodelan yang akan memperhatikan kelas kelas minoritas tanpa harus mengubah data. hal ini sangat efisien, utamanya jika mengolah data yang sangat besar dikarenakan minimnya ubah data asal asli

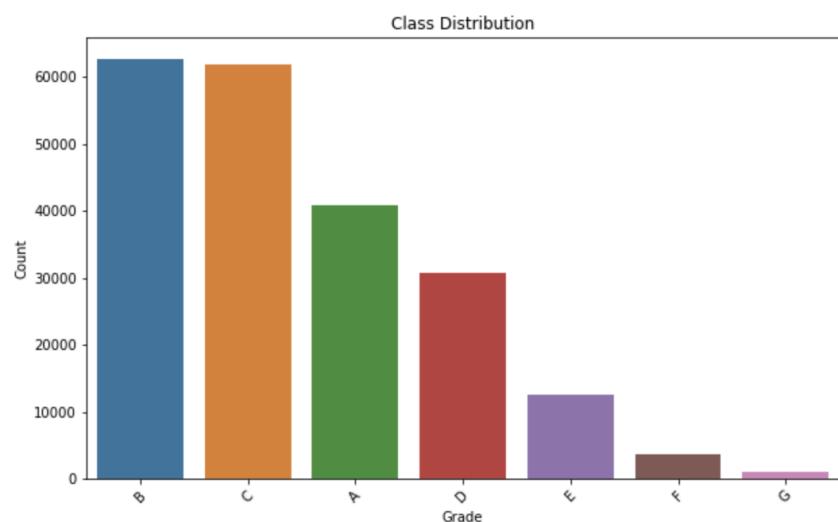
```
small_df = df_accepted.sample(frac=0.1, random_state=42) # random_state  
for reproducibility
```

```
# Proceed with small_df for your experiments  
X = small_df.drop('grade_encoded', axis=1)  
y = small_df['grade_encoded']
```

Kode ini bertujuan untuk **mengambil subset (sampel) dari dataset df_accepted** agar dapat digunakan dalam eksperimen atau pelatihan model dengan data yang lebih kecil.

```
# Plot class frequencies  
plt.figure(figsize=(10, 6))  
sns.countplot(x='grade_encoded',  
               data=small_df,  
               order=small_df['grade_encoded'].value_counts().index)  
plt.title('Class Distribution')  
plt.xlabel('Grade')  
plt.ylabel('Count')  
plt.xticks(rotation=45)  
plt.show()
```

Kode ini bertujuan untuk mengevaluasi distribusi kelas pada variabel grade_encoded dalam subset data **small_df**, yang diambil sebagai sampel 10% dari dataset utama **df_accepted**. Dengan menggunakan **sns.countplot()**, kode ini membuat histogram yang menunjukkan jumlah observasi dalam setiap kategori **grade_encoded**. Urutan tampilan kelas ditentukan berdasarkan jumlah kemunculannya dalam data.



Dari output yang ditampilkan, terlihat ternyata distribusi kelas

tetap tidak seimbang, dengan kelas B dan C memiliki jumlah observasi terbanyak, sementara kelas F dan G memiliki jumlah yang jauh lebih sedikit.

```
# For classification target (e.g., 'label')
class_counts = small_df['grade_encoded'].value_counts()
print("Class Distribution:")
print(class_counts)

# Normalized percentages
class_percent = small_df['grade_encoded'].value_counts(normalize=True) *
100
print("\nClass Percentages:")
print(class_percent.round(2))
```

Kode ini bertujuan untuk mengevaluasi distribusi kelas pada variabel target grade_encoded dalam subset data small_df. Pertama, kode menghitung jumlah observasi untuk setiap kelas menggunakan value_counts() dan mencetak hasilnya. Selanjutnya, kode menghitung persentase masing-masing kelas dengan mengaktifkan parameter normalize=True, lalu mengalikannya dengan 100 untuk mendapatkan hasil dalam bentuk persentase.

```
Class Distribution:
grade
B    62749
C    61943
A    40824
D    30806
E    12552
F     3769
G     1064
Name: count, dtype: int64

Class Percentages:
grade
B    29.36
C    28.99
A    19.10
D    14.42
E     5.87
F     1.76
G     0.50
Name: proportion, dtype: float64
```

Dari output yang ditampilkan, terlihat bahwa distribusi kelas tidak merata. Kelas B dan C memiliki jumlah observasi tertinggi, masing-masing sekitar 29.36% dan 28.99%, sedangkan kelas G memiliki jumlah yang sangat kecil, hanya 0.50%. Ketidakseimbangan ini dapat berdampak pada performa model klasifikasi, karena model cenderung lebih akurat dalam memprediksi kelas mayoritas dan kesulitan dalam mengenali kelas minoritas.

```
majority_count = class_counts.max()
minority_count = class_counts.min()
imbalance_ratio = majority_count / minority_count
print(f"\nImbalance Ratio: {imbalance_ratio:.1f}:1")
```

Kode ini bertujuan untuk menghitung rasio ketidakseimbangan (*imbalance ratio*) antara kelas mayoritas dan kelas minoritas dalam variabel target `grade_encoded`. Pertama, kode mencari jumlah sampel terbesar dalam suatu kelas dengan `class_counts.max()`, yang mewakili kelas mayoritas. Kemudian, kode mencari jumlah sampel terkecil dalam suatu kelas menggunakan `class_counts.min()`, yang mewakili kelas minoritas. Selanjutnya, rasio ketidakseimbangan dihitung dengan membagi jumlah sampel kelas mayoritas dengan jumlah sampel kelas minoritas. Hasilnya dicetak dalam format yang menunjukkan perbandingan terhadap 1.

Imbalance Ratio: 59.0:1

```
from sklearn.utils.class_weight import compute_class_weight
import numpy as np

# Get class weights
classes = np.unique(small_df['grade_encoded'])      # y is your target
variable (e.g., 'grade_encoded')
y = small_df['grade_encoded']
weights = compute_class_weight('balanced', classes=classes, y=y)
```

```
# Convert to dictionary format
class_weights = dict(zip(classes, weights))
print("Class Weights:", class_weights)
```

Kode ini bertujuan untuk menghitung bobot kelas (*class weights*) guna menangani ketidakseimbangan kelas pada variabel target `grade_encoded`. Ketika data memiliki distribusi kelas yang tidak merata, model pembelajaran mesin cenderung lebih memprioritaskan kelas mayoritas. Untuk mengatasi hal ini, bobot kelas dapat dihitung menggunakan fungsi `compute_class_weight` dari `sklearn.utils.class_weight`, yang secara otomatis memberikan bobot lebih tinggi untuk kelas dengan jumlah sampel lebih sedikit.

```
Class Weights: {'A': 0.7478339072254416, 'B': 0.48653478826071217,
'C': 0.4928655607344079, 'D': 0.9910267944092523, 'E':
2.4322475644177364, 'F': 8.100178145017624, 'G': 28.693206229860365}
```

Import Library

```
from sklearn.model_selection import train_test_split
from lightgbm import LGBMClassifier
import numpy as np

from lightgbm import LGBMClassifier
```

Menyiapkan Data dan Membagi Train/Test

```
X = small_df[['loan_amnt', 'annual_inc']]
y = small_df['grade_encoded']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
```

```
y
```

```
2016079      B
1470901      C
747210       C
85451        C
1510742      B
...
2095591      C
1573693      C
1727315      A
725240       B
1978732      D
Name: grade, Length: 213707, dtype: object
```

```
from sklearn.utils.class_weight import compute_sample_weight
```

```
sample_weights = compute_sample_weight(class_weights, y)
```

Metode class weighting menggunakan **Random Forest**

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Baseline model (NO weighting)
model_baseline = RandomForestClassifier(random_state=42)
model_baseline.fit(X, y) # No weights

# Weighted model (WITH your class weights)
model_weighted = RandomForestClassifier(random_state=42)
model_weighted.fit(X, y, sample_weight=sample_weights) # Your weights
```

Kode ini bertujuan untuk menerapkan metode **class weighting** menggunakan algoritma **Random Forest** guna menangani ketidakseimbangan kelas dalam dataset. Pertama, kode mengimpor RandomForestClassifier dari sklearn.ensemble sebagai model pembelajaran mesin dan classification_report serta confusion_matrix dari sklearn.metrics untuk evaluasi model.

Selanjutnya, dua model Random Forest dibuat:

1. **Baseline model (tanpa class weighting)** – Model pertama (model_baseline) dibuat tanpa mempertimbangkan bobot kelas, sehingga setiap sampel dianggap memiliki kepentingan yang sama.
2. **Weighted model (dengan class weighting)** – Model kedua (model_weighted) dilatih menggunakan **sample weights**, yaitu bobot yang disesuaikan berdasarkan distribusi kelas untuk menangani ketidakseimbangan data.



RandomForestClassifier

RandomForestClassifier(random_state=42)

Output yang ditampilkan berupa dua baris yang menunjukkan bahwa model RandomForestClassifier telah berhasil dibuat dengan parameter random_state=42. Ini memastikan reproducibilitas hasil dengan mengontrol seed acak. Perbedaan utama antara kedua model ini akan terlihat pada hasil evaluasi, di mana model berbobot (weighted) seharusnya lebih baik dalam menangani kelas minoritas dibandingkan model tanpa bobot.

```
# Predictions
y_pred_baseline = model_baseline.predict(X)
y_pred_weighted = model_weighted.predict(X)

# Print metrics
print("== Baseline Model (No Weighting) ==")
print(classification_report(y, y_pred_baseline))

print("\n== Weighted Model ==")
print(classification_report(y, y_pred_weighted))
```

Kode ini bertujuan untuk membandingkan performa model **Random Forest** tanpa class weighting dan dengan class weighting menggunakan metrik evaluasi **classification report**. Pertama, model baseline (model_baseline) dan model berbobot (model_weighted) digunakan untuk membuat prediksi terhadap dataset X. Selanjutnya, hasil prediksi dibandingkan dengan nilai asli y menggunakan classification_report, yang menampilkan metrik **precision**, **recall**, **f1-score**, dan **support** untuk setiap kelas.

== Baseline Model (No Weighting) ==				
	precision	recall	f1-score	support
A	0.57	0.38	0.46	40824
B	0.48	0.67	0.56	62749
C	0.50	0.59	0.54	61943
D	0.73	0.38	0.50	30806
E	0.77	0.40	0.53	12552
F	0.84	0.40	0.54	3769
G	0.87	0.42	0.57	1064
accuracy			0.53	213707
macro avg	0.68	0.46	0.53	213707
weighted avg	0.56	0.53	0.52	213707
== Weighted Model ==				
	precision	recall	f1-score	support
A	0.42	0.57	0.49	40824
B	0.59	0.43	0.50	62749
C	0.66	0.38	0.48	61943
D	0.55	0.44	0.49	30806
E	0.37	0.57	0.45	12552
F	0.16	0.75	0.26	3769
...				
accuracy			0.46	213707
macro avg	0.40	0.58	0.40	213707
weighted avg	0.55	0.46	0.48	213707

Dari output yang ditampilkan, dapat dilihat bahwa:

- Baseline Model (No Weighting): Model ini memiliki precision lebih tinggi, tetapi recall lebih rendah, terutama untuk kelas yang lebih jarang muncul (misalnya kelas G). Ini menunjukkan

bahwa model lebih sering salah mengklasifikasikan kelas minoritas.

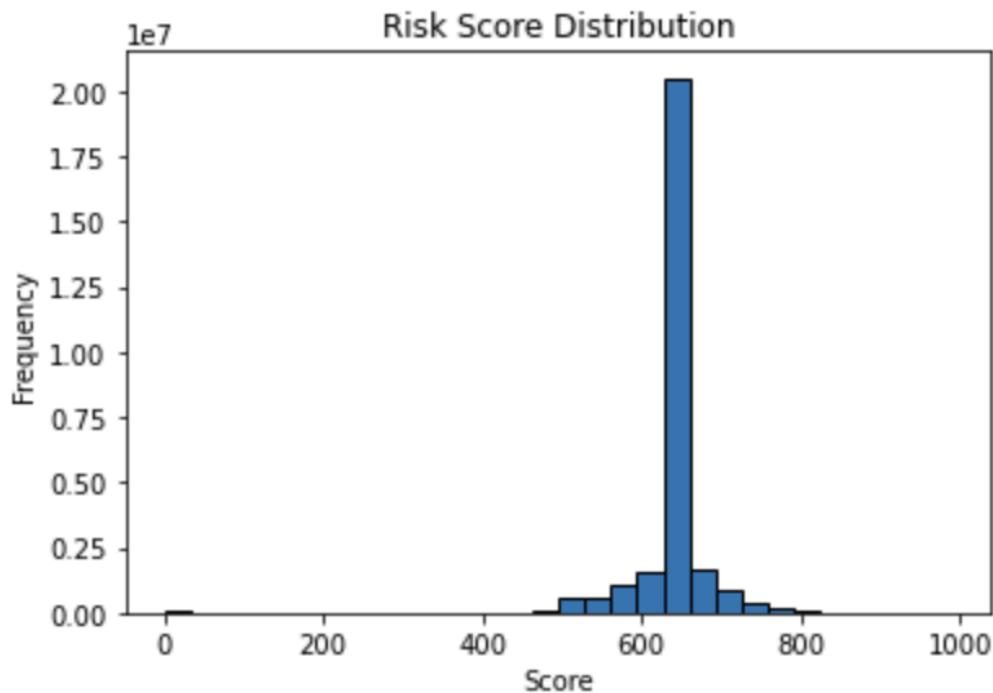
- Weighted Model: Model ini memiliki recall yang lebih tinggi, terutama pada kelas dengan jumlah sampel lebih sedikit (misalnya kelas F dan G), yang menunjukkan bahwa metode class weighting membantu model mengenali kelas yang jarang muncul. Namun, ini terjadi dengan trade-off pada precision, yang berarti model cenderung lebih banyak salah positif.

Rejected_2007_to_2018Q4.csv

df_rejected.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27646225 entries, 0 to 27646224
Data columns (total 9 columns):
 #   Column           Dtype  
 --- 
 0   Amount Requested    float64
 1   Application Date   object  
 2   Loan Title          object  
 3   Risk_Score          int16  
 4   Debt-To-Income Ratio object  
 5   Zip Code            object  
 6   State               object  
 7   Employment Length   object  
 8   Policy Code         int8  
dtypes: float64(1), int16(1), int8(1), object(6)
memory usage: 1.5+ GB
```

```
plt.hist(df_rejected['Risk_Score'], bins=30, edgecolor='black')
plt.title("Risk Score Distribution")
plt.xlabel("Score")
plt.ylabel("Frequency")
plt.show()
```



Output yang ditampilkan berupa histogram distribusi `Risk_Score`, yang menunjukkan bahwa sebagian besar nilai skor risiko terkonsentrasi di sekitar 600 dengan distribusi yang sangat tidak merata. Hal ini mengindikasikan adanya ketidakseimbangan kelas yang signifikan dalam dataset, yang dapat memengaruhi kinerja model prediktif.

```
# Take 10% of your data (adjust fraction as needed)
small_df_rejected = df_rejected.sample(frac=0.1, random_state=42) # random_state for reproducibility

# small_df_rejected.drop('Risk_Score_processed', axis=1)
# small_df_rejected.drop('Risk_Score_scaled', axis=1)
# small_df_rejected.drop('Risk_Scaled', axis=1)

# Proceed with small_df for your experiments
X = small_df_rejected.drop('risk_score', axis=1)
y = small_df_rejected['risk_score']
```

Kode ini bertujuan untuk mengambil sampel acak sebesar 10% dari dataset `df_rejected` menggunakan `sample()` dengan parameter `random_state=42` untuk memastikan reproduksibilitas.

```
# Print basic statistics
print("Statistical Summary:")
print(df_rejected['risk_score'].describe())
```

```
# Additional metrics
print("\nSkewness:", round(small_df_rejected['risk_score'].skew(), 2))
# Measure of asymmetry
print("Kurtosis:", round(small_df_rejected['risk_score'].kurt(), 2))    #
Measure of tail heaviness
```

Kode ini bertujuan untuk menganalisis distribusi variabel `risk_score` dengan menampilkan ringkasan statistik dasar, seperti jumlah data, rata-rata, standar deviasi, nilai minimum dan maksimum, serta persentil. Selain itu, kode juga menghitung nilai skewness untuk mengetahui tingkat keasimetrian distribusi dan kurtosis untuk mengukur seberapa berat ekor distribusi dibandingkan dengan distribusi normal.

Statistical Summary:

```
count      2.764622e+07
mean      6.340770e+02
std       5.190879e+01
min       0.000000e+00
25%       6.370000e+02
50%       6.370000e+02
75%       6.370000e+02
max       9.900000e+02
```

Name: Risk_Score, dtype: float64

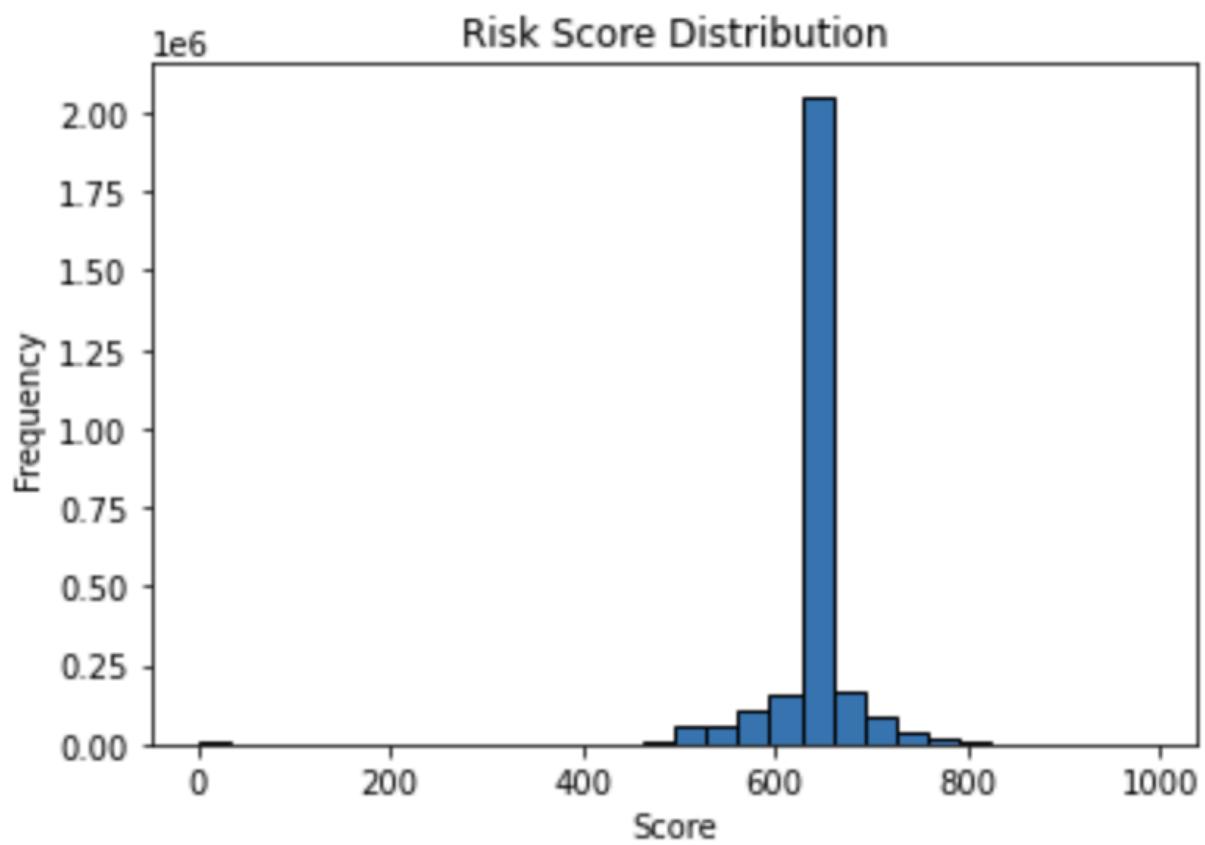
Skewness: -5.66

Kurtosis: 69.96

Hasil output menunjukkan bahwa distribusi `risk_score` memiliki skewness sebesar -5.66, yang berarti distribusi sangat condong ke kiri. Selain itu, nilai kurtosis yang sangat tinggi, yaitu 69.96, menunjukkan bahwa distribusi memiliki ekor yang jauh lebih berat dibandingkan distribusi normal. Kondisi ini mengindikasikan adanya

ketidakseimbangan data yang signifikan, sehingga diperlukan teknik penyeimbangan lanjutan

```
plt.hist(small_df_rejected['Risk_Score'], bins=30, edgecolor='black')
plt.title("Risk Score Distribution")
plt.xlabel("Score")
plt.ylabel("Frequency")
plt.show()
```

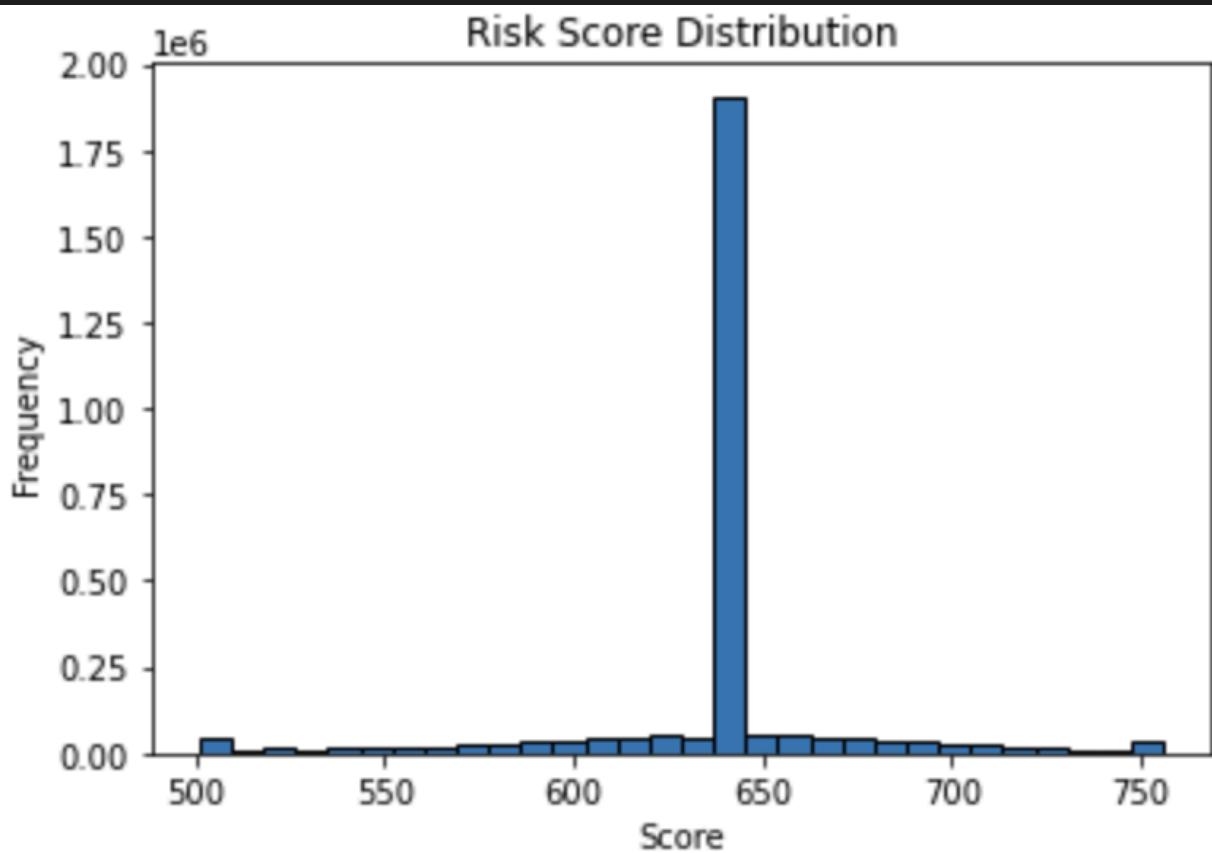


Menggunakan winsorize

```
from scipy.stats.mstats import winsorize
# Clip top/bottom 1%
```

```
small_df_rejected[ 'Risk_Score_processed' ] =  
winsorize(small_df_rejected[ 'Risk_Score' ], limits=[0.01, 0.01])
```

```
plt.hist(small_df_rejected[ 'Risk_Score_processed' ], bins=30,  
edgecolor='black')  
plt.title("Risk Score Distribution")  
plt.xlabel("Score")  
plt.ylabel("Frequency")  
plt.show()
```



Histogram hasil setelah winsorization menunjukkan bahwa distribusi Risk_Score_processed masih memiliki konsentrasi data yang sangat tinggi pada nilai tertentu, kemungkinan besar karena nilai yang dipangkas digantikan dengan nilai batas yang lebih rendah atau lebih tinggi dalam distribusi.

```
# Print basic statistics  
  
print("Statistical Summary:")  
print(small_df_rejected[ 'Risk_Score_processed' ].describe())  
  
# Additional metrics  
  
print("\nSkewness:",  
round(small_df_rejected[ 'Risk_Score_processed' ].skew(), 2))      #  
Measure of asymmetry  
  
print("Kurtosis:",  
round(small_df_rejected[ 'Risk_Score_processed' ].kurt(), 2))      #  
Measure of tail heaviness
```

Statistical Summary:

count	2.764622e+06
mean	6.353511e+02
std	3.636900e+01
min	5.010000e+02
25%	6.370000e+02
50%	6.370000e+02
75%	6.370000e+02
max	7.560000e+02

Name: Risk_Score_processed, dtype: float64

Skewness: -0.61

Skewness: -0.61

Kurtosis: 4.84

Histogram setelah proses winsorization menunjukkan bahwa distribusi **Risk_Score_processed** masih memiliki konsentrasi data yang sangat tinggi di satu nilai tertentu, yang kemungkinan besar disebabkan oleh proses winsorization yang menggantikan nilai ekstrem dengan batas tertentu. Hal ini menyebabkan sebagian besar data terkonsentrasi pada satu titik.

Namun, setelah dilakukan evaluasi statistik deskriptif, terlihat bahwa skewness yang awalnya jauh dari nol telah membaik menjadi -0.61, yang menunjukkan distribusi lebih seimbang. Selain itu, kurtosis juga turun dari angka yang sangat tinggi menjadi 4.84, yang menandakan distribusi lebih normal dan kurang dipengaruhi oleh nilai ekstrem. Perubahan ini cukup baik karena mengurangi dampak outlier, membuat distribusi data lebih stabil untuk analisis lebih lanjut.

```
from sklearn.preprocessing import RobustScaler
# A. Standard Scaling (if using linear models/NNs)
from sklearn.preprocessing import StandardScaler

small_df_rejected['Risk_Scaled'] = StandardScaler().fit_transform(small_df_rejected[['Risk_Score_processed']])

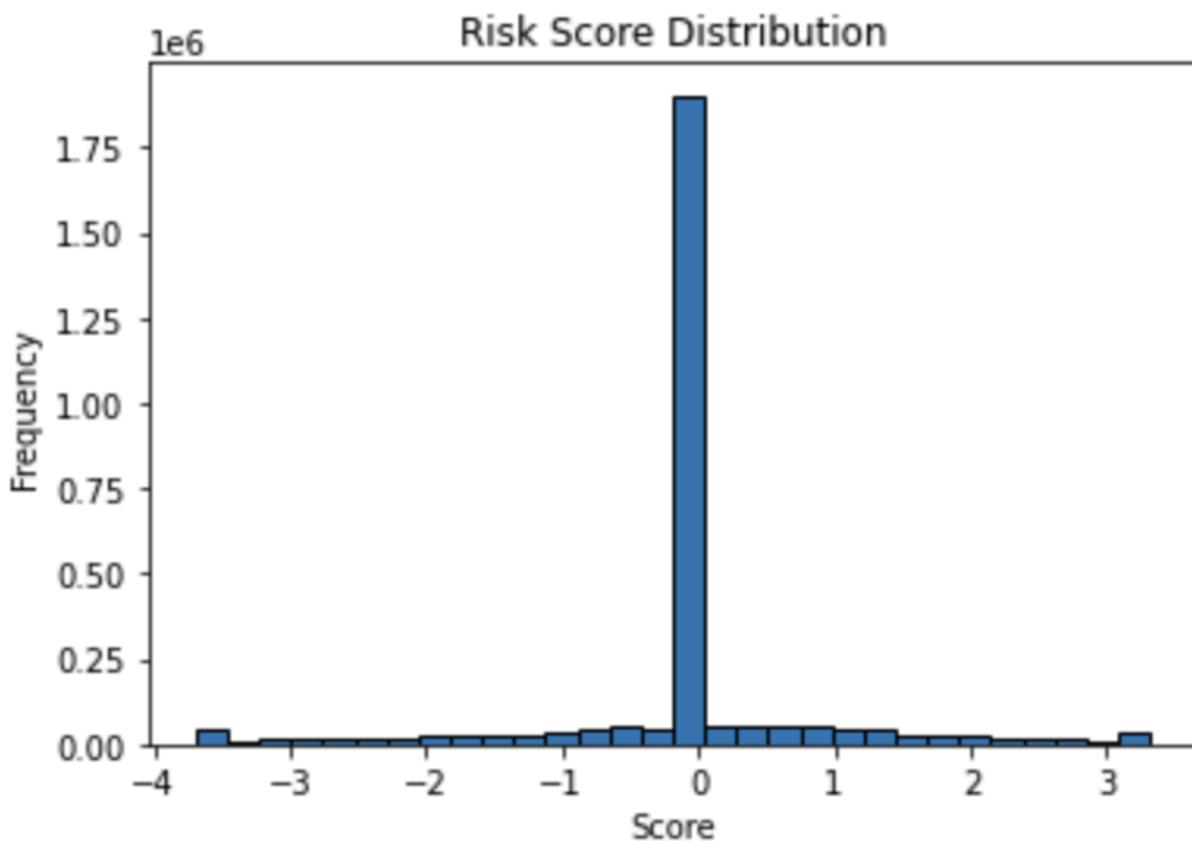
```

Kode ini bertujuan untuk melakukan **scaling** (penyesuaian skala) terhadap variabel Risk_Score_processed dalam dataset small_df_rejected. Scaling digunakan untuk memastikan bahwa variabel memiliki distribusi dengan skala yang lebih sebanding, sehingga lebih optimal untuk digunakan dalam model pembelajaran mesin, terutama model yang berbasis regresi atau jaringan saraf.

```
plt.hist(small_df_rejected['Risk_Scaled'], bins=30,
edgecolor='black')
plt.title("Risk Score Distribution")
plt.xlabel("Score")
plt.ylabel("Frequency")
```

```
plt.show()
```

Kode ini bertujuan untuk memvisualisasikan distribusi variabel **Risk_Scaled**, yang merupakan hasil transformasi dari **Risk_Score_processed** menggunakan **StandardScaler**. Histogram dibuat menggunakan fungsi `plt.hist()`, dengan 30 bins dan tepi berwarna hitam untuk memperjelas batas antar batang. Judul grafik diberikan melalui `plt.title()`, sementara sumbu X dan Y diberi label "Score" dan "Frequency" untuk memperjelas makna visualisasi. Terakhir, `plt.show()` digunakan untuk menampilkan histogram.



Hasil output menunjukkan bahwa mayoritas data terkonsentrasi di sekitar nilai **0**, yang sesuai dengan sifat **StandardScaler** yang menstandarisasi data ke dalam distribusi dengan **mean = 0** dan **standar deviasi = 1**. Namun, distribusi tetap menunjukkan adanya skewness atau pencilan, yang terlihat dari beberapa nilai yang tersebar di sekitar rentang -4 hingga 3. Hal ini mengindikasikan bahwa meskipun telah dinormalisasi, data awal kemungkinan memiliki distribusi yang sangat tidak merata.

```
# Print basic statistics
print("Statistical Summary:")
print(small_df_rejected['Risk_Scaled'].describe())

# Additional metrics
print("\nSkewness:", round(small_df_rejected['Risk_Scaled'].skew(), 2))
# Measure of asymmetry
print("Kurtosis:", round(small_df_rejected['Risk_Scaled'].kurt(), 2))
# Measure of tail heaviness
```

Statistical Summary:

count	2.764622e+06
mean	1.047402e-15
std	1.000000e+00
min	-3.694112e+00
25%	4.533707e-02
50%	4.533707e-02
75%	4.533707e-02
max	3.317355e+00

Name: Risk_Scaled, **dtype:** float64

Skewness: -0.61

Kurtosis: 4.84

Output menunjukkan ringkasan statistik dari variabel Risk_Scaled setelah standardisasi, dengan 2.76 juta data, rata-rata mendekati 0, dan standar deviasi 1, sesuai dengan transformasi StandardScaler. Nilai minimum dan maksimum berkisar antara -3.69 hingga 3.32, sementara kuartil menunjukkan banyaknya data terkonsentrasi di

sekitar 0.045. Skewness sebesar -0.61 menunjukkan distribusi sedikit condong ke kiri, dan Kurtosis sebesar 4.84 mengindikasikan distribusi memiliki puncak tajam serta ekor yang lebih berat, yang bisa menandakan adanya pencilan.

```
from scipy.stats.mstats import winsorize  
small_df_rejected['Risk_Winsorized'] =  
winsorize(small_df_rejected['Risk_Scaled'], limits=[0.01, 0.01])
```

```
# Print basic statistics  
  
print("Statistical Summary:")  
print(small_df_rejected[ 'Risk_Winsorized' ].describe())  
  
# Additional metrics  
  
print("\nSkewness:",  
round(small_df_rejected[ 'Risk_Winsorized' ].skew(), 2)) # Measure of  
asymmetry  
print("Kurtosis:", round(small_df_rejected[ 'Risk_Winsorized' ].kurt(),  
2)) # Measure of tail heaviness
```

Statistical Summary:

count	2.764622e+06
mean	1.047402e-15
std	1.000000e+00
min	-3.694112e+00
25%	4.533707e-02
50%	4.533707e-02
75%	4.533707e-02
max	3.317355e+00

Name: Risk_Winsorized, dtype: float64

```
Skewness: -0.61  
Kurtosis: 4.84
```

Proses Winsorization dilakukan dengan memangkas nilai ekstrem pada batas 1% terendah dan tertinggi untuk variabel Risk_Scaled, menghasilkan variabel baru Risk_Winsorized. Namun, hasil statistik menunjukkan bahwa skewness (-0.61) dan kurtosis (4.84) tetap sama, menandakan distribusi data tidak mengalami perubahan signifikan dalam hal kemiringan dan ekor distribusi. Selain itu, mean (≈ 0) dan standar deviasi (≈ 1) tetap seimbang, menunjukkan bahwa data sudah cukup terdistribusi dengan baik setelah standardisasi awal. **Oleh karena itu, Winsorization saja sudah cukup untuk menangani nilai ekstrem tanpa perlu metode tambahan.**

#Dataset Home Credit

```
application_train
```

Hasil analisis dari kode dibawah :

Berdasarkan hasil analisis pada dataset Application Train, distribusi kelas dalam variabel target yaitu AMT_CREDIT menunjukkan keseimbangan yang baik. Hal ini ditunjukkan oleh **nilai skewness dan kurtosis yang berada dalam rentang yang wajar**, menandakan bahwa distribusi data tidak terlalu condong ke satu sisi (asimetri rendah) dan tidak memiliki ekor yang terlalu berat. Dengan demikian, tidak diperlukan teknik penyeimbangan data seperti SMOTE, class weighting, atau undersampling/oversampling. Kondisi ini mengindikasikan bahwa **dataset sudah siap digunakan untuk proses prediksi** tanpa perlu penyesuaian lebih lanjut terhadap distribusi kelas.

```
df_application_train = downcasting(df_application_train)
```

```
df_application_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 86 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   SK_ID_CURR       307511 non-null   int32  
 1   TARGET           307511 non-null   int8   
 2   NAME_CONTRACT_TYPE 307511 non-null   object  
 3   CODE_GENDER      307511 non-null   object  
 4   FLAG_OWN_CAR     307511 non-null   object  
 5   FLAG_OWN_REALTY  307511 non-null   object  
 6   CNT_CHILDREN     307511 non-null   int8   
 7   AMT_INCOME_TOTAL 307511 non-null   float64 
 8   AMT_CREDIT        307511 non-null   float32 
 9   AMT_ANNUITY       307511 non-null   float32 
 10  AMT_GOODS_PRICE   307511 non-null   float32 
 11  NAME_TYPE_SUITE   307511 non-null   object  
 12  NAME_INCOME_TYPE  307511 non-null   object  
 13  NAME_EDUCATION_TYPE 307511 non-null   object  
 14  NAME_FAMILY_STATUS 307511 non-null   object  
 15  NAME_HOUSING_TYPE 307511 non-null   object  
 16  REGION_POPULATION_RELATIVE 307511 non-null   float32 
 17  DAYS_BIRTH        307511 non-null   int16  
 18  DAYS_EMPLOYED     307511 non-null   int32  
 19  DAYS_REGISTRATION 307511 non-null   float32 
 ...
 84  FLAG_DOCUMENT_21   307511 non-null   int8   
 85  AMT_REQ_CREDIT_BUREAU_QRT  307511 non-null   int16  
dtypes: float32(23), float64(1), int16(5), int32(2), int8(39), object(16)
memory usage: 83.6+ MB
```

```
df_application_train[ 'AMT_CREDIT' ]
```

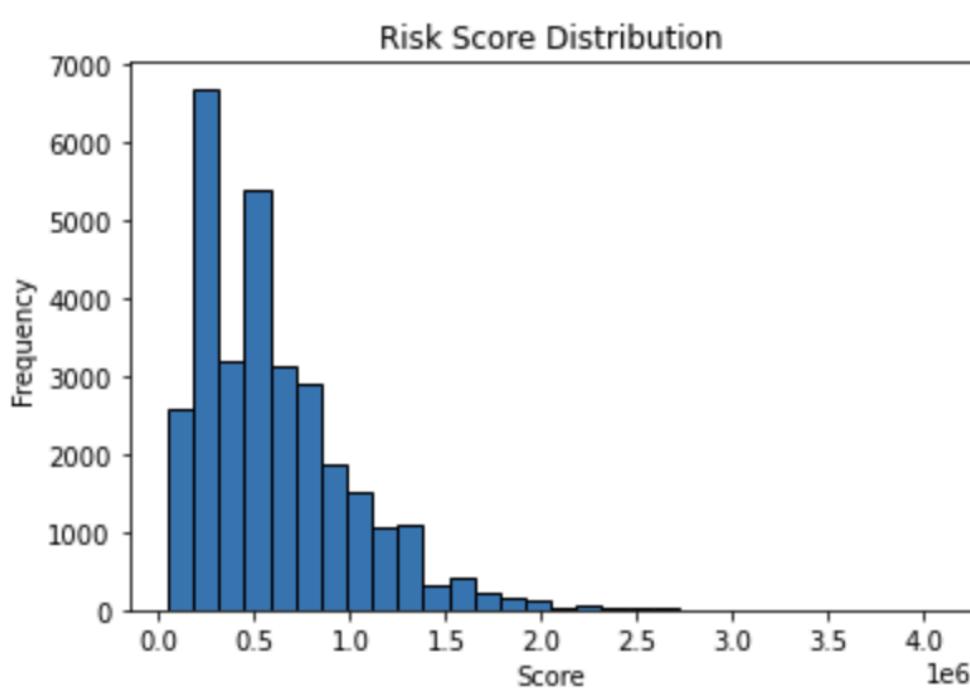
```
0      406597.5
1      1293502.5
2      135000.0
3      312682.5
4      513000.0
      ...
307506    254700.0
307507    269550.0
307508    677664.0
307509    370107.0
307510    675000.0
Name: AMT_CREDIT, Length: 307511, dtype: float32
```

```
# Take 10% of your data (adjust fraction as needed)
small_df_application_train = df_application_train.sample(frac=0.1, random_state=42)
# random_state for reproducibility

# small_df_rejected.drop('Risk_Score_processed', axis=1)
# small_df_rejected.drop('Risk_Score_scaled', axis=1)
# small_df_rejected.drop('Risk_Scaled', axis=1)

# Proceed with small_df for your experiments
X = df_application_train.drop('AMT_CREDIT', axis=1)
y = df_application_train['AMT_CREDIT']
```

```
plt.hist(small_df_application_train['AMT_CREDIT'], bins=30, edgecolor='black')
plt.title("Risk Score Distribution")
plt.xlabel("Score")
plt.ylabel("Frequency")
plt.show()
```



```
# Print basic statistics
print("Statistical Summary:")
print(small_df_application_train['AMT_CREDIT'].describe())

# Additional metrics
print("\nSkewness:", round(small_df_application_train['AMT_CREDIT'].skew(), 2)) # Measure of asymmetry
print("Kurtosis:", round(small_df_application_train['AMT_CREDIT'].kurt(), 2)) # Measure of tail heaviness
```

```
Statistical Summary:  
count      3.075100e+04  
mean       5.979197e+05  
std        4.023701e+05  
min        4.500000e+04  
25%        2.700000e+05  
50%        5.108535e+05  
75%        8.086500e+05  
max        4.050000e+06  
Name: AMT_CREDIT, dtype: float64
```

Skewness: 1.29
Kurtosis: 2.34

```
# Print basic statistics  
print("Statistical Summary:")  
print(small_df_application_train['AMT_CREDIT'].describe())  
  
# Additional metrics  
print("\nSkewness:", round(small_df_application_train['AMT_CREDIT'].skew(), 2))  #  
Measure of asymmetry  
print("Kurtosis:", round(small_df_application_train['AMT_CREDIT'].kurt(), 2))    #  
Measure of tail heaviness
```

```
Statistical Summary:  
count      3.075100e+04  
mean       5.979197e+05  
std        4.023701e+05  
min        4.500000e+04  
25%        2.700000e+05  
50%        5.108535e+05  
75%        8.086500e+05  
max        4.050000e+06  
Name: AMT_CREDIT, dtype: float64
```

Skewness: 1.29

Kurtosis: 2.34

Berdasarkan hasil analisis pada dataset Application Train, distribusi kelas dalam variabel target yaitu AMT_CREDIT menunjukkan keseimbangan yang baik. Hal ini ditunjukkan oleh **nilai skewness dan kurtosis yang berada dalam rentang yang wajar**, menandakan bahwa distribusi data tidak terlalu condong ke satu sisi (asimetri rendah) dan tidak memiliki ekor yang terlalu berat. Dengan demikian, tidak diperlukan teknik penyeimbangan data seperti SMOTE, class weighting, atau undersampling/oversampling. Kondisi ini mengindikasikan bahwa **dataset sudah siap digunakan untuk proses prediksi** tanpa perlu penyesuaian lebih lanjut terhadap distribusi kelas.

installments_payments

```
df_installments_payments = downcasting(df_installments_payments)
```

```
df_installments_payments
```

SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	DAYS_INSTALMENT	DAYS_ENTRY_PAYMENT	AMT_INSTALMENT	AMT_PAYMENT
1054186	161674	1	6	-1180	-1187	6948.360	6948.36
1330831	151639	0	34	-2156	-2156	1716.525	1716.52
2085231	193053	2	1	-63	-63	25425.000	25425.00
2452527	199697	1	3	-2418	-2426	24350.130	24350.13
2714724	167756	1	2	-1383	-1366	2165.040	2160.51
...
2006721	442291	1	3	-1311	-1318	2934.225	2934.22
1126000	428449	0	12	-301	-302	6793.470	6750.00
1519070	444122	1	5	-399	-407	4363.830	4363.83
2784672	444977	0	4	-157	-157	373.005	373.00
2653119	423205	1	14	-975	-974	12503.385	12503.38

tidak memiliki variabel yang cocok sebagai target untuk analisis keseimbangan kelas. Semua fitur yang tersedia dalam dataset merupakan variabel numerik yang bersifat kontinu, seperti jumlah cicilan, jumlah pembayaran, dan hari keterlambatan. Keseimbangan kelas umumnya dianalisis pada variabel kategorikal dengan kelas yang jelas, seperti "lulus/gagal" atau "fraud/non-fraud." Karena tidak ada variabel target yang sesuai, analisis ketidakseimbangan kelas tidak dapat diterapkan pada dataset ini.