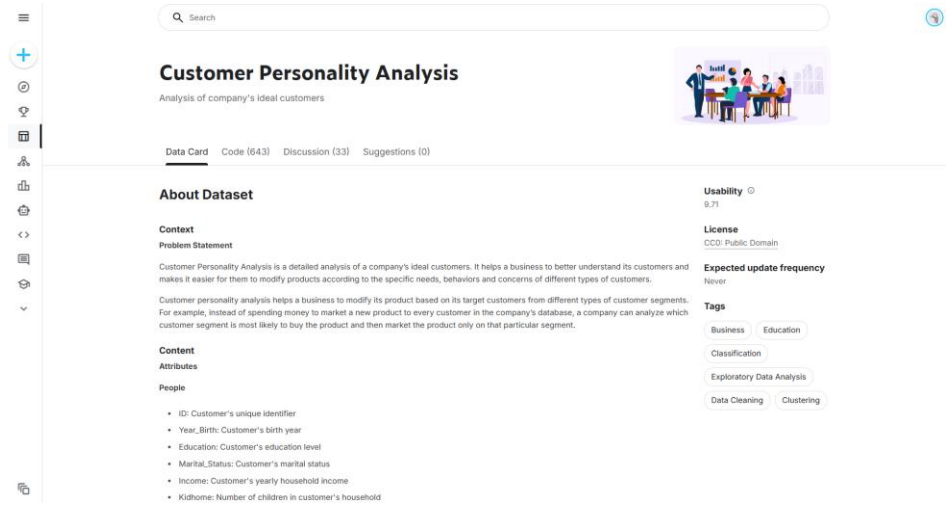
Tugas Proyek Praktik: Pre-processing Dataset Customer Personality Analysis

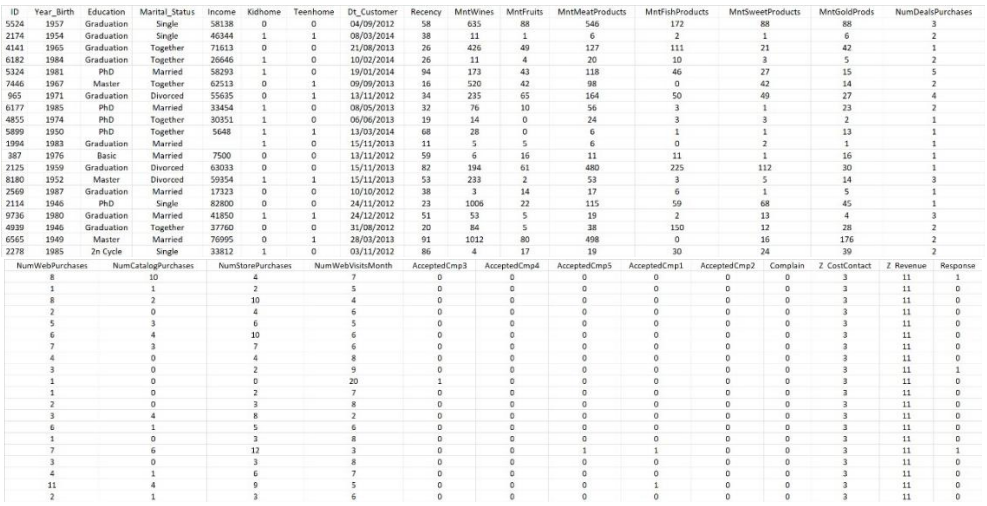
A. Deskripsi Dataset

Dataset yang digunakan dalam proyek ini adalah **"Customer Personality Analysis"**, yang diperoleh dari platform Kaggle (https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis/data).



Dataset ini menyediakan data yang berfokus pada analisis personalitas pelanggan, yaitu studi mendalam tentang target pelanggan ideal sebuah perusahaan. Tujuan dari analisis ini adalah untuk membantu bisnis lebih memahami pelanggan mereka, sehingga memudahkan modifikasi produk atau strategi pemasaran agar sesuai dengan kebutuhan dan perilaku segmen pelanggan tertentu.Dataset ini berisi 2240 baris (mewakili pelanggan individu) dan 29 fitur (kolom). Fitur-fitur ini terbagi menjadi empat kategori utama:

- People: Informasi demografis dan pribadi pelanggan, seperti ID, Year_Birth, Education, Marital_Status, Income, jumlah tanggungan (Kidhome, Teenhome), tanggal pendaftaran (Dt_Customer), Recency (hari sejak pembelian terakhir), dan Complain.
- Products: Jumlah total yang dihabiskan pelanggan dalam 2 tahun terakhir pada berbagai kategori produk, seperti MntWines, MntFruits, MntMeatProducts, MntFishProducts, MntSweetProducts, dan MntGoldProds.
- Place: Saluran tempat pelanggan melakukan pembelian, seperti NumWebPurchases, NumCatalogPurchases, dan NumStorePurchases, serta NumWebVisitsMonth (kunjungan ke situs web).
- Promotion: Riwayat interaksi pelanggan dengan promosi, termasuk jumlah pembelian dengan diskon (NumDealsPurchases), respons terhadap kampanye sebelumnya (AcceptedCmp1 hingga AcceptedCmp5), dan respons terhadap kampanye terakhir (Response).



ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	NumDealsPurchases
5524	1957	Graduation	Single	58138	0	0	04/09/2012	58	635	88	546	172	88	88	3
2174	1954	Graduation	Single	46344	1	1	08/03/2014	38	11	1	6	2	1	6	2
4141	1965	Graduation	Together	71613	0	0	21/08/2013	26	426	49	127	111	21	42	1
6182	1984	Graduation	Together	26646	1	0	10/02/2014	26	11	4	20	10	3	5	2
5324	1981	PhD	Married	58293	1	0	19/01/2014	94	173	43	118	46	27	15	5
7446	1967	Master	Together	62513	0	1	09/09/2013	16	520	42	98	0	42	14	2
965	1971	Graduation	Divorced	55635	0	1	13/11/2012	34	235	65	164	50	49	27	4
6177	1985	PhD	Married	33454	1	0	08/05/2013	32	76	10	56	3	1	23	2
4855	1974	PhD	Together	30351	1	0	06/06/2013	19	14	0	24	3	3	2	1
5899	1950	PhD	Together	5648	1	1	13/03/2014	68	28	0	6	1	1	13	1
1994	1983	Graduation	Married		1	0	15/11/2013	11	5	5	6	0	2	1	1
387	1976	Basic	Married	7500	0	0	13/11/2012	59	6	16	11	11	1	16	1
2125	1959	Graduation	Divorced	63033	0	0	15/11/2013	82	194	61	480	225	112	30	1
8180	1952	Master	Divorced	59354	1	1	15/11/2013	53	233	2	53	3	5	14	3
2569	1987	Graduation	Married	17323	0	0	10/10/2012	38	3	14	17	6	1	5	1
2114	1946	PhD	Single	82800	0	0	24/11/2012	23	1006	22	115	59	68	45	1
9736	1980	Graduation	Married	41850	1	1	24/12/2012	51	53	5	19	2	13	4	3
4939	1946	Graduation	Together	37760	0	0	31/08/2012	20	84	5	38	150	12	28	2
6565	1949	Master	Married	76995	0	1	28/03/2013	91	1012	80	498	0	16	176	2
2278	1985	2n Cycle	Single	33812	1	0	03/11/2012	86	4	17	19	30	24	39	2

NumWebPurchases	NumCatalogPurchases	NumStorePurchases	NumWebVisitsMonth	AcceptedCmp3	AcceptedCmp4	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain	Z_CostContact	Z_Revenue	Response
8	10	4	7	0	0	0	0	0	0	3	11	1
1	1	2	5	0	0	0	0	0	0	3	11	0
8	2	10	4	0	0	0	0	0	0	3	11	0
2	0	4	6	0	0	0	0	0	0	3	11	0
5	3	6	5	0	0	0	0	0	0	3	11	0
6	4	10	6	0	0	0	0	0	0	3	11	0
7	3	7	6	0	0	0	0	0	0	3	11	0
4	0	4	8	0	0	0	0	0	0	3	11	0
3	0	2	9	0	0	0	0	0	0	3	11	1
1	0	0	20	1	0	0	0	0	0	3	11	0
1	0	2	7	0	0	0	0	0	0	3	11	0
2	0	3	8	0	0	0	0	0	0	3	11	0
3	4	8	2	0	0	0	0	0	0	3	11	0
6	1	5	6	0	0	0	0	0	0	3	11	0
1	0	3	8	0	0	0	0	0	0	3	11	0
7	6	12	3	0	0	1	1	0	0	3	11	1
3	0	3	8	0	0	0	0	0	0	3	11	0
4	1	6	7	0	0	0	0	0	0	3	11	0
11	4	9	5	0	0	0	1	0	0	3	11	0
2	1	3	6	0	0	0	0	0	0	3	11	0

B. Tujuan

Tujuan utama dari proyek pra-pemrosesan ini adalah untuk mengubah data mentah (*raw data*) dari dataset "Customer Personality Analysis" menjadi data yang bersih, terstruktur, dan siap digunakan untuk pelatihan model *machine learning*. Secara spesifik, data ini disiapkan untuk mendukung dua skenario *supervised learning* utama, yaitu Tugas Klasifikasi (untuk memprediksi apakah pelanggan akan merespons kampanye terakhir, menggunakan kolom Response) dan Tugas Regresi (untuk memprediksi nilai numerik kontinu, seperti Income). Untuk mencapai hal ini, proyek akan menerapkan kerangka kerja pra-pemrosesan data secara sistematis, yang mencakup 6 langkah: Inspeksi Data, Penanganan *Missing Values*, *Encoding* Data Kategorikal, Penanganan *Outlier*, dan Penskalaan Fitur (*Feature Scaling*).

C. Langkah-Langkah Pra-pemrosesan Data I: Data Cleaning dan Data Transformation

Pada bagian ini, akan dijelaskan 6 langkah sistematis yang dilakukan dalam proses pra-pemrosesan data. Proses ini dibagi menjadi dua aktivitas utama: Data Cleaning dan Data Transformation. Aktivitas Data Cleaning berfokus pada perbaikan data "kotor". Berdasarkan inspeksi, ini mencakup penanganan 24 nilai yang hilang (*missing values*) pada kolom Income dan mengatasi anomali data (*outlier*) pada kolom Year_Birth dan Income. Selanjutnya, aktivitas Data Transformation berfokus pada pengubahan format data agar siap untuk *modelling*. Ini melibatkan *encoding* data kategorikal (Teks) seperti Education dan Marital_Status menjadi format numerik, serta menyeragamkan rentang nilai fitur yang berbeda (seperti Income dan Kidhome) melalui *feature scaling* (penskalaan).

1. Pengumpulan dan Inspeksi Data

Langkah pertama ini berfokus pada pemuatan dataset ke dalam *environment* kerja dan melakukan inspeksi awal. Tujuannya adalah untuk memahami struktur data mentah dan mengidentifikasi masalah kualitas data, seperti tipe data yang tidak konsisten atau data yang tidak lengkap.

1.1. Inisialisasi Environment

Sebagai langkah awal, semua *library* yang diperlukan untuk analisis data, manipulasi, dan visualisasi diimpor, termasuk Pandas, Numpy, Matplotlib, dan Seaborn.

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

✓ 0.0s Python

1.2. Pemuatan dan Inspeksi Visual Data

Dataset marketing_campaign.csv dimuat ke dalam *dataframe* Pandas. Lima baris pertama data ditampilkan untuk memastikan data telah dimuat dengan benar dan untuk mendapatkan gambaran kontekstual awal.

```python
# Tahap 1: Pengumpulan dan Inspeksi Data

# 1. Identifikasi: Muat dataset
df = pd.read_csv('marketing_campaign.csv', sep='\t')

# 2. Inspeksi Awal - 5 baris pertama
print("5 Baris Pertama")
print(df.head())
print("\n")
```

✓ 0.0s Python

Output ini memberikan gambaran visual awal mengenai 29 fitur dan format data di setiap kolom.

```
5 Baris Pertama
     ID  Year_Birth   Education Marital_Status   Income  Kidhome  Teenhome  \
0  5524        1957  Graduation         Single  58138.0        0         0
1  2174        1954  Graduation         Single  46344.0        1         1
2  4141        1965  Graduation       Together  71613.0        0         0
3  6182        1984  Graduation       Together  26646.0        1         0
4  5324        1981         PhD        Married  58293.0        1         0

  Dt_Customer  Recency  MntWines  ...  NumWebVisitsMonth  AcceptedCmp3  \
0  04-09-2012       58       635  ...                  7             0
1  08-03-2014       38        11  ...                  5             0
2  21-08-2013       26       426  ...                  4             0
3  10-02-2014       26        11  ...                  6             0
4  19-01-2014       94       173  ...                  5             0

   AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Complain  \
0             0             0             0             0         0
1             0             0             0             0         0
2             0             0             0             0         0
3             0             0             0             0         0
4             0             0             0             0         0

   Z_CostContact  Z_Revenue  Response
0              3         11         1
1              3         11         0
2              3         11         0
3              3         11         0
4              3         11         0

[5 rows x 29 columns]
```

1.3. Inspeksi Tipe Data

Inspeksi dilanjutkan dengan memeriksa rangkuman teknis dari *dataframe*, termasuk tipe data setiap kolom dan jumlah nilai non-null. Ini adalah langkah krusial untuk mengidentifikasi kolom mana yang bersifat numerik dan mana yang non-numerik (Teks).

```python
# 3. Inspeksi Awal - Periksa tipe data
print("Tipe Data")
print(df.info())
print("\n")
```

✓ 0.0s Python

Output menunjukkan bahwa dataset memiliki 2240 baris. Ditemukan tiga kolom dengan tipe object (Teks), yaitu Education, Marital_Status, dan Dt_Customer. Temuan ini mengindikasikan perlunya *encoding* di langkah selanjutnya.

```
Tipe Data
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2240 non-null   int64
 1   Year_Birth           2240 non-null   float64
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
 4   Income               2240 non-null   float64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64
 9   MntWines             2240 non-null   int64
 10  MntFruits            2240 non-null   int64
 11  MntMeatProducts      2240 non-null   int64
 12  MntFishProducts      2240 non-null   int64
 13  MntSweetProducts     2240 non-null   int64
 14  MntGoldProds         2240 non-null   int64
 15  NumDealsPurchases    2240 non-null   int64
 16  NumWebPurchases      2240 non-null   int64
 17  NumCatalogPurchases  2240 non-null   int64
 18  NumStorePurchases    2240 non-null   int64
 19  NumWebVisitsMonth    2240 non-null   int64
 20  AcceptedCmp3         2240 non-null   int64
 21  AcceptedCmp4         2240 non-null   int64
 22  AcceptedCmp5         2240 non-null   int64
 23  AcceptedCmp1         2240 non-null   int64
 24  AcceptedCmp2         2240 non-null   int64
 25  Complain             2240 non-null   int64
 26  Z_CostContact        2240 non-null   int64
 27  Z_Revenue            2240 non-null   int64
 28  Response             2240 non-null   int64
dtypes: float64(2), int64(24), object(3)
memory usage: 507.6+ KB
None
```

1.4. Inspeksi Nilai Hilang (Missing Values)

Terakhir, dilakukan pemeriksaan kelengkapan data untuk mengidentifikasi *missing values* (data yang tidak lengkap) di setiap kolom.

```python
# 4. Inspeksi Awal - Hitung jumlah nilai hilang
print("Jumlah Nilai Hilang (Awal): ")
print(df.isnull().sum())
```

✓ 0.0s Python

Hasil inspeksi ini sangat penting. Ditemukan bahwa kolom Income memiliki **24 nilai yang hilang (missing values)**, yang akan menjadi fokus pada langkah pembersihan data berikutnya.

```
Jumlah Nilai Hilang (Awal):
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                24
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
```

2. Penanganan Nilai Hilang (Missing Values)

Berdasarkan temuan di Langkah 1, dataset ini memiliki 24 nilai yang hilang (incomplete) pada kolom Income. Langkah ini berfokus pada penanganan masalah tersebut menggunakan metode imputasi.

2.1. Perhitungan Metode Imputasi (Median)

Metode yang dipilih untuk imputasi adalah **Median** (nilai tengah) dari kolom Income.

```python
# Tahap 2: Penanganan Nilai Hilang (Missing Values)

# 1. Hitung Median dari kolom 'Income'
# .median() secara otomatis mengabaikan nilai NaN saat perhitungan
median_income = df['Income'].median()
print(f"Median 'Income' yang akan digunakan untuk imputasi: {median_income}")
```

✓ 0.0s Python

```
Median 'Income' yang akan digunakan untuk imputasi: 51381.5
```

Pendapatan (Income) adalah fitur yang rentan terhadap *outlier* (nilai ekstrem yang sangat tinggi). Jika menggunakan *Mean* (rata-rata), nilai *outlier* tersebut akan menggeser (mendistorsi) nilai imputasi. **Median** dipilih karena bersifat *robust* (kuat) terhadap *outlier*, sehingga menghasilkan nilai pengganti yang lebih representatif untuk mayoritas data.

2.2. Penerapan Imputasi

Nilai median yang telah dihitung kemudian digunakan untuk mengisi 24 baris yang hilang tersebut menggunakan fungsi .fillna().

```python
# 2. Terapkan Imputasi Median
# Kita mengisi 24 nilai yang hilang (NaN) dengan nilai median
df['Income'] = df['Income'].fillna(median_income)

# 3. Verifikasi Hasil
print("\nVerifikasi Setelah Imputasi")
print("Jumlah nilai hilang di kolom 'Income' sekarang:")
print(df['Income'].isnull().sum())
```

✓ 0.0s Python

```
Verifikasi Setelah Imputasi
Jumlah nilai hilang di kolom 'Income' sekarang:
0
```

Output ini mengonfirmasi bahwa 24 nilai yang hilang pada kolom Income telah berhasil diisi.

2.3. Verifikasi Akhir Kelengkapan Data

Sebagai verifikasi akhir, dilakukan pemeriksaan nilai hilang pada *keseluruhan* dataset untuk memastikan tidak ada lagi data yang hilang sebelum melanjutkan ke langkah berikutnya.

```python
# 4. Hitung kembali jumlah nilai hilang
print("Jumlah Nilai Hilang (setelah di Imputasi): ")
print(df.isnull().sum())
```

✓ 0.0s Python

Output ini menunjukkan bahwa dataset sekarang telah lengkap (0 nilai hilang di semua kolom), dan proses Data Cleaning untuk *missing values* telah selesai.

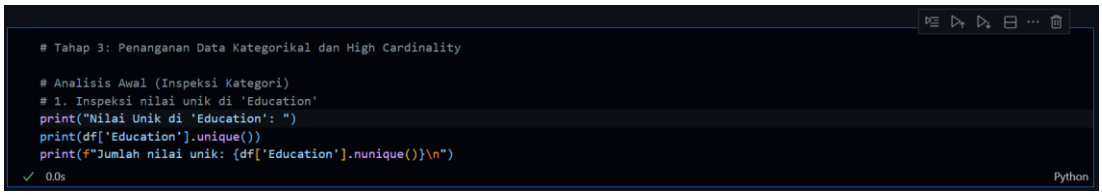
```
Jumlah Nilai Hilang (setelah di Imputasi):
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                 0
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
```

3. Penanganan Data Kategorikal dan High Cardinality

Langkah ini berfokus pada aktivitas **"Feature Engineering"**. Berdasarkan temuan di Langkah 1, kolom Education dan Marital_Status adalah data object (Teks). Algoritma *machine learning* tidak dapat memproses data teks, sehingga kita harus mengubahnya (meng-encode) menjadi format numerik.

3.1. Inspeksi Fitur Kategorikal (Education)

Pertama, dilakukan inspeksi untuk mengidentifikasi semua nilai unik dalam kolom Education untuk memahami variasinya.

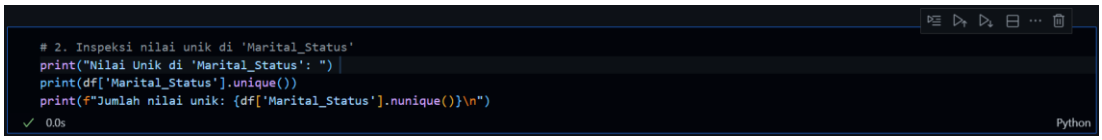


Ditemukan 5 kategori unik. Ini adalah jumlah kardinalitas yang rendah dan dapat langsung di-encode.

```
Nilai Unik di 'Education':
['Graduation' 'PhD' 'Master' 'Basic' '2n Cycle']
Jumlah nilai unik: 5
```

3.2. Inspeksi Fitur Kategorikal (Marital_Status)

Inspeksi serupa dilakukan pada kolom Marital_Status.

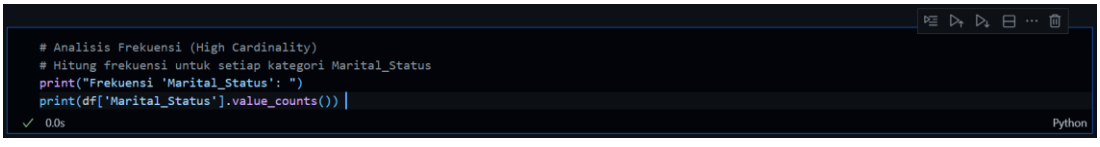


Ditemukan 8 kategori unik. Selain kategori standar (misal, 'Married', 'Single'), terdapat beberapa nilai yang berpotensi anomali atau sangat jarang, seperti 'Alone', 'Absurd', dan 'YOLO'.

```
Nilai Unik di 'Marital_Status':
['Single' 'Together' 'Married' 'Divorced' 'Widow' 'Lain-Lain']
Jumlah nilai unik: 6
```

3.3. Analisis Frekuensi (Marital_Status)

Untuk menangani kategori yang berpotensi anomali tersebut, dilakukan analisis frekuensi untuk memverifikasi seberapa sering kategori tersebut muncul.




```
Frekuensi 'Marital_Status':
Marital_Status
Married     864
Together    580
Single      480
Divorced    232
Widow        77
Alone         3
Absurd        2
YOLO          2
Name: count, dtype: int64
```

Output ini mengonfirmasi bahwa kategori 'Alone' (3), 'Absurd' (2), dan 'YTODO' (2) memiliki frekuensi yang sangat rendah (jauh di bawah 1% dari total data). Mempertahankan kategori ini dapat menambah *noise* pada model. Sesuai kerangka kerja, kategori-kategori ini akan dikelompokkan.

3.4. Penerapan Pengelompokan dan One-Hot Encoding

Strategi berikut diterapkan:

- **Grouping:** Kategori langka ('Alone', 'Absurd', 'YOLO') dikelompokkan menjadi satu kategori baru, '**Lain-Lain**'.
- **Encoding:** Metode **One-Hot Encoding** (pd.get_dummies) diterapkan pada kedua kolom (Education dan Marital_Status yang telah bersih). Metode ini membuat kolom biner (dummy) baru untuk setiap kategori, mengubah data teks menjadi format numerik (True/False, yang setara dengan 1/0) yang siap untuk model.

```
# 1. Penanganan High Cardinality: Kelompokkan kategori langka di 'Marital_Status'
# Berdasarkan inspeksi, kita kelompokkan nilai-nilai anomali/langka
rare_categories = ['Alone', 'Absurd', 'YOLO']
df['Marital_Status'] = df['Marital_Status'].replace(rare_categories, 'Lain-Lain')

# Verifikasi hasil pengelompokan
print("\n'Marital_Status' Setelah Pengelompokan: ")
print(df['Marital_Status'].unique())
```

✓ 0.0s Python

```
# 2. Terapkan One-Hot Encoding
# Kita menggunakan pd.get_dummies() untuk membuat kolom baru untuk setiap kategori
# 'drop_first=True' adalah praktik yang baik untuk menghindari multikolinearitas
# (Ini akan dibahas di agenda Feature Selection Anda)
df_encoded = pd.get_dummies(df, columns=['Education', 'Marital_Status'], drop_first=True)

# 3. Verifikasi Hasil
print("\nDataframe Setelah One-Hot Encoding (5 Baris Pertama): ")
print(df_encoded.head())
```

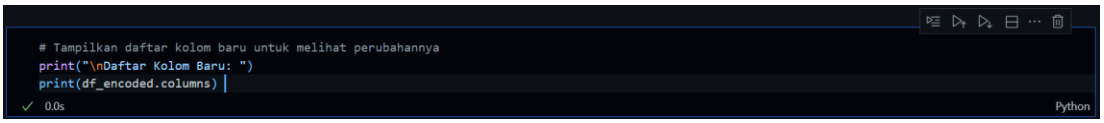
✓ 0.0s Python

```
'Marital_Status' Setelah Pengelompokan:
['Single' 'Together' 'Married' 'Divorced' 'Widow' 'Lain-Lain']
Dataframe Setelah One-Hot Encoding (5 Baris Pertama):
     ID  Year_Birth   Income  Kidhome  Teenhome Dt_Customer  Recency  \
0  5524        1957  58138.0        0         0  04-09-2012       58
1  2174        1954  46344.0        1         1  08-03-2014       38
2  4141        1965  71613.0        0         0  21-08-2013       26
3  6182        1984  26646.0        1         0  10-02-2014       26
4  5324        1981  58293.0        1         0  19-01-2014       94

   MntWines  MntFruits  MntMeatProducts  ...  Response  Education_Basic  \
0       635         88              546  ...         1            False
1        11          1                6  ...         0            False
2       426         49              127  ...         0            False
3        11          4               20  ...         0            False
4       173         43              118  ...         0            False

   Education_Graduation  Education_Master  Education_PhD  \
0                  True             False          False
1                  True             False          False
2                  True             False          False
3                  True             False          False
4                 False             False           True

   Marital_Status_Lain-Lain  Marital_Status_Married  Marital_Status_Single  \
0                     False                  False                   True
1                     False                  False                   True
2                     False                  False                  False
3                     False                  False                  False
4                     False                   True                  False

   Marital_Status_Together  Marital_Status_Widow
0                    False                 False
1                    False                 False
2                     True                 False
3                     True                 False
4                    False                 False

[5 rows x 36 columns]
```

Hasil .head() menunjukkan *dataframe* baru bernama df_encoded, di mana kolom Education dan Marital_Status yang asli telah digantikan oleh kolom-kolom biner baru (misal, Education_PhD, Marital_Status_Married, dll.).

3.5. Verifikasi Struktur Kolom Baru

Terakhir, daftar kolom ditampilkan untuk mengonfirmasi struktur *dataframe* yang baru setelah *encoding*.



```
# Tampilkan daftar kolom baru untuk melihat perubahannya
print("\nDaftar Kolom Baru: ")
print(df_encoded.columns)
```

0.0s Python

Dataset kini memiliki 36 kolom dan siap untuk langkah penanganan *outlier*.

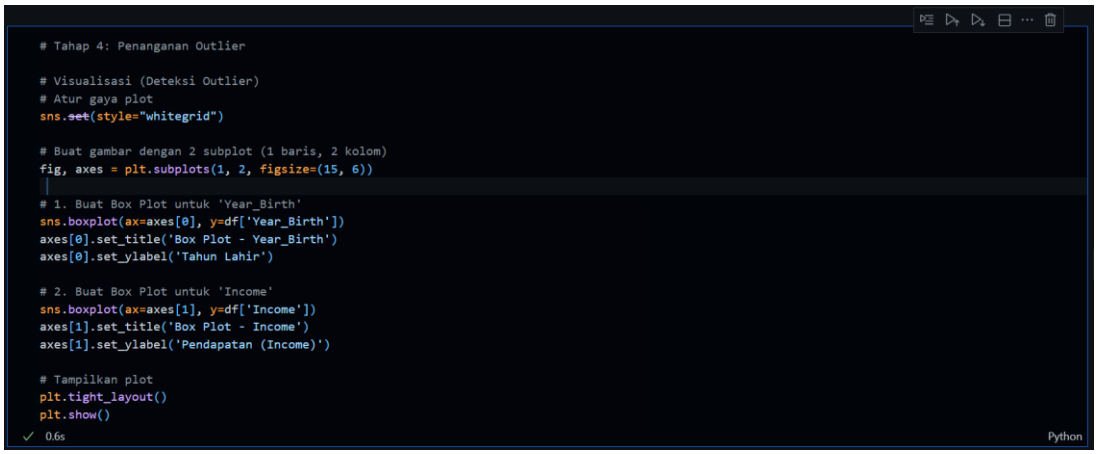
```
Daftar Kolom Baru:
Index(['ID', 'Year_Birth', 'Income', 'Kidhome', 'Teenhome', 'Dt_Customer',
       'Recency', 'MntWines', 'MntFruits', 'MntMeatProducts',
       'MntFishProducts', 'MntSweetProducts', 'MntGoldProds',
       'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases',
       'NumStorePurchases', 'NumWebVisitsMonth', 'AcceptedCmp3',
       'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2',
       'Complain', 'Z_CostContact', 'Z_Revenue', 'Response', 'Education_Basic',
       'Education_Graduation', 'Education_Master', 'Education_PhD',
       'Marital_Status_Lain-Lain', 'Marital_Status_Married',
       'Marital_Status_Single', 'Marital_Status_Together',
       'Marital_Status_Widow'],
      dtype='object')
```

4. Penanganan Outlier

Langkah ini melanjutkan aktivitas **Data Cleaning** dengan fokus pada data "noisy" atau *outlier*. *Outlier* adalah nilai-nilai ekstrem yang berada jauh di luar rentang normal data dan dapat merusak akurasi serta stabilitas model matematis.

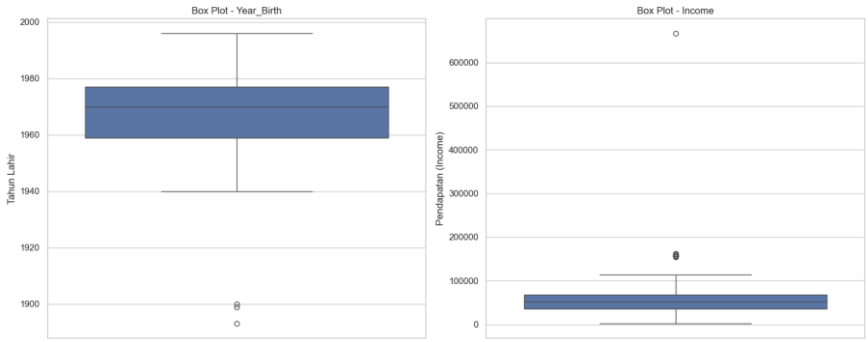
4.1. Visualisasi (Deteksi Outlier)

Metode yang paling efektif untuk mendeteksi *outlier* adalah melalui visualisasi. **Box Plot** digunakan untuk memvisualisasikan distribusi dan mengidentifikasi anomali pada fitur-fitur yang rentan, yaitu Year_Birth (untuk anomali usia) dan Income (untuk pendapatan ekstrem).



```
# Tahap 4: Penanganan Outlier

# Visualisasi (Deteksi Outlier)
# Atur gaya plot
sns.set(style="whitegrid")

# Buat gambar dengan 2 subplot (1 baris, 2 kolom)
fig, axes = plt.subplots(1, 2, figsize=(15, 6))

# 1. Buat Box Plot untuk 'Year_Birth'
sns.boxplot(ax=axes[0], y=df['Year_Birth'])
axes[0].set_title('Box Plot - Year_Birth')
axes[0].set_ylabel('Tahun Lahir')

# 2. Buat Box Plot untuk 'Income'
sns.boxplot(ax=axes[1], y=df['Income'])
axes[1].set_title('Box Plot - Income')
axes[1].set_ylabel('Pendapatan (Income)')

# Tampilkan plot
plt.tight_layout()
plt.show()
```

0.6s Python



Plot di atas dengan jelas mengonfirmasi adanya *outlier*.

1. **Year_Birth:** Terlihat beberapa titik data di bawah "pagar" (whisker) bawah, menunjukkan nilai-nilai anomali (tahun lahir sekitar 1900 atau lebih tua).
2. **Income:** Terlihat beberapa titik di atas pagar atas, termasuk satu *outlier* yang sangat ekstrem (di atas 600.000).

4.2. Penerapan Capping (Pembatasan Outlier)

Sesuai kerangka kerja, *outlier* ini tidak dihapus (untuk menghindari kehilangan data), melainkan ditangani dengan metode **Capping** (pembatasan). Metode ini menggunakan **1.5 x IQR (Interquartile Range)** untuk menentukan "batas pagar" yang wajar.

Nilai yang berada di luar batas pagar ini akan diganti dengan nilai batas pagar itu sendiri.

```python
# Terapkan Capping untuk 'Year_Birth' (Menangani usia anomali)
Q1_year = df['Year_Birth'].quantile(0.25)
Q3_year = df['Year_Birth'].quantile(0.75)
IQR_year = Q3_year - Q1_year
lower_bound_year = Q1_year - 1.5 * IQR_year
upper_bound_year = Q3_year + 1.5 * IQR_year

print(f"Capping untuk 'Year_Birth': ")
print(f"Batas Bawah: {lower_bound_year}, Batas Atas: {upper_bound_year}")

# Terapkan Capping
# Kita hanya cap pada batas bawah (usia terlalu tua)
df['Year_Birth'] = np.where(
    df['Year_Birth'] < lower_bound_year,
    lower_bound_year,
    df['Year_Birth']
)

# Terapkan Capping untuk 'Income' (Menangani pendapatan ekstrem)
Q1_income = df['Income'].quantile(0.25)
Q3_income = df['Income'].quantile(0.75)
IQR_income = Q3_income - Q1_income
lower_bound_income = Q1_income - 1.5 * IQR_income
upper_bound_income = Q3_income + 1.5 * IQR_income

print(f"\nCapping untuk 'Income': ")
print(f"Batas Bawah: {lower_bound_income}, Batas Atas: {upper_bound_income}")

# Terapkan Capping
# Kita hanya cap pada batas atas (pendapatan terlalu tinggi)
df['Income'] = np.where(
    df['Income'] > upper_bound_income,
    upper_bound_income,
    df['Income']
)

print("\nOutlier telah ditangani (di-cap): ")
```

✓ 0.0s Python

```
Capping untuk 'Year_Birth':
Batas Bawah: 1932.0, Batas Atas: 2004.0

Capping untuk 'Income':
Batas Bawah: -13587.75, Batas Atas: 117416.25

Outlier telah ditangani (di-cap)
```

Batas bawah untuk Year_Birth dihitung sekitar **1932.0**, dan batas atas untuk Income dihitung sekitar **117416.25**. *Outlier* yang ekstrem (seperti pendapatan > 600k) mungkin merepresentasikan data yang sah

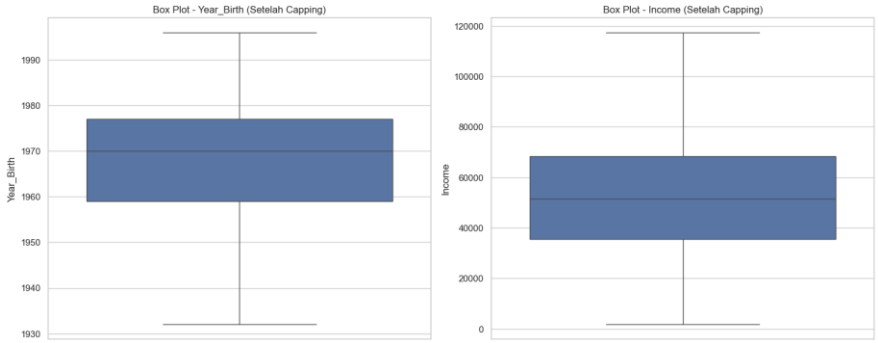
(misal, kesenjangan sosial). Namun, *capping* tetap dilakukan sebagai **keputusan teknis** untuk melindungi model dari *noise* statistik, yang dapat merusak perhitungan matematis dan membuat model menjadi tidak stabil.

4.3. Verifikasi Hasil Capping

Box Plot dibuat ulang untuk memverifikasi bahwa *outlier* telah berhasil ditangani.

```
# Verifikasi: Kita bisa membuat ulang box plot untuk melihat perubahannya
print("--- Verifikasi Box Plot (Setelah Capping) ---")
fig, axes = plt.subplots(1, 2, figsize=(15, 6))
sns.boxplot(ax=axes[0], y=df['Year_Birth']).set_title('Box Plot - Year_Birth (Setelah Capping)')
sns.boxplot(ax=axes[1], y=df['Income']).set_title('Box Plot - Income (Setelah Capping)')
plt.tight_layout()

plt.show()
```

✓ 0.3s Python



Plot verifikasi di atas menunjukkan bahwa data sekarang jauh lebih bersih. Titik-titik *outlier* ekstrem telah "ditarik" masuk ke batas pagar (whisker) plot, membuat distribusi data lebih padat dan siap untuk langkah penskalaan.

5. Penskalaan Fitur

Langkah ini adalah bagian akhir dari **"Data Transformation"**. Tujuannya adalah untuk menyeragamkan rentang nilai dari semua fitur numerik.

Fitur-fitur seperti Income (puluhan ribu) dan Kidhome (0, 1, 2) memiliki skala yang sangat berbeda. Perbedaan skala ini dapat menyebabkan model *machine learning* (seperti K-NN yang berbasis jarak) memberikan bobot yang tidak adil pada fitur dengan nilai besar. Penskalaan (Standardisasi) mengubah semua fitur ke skala yang sama (rata-rata 0, standar deviasi 1) sehingga perbandingannya menjadi "apple to apple".

5.1. Persiapan Data (Train-Test Split)

Sesuai kerangka kerja, sangat penting untuk memisahkan data menjadi set Latih dan set Uji **sebelum** menerapkan penskalaan. Ini adalah praktik terbaik untuk mencegah *data leakage*, di mana statistik dari data uji (yang seharusnya "tidak terlihat") "bocor" dan memengaruhi parameter penskalaan yang dipelajari dari data latih.

```python
# Tahap 5: Penskalaan Fitur

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Persiapan Sebelum Scaling
# 1. Mulai dari df_encoded (hasil Langkah 3) dan hapus kolom yang tidak relevan
# Kita membuat variabel baru agar jelas
df_processed = df_encoded.drop(['ID', 'Dt_Customer', 'Z_CostContact', 'Z_Revenue'], axis=1)

# 2. Pisahkan Fitur (X) dan Target (y)
X = df_processed.drop('Response', axis=1)
y = df_processed['Response']

# 3. Bagi Data (Train-Test Split)
# Kita tetap menggunakan stratify=y karena data kita Imbalanced
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# 4. Identifikasi kolom yang akan di-scale
# Kita pilih semua kolom yang tipe datanya float atau int, bukan biner
numerical_cols = [
    col for col in X_train.columns
    if X_train[col].dtype in ['int64', 'float64']
    and X_train[col].nunique() > 2
]

print(f"Bentuk X_train: {X_train.shape}")
print(f"Bentuk X_test: {X_test.shape}")
print("\nKolom yang akan di-scale:")
print(numerical_cols)
```

✓ 0.0s 暟 Open 'numerical_cols' in Data Wrangler Python

```
Bentuk X_train: (1792, 31)
Bentuk X_test: (448, 31)

Kolom yang akan di-scale:
['Year_Birth', 'Income', 'Kidhome', 'Teenhome', 'Recency', 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
'NumWebVisitsMonth']
```

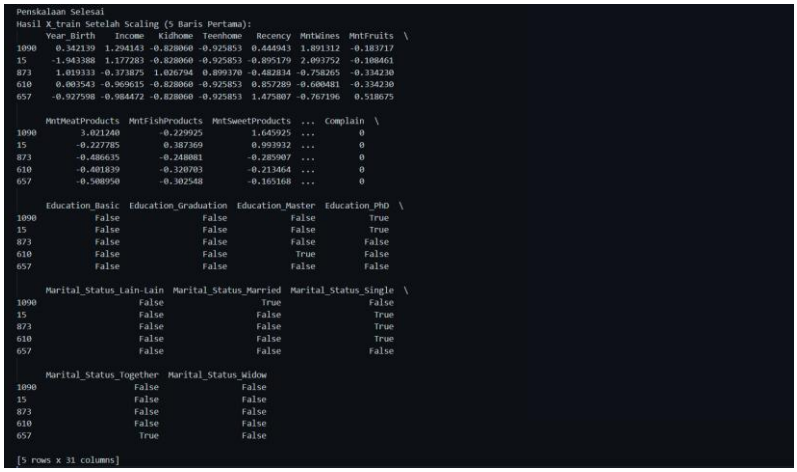
Pada langkah ini, kolom-kolom yang tidak relevan (seperti ID dan konstanta Z_...) dibuang. Data kemudian dibagi menjadi 80% data latih (1792 baris) dan 20% data uji (448 baris). Parameter **stratify=y** digunakan secara sengaja. Ini memastikan bahwa rasio kelas target Response yang **tidak seimbang** (1906 vs 334) tetap terjaga secara proporsional di kedua set data (latih dan uji). Terakhir, daftar numerical_cols dibuat untuk memisahkan kolom numerik kontinu (yang akan di-scale) dari kolom biner/encoded (yang sudah dalam skala 0/1 dan tidak perlu di-scale).

5.2. Penerapan Standardisasi

Metode **StandardScaler** (Z-score normalization) dipilih. Metode ini akan "belajar" (menggunakan .fit()) nilai rata-rata dan standar deviasi **hanya** dari X_train, dan kemudian menerapkan transformasi tersebut (.transform()) ke X_train dan X_test.

```python
# Menerapkan Penskalaan Fitur

# 1. Inisialisasi StandardScaler
scaler = StandardScaler()

# 2. Terapkan Penskalaan (fit hanya pada data latih)
# Sekarang scaler.fit() hanya akan menerima kolom numerik yang benar
scaler.fit(X_train[numerical_cols])

# 3. Terapkan (transform) pada Data Latih dan Data Uji
X_train[numerical_cols] = scaler.transform(X_train[numerical_cols])
X_test[numerical_cols] = scaler.transform(X_test[numerical_cols])

print("\nPenskalaan Selesai")

# Verifikasi: Tampilkan 5 baris pertama dari data latih yang sudah di-scale
print("Hasil X_train Setelah Scaling (5 Baris Pertama): ")
print(X_train.head())
```

✓ 0.0s Python



```
Penskalaan Selesai
Hasil X_train Setelah Scaling (5 Baris Pertama):
      Year_Birth    Income   Kidhome  Teenhome   Recency  MntWines  MntFruits  \
1090    0.342139  1.294143 -0.828060 -0.925853  0.444943  1.891312  -0.183717
15     -1.943388  1.177283 -0.828060 -0.925853 -0.895179  2.093752  -0.108461
873     1.019333 -0.373875  1.026794  0.899370 -0.482834 -0.758265  -0.334230
610     0.003543 -0.969615 -0.828060 -0.925853  0.857289 -0.600481  -0.334230
657    -0.927598 -0.984472 -0.828060 -0.925853  1.475807 -0.767196   0.518675

      MntMeatProducts  MntFishProducts  MntSweetProducts  ...  Complain  \
1090         3.021240        -0.229925          1.645925  ...         0
15          -0.227785         0.387369          0.993932  ...         0
873         -0.486635        -0.248081         -0.285907  ...         0
610         -0.401839        -0.320703         -0.213464  ...         0
657         -0.508950        -0.302548         -0.165168  ...         0

      Education_Basic  Education_Graduation  Education_Master  Education_PhD  \
1090            False                 False             False           True
15              False                 False             False           True
873             False                 False             False          False
610             False                 False              True          False
657             False                 False             False          False

      Marital_Status_Lain-Lain  Marital_Status_Married  Marital_Status_Single  \
1090                     False                   True                  False
15                       False                  False                   True
873                      False                  False                   True
610                      False                  False                   True
657                      False                  False                  False

      Marital_Status_Together  Marital_Status_Widow
1090                    False                 False
15                      False                 False
873                     False                 False
610                     False                 False
657                      True                 False

[5 rows x 31 columns]
```

Output .head() terakhir menunjukkan bahwa proses penskalaan telah berhasil. Kolom-kolom numerik (seperti Year_Birth, Income, MntWines, dll.) kini memiliki nilai-nilai baru yang terstandardisasi (misal, 0.342139, 1.294143, 1.891312), yang menunjukkan bahwa semua fitur tersebut sekarang berada dalam skala yang seragam dan siap untuk *modelling*.

6. Kesimpulan

Sebagai kesimpulan, pra-pemrosesan ini telah berhasil mengubah dataset "Customer Personality Analysis" yang mentah menjadi data yang bersih dan siap untuk *modelling*. Serangkaian metode teknis telah diterapkan dengan justifikasi yang jelas: Imputasi Median dipilih untuk menangani 24 *missing values* di Income karena sifatnya yang *robust* terhadap *outlier*; One-Hot Encoding (disertai *grouping* kategori langka) digunakan untuk mengubah data Teks (Education, Marital_Status) menjadi format biner yang dapat dibaca model; Capping (1.5*IQR) diterapkan sebagai keputusan teknis untuk membatasi *noise* statistik dari *outlier* ekstrem di Year_Birth dan Income; dan terakhir, StandardScaler diterapkan setelah *train-test split* (dengan *stratify*) untuk menyeragamkan skala fitur, yang krusial untuk algoritma yang sensitif terhadap jarak seperti K-NN.