User segmentation

user segmentation dilakukan untuk mengelompokan user kedalam beberapa grup berdasarkan geografi, demografi, budaya dan kebiasaan. pengelompokan ini diharapkan dapat membantu meningkatkan penjualan, pengunaan user ataupun pencegahan agar user tidak meninggalkan produk/platform.

FRM Segmentation

RFM segmentation adalah salah satu teknik segmentasi pengguna berdasarkan 3 karakteristik utama dari kebiasaan transaksi: kebaruan(resensi), frequency, moneter

```
import pandas as pd
import numpy as np
import datetime as dt
```

```
df = pd.read_csv('Salinan Salinan Online Retail Data.csv')
df
```

	order_id	product_code	product_name	quantity	orc
0	493410	TEST001	This is a test product.	5	20
1	C493411	21539	RETRO SPOTS BUTTER DISH	-1	20
2	493412	TEST001	This is a test product.	5	20
3	493413	21724	PANDA AND BUNNIES STICKER SHEET	1	20
4	493413	84578	ELEPHANT TOY WITH BLUE T-SHIRT	1	20
...	...	...	...	...	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 461773 entries, 0 to 461772
Data columns (total 7 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   order_id      461773 non-null  object
 1   product_code  461773 non-null  object
 2   product_name  459055 non-null  object
 3   quantity      461773 non-null  int64
 4   order_date    461773 non-null  object
 5   price         461773 non-null  float64
 6   customer_id   360853 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 24.7+ MB
```

Data Cleaning

```
df_clean = df.copy()
# membuat kolom date
df_clean['date'] = pd.to_datetime(df_clean['order_date']).dt.date.astype('datetime64')

# menghapus semua baris tanpa customer_id
df_clean= df_clean[~df_clean['customer_id'].isna()]

# menghapus semua baris tanpa product_name
df_clean = df_clean[~df_clean['product_name'].isna()]

# membuat semua product_name berhuruf kecil
```

```
df_clean['product_name'] = df_clean['product_name'].str.lower()

# menghapus semua baris dengan product_code atau product_name test
df_clean = df_clean[(~df_clean['product_code'].str.lower().str.contains('test')) | (~df_clean['product_name'].str.contains('test'))]

# membuat kolom order_status dengan nilai 'cancelled' jika order_id diawali dengan huruf 'C' dan 'delivered' jika tanpa 'C'
df_clean['order_status'] = np.where(df_clean['order_id'].str[:1] == 'C', 'cancelled', 'delivered')

# mengubah nilai quantity yang negatif menjadi positif
df_clean['quantity'] = df_clean['quantity'].abs()

# menghapus baris dengan price ernilai negatif
df_clean = df_clean[df_clean['quantity']>0]

# membuat nilai amount, yaitu perkalian antara quantity dan price
df_clean['amount'] = df_clean['quantity'] * df_clean['price']

# mengganti product_name dari product_code yang memiliki beerapa product_name dengan salah satu product_name nya yang sering muncul
most_freq_product_name = df_clean.groupby(['product_code', 'product_name'], as_index=False).agg(order_cnt=('order_id','nunique')).sort_va
most_freq_product_name['rank'] = most_freq_product_name.groupby('product_code')['order_cnt'].rank(method='first', ascending=False)
most_freq_product_name = most_freq_product_name[most_freq_product_name['rank']==1].drop(columns=['order_cnt','rank'])
df_clean = df_clean.merge(most_freq_product_name.rename(columns={'product_name':'most_freq_product_name'}), how='left', on='product_code'
df_clean['product_name'] = df_clean['most_freq_product_name']
df_clean = df_clean.drop(columns='most_freq_product_name')

# mengkonversi customer_id menjadi string
df_clean['customer_id'] = df_clean['customer_id'].astype(str)

# menghapus outlier
from scipy import stats
df_clean = df_clean[(np.abs(stats.zscore(df_clean[['quantity','amount']]))<3).all(axis=1)]
df_clean = df_clean.reset_index(drop=True)
df_clean
```

	order_id	product_code	product_name	quantity	order_dat
0	C493411	21539	red retrospot butter dish	1	2010-01-0 09:43:0
1	493414	21844	red retrospot mug	36	2010-01-0 10:28:0
2	493414	21533	retro spot large milk jug	12	2010-01-0 10:28:0
3	493414	37508	new england ceramic cake server	2	2010-01-0 10:28:0
4	493414	35001G	hand open shape gold	2	2010-01-0 10:28:0
...	...	...	...	...	.

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 358482 entries, 0 to 358481
Data columns (total 10 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   order_id      358482 non-null  object
 1   product_code  358482 non-null  object
 2   product_name  358482 non-null  object
 3   quantity      358482 non-null  int64
 4   order_date    358482 non-null  object
 5   price         358482 non-null  float64
 6   customer_id   358482 non-null  object
 7   date          358482 non-null  datetime64[ns]
 8   order_status  358482 non-null  object
 9   amount        358482 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(1), object(6)
memory usage: 27.4+ MB
```

Membuat RFM Segmentation

Agregat data transaksi ke bentuk summary total transaksi (order), total nilai order (order value), tanggal order terakhir dari setiap pengguna

```
df_user = df_clean.groupby('customer_id', as_index=False).agg(order_cnt=('order_id','nunique'),
                                                              max_order_date=('date','max'),
                                                              total_order_value=('amount', 'sum'))
df_user
```

	customer_id	order_cnt	max_order_date	total_order_value
0	12346.0	5	2010-10-04	602.40
1	12608.0	1	2010-10-31	415.79
2	12745.0	2	2010-08-10	723.85
3	12746.0	2	2010-06-30	266.35
4	12747.0	19	2010-12-13	4094.79
...	...	...	...	...
3884	18283.0	6	2010-11-22	641.77
3885	18284.0	2	2010-10-06	486.68
3886	18285.0	1	2010-02-17	427.00
3887	18286.0	2	2010-08-20	941.48

Kolom jumlah hari sejak order terakhir

```
today = df_clean['date'].max()
df_user['day_since_last_order'] = (today - df_user['max_order_date']).dt.days
df_user
```

	customer_id	order_cnt	max_order_date	total_order_value
0	12346.0	5	2010-10-04	602.40
1	12608.0	1	2010-10-31	415.79
2	12745.0	2	2010-08-10	723.85
3	12746.0	2	2010-06-30	266.35
4	12747.0	19	2010-12-13	4094.79
...	...	...	...	...
3884	18283.0	6	2010-11-22	641.77
3885	18284.0	2	2010-10-06	486.68
3886	18285.0	1	2010-02-17	427.00

```
df_user.describe()
```

Membuat binning dari jumlah hari sejak order terakhir yang terdiri dari 5 bins, dengan batasan min,p20,p40,p60,p80,max dan beri 1 sampai 5 dari bin tertinggi ke rendah sebagai skor recency

```
df_user['recency_score'] = pd.cut(df_user['day_since_last_order'],
                                  bins = [
                                          df_user['day_since_last_order'].min(),
                                          np.percentile(df_user['day_since_last_order'], 20),
                                          np.percentile(df_user['day_since_last_order'], 40),
                                          np.percentile(df_user['day_since_last_order'], 60),
                                          np.percentile(df_user['day_since_last_order'], 80),
                                          df_user['day_since_last_order'].max()
                                         ], labels=[5, 4, 3, 2, 1],include_lowest=True).astype(int)
df_user
```

	customer_id	order_cnt	max_order_date	total_order_value
0	12346.0	5	2010-10-04	602.40
1	12608.0	1	2010-10-31	415.79
2	12745.0	2	2010-08-10	723.85
3	12746.0	2	2010-06-30	266.35
4	12747.0	19	2010-12-13	4094.79
...	...	...	...	...
3884	18283.0	6	2010-11-22	641.77
3885	18284.0	2	2010-10-06	486.68
3886	18285.0	1	2010-02-17	427.00

Buat binning dari total transaksi (order) yang terdiri dari 5 bins dengan batas-batasnya dan beri label 1 sampai 5 dari bin terendah ke tertinggi sebagai skor frequency

```
df_user['frequency_score'] = pd.cut(df_user['order_cnt'],
                                    bins=[
                                        0,
                                        np.percentile(df_user['order_cnt'], 20),
                                        np.percentile(df_user['order_cnt'], 40),
                                        np.percentile(df_user['order_cnt'], 60),
                                        np.percentile(df_user['order_cnt'], 80),
                                        df_user['order_cnt'].max()
                                    ], labels=[1, 2, 3, 4, 5], include_lowest=True).astype(int)
df_user
```

	customer_id	order_cnt	max_order_date	total_order_value
0	12346.0	5	2010-10-04	602.40
1	12608.0	1	2010-10-31	415.79
2	12745.0	2	2010-08-10	723.85
3	12746.0	2	2010-06-30	266.35
4	12747.0	19	2010-12-13	4094.79
...	...	...	...	...
3884	18283.0	6	2010-11-22	641.77
3885	18284.0	2	2010-10-06	486.68
3886	18285.0	1	2010-02-17	427.00

Buat bins dari total nilai order (order value) yang terdiri dari 5 bins sebagai skor monetary

```
df_user['monetary_score'] = pd.cut(df_user['total_order_value'],
                                   bins=[
                                       df_user['total_order_value'].min(),
                                       np.percentile(df_user['total_order_value'], 20),
                                       np.percentile(df_user['total_order_value'], 40),
                                       np.percentile(df_user['total_order_value'], 60),
                                       np.percentile(df_user['total_order_value'], 80),
                                       df_user['total_order_value'].max()
                                   ], labels=[1, 2, 3, 4, 5], include_lowest=True).astype(int)
df_user
```

	customer_id	order_cnt	max_order_date	total_order_value
0	12346.0	5	2010-10-04	602.40
1	12608.0	1	2010-10-31	415.79
2	12745.0	2	2010-08-10	723.85
3	12746.0	2	2010-06-30	266.35
4	12747.0	19	2010-12-13	4094.79
...	...	...	...	...
3884	18283.0	6	2010-11-22	641.77
3885	18284.0	2	2010-10-06	486.68
3886	18285.0	1	2010-02-17	427.00

```
df_user.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3889 entries, 0 to 3888
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   customer_id           3889 non-null   object
 1   order_cnt             3889 non-null   int64
 2   max_order_date        3889 non-null   datetime64[ns]
 3   total_order_value     3889 non-null   float64
 4   day_since_last_order  3889 non-null   int64
 5   recency_score         3889 non-null   int32
 6   frequency_score       3889 non-null   int32
 7   monetary_score        3889 non-null   int32
dtypes: datetime64[ns](1), float64(1), int32(3), int64(2), object(1)
memory usage: 197.6+ KB
```

```
df_user['segment'] = np.select(
    [(df_user['recency_score']==5) & (df_user['frequency_score']>=4),
     (df_user['recency_score'].between(3, 4)) & (df_user['frequency_score']>=4),
     (df_user['recency_score']>=4) & (df_user['frequency_score'].between(2, 3)),
     (df_user['recency_score']<=2) & (df_user['frequency_score']==5),
     (df_user['recency_score']==3) & (df_user['frequency_score']==3),
     (df_user['recency_score']==5) & (df_user['frequency_score']==1),
     (df_user['recency_score']==4) & (df_user['frequency_score']==1),
     (df_user['recency_score']<=2) & (df_user['frequency_score'].between(3, 4)),
     (df_user['recency_score']==3) & (df_user['frequency_score']<=2),
     (df_user['recency_score']<=2) & (df_user['frequency_score']<=2)],
    ['01-Champion', '02-Loyal Customers', '03-Potential Loyalists', "04-Can't Lose Them", '05-Need Attention',
     '06-New Customers', '07-Promising', '08-At Risk', '09-About to Sleep', '10-Hibernating']
)
df_user
```

	customer_id	order_cnt	max_order_date	total_order_value
0	12346.0	5	2010-10-04	602.40
1	12608.0	1	2010-10-31	415.79
2	12745.0	2	2010-08-10	723.85
3	12746.0	2	2010-06-30	266.35

Tampilkan summary dari RFM segmentation (poin 8) berupa banyaknya pengguna, rata-rata dan median dari total order, total order value, dan jumlah hari sejak order terakhir

```
summary = pd.pivot_table(df_user, index='segment',
               values=['customer_id','day_since_last_order','order_cnt','total_order_value'],
               aggfunc={'customer_id': pd.Series.nunique,
                        'day_since_last_order': [np.mean, np.median],
                        'order_cnt': [np.mean, np.median],
                        'total_order_value': [np.mean, np.median]})
summary['pct_unique'] = (summary['customer_id'] / summary['customer_id'].sum() * 100).round(1)
summary
```

	customer_id	day_since_last_order		order_cnt	
	nunique	mean	median	mean	median
segment					
01-Champion	550	10.618182	9.5	15.467273	10.0
02-Loyal Customers	546	40.864469	37.0	8.767399	7.0
03-Potential Loyalists	523	23.573614	24.0	2.829828	3.0
04-Can't Lose Them	64	121.984375	112.5	11.375000	9.5

Kesimpulan analisis

Pengguna paling banyak berada pada segmen Hibernating (1060 atau 27.3%), Champion (550 atau 14.1%) dan loyal Customer(546 atau 14.0%).

Program khusus yang fokus pada urgensi bertransaksi untuk Loyal Customers(546 atau 14.0%) dapat dibuat untuk membuat mereka bertransaksi kembali dalam waktu dekat sehingga bisa naik ke segmen Champion

Program khusus yang fokus pada jumlah transaksi untuk Potential Loyalists (52 atau 13.4%) dapat dibuat untuk membuat mereka lebih sering bertransaksi sehingga bisa naik ke segmen Champion

Program khusus untuk Hibernating (1060 atau 27.3%) dapat dibuat untuk membuat mereka kembali bertransaksi walaupun belum begitu sering sehingga bisa naik ke segmen new Customers atau bagkan potential loyalists