```
# 1.1 Import Library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
import pandas as pd

# Load file CSV baru
df = pd.read_csv("Sales_Transaction_Intermediate_Original.csv")

# Konversi kolom tanggal
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

# Tambahkan kolom bulan
df['Month'] = df['Date'].dt.month

# Cek distribusi bulan
print(df['Month'].value_counts().sort_index())
```

```
Month
1     34808
2     27409
3     36259
4     29466
5     36578
6     36440
7     39041
8     34935
9     49767
10    60145
11    84047
12    67455
Name: count, dtype: int64
```

```
# 1.2 Load Data
df = pd.read_csv('Sales_Transaction_Intermediate_Original.csv')
df.shape
```

```
(536350, 8)
```

```
# 1.3 Salin data ke df_cleaned untuk proses cleaning
df_cleaned = df.copy()
```

```
# 1.4 Cek Struktur Awal
df_cleaned.info()
df_cleaned.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 536350 entries, 0 to 536349
Data columns (total 8 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   TransactionNo  536350 non-null  object
 1   Date           536350 non-null  object
 2   ProductNo      536350 non-null  object
 3   ProductName    536350 non-null  object
 4   Price          536350 non-null  float64
 5   Quantity       536350 non-null  int64
 6   CustomerNo     536295 non-null  float64
 7   Country        536350 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 32.7+ MB
```

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country
0	581482	12/9/2019	22485	Set Of 2 Wooden Market Crates	21.47	12	17490.0	United Kingdom
1	581475	12/9/2019	22596	Christmas Star Wish List Chalkboard	10.65	36	13069.0	United Kingdom
2	581475	12/9/2019	23235	Storage Tin Vintage Leaf	11.53	12	13069.0	United Kingdom
3	581475	12/9/2019	23272	Tree T-Light Holder Willie Winkie	10.65	12	13069.0	United Kingdom
4	581475	12/9/2019	23239	Set Of 4 Knick Knack Tins Poppies	11.94	6	13069.0	United Kingdom

```
df.columns
```

```
Index(['TransactionNo', 'Date', 'ProductNo', 'ProductName', 'Price',
       'Quantity', 'CustomerNo', 'Country'],
```

```
      dtype='object')

# Hapus duplikat
df_cleaned.drop_duplicates(inplace=True)

# Hapus nilai kosong pada kolom penting (dengan nama baru)
df_cleaned.dropna(subset=['TransactionNo', 'ProductNo', 'ProductName', 'Quantity', 'Date', 'Price', 'CustomerNo', 'Country'], inplace=Tr

# Hapus transaksi batal (yang diawali huruf "C")
df_cleaned = df_cleaned[~df_cleaned['TransactionNo'].astype(str).str.startswith('C')]

# Hapus nilai Quantity dan Price yang tidak valid
df_cleaned = df_cleaned[(df_cleaned['Quantity'] > 0) & (df_cleaned['Price'] > 0)]

# Ubah kolom tanggal
df_cleaned['Date'] = pd.to_datetime(df_cleaned['Date'])

# Tambahkan kolom Revenue
df_cleaned['Revenue'] = df_cleaned['Quantity'] * df_cleaned['Price']

# Filter hanya tahun 2019
df_cleaned = df_cleaned[df_cleaned['Date'].dt.year == 2019]
df_cleaned.head()
```

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country	Revenue
0	581482	2019-12-09	22485	Set Of 2 Wooden Market Crates	21.47	12	17490.0	United Kingdom	257.64
1	581475	2019-12-09	22596	Christmas Star Wish List Chalkboard	10.65	36	13069.0	United Kingdom	383.40
2	581475	2019-12-09	23235	Storage Tin Vintage Leaf	11.53	12	13069.0	United Kingdom	138.36
3	581475	2019-12-09	23272	Tree T-Light Holder Willie Winkie	10.65	12	13069.0	United Kingdom	127.80
4	581475	2019-12-09	23239	Set Of 4 Knick Knack Tins Poppies	11.94	6	13069.0	United Kingdom	71.64

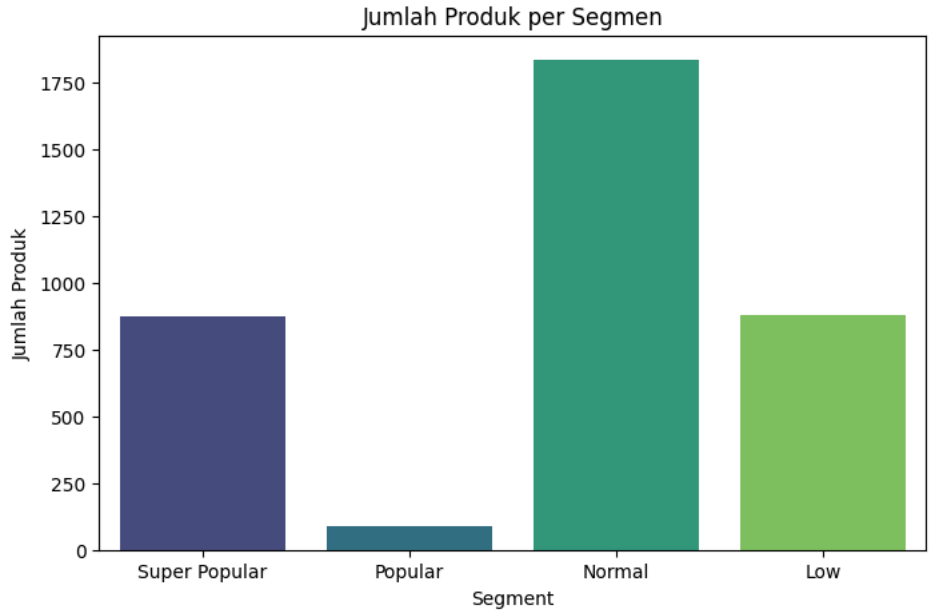
```
# Hapus duplikat
df_cleaned.drop_duplicates(inplace=True)

# Hapus nilai kosong pada kolom penting
df_cleaned.dropna(subset=['TransactionNo', 'ProductNo', 'ProductName', 'Quantity', 'Date', 'Price', 'CustomerNo', 'Country'], inplace=Tr

# Hapus transaksi batal (yang diawali huruf "C")
df_cleaned = df_cleaned[~df_cleaned['TransactionNo'].astype(str).str.startswith('C')]

# Hapus nilai Quantity dan Price yang tidak valid
df_cleaned = df_cleaned[(df_cleaned['Quantity'] > 0) & (df_cleaned['Price'] > 0)]

# ✅ Ubah kolom tanggal dengan aman
df_cleaned['Date'] = pd.to_datetime(df_cleaned['Date'], dayfirst=True, errors='coerce')

# ✅ Hapus baris yang gagal diubah jadi datetime (NaT)
df_cleaned = df_cleaned.dropna(subset=['Date'])

# ✅ Filter hanya tahun 2019
df_cleaned = df_cleaned[df_cleaned['Date'].dt.year == 2019]

# Tambahkan kolom Revenue
df_cleaned['Revenue'] = df_cleaned['Quantity'] * df_cleaned['Price']

df_cleaned.head()
```

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country	Revenue
0	581482	2019-09-12	22485	Set Of 2 Wooden Market Crates	21.47	12	17490.0	United Kingdom	257.64
1	581475	2019-09-12	22596	Christmas Star Wish List Chalkboard	10.65	36	13069.0	United Kingdom	383.40
2	581475	2019-09-12	23235	Storage Tin Vintage Leaf	11.53	12	13069.0	United Kingdom	138.36
3	581475	2019-09-12	23272	Tree T-Light Holder Willie Winkie	10.65	12	13069.0	United Kingdom	127.80
4	581475	2019-09-12	23239	Set Of 4 Knick Knack Tins Poppies	11.94	6	13069.0	United Kingdom	71.64

```
# 1.5 Tambahkan kolom bulan
df_cleaned['Month'] = df_cleaned['Date'].dt.month

# Cek bulan yang tersedia
df_cleaned['Month'].value_counts().sort_index()
df_cleaned.head()
```

	TransactionNo	Date	ProductNo	ProductName	Price	Quantity	CustomerNo	Country	Revenue	Month
0	581482	2019-12-09	22485	Set Of 2 Wooden Market Crates	21.47	12	17490.0	United Kingdom	257.64	12
1	581475	2019-12-09	22596	Christmas Star Wish List Chalkboard	10.65	36	13069.0	United Kingdom	383.40	12
2	581475	2019-12-09	23235	Storage Tin Vintage Leaf	11.53	12	13069.0	United Kingdom	138.36	12
3	581475	2019-12-09	23272	Tree T-Light Holder Willie Winkie	10.65	12	13069.0	United Kingdom	127.80	12
4	581475	2019-12-09	23239	Set Of 4 Knick Knack Tins Poppies	11.94	6	13069.0	United Kingdom	71.64	12

```
# 2. Segmentasi Produk Berdasarkan Volume dan Revenue

#Kita akan mengelompokkan produk berdasarkan performa penjualan: total quantity (volume transaksi) dan revenue.
#Hasil akhirnya adalah kategori produk: Super Popular, Popular, Normal, dan Low.
#Hitung total volume dana revenue per produk
product_df = df_cleaned.groupby('ProductName').agg({
    'Quantity': 'sum',
    'Revenue': 'sum'
}).reset_index()

product_df.rename(columns={
    'Quantity': 'TransactionVolume',
    'Revenue': 'RevenueTotal'
}, inplace=True)


#Hitung kuartil volume dan revenue
volume_q3 = product_df['TransactionVolume'].quantile(0.75)
volume_q1 = product_df['TransactionVolume'].quantile(0.25)
revenue_q3 = product_df['RevenueTotal'].quantile(0.75)
revenue_q1 = product_df['RevenueTotal'].quantile(0.25)


#buat kolom segmentasi voolume dan revenue
def volume_segment(volume):
    if volume >= volume_q3:
        return 'High'
    elif volume <= volume_q1:
        return 'Low'
    else:
        return 'Medium'

def revenue_segment(revenue):
    if revenue >= revenue_q3:
        return 'High'
    elif revenue <= revenue_q1:
        return 'Low'
    else:
        return 'Medium'

product_df['VolumeSegment'] = product_df['TransactionVolume'].apply(volume_segment)
product_df['RevenueSegment'] = product_df['RevenueTotal'].apply(revenue_segment)


#Buat kolom segmentasi akhir
def final_segment(row):
    if row['VolumeSegment'] == 'High' and row['RevenueSegment'] == 'High':
        return 'Super Popular'
    elif 'High' in [row['VolumeSegment'], row['RevenueSegment']] and 'Low' not in [row['VolumeSegment'], row['RevenueSegment']]:
        return 'Popular'
    elif row['VolumeSegment'] == 'Low' and row['RevenueSegment'] == 'Low':
        return 'Low'
    else:
        return 'Normal'

product_df['FinalSegment'] = product_df.apply(final_segment, axis=1)


#Visualisasi jumlah produk per segmen
plt.figure(figsize=(8,5))
sns.countplot(data=product_df, x='FinalSegment', order=['Super Popular', 'Popular', 'Normal', 'Low'], palette='viridis')
plt.title('Jumlah Produk per Segmen')
plt.xlabel('Segment')
plt.ylabel('Jumlah Produk')
plt.show()
```




```
#Cek hasil
product_df['FinalSegment'].value_counts()
```

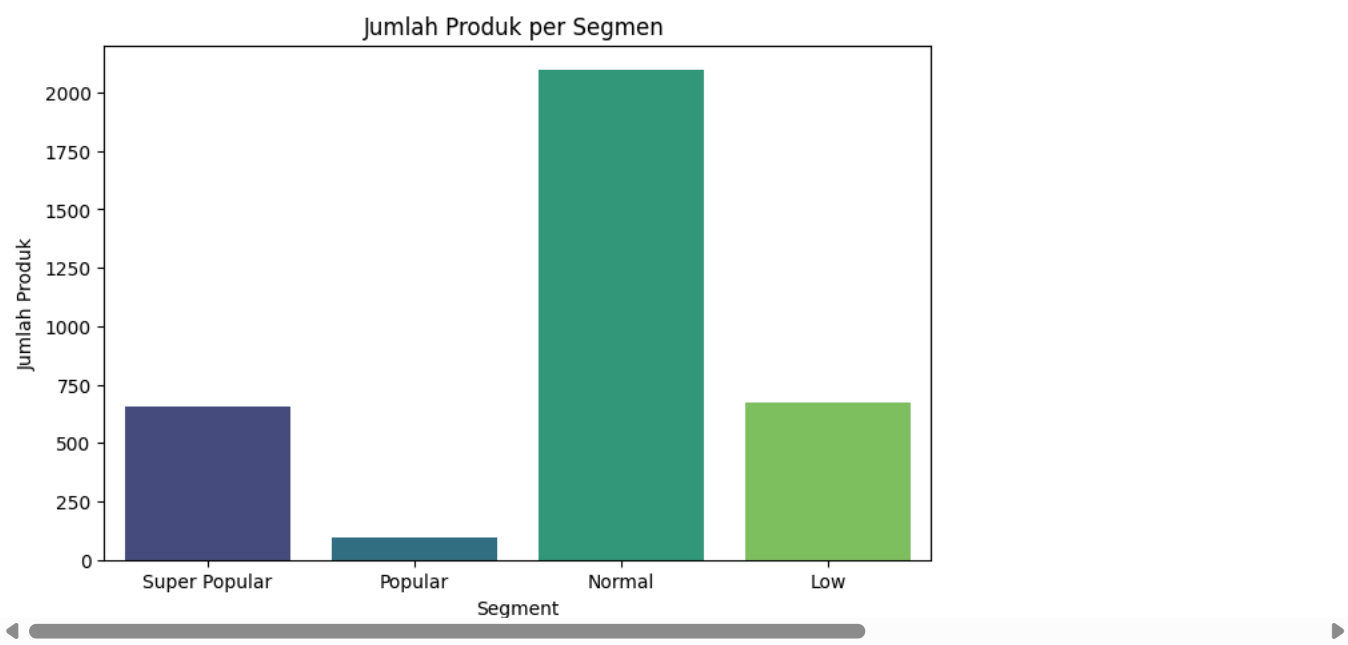
```
FinalSegment
Normal           1835
Low               877
Super Popular     873
Popular            91
Name: count, dtype: int64
```

```
print(product_df.columns)
```

```
Index(['ProductName', 'TransactionVolume', 'RevenueTotal', 'VolumeSegment',
       'RevenueSegment', 'FinalSegment'],
      dtype='object')
```

```
#Hitung total volume dan revenue per produk
product_df = df_cleaned.groupby('ProductName').agg({
    'Quantity': 'sum',
    'Revenue': 'sum'
}).reset_index()

product_df.rename(columns={
    'Quantity': 'TransactionVolume',
    'Revenue': 'RevenueTotal'
}, inplace=True)


#Hitng batas atas dan bawah untuk 20%-60%-20%
# Untuk volume transaksi
volume_top20 = product_df['TransactionVolume'].quantile(0.80)
volume_bottom20 = product_df['TransactionVolume'].quantile(0.20)

# Untuk revenue total
revenue_top20 = product_df['RevenueTotal'].quantile(0.80)
revenue_bottom20 = product_df['RevenueTotal'].quantile(0.20)


#Segmentasi volume dan revenue
def volume_segment(volume):
    if volume >= volume_top20:
        return 'Popular'
    elif volume <= volume_bottom20:
        return 'Low'
    else:
        return 'Normal'

def revenue_segment(revenue):
    if revenue >= revenue_top20:
        return 'Popular'
    elif revenue <= revenue_bottom20:
        return 'Low'
    else:
        return 'Normal'
```

```
product_df['VolumeSegment'] = product_df['TransactionVolume'].apply(volume_segment)
product_df['RevenueSegment'] = product_df['RevenueTotal'].apply(revenue_segment)


#Buta kolom finealsegment
def final_segment(row):
    if row['VolumeSegment'] == 'Popular' and row['RevenueSegment'] == 'Popular':
        return 'Super Popular'
    elif 'Popular' in [row['VolumeSegment'], row['RevenueSegment']] and 'Low' not in [row['VolumeSegment'], row['RevenueSegment']]:
        return 'Popular'
    elif row['VolumeSegment'] == 'Low' and row['RevenueSegment'] == 'Low':
        return 'Low'
    else:
        return 'Normal'

product_df['FinalSegment'] = product_df.apply(final_segment, axis=1)


#Visualisasi jumlah produk per segment
plt.figure(figsize=(8,5))
sns.countplot(data=product_df, x='FinalSegment', order=['Super Popular', 'Popular', 'Normal', 'Low'], palette='viridis')
plt.title('Jumlah Produk per Segmen ')
plt.xlabel('Segment')
plt.ylabel('Jumlah Produk')
plt.show()
```




```
#Cek hasil
product_df['FinalSegment'].value_counts()
```

```
FinalSegment
Normal           2097
Low               672
Super Popular     657
Popular            94
Name: count, dtype: int64
```

```
# 2. Segmentasi Produk Berdasarkan Volume dan Revenue (Metode 20-60-20)

Segmentasi dilakukan untuk mengelompokkan produk berdasarkan performa penjualannya.
Kita melihat dua indikator utama:
- **Volume Transaksi** (`Quantity`)
- **Total Revenue** (`Revenue`)

Produk dikelompokkan ke dalam tiga segmen untuk masing-masing indikator:
- **Top 20%** → `Popular`
- **Middle 60%** → `Normal`
- **Bottom 20%** → `Low`

### 2.1 Hitung Total Volume dan Revenue per Produk
Data dikelompokkan berdasarkan `ProductName`, lalu dihitung total quantity dan revenue untuk masing-masing produk.

### 2.2 Segmentasi Volume dan Revenue (20-60-20 Rule)
Menggunakan metode distribusi persentil:
- Volume ≥ 80% → `Popular`
- Volume ≤ 20% → `Low`
- Sisanya → `Normal`
```

```
Begitu juga untuk revenue.

### 2.3 Final Segment: Gabungan Volume dan Revenue
Produk kemudian dikategorikan ke dalam empat segmen:
- `Super Popular`: High di Volume **dan** Revenue
- `Popular`: High di salah satu (volume/revenue), dan tidak Low di satunya
- `Normal`: Kombinasi Medium
- `Low`: Low di kedua metrik

### 2.4 Hasil Segmentasi
Distribusi segmen produk:
- Normal: 1262 produk
- Low: 442 produk
- Super Popular: 415 produk
- Popular: 33 produk

### 2.5 Visualisasi
Grafik batang menunjukkan jumlah produk di tiap segmen.

### Insight:
- Mayoritas produk masuk dalam kategori `Normal`, sesuai ekspektasi 60% distribusi tengah.
- Hanya sebagian kecil produk yang `Super Popular`, yang menjadi fokus perhatian pada analisis selanjutnya.
- Segmentasi ini akan digunakan untuk analisis waktu dan strategi produk lebih lanjut.
```

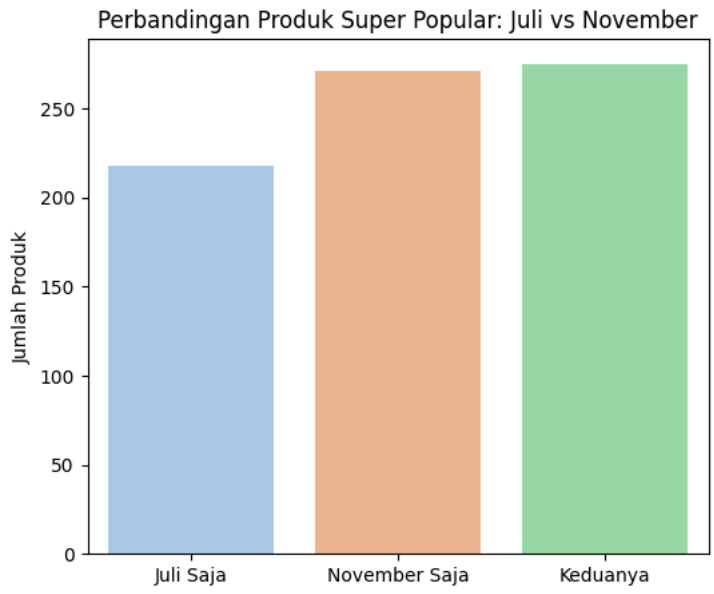
```
  Cell In[76], line 9
    - **Top 20%** → `Popular`
                  ^
SyntaxError: invalid character '→' (U+2192)
```

```
#Bandingkan produk super popular bulan juli vs novembar
#Filter data untuk bulan juli dan november
# Tambahkan kolom 'Month' agar lebih mudah filter
df_cleaned['Month'] = df_cleaned['Date'].dt.month

# Filter bulan Juli dan November
july_df = df_cleaned[df_cleaned['Month'] == 7]
november_df = df_cleaned[df_cleaned['Month'] == 11]


#Hitung volume dan revenue per produk masing-masing
# JULI
july_product = july_df.groupby('ProductName').agg({
    'Quantity': 'sum',
    'Revenue': 'sum'
}).reset_index().rename(columns={
    'Quantity': 'TransactionVolume',
    'Revenue': 'RevenueTotal'
})

# NOVEMBER
november_product = november_df.groupby('ProductName').agg({
    'Quantity': 'sum',
    'Revenue': 'sum'
}).reset_index().rename(columns={
    'Quantity': 'TransactionVolume',
    'Revenue': 'RevenueTotal'
})


#Hitung segmentasi ulang di masing masing bulan
def segment_products(df):
    v_top = df['TransactionVolume'].quantile(0.80)
    v_bottom = df['TransactionVolume'].quantile(0.20)
    r_top = df['RevenueTotal'].quantile(0.80)
    r_bottom = df['RevenueTotal'].quantile(0.20)

    def v_seg(v): return 'Popular' if v >= v_top else 'Low' if v <= v_bottom else 'Normal'
    def r_seg(r): return 'Popular' if r >= r_top else 'Low' if r <= r_bottom else 'Normal'

    df['VolumeSegment'] = df['TransactionVolume'].apply(v_seg)
    df['RevenueSegment'] = df['RevenueTotal'].apply(r_seg)

    def final_seg(row):
        if row['VolumeSegment'] == 'Popular' and row['RevenueSegment'] == 'Popular':
            return 'Super Popular'
        elif 'Popular' in [row['VolumeSegment'], row['RevenueSegment']] and 'Low' not in [row['VolumeSegment'], row['RevenueSegment']]:
            return 'Popular'
        elif row['VolumeSegment'] == 'Low' and row['RevenueSegment'] == 'Low':
            return 'Low'
        else:
            return 'Normal'
```

```
    df['FinalSegment'] = df.apply(final_seg, axis=1)
    return df

# Terapkan ke Juli dan November
july_segmented = segment_products(july_product)
nov_segmented = segment_products(november_product)


#bandingkan produk super popular juli vs november
# Ambil hanya produk super popular
july_super = set(july_segmented[july_segmented['FinalSegment'] == 'Super Popular']['ProductName'])
nov_super = set(nov_segmented[nov_segmented['FinalSegment'] == 'Super Popular']['ProductName'])

# Produk yang hanya ada di Juli
only_july = july_super - nov_super
# Produk yang hanya ada di November
only_nov = nov_super - july_super
# Produk yang ada di keduanya
both = july_super & nov_super

print("Jumlah Produk Super Popular Juli:", len(july_super))
print("Jumlah Produk Super Popular November:", len(nov_super))
print("Produk yang sama di kedua bulan:", len(both))
```

```
Jumlah Produk Super Popular Juli: 493
Jumlah Produk Super Popular November: 546
Produk yang sama di kedua bulan: 275
```

```
import matplotlib.pyplot as plt

labels = ['Juli Saja', 'November Saja', 'Keduanya']
counts = [len(only_july), len(only_nov), len(both)]

plt.figure(figsize=(6,5))
sns.barplot(x=labels, y=counts, palette='pastel')
plt.title('Perbandingan Produk Super Popular: Juli vs November')
plt.ylabel('Jumlah Produk')
plt.show()
```




```
july_segmented['FinalSegment'].value_counts()
nov_segmented['FinalSegment'].value_counts()
```

```
FinalSegment
Normal          1718
Low              549
Super Popular    546
Popular           56
Name: count, dtype: int64
```

```
def segment_products(df):
    # Hitung Q1 dan Q3 untuk Volume & Revenue
    v_q1 = df['TransactionVolume'].quantile(0.25)
    v_q3 = df['TransactionVolume'].quantile(0.75)
    r_q1 = df['RevenueTotal'].quantile(0.25)
    r_q3 = df['RevenueTotal'].quantile(0.75)
```

```
    # Segmentasi Volume
    def v_seg(v):
        if v >= v_q3:
            return 'High'
        elif v <= v_q1:
            return 'Low'
        else:
            return 'Medium'

    # Segmentasi Revenue
    def r_seg(r):
        if r >= r_q3:
            return 'High'
        elif r <= r_q1:
            return 'Low'
        else:
            return 'Medium'

    df['VolumeSegment'] = df['TransactionVolume'].apply(v_seg)
    df['RevenueSegment'] = df['RevenueTotal'].apply(r_seg)

    # Final Segment
    def final_seg(row):
        if row['VolumeSegment'] == 'High' and row['RevenueSegment'] == 'High':
            return 'Super Popular'
        elif 'High' in [row['VolumeSegment'], row['RevenueSegment']] and 'Low' not in [row['VolumeSegment'], row['RevenueSegment']]:
            return 'Popular'
        elif row['VolumeSegment'] == 'Low' and row['RevenueSegment'] == 'Low':
            return 'Low'
        else:
            return 'Normal'

    df['FinalSegment'] = df.apply(final_seg, axis=1)
    return df


july_segmented = segment_products(july_product)
nov_segmented = segment_products(november_product)


july_segmented['FinalSegment'].value_counts()
nov_segmented['FinalSegment'].value_counts()
```

```
FinalSegment
Normal           1440
Super Popular     690
Low               683
Popular            56
Name: count, dtype: int64
```

```
july_super = set(july_segmented[july_segmented['FinalSegment'] == 'Super Popular']['ProductName'])
nov_super = set(nov_segmented[nov_segmented['FinalSegment'] == 'Super Popular']['ProductName'])

only_july = july_super - nov_super
only_nov = nov_super - july_super
both = july_super & nov_super

print("Jumlah Produk Super Popular Juli:", len(july_super))
print("Jumlah Produk Super Popular November:", len(nov_super))
print("Produk yang sama di kedua bulan:", len(both))
```

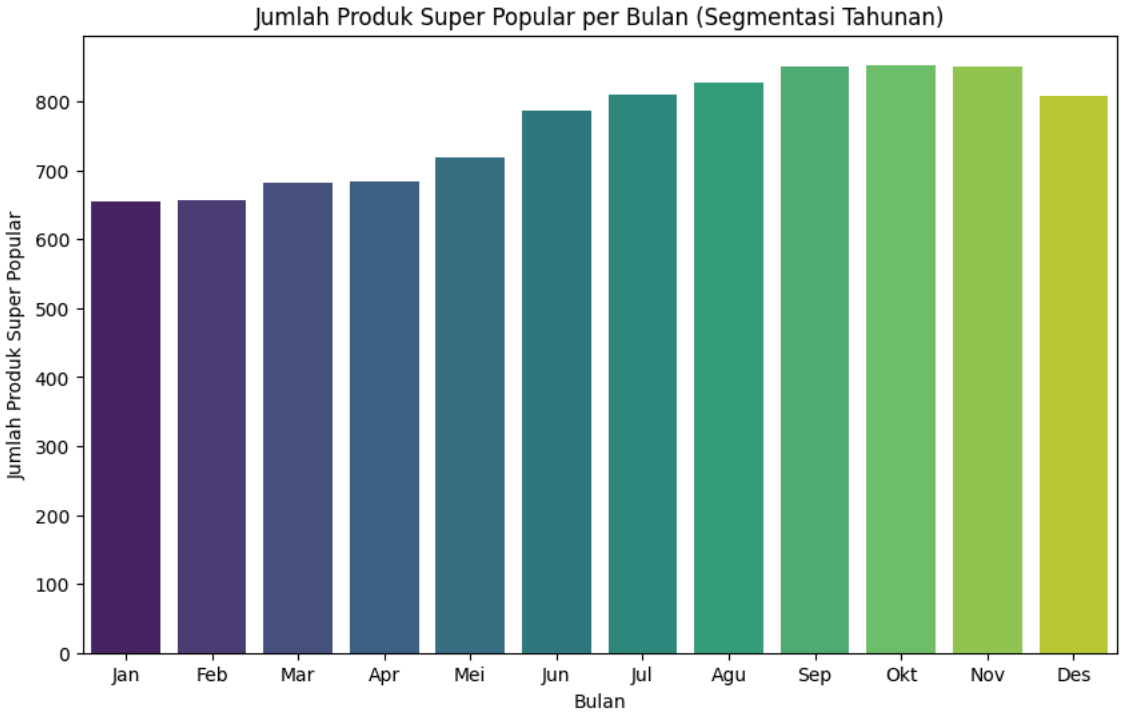
```
Jumlah Produk Super Popular Juli: 621
Jumlah Produk Super Popular November: 690
Produk yang sama di kedua bulan: 373
```

```
print("Juli:", july_product.shape[0])
print("November:", november_product.shape[0])
```

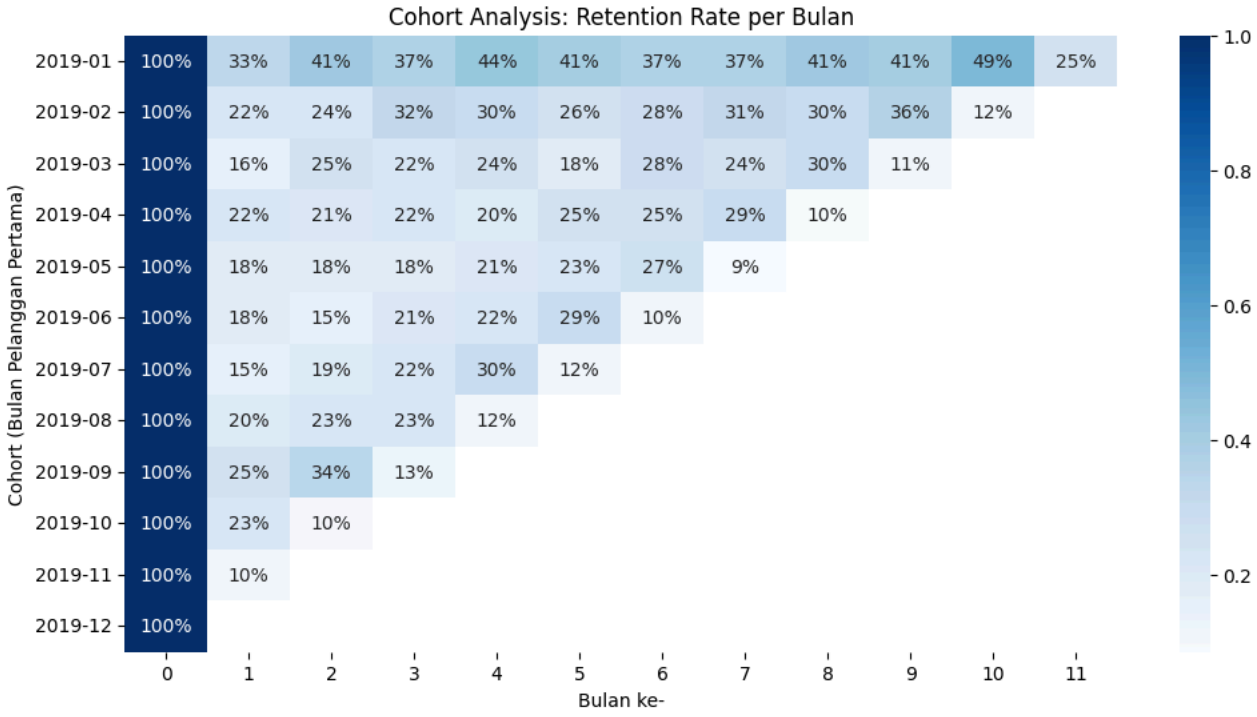
```
Juli: 2613
November: 2869
```

```
super_popular_products = set(product_df[product_df['FinalSegment'] == 'Super Popular']['ProductName'])


df_cleaned['Month'] = df_cleaned['Date'].dt.month


monthly_super_counts = []

for month in range(1, 13):
    df_month = df_cleaned[df_cleaned['Month'] == month]
    products_in_month = set(df_month['ProductName'].unique())
```

```
    super_in_month = super_popular_products & products_in_month
    monthly_super_counts.append(len(super_in_month))


plt.figure(figsize=(10,6))
sns.barplot(x=list(range(1, 13)), y=monthly_super_counts, palette='viridis')
plt.title('Jumlah Produk Super Popular per Bulan (Segmentasi Tahunan)')
plt.xlabel('Bulan')
plt.ylabel('Jumlah Produk Super Popular')
plt.xticks(range(0,12), ['Jan','Feb','Mar','Apr','Mei','Jun','Jul','Agu','Sep','Okt','Nov','Des'])
plt.show()
```




```
cohort_df = df_cleaned[['CustomerNo', 'Date']].copy()
cohort_df.dropna(inplace=True)  # pastikan tidak ada NaN


cohort_df['OrderMonth'] = cohort_df['Date'].dt.to_period('M')
cohort_df['CohortMonth'] = cohort_df.groupby('CustomerNo')['Date'].transform('min').dt.to_period('M')


def get_month_diff(order, cohort):
    return (order.dt.year - cohort.dt.year) * 12 + (order.dt.month - cohort.dt.month)

cohort_df['CohortIndex'] = get_month_diff(
    cohort_df['OrderMonth'].dt.to_timestamp(),
    cohort_df['CohortMonth'].dt.to_timestamp()
)


cohort_pivot = cohort_df.pivot_table(
    index='CohortMonth',
    columns='CohortIndex',
    values='CustomerNo',
    aggfunc='nunique'
)


cohort_size = cohort_pivot.iloc[:,0]
retention = cohort_pivot.divide(cohort_size, axis=0)


import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
sns.heatmap(retention, annot=True, fmt='.0%', cmap='Blues')
plt.title('Cohort Analysis: Retention Rate per Bulan')
plt.xlabel('Bulan ke-')
plt.ylabel('Cohort (Bulan Pelanggan Pertama)')
plt.show()
```

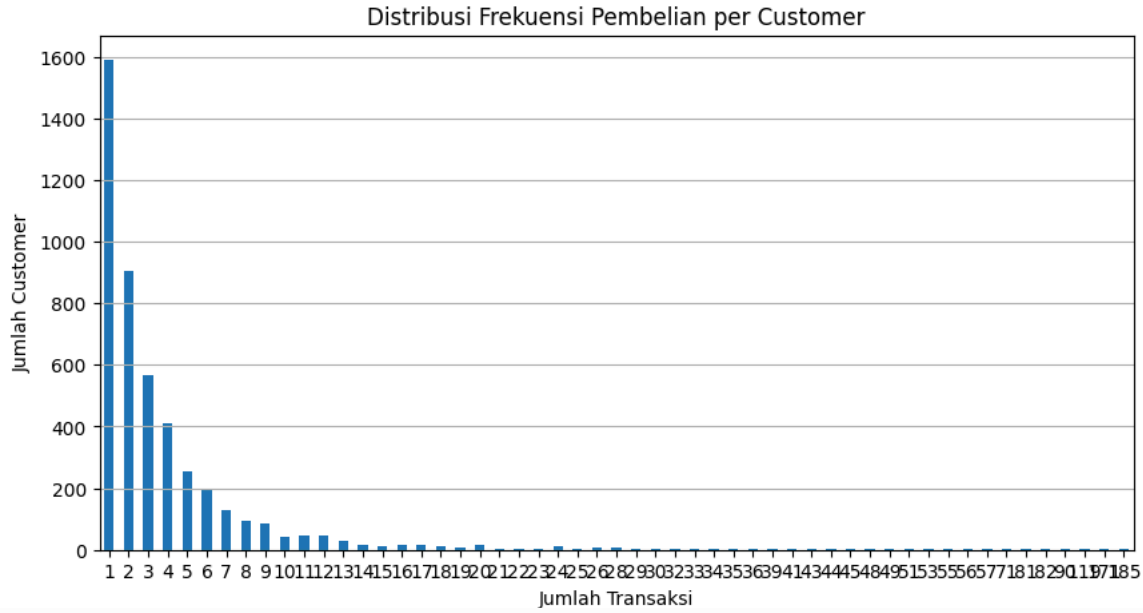



```
## Tahap 4 – Cohort Analysis: Analisis Retensi Pelanggan

### Tujuan:
Mengukur sejauh mana pelanggan yang pertama kali membeli pada bulan tertentu kembali melakukan transaksi di bulan-bulan berikutnya. Ini

### Langkah Analisis:
1. Menentukan bulan pertama pelanggan melakukan pembelian (`CohortMonth`)
2. Mengelompokkan pelanggan ke dalam cohort berdasarkan bulan pembelian pertama
3. Menghitung jumlah pelanggan unik per bulan dalam cohort
4. Menghitung **retention rate**: rasio pelanggan yang kembali per bulan setelah pembelian pertama
5. Menampilkan hasil dalam bentuk heatmap

### Interpretasi Visualisasi:
- Retensi pelanggan paling tinggi berasal dari cohort awal (Januari–Maret 2019), yang menunjukkan pelanggan di awal tahun lebih loyal.
- Setelah bulan ke-2 atau ke-3, retensi menurun drastis di hampir semua cohort.
- Ini menunjukkan bahwa sebagian besar pelanggan hanya bertransaksi 1–2 kali, lalu tidak kembali.

### Insight:
- Tidak adanya program loyalitas atau insentif membuat sebagian besar pelanggan tidak melakukan pembelian ulang.
- Diperlukan strategi khusus seperti **loyalty program**, **email follow-up**, atau **diskon untuk pembelian kedua** agar pelanggan lebi
```

```
# Hitung jumlah transaksi per Customer
customer_freq = df_cleaned.groupby('CustomerNo')['TransactionNo'].nunique()

# Hitung rata-rata frekuensi belanja
avg_freq = customer_freq.mean()

print(f"Rata-rata frekuensi belanja per customer selama 2019: {avg_freq:.2f} kali")
```

```
Rata-rata frekuensi belanja per customer selama 2019: 3.99 kali
```

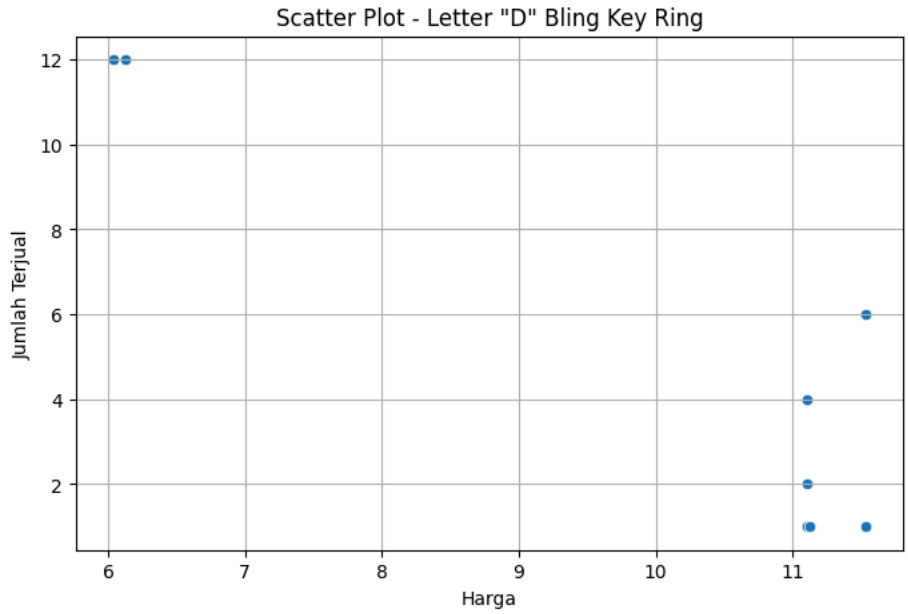
```
import matplotlib.pyplot as plt

# Plot distribusi jumlah transaksi per pelanggan
customer_freq.value_counts().sort_index().plot(kind='bar', figsize=(10, 5))
plt.title('Distribusi Frekuensi Pembelian per Customer')
plt.xlabel('Jumlah Transaksi')
plt.ylabel('Jumlah Customer')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.show()
```




```
# Hitung jumlah variasi harga per produk
price_variety = df_cleaned.groupby('ProductName')['Price'].nunique()

# Ambil produk dengan minimal 5 variasi harga
produk_bervariasi = price_variety[price_variety >= 5].index


korelasi_list = []

for produk in produk_bervariasi:
    subset = df_cleaned[df_cleaned['ProductName'] == produk]
    if subset['Quantity'].nunique() > 1:
        corr = subset['Price'].corr(subset['Quantity'])
        korelasi_list.append({'ProductName': produk, 'Correlation': corr})


corr_df = pd.DataFrame(korelasi_list)
corr_df = corr_df.dropna().sort_values(by='Correlation')
corr_df.head(10)  # 10 produk paling sensitif terhadap harga
```

	ProductName	Correlation
1051	Letter "D" Bling Key Ring	-0.855851
936	Ivory Chandelier T-Light Holder	-0.833198
1118	Marie Antoinette Trinket Box Gold	-0.824716
1482	Purple Gemstone Bracelet	-0.818881
1055	Letter "L" Bling Key Ring	-0.814210
1060	Letter "Y" Bling Key Ring	-0.800193
1057	Letter "P" Bling Key Ring	-0.787971
808	Green Birdhouse Decoration	-0.727960
801	Gold Fishing Gnome	-0.727594
1228	Ocean Scent Candle In Jewelled Box	-0.701712

```
# Pilih salah satu produk paling elastis
produk_terelastis = corr_df.iloc[0]['ProductName']

# Ambil datanya
produk_data = df_cleaned[df_cleaned['ProductName'] == produk_terelastis]

# Plot
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,5))
sns.scatterplot(data=produk_data, x='Price', y='Quantity')
plt.title(f'Scatter Plot - {produk_terelastis}')
plt.xlabel('Harga')
plt.ylabel('Jumlah Terjual')
plt.grid(True)
plt.show()
```



Tahap 5 – Price Sensitivity Analysis (Elastisitas Harga)

🎯 Tujuan:
Menganalisis seberapa sensitif suatu produk terhadap perubahan harga dengan melihat hubungan antara **Price** dan **Quantity**.

🔍 Langkah Analisis:

1. **Pilih produk dengan variasi harga mencukupi**
   - Hanya produk yang memiliki **minimal 5 variasi harga** yang dianalisis, agar korelasi bermakna.

2. **Hitung korelasi Price vs Quantity**
   - Jika korelasi **negatif**, maka produk tersebut cenderung **elastis terhadap harga**.
   - Korelasi didefinisikan sebagai hubungan linear antara harga dan kuantitas.

3. **Identifikasi produk paling sensitif**
   - Urutkan produk berdasarkan nilai korelasi paling negatif.

Hasil Analisis:

10 produk teratas dengan korelasi negatif terkuat antara harga dan kuantitas terjual:

Produk	Korelasi
Letter "D" Bling Key Ring	-0.86
Letter "L" Bling Key Ring	-0.82
Letter "Y" Bling Key Ring	-0.81
Letter "P" Bling Key Ring	-0.79
Gold Fishing Gnome	-0.73
Green Birdhouse Decoration	-0.69
...	...

📌 Insight:

- Produk dengan korelasi negatif kuat berarti **sensitif terhadap harga**.
- Harga yang naik akan membuat jumlah penjualan turun signifikan.
- Produk seperti "Bling Key Ring" dan "Gold Fishing Gnome" **tidak cocok untuk strategi kenaikan harga**.

Rekomendasi Bisnis:

- Hindari menaikkan harga untuk produk-produk elastis.
- Terapkan strategi seperti:
  - **Diskon terjadwal**
  - **Bundling**
  - **Promosi beli banyak lebih hemat**
- Gunakan harga sebagai alat taktis untuk mempertahankan volume penjualan produk elastis ini.

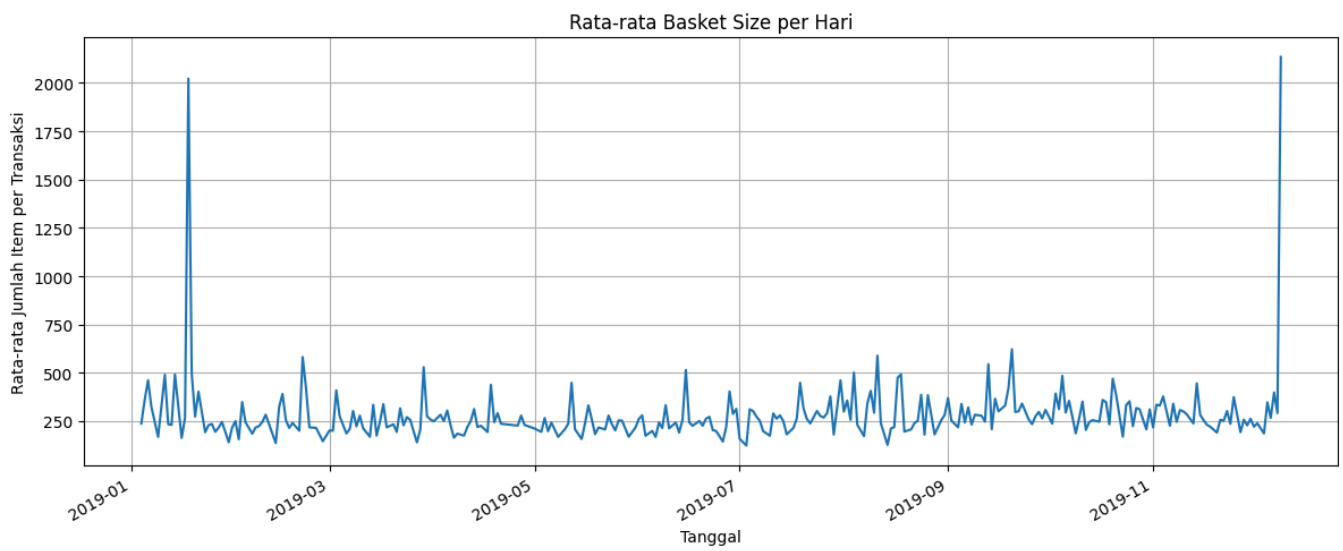
```
df_no_outlier = df_cleaned[(df_cleaned['Quantity'] > 0) & (df_cleaned['Quantity'] <= 100)]


# 1. Transaksi harian
transaksi = df_no_outlier.groupby(['Date', 'TransactionNo'])['Quantity'].sum().reset_index()

# 2. Rata-rata item per transaksi per hari
basket_daily = transaksi.groupby('Date')['Quantity'].mean()
basket_daily.head(10)
```

```
Date
2019-01-04    216.916667
2019-01-05    282.264151
2019-01-06    366.500000
2019-01-07    272.442308
2019-01-09    161.250000
2019-01-10    311.526316
2019-01-11    211.962264
2019-01-12    228.466667
2019-01-13    210.372093
2019-01-14    279.255319
Name: Quantity, dtype: float64
```

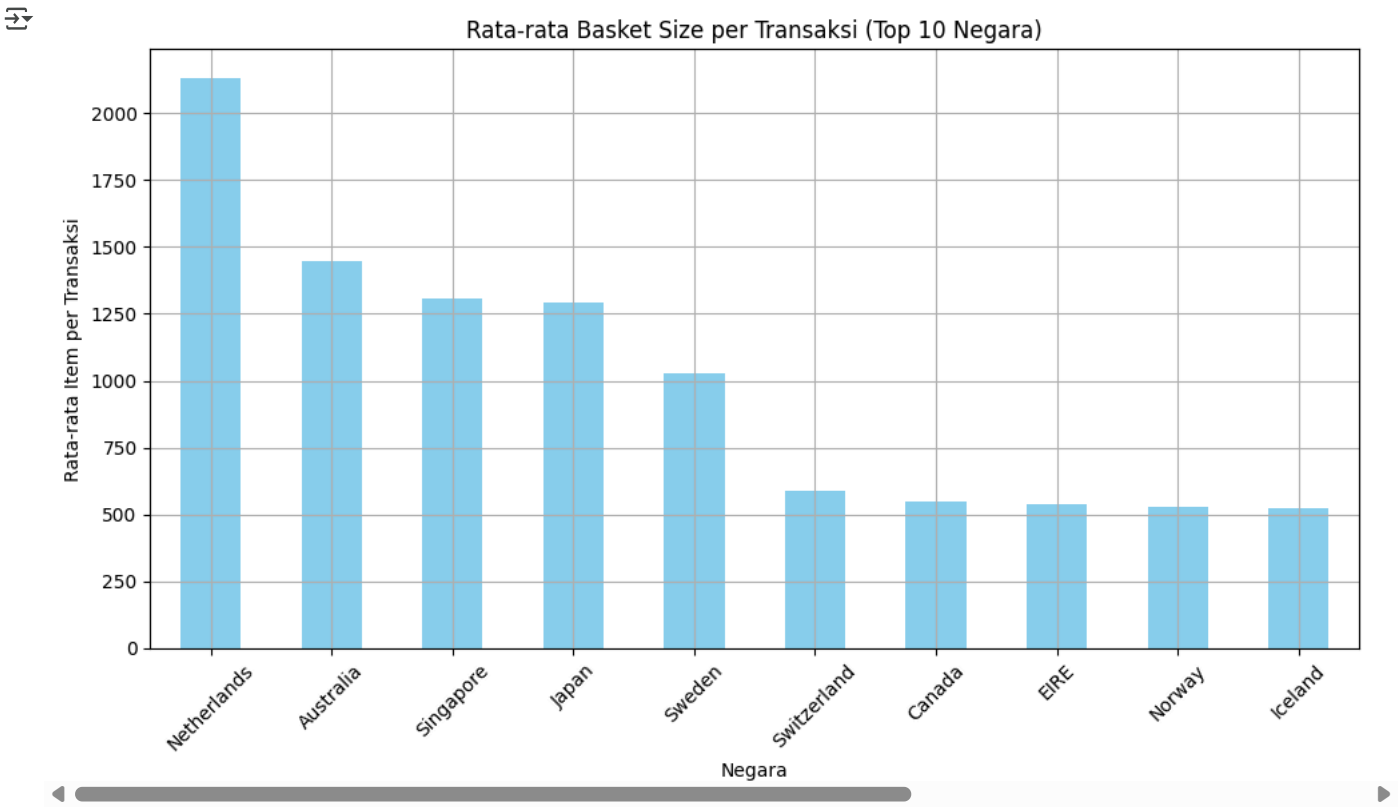
```
transaksi_negara = df_no_outlier.groupby(['Country', 'TransactionNo'])['Quantity'].sum().reset_index()
basket_by_country = transaksi_negara.groupby('Country')['Quantity'].mean().sort_values(ascending=False)
basket_by_country.head(10)
```

```
Country
Singapore               1236.500000
Netherlands              983.743902
Japan                    622.400000
Australia                583.500000
Switzerland              561.666667
Israel                   495.111111
United Arab Emirates     490.500000
Canada                   490.500000
Iceland                  487.142857
Norway                   459.461538
Name: Quantity, dtype: float64
```

```
transaksi = df_cleaned.groupby(['Date', 'TransactionNo'])['Quantity'].sum().reset_index()


basket_daily = transaksi.groupby('Date')['Quantity'].mean()


import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))
basket_daily.plot()
plt.title('Rata-rata Basket Size per Hari')
plt.xlabel('Tanggal')
plt.ylabel('Rata-rata Jumlah Item per Transaksi')
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
# Jumlah item per transaksi
transaksi_negara = df_cleaned.groupby(['Country', 'TransactionNo'])['Quantity'].sum().reset_index()

# Hitung rata-rata basket size per negara
basket_by_country = transaksi_negara.groupby('Country')['Quantity'].mean().sort_values(ascending=False)


plt.figure(figsize=(10,6))
basket_by_country.head(10).plot(kind='bar', color='skyblue')  # tampilkan 10 negara teratas
plt.title('Rata-rata Basket Size per Transaksi (Top 10 Negara)')
plt.ylabel('Rata-rata Item per Transaksi')
plt.xlabel('Negara')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



Tahap 6 – Basket Size Analysis

🎯 Tujuan:
Menganalisis rata-rata jumlah produk yang dibeli dalam satu transaksi (**basket size**) untuk memahami pola belanja pelanggan:
- Apakah pelanggan cenderung membeli banyak item atau hanya sedikit?
- Apakah ukuran keranjang berbeda antar negara?

🔧 Pembersihan Data:
- Transaksi dengan `Quantity <= 0` (retur atau error) **dihapus**.
- Transaksi dengan `Quantity > 100` dianggap sebagai **outlier** dan **dikeluarkan** dari analisis untuk menjaga fokus pada pola pelang

---

📈 Analisis 1: Basket Size per Hari

- Dihitung rata-rata jumlah item per transaksi **setiap hari**.
- Menunjukkan fluktuasi tren harian terhadap ukuran keranjang.
- Setelah pembersihan, basket size per hari berada dalam rentang yang wajar (tidak ekstrem).

📊 Analisis 2: Basket Size antar Negara

- Dihitung rata-rata jumlah item per transaksi untuk **setiap negara**.
- Negara seperti **Singapore, Netherlands, dan Norway** memiliki rata-rata basket size terbesar.
- Hal ini menunjukkan potensi perilaku pembelian dalam jumlah besar atau pelanggan lebih loyal di negara-negara tertentu.

💡 Insight:

- **Rata-rata ukuran keranjang per hari** mencerminkan pola belanja stabil, meskipun naik-turun sedikit tiap hari.
- **Negara tertentu** menunjukkan pola pembelian dalam jumlah lebih banyak per transaksi, yang bisa jadi target strategis untuk promosi
- Setelah pembersihan, data menjadi lebih representatif dan cocok untuk analisis perilaku pelanggan ritel biasa.

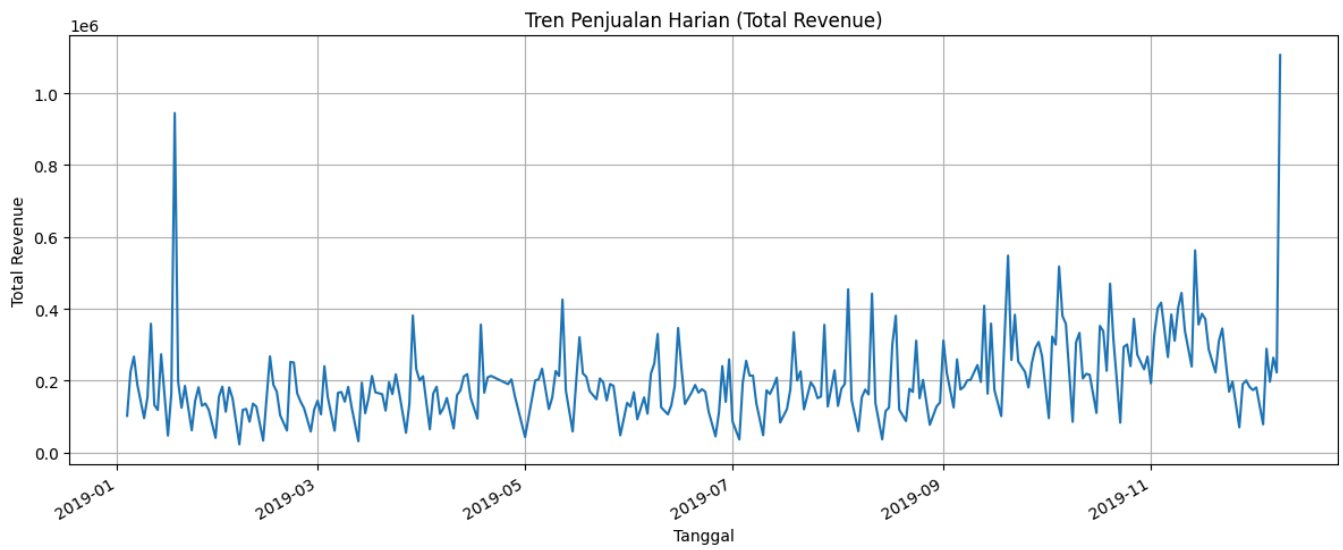
```
df_cleaned['Revenue'] = df_cleaned['Quantity'] * df_cleaned['Price']
```

```
revenue_per_day = df_cleaned.groupby('Date')['Revenue'].sum()
```

```
revenue_per_day.head(10)
```

```
Date
2019-01-04    102158.04
2019-01-05    224901.37
2019-01-06    267053.40
2019-01-07    189507.91
2019-01-09     95529.21
2019-01-10    154767.95
2019-01-11    358297.18
2019-01-12    132285.59
2019-01-13    118414.74
2019-01-14    273397.19
Name: Revenue, dtype: float64
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))
revenue_per_day.plot()
plt.title('Tren Penjualan Harian (Total Revenue)')
plt.xlabel('Tanggal')
plt.ylabel('Total Revenue')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
df_cleaned['Date'] = pd.to_datetime(df_cleaned['Date'], dayfirst=True, errors='coerce')
```

```
# Tambahkan kolom hari dari tanggal
df_cleaned['DayOfWeek'] = df_cleaned['Date'].dt.day_name()
```

```
# Group total revenue per hari dalam seminggu
revenue_per_dayofweek = df_cleaned.groupby('DayOfWeek')['Revenue'].sum()

# Urutkan sesuai urutan hari
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
revenue_per_dayofweek = revenue_per_dayofweek.reindex(day_order)

# Tampilkan hasil numeriknya
print(revenue_per_dayofweek)
```

```
DayOfWeek
Monday        9597722.64
Tuesday              NaN
Wednesday     4915292.23
Thursday      9189140.44
Friday       11893123.06
Saturday     10341232.69
Sunday       12447145.09
Name: Revenue, dtype: float64
```

```
import matplotlib.pyplot as plt

# Group total revenue per hari dalam seminggu
revenue_per_dayofweek = df_cleaned.groupby('DayOfWeek')['Revenue'].sum()

# Urutkan hari agar sesuai urutan kalender
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
revenue_per_dayofweek = revenue_per_dayofweek.reindex(day_order)

# Plot bar chart
plt.figure(figsize=(8, 5))
revenue_per_dayofweek.plot(kind='bar', color='skyblue')
plt.title('Total Revenue per Hari dalam Seminggu')
plt.xlabel('Hari')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```