Statprob24_FP

December 19, 2024

1 Table of Content

2 Import Dataset

[6]:
```
!pip install -q kaggle
```

[7]:
```
!mkdir ~/.kaggle
```

[8]:
```
!cp kaggle.json ~/.kaggle
```

```
cp: cannot stat 'kaggle.json': No such file or directory
```

[9]:
```
!chmod 600 ~/.kaggle/kaggle.json
```

```
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
```

[10]:
```
!kaggle datasets list
```

```
Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 5, in <module>
    from kaggle.cli import main
  File "/usr/local/lib/python3.10/dist-packages/kaggle/__init__.py", line 7, in
<module>
    api.authenticate()
  File "/usr/local/lib/python3.10/dist-
packages/kaggle/api/kaggle_api_extended.py", line 407, in authenticate
    raise IOError('Could not find {}. Make sure it\'s located in'
OSError: Could not find kaggle.json. Make sure it's located in /root/.kaggle. Or
use the environment method. See setup instructions at
https://github.com/Kaggle/kaggle-api/
```

[11]:
```
!kaggle datasets download -d 'muhdaniyal/supermarket-sales-cleaned-dataset'
```

```
Dataset URL: https://www.kaggle.com/datasets/muhdaniyal/supermarket-sales-
cleaned-dataset
License(s): apache-2.0
Downloading supermarket-sales-cleaned-dataset.zip to /content
  0% 0.00/38.7k [00:00<?, ?B/s]
100% 38.7k/38.7k [00:00<00:00, 34.9MB/s]
```

[12]:
```
import zipfile
dataset_zip = zipfile.ZipFile('/content/supermarket-sales-cleaned-dataset.zip',␣
 ↪'r')

dataset_zip.extractall()

dataset_zip.close()
```

[13]:
```
import pandas as pd
```

[14]:
```
# fungsi untuk display table
def show_df(df, row=5):
  print(df.head(row))
  print("\n")

  rows, columns = df.shape
```

```
  print(f"Jumlah baris: {rows}")
  print(f"Jumlah kolom: {columns}")
  print(f"Kolom: {df.columns.tolist()}")
```

[15]:
```
df_supermarket = pd.read_csv("/content/Supermarket Sales Cleaned.csv",␣
 ↪delimiter=",", encoding='latin1', on_bad_lines='skip')
show_df(df_supermarket)
```

```
    Invoice ID Branch       City Customer type  Gender  \
0  750-67-8428      A     Yangon        Member  Female
1  226-31-3081      C  Naypyitaw        Normal  Female
2  631-41-3108      A     Yangon        Normal    Male
3  123-19-1176      A     Yangon        Member    Male
4  373-73-7910      A     Yangon        Normal    Male

             Product line  Unit price  Quantity   Tax 5%     Total  \
0       Health and beauty       74.69         7  26.1415  548.9715
1  Electronic accessories       15.28         5   3.8200   80.2200
2      Home and lifestyle       46.33         7  16.2155  340.5255
3       Health and beauty       58.22         8  23.2880  489.0480
4       Sports and travel       86.31         7  30.2085  634.3785

         Date   Time      Payment    cogs  gross margin percentage  \
0  2019-01-05  13:08      Ewallet  522.83                 4.761905
1  2019-03-08  10:29         Cash   76.40                 4.761905
2  2019-03-03  13:23  Credit card  324.31                 4.761905
3  2019-01-27  20:33      Ewallet  465.76                 4.761905
4  2019-02-08  10:37      Ewallet  604.17                 4.761905

   gross income  Rating  Day  Month  Year
0       26.1415     9.1    5      1  2019
1        3.8200     9.6    8      3  2019
2       16.2155     7.4    3      3  2019
3       23.2880     8.4   27      1  2019
4       30.2085     5.3    8      2  2019


Jumlah baris: 1000
Jumlah kolom: 20
Kolom: ['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender', 'Product
line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date', 'Time', 'Payment',
'cogs', 'gross margin percentage', 'gross income', 'Rating', 'Day', 'Month',
'Year']
```

2.1 Pre-Processing

[16]:
```
# Daftar kolom yang ingin dihapus
columns_to_drop = ['City', 'Customer type', 'Gender', 'cogs', 'gross margin␣
 ↪percentage', 'gross income']

# Menghapus kolom dari df_supermarket
df_supermarket = df_supermarket.drop(columns=columns_to_drop)

show_df(df_supermarket)
```

```
    Invoice ID Branch            Product line  Unit price  Quantity   Tax 5%  \
0  750-67-8428      A       Health and beauty       74.69         7  26.1415
1  226-31-3081      C  Electronic accessories       15.28         5   3.8200
2  631-41-3108      A      Home and lifestyle       46.33         7  16.2155
3  123-19-1176      A       Health and beauty       58.22         8  23.2880
4  373-73-7910      A       Sports and travel       86.31         7  30.2085

      Total        Date   Time      Payment  Rating  Day  Month  Year
0  548.9715  2019-01-05  13:08      Ewallet     9.1    5      1  2019
1   80.2200  2019-03-08  10:29         Cash     9.6    8      3  2019
2  340.5255  2019-03-03  13:23  Credit card     7.4    3      3  2019
3  489.0480  2019-01-27  20:33      Ewallet     8.4   27      1  2019
4  634.3785  2019-02-08  10:37      Ewallet     5.3    8      2  2019


Jumlah baris: 1000
Jumlah kolom: 14
Kolom: ['Invoice ID', 'Branch', 'Product line', 'Unit price', 'Quantity', 'Tax
5%', 'Total', 'Date', 'Time', 'Payment', 'Rating', 'Day', 'Month', 'Year']
```

[17]:
```
df_supermarket
```

[17]:
```
      Invoice ID Branch            Product line  Unit price  Quantity  \
0    750-67-8428      A       Health and beauty       74.69         7
1    226-31-3081      C  Electronic accessories       15.28         5
2    631-41-3108      A      Home and lifestyle       46.33         7
3    123-19-1176      A       Health and beauty       58.22         8
4    373-73-7910      A       Sports and travel       86.31         7
..           …    …                       …         …       …
995  233-67-5758      C       Health and beauty       40.35         1
996  303-96-2227      B      Home and lifestyle       97.38        10
997  727-02-1313      A      Food and beverages       31.84         1
998  347-56-2442      A      Home and lifestyle       65.82         1
999  849-09-3807      A     Fashion accessories       88.34         7

      Tax 5%      Total        Date   Time      Payment  Rating  Day  Month  \
```

```
0    26.1415   548.9715  2019-01-05  13:08      Ewallet     9.1    5      1
1     3.8200    80.2200  2019-03-08  10:29         Cash     9.6    8      3
2    16.2155   340.5255  2019-03-03  13:23  Credit card     7.4    3      3
3    23.2880   489.0480  2019-01-27  20:33      Ewallet     8.4   27      1
4    30.2085   634.3785  2019-02-08  10:37      Ewallet     5.3    8      2
..       …         …         …     …           …      …  …    …
995   2.0175    42.3675  2019-01-29  13:46      Ewallet     6.2   29      1
996  48.6900  1022.4900  2019-03-02  17:16      Ewallet     4.4    2      3
997   1.5920    33.4320  2019-02-09  13:22         Cash     7.7    9      2
998   3.2910    69.1110  2019-02-22  15:33         Cash     4.1   22      2
999  30.9190   649.2990  2019-02-18  13:28         Cash     6.6   18      2

     Year
0    2019
1    2019
2    2019
3    2019
4    2019
..    …
995  2019
996  2019
997  2019
998  2019
999  2019

[1000 rows x 14 columns]
```

3 Analisis Distribusi Data

[18]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

3.1 1. Analisis Distribusi Data Berdasarkan Tahun

1.1 Mengelompokkan Data Berdasarkan Tahun

[19]:
```
# Mengelompokkan data berdasarkan Tahun dan menghitung total penjualan per tahun
df_year_sales = df_supermarket.groupby('Year').agg(
    total_sales=('Total', 'sum'),  # Menghitung total penjualan per tahun
    average_rating=('Rating', 'mean')  # Menghitung rata-rata rating per tahun
).reset_index()

# Menampilkan hasilnya
show_df(df_year_sales)
```

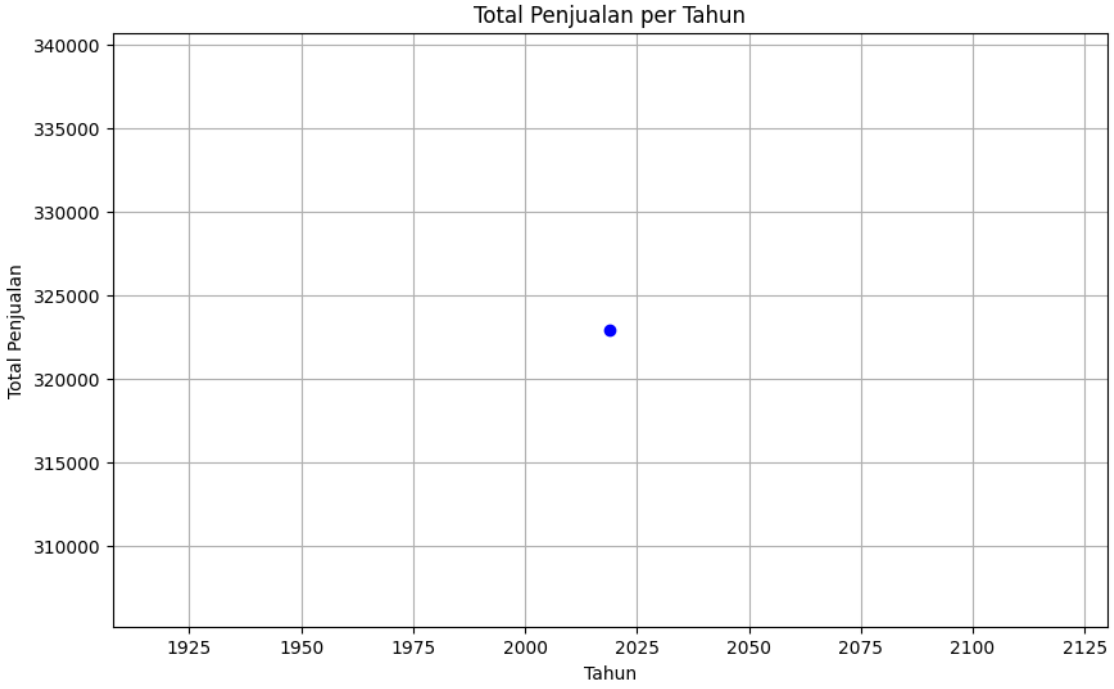
```
   Year  total_sales  average_rating
0  2019    322966.749          6.9727
```

```
Jumlah baris: 1
Jumlah kolom: 3
Kolom: ['Year', 'total_sales', 'average_rating']
```

- groupby('Year'): Mengelompokkan data berdasarkan kolom Year. agg(): Menggunakan agregasi untuk menghitung total penjualan (sum) dan rata-rata rating (mean) per tahun.
- reset_index(): Mengubah hasil agregasi menjadi dataframe kembali.

1.2 Menganalisis Total Penjualan per Tahun

[20]:
```
# Plot total penjualan per tahun
plt.figure(figsize=(10, 6))
plt.plot(df_year_sales['Year'], df_year_sales['total_sales'], marker='o',␣
 ↪color='b')
plt.title('Total Penjualan per Tahun')
plt.xlabel('Tahun')
plt.ylabel('Total Penjualan')
plt.grid(True)
plt.show()
```



Visualisasi ini akan menunjukkan bagaimana total penjualan berubah dari tahun ke tahun. Ini berguna untuk melihat tren peningkatan atau penurunan penjualan dari waktu ke waktu.

1.3 Menghitung Total Penjualan Berdasarkan Tahun dan Cabang

[21]:
```
# Mengelompokkan data berdasarkan Tahun dan Cabang
df_year_branch_sales = df_supermarket.groupby(['Year', 'Branch']).agg(
    total_sales=('Total', 'sum')
).reset_index()

# Menampilkan hasilnya
show_df(df_year_branch_sales)
```

```
   Year Branch  total_sales
0  2019      A  106200.3705
1  2019      B  106197.6720
2  2019      C  110568.7065


Jumlah baris: 3
Jumlah kolom: 3
Kolom: ['Year', 'Branch', 'total_sales']
```

- groupby([‘Year’, ‘Branch’]): Mengelompokkan data berdasarkan tahun dan cabang.
- Agregasi dilakukan dengan menghitung total penjualan per kombinasi tahun dan cabang.

1.4 Analisis Rating per Tahun

[22]:
```
# Mengelompokkan data berdasarkan Tahun dan menghitung rata-rata Rating
df_year_rating = df_supermarket.groupby('Year').agg(
    average_rating=('Rating', 'mean')
).reset_index()

# Menampilkan hasilnya
show_df(df_year_rating)
```

```
   Year  average_rating
0  2019          6.9727


Jumlah baris: 1
Jumlah kolom: 2
Kolom: ['Year', 'average_rating']
```

1. Analisis Total Penjualan per Tahun Total Penjualan untuk tahun 2019 adalah 322,966.749. Nilai ini menggambarkan total keseluruhan penjualan yang tercatat pada tahun tersebut. Visualisasi Tren Total Penjualan per Tahun Dalam visualisasi yang dibuat dengan plot garis, total penjualan per tahun akan menunjukkan bagaimana penjualan berkembang dari tahun ke tahun. Namun, karena hanya ada data untuk satu tahun (2019), visualisasi ini tidak menunjukkan tren peningkatan atau penurunan karena hanya satu titik data yang tersedia.
2. Analisis Penjualan Berdasarkan Cabang Pengelompokan Berdasarkan Tahun dan Cabang: Tabel ini mengelompokkan total penjualan per cabang pada tahun 2019:

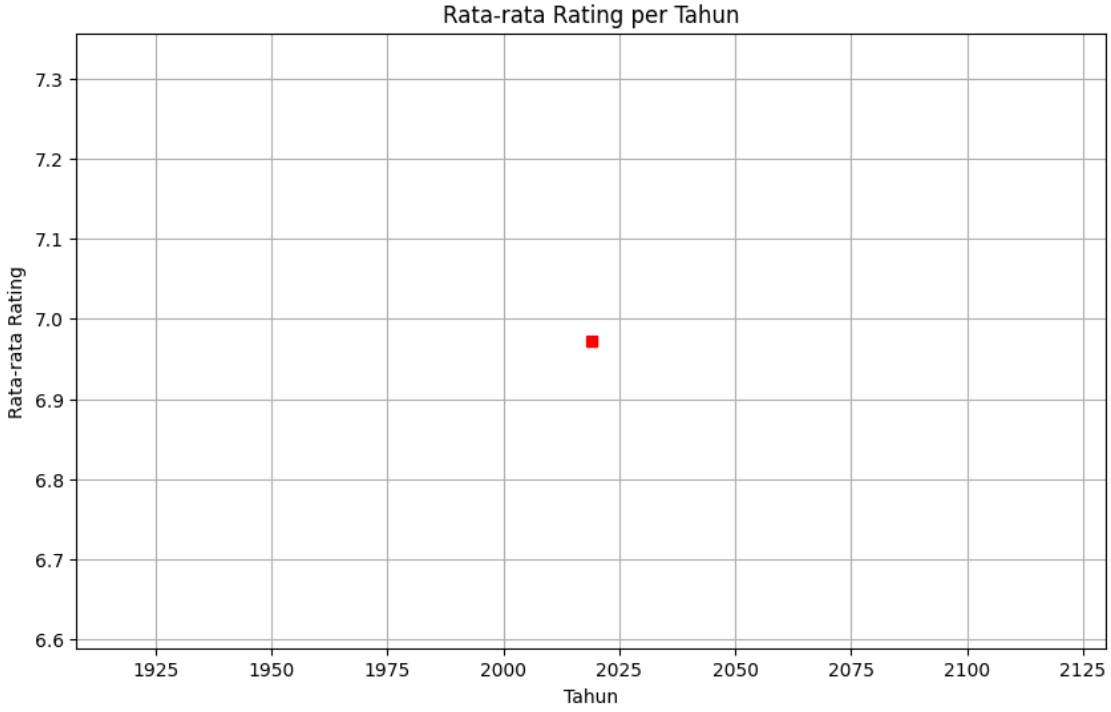
Cabang A: 106,200.37 Cabang B: 106,197.67 Cabang C: 110,568.71 Analisis:

Distribusi Penjualan Antara Cabang: Total penjualan untuk ketiga cabang hampir sama, dengan Cabang C sedikit lebih tinggi dibandingkan Cabang A dan Cabang B. Keseimbangan Penjualan: Penjualan antar cabang cukup seimbang, dengan selisih yang tidak terlalu besar antara cabang satu dengan lainnya. Ini menunjukkan bahwa tidak ada cabang yang jauh lebih dominan dari yang lainnya dalam hal kontribusi penjualan.

3. Analisis Rata-rata Rating per Tahun Rata-rata Rating untuk tahun 2019 adalah 6.9727. Ini menunjukkan penilaian rata-rata yang diterima oleh produk atau layanan pada tahun tersebut. Rating ini memberi gambaran umum tentang kepuasan pelanggan, meskipun rating 6.97 masih di bawah skala 7, yang menunjukkan adanya ruang untuk perbaikan.

1.5 Visualisasi Rating per Tahun

[23]:
```
# Plot rata-rata rating per tahun
plt.figure(figsize=(10, 6))
plt.plot(df_year_rating['Year'], df_year_rating['average_rating'], marker='s',␣
 ↪color='r')
plt.title('Rata-rata Rating per Tahun')
plt.xlabel('Tahun')
plt.ylabel('Rata-rata Rating')
plt.grid(True)
plt.show()
```



Dari data diatas dapat disimpulkan bahwa data penjualan berdasarkan tahun hanya terjadi di 2019 dengan rata rata rating 6.9727 dan penjualan terbanyak di brand C

3.2 2. Analisis Distribusi Data Berdasarkan Bulan

[24]:
```
# Menambahkan kolom 'Month' dari kolom 'Date'
df_supermarket['Month'] = pd.to_datetime(df_supermarket['Date']).dt.month
```

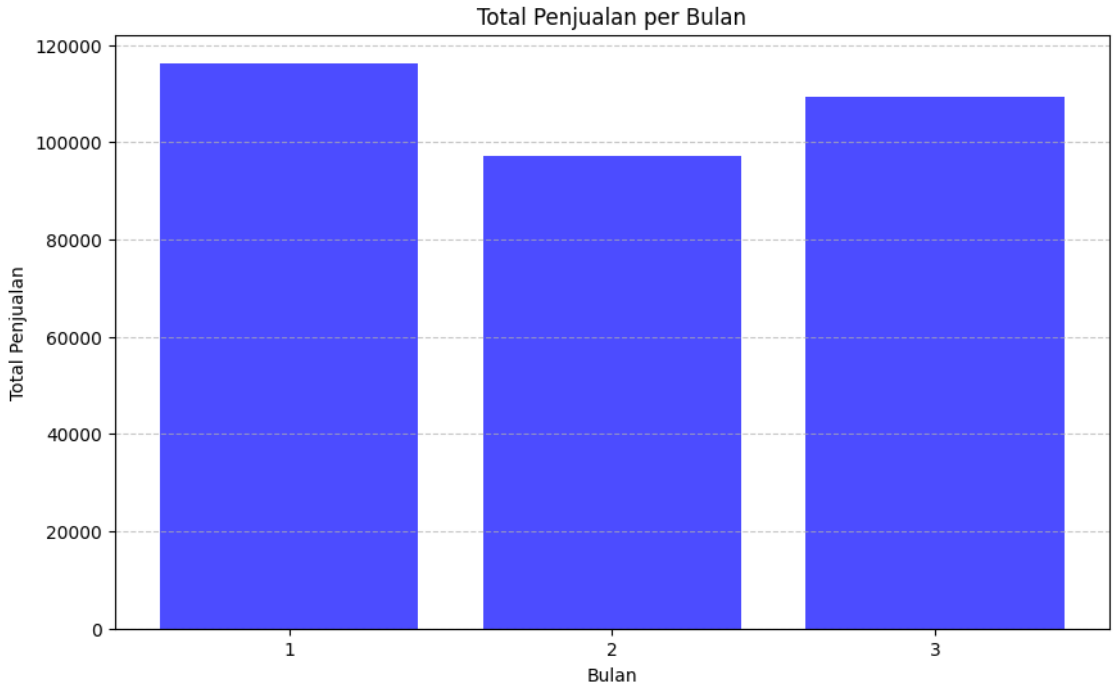
2.1 Mengelompokkan Data Berdasarkan Bulan

[25]:
```
df_month_sales = df_supermarket.groupby('Month').agg(
    total_sales=('Total', 'sum'),  # Total penjualan per bulan
    average_rating=('Rating', 'mean')  # Rata-rata rating per bulan
).reset_index()
```

- groupby('Month'): Mengelompokkan data berdasarkan kolom Month.
- agg(): Digunakan untuk melakukan agregasi pada data. Di sini kita menghitung: total_sales: Total penjualan setiap bulan. average_rating: Rata-rata rating per bulan.
- reset_index(): Mengembalikan indeks DataFrame setelah dilakukan pengelompokan.

2.2 Visualisasi Total Penjualan per Bulan

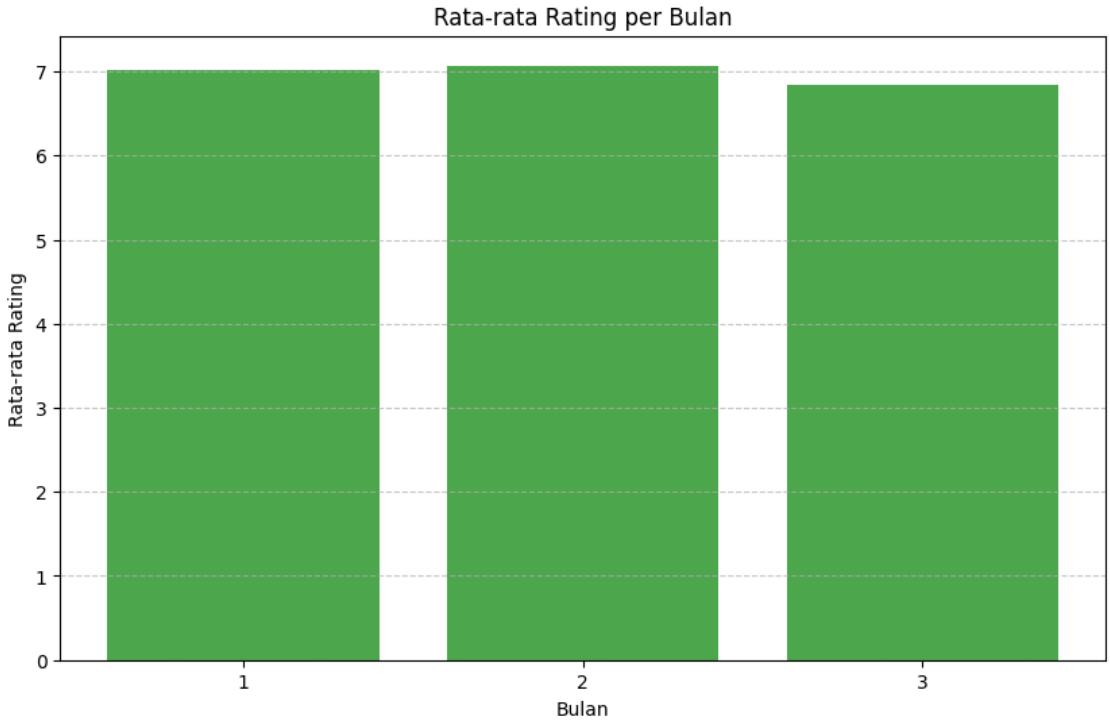
[26]:
```
plt.figure(figsize=(10, 6))
plt.bar(df_month_sales['Month'], df_month_sales['total_sales'], color='b',␣
↪alpha=0.7)
plt.title('Total Penjualan per Bulan')
plt.xlabel('Bulan')
plt.ylabel('Total Penjualan')
plt.xticks(df_month_sales['Month'])  # Menampilkan bulan pada sumbu x
plt.grid(True, axis='y', linestyle='--', alpha=0.7)
plt.show()
```



- plt.bar(): Membuat grafik batang untuk total penjualan.
- alpha=0.7: Memberikan transparansi pada batang agar lebih estetik.
- plt.xticks(): Menampilkan bulan sebagai label pada sumbu x.
- plt.grid(): Menambahkan garis bantu pada sumbu y agar lebih mudah membaca grafik.

2.3 Visualisasi Rata-rata Rating per Bulan

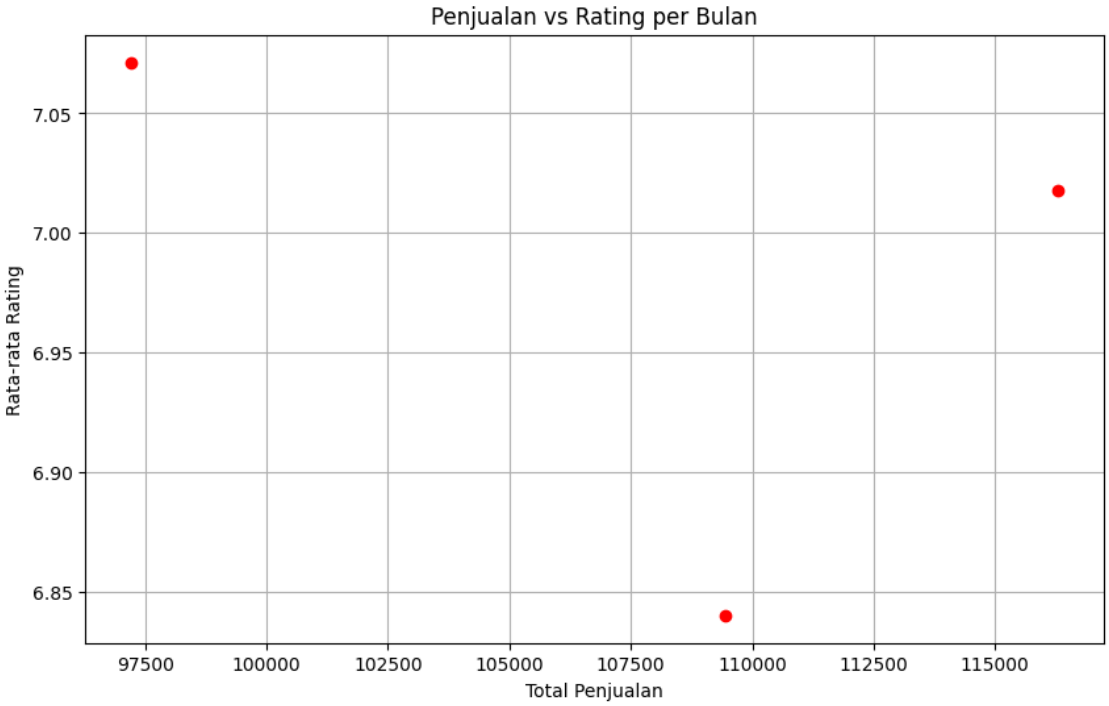
[27]:
```
# Grafik batang untuk rata-rata rating per bulan
plt.figure(figsize=(10, 6))
plt.bar(df_month_sales['Month'], df_month_sales['average_rating'], color='g',␣
 ↪alpha=0.7)
plt.title('Rata-rata Rating per Bulan')
plt.xlabel('Bulan')
plt.ylabel('Rata-rata Rating')
plt.xticks(df_month_sales['Month'])  # Menampilkan bulan pada sumbu x
plt.grid(True, axis='y', linestyle='--', alpha=0.7)
plt.show()
```



- Menampilkan rata-rata rating per bulan untuk melihat apakah ada perubahan kualitas produk dari bulan ke bulan.

2.4 Analisis Penjualan

[28]:
```
# Scatter plot antara total penjualan dan rating per bulan
plt.figure(figsize=(10, 6))
plt.scatter(df_month_sales['total_sales'], df_month_sales['average_rating'],␣
 ↪color='r')
plt.title('Penjualan vs Rating per Bulan')
plt.xlabel('Total Penjualan')
plt.ylabel('Rata-rata Rating')
plt.grid(True)
plt.show()
```



Scatter Plot: Ini membantu melihat apakah ada pola atau hubungan langsung antara total penjualan dan rata-rata rating.

- pd.to_datetime(df_supermarket[‘Date’]): Mengonversi kolom tanggal menjadi format tanggal yang bisa diproses lebih lanjut.
- .dt.day: Mengambil informasi hari dari tanggal yang sudah diubah.

3.3 3. Analisis Berdasarkan Total Penjualan

[29]:
```
# Deskripsi statistik untuk total penjualan
total_stats = df_supermarket['Total'].describe()
print(total_stats)
```

```
count    1000.000000
mean      322.966749
```

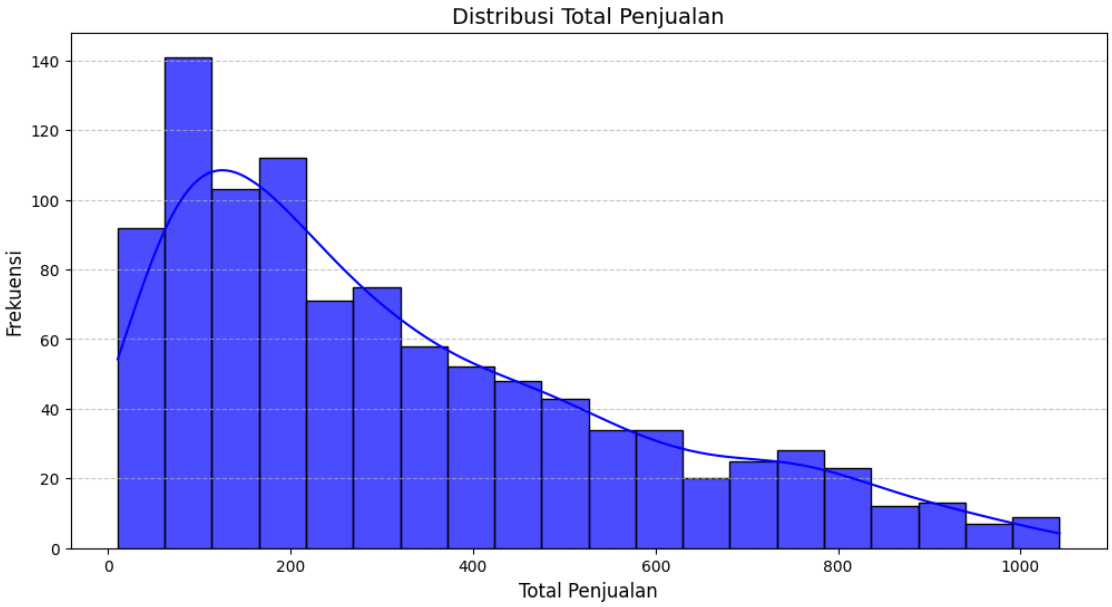
```
std        245.885335
min         10.678500
25%        124.422375
50%        253.848000
75%        471.350250
max       1042.650000
Name: Total, dtype: float64
```

Hasil dari describe() akan memberikan metrik penting, seperti:

- Count: Jumlah transaksi.
- Mean: Rata-rata total penjualan.
- Std: Standar deviasi (keragaman data).
- Min: Total penjualan terkecil.
- 25%, 50%, 75%: Kuartil, yang menunjukkan distribusi data.
- Max: Total penjualan terbesar.

3.1 Visualisasi Distribusi Total Penjualan

[30]:
```
# Histogram distribusi total penjualan
plt.figure(figsize=(12, 6))
sns.histplot(df_supermarket['Total'], bins=20, kde=True, color='blue', alpha=0.
↪7)
plt.title('Distribusi Total Penjualan', fontsize=14)
plt.xlabel('Total Penjualan', fontsize=12)
plt.ylabel('Frekuensi', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



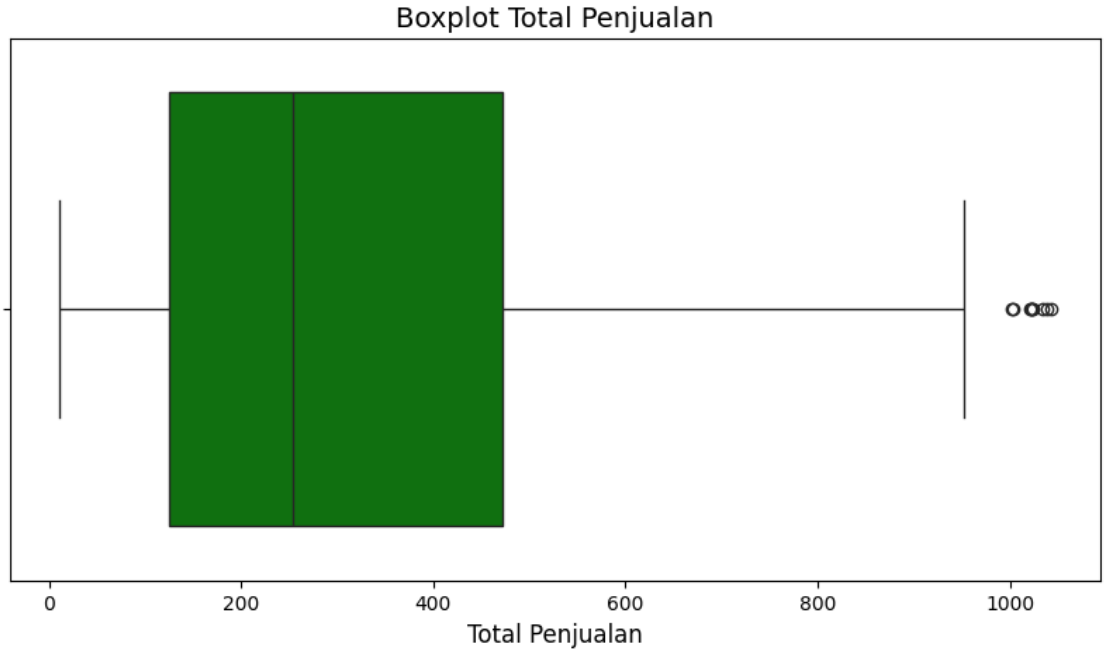
Penjelasan:

- Histogram: Membagi data total penjualan menjadi beberapa "bin" untuk melihat frekuensi setiap rentang.
- KDE (Kernel Density Estimation): Menunjukkan garis halus yang mewakili distribusi data.

Analisis Visual:

- Rentang nilai penjualan yang sering terjadi (modus).
- Identifikasi apakah distribusi normal, miring ke kanan/kiri, atau ada outlier.

[31]:
```
# Boxplot untuk total penjualan
plt.figure(figsize=(10, 5))
sns.boxplot(x=df_supermarket['Total'], color='green')
plt.title('Boxplot Total Penjualan', fontsize=14)
plt.xlabel('Total Penjualan', fontsize=12)
plt.show()
```



Penjelasan:

- Boxplot: Menampilkan nilai minimum, kuartil, median, maksimum, dan outlier.
- Outlier: Titik yang berada di luar "whisker" (batas normal data).

Analisis Visual:

- Identifikasi transaksi dengan penjualan sangat tinggi atau sangat rendah.
- Median (nilai tengah) dibandingkan dengan rata-rata (dari describe()).

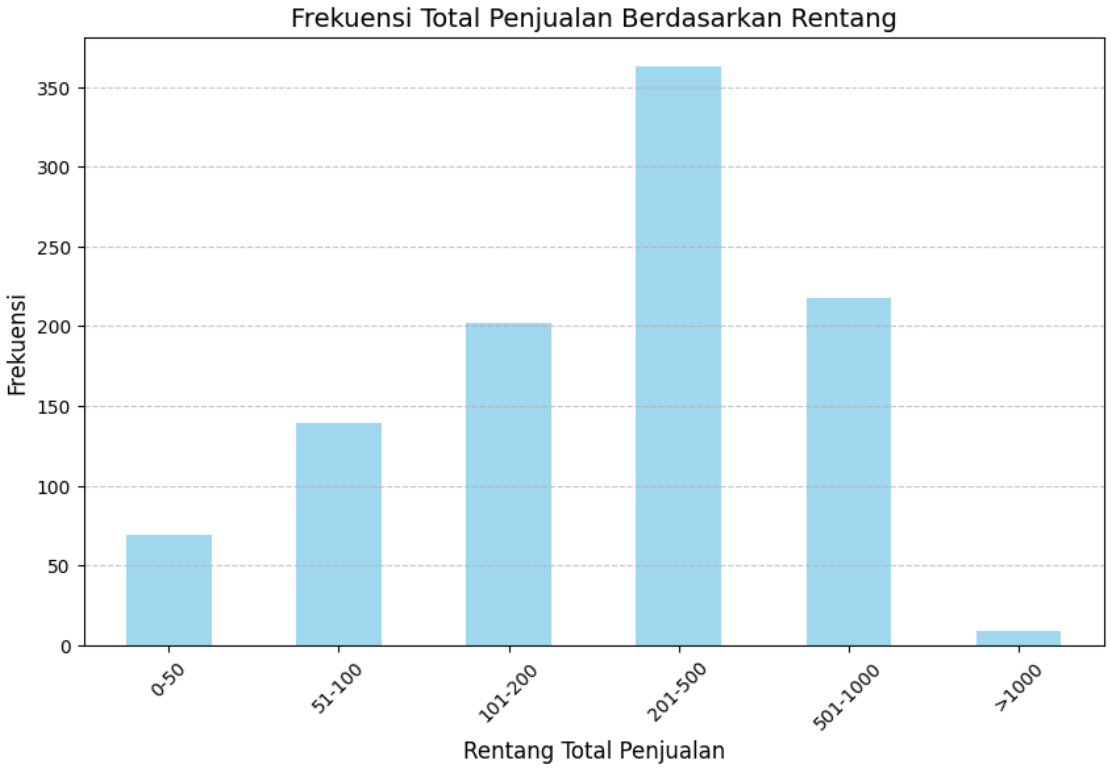
3.2 Pengelompokan Berdasarkan Rentang Total Penjualan

[32]:
```
# Membuat kategori berdasarkan rentang penjualan
bins = [0, 50, 100, 200, 500, 1000, df_supermarket['Total'].max()]
labels = ['0-50', '51-100', '101-200', '201-500', '501-1000', '>1000']
df_supermarket['Total_Range'] = pd.cut(df_supermarket['Total'], bins=bins,␣
 ↪labels=labels)

# Menghitung frekuensi setiap kategori
range_sales = df_supermarket['Total_Range'].value_counts().sort_index()
print(range_sales)
```

```
Total_Range
0-50          69
51-100       139
101-200      202
201-500      363
501-1000     218
>1000          9
Name: count, dtype: int64
```

- pd.cut: Membagi total penjualan ke dalam rentang tertentu.
- value_counts(): Menghitung jumlah transaksi di setiap rentang.

[33]:
```
# Bar plot untuk frekuensi penjualan berdasarkan rentang
plt.figure(figsize=(10, 6))
range_sales.plot(kind='bar', color='skyblue', alpha=0.8)
plt.title('Frekuensi Total Penjualan Berdasarkan Rentang', fontsize=14)
plt.xlabel('Rentang Total Penjualan', fontsize=12)
plt.ylabel('Frekuensi', fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

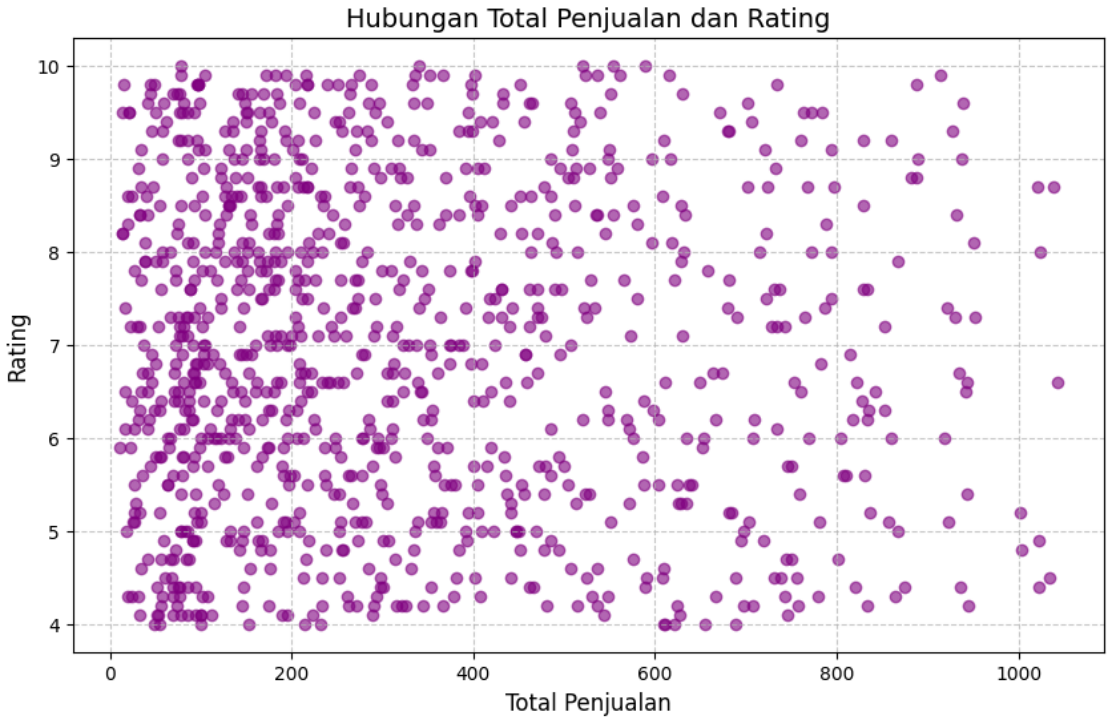


Analisis Visual:

- Rentang penjualan dengan frekuensi tertinggi.
- Apakah lebih banyak transaksi dengan nilai kecil

3.3 Hubungan Total Penjualan dengan Variabel Lain

[34]:
```
# Scatter plot antara total penjualan dan rating
plt.figure(figsize=(10, 6))
plt.scatter(df_supermarket['Total'], df_supermarket['Rating'], alpha=0.6,␣
 ↪color='purple')
plt.title('Hubungan Total Penjualan dan Rating', fontsize=14)
plt.xlabel('Total Penjualan', fontsize=12)
plt.ylabel('Rating', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```



Analisis Visual:

- Apakah ada korelasi antara total penjualan dan rating?
- Tren apakah transaksi dengan penjualan tinggi mendapatkan rating lebih baik.

1. Statistik Deskriptif Total Penjualan Jumlah Data (Count): Terdapat 1000 transaksi yang tercatat dengan nilai total penjualan. Rata-rata (Mean): Rata-rata total penjualan per transaksi adalah 322.97, yang menunjukkan bahwa, secara umum, total penjualan dalam transaksi ini berputar di sekitar angka tersebut. Standar Deviasi (Std): Nilai standar deviasi sebesar 245.89 menunjukkan keragaman yang besar dalam nilai total penjualan. Artinya, ada beberapa transaksi dengan total yang jauh lebih tinggi atau lebih rendah dari rata-rata. Nilai Minimum (Min): Total penjualan terkecil adalah 10.68, yang menunjukkan adanya transaksi dengan nilai penjualan sangat rendah. Kuartil (25%, 50%, 75%): 25% dari transaksi memiliki total penjualan hingga 124.42. 50% (Median) dari transaksi memiliki total penjualan sekitar 253.85. 75% dari transaksi memiliki total penjualan hingga 471.35, yang menunjukkan bahwa 25% transaksi teratas memiliki total lebih tinggi dari angka tersebut. Nilai Maksimum (Max): Nilai total penjualan tertinggi adalah 1042.65, yang mungkin menunjukkan transaksi dengan volume penjualan yang sangat tinggi.

2. Analisis Berdasarkan Statistik Deskriptif Total Penjualan Lebih Tinggi pada Pembelian Besar: Dengan nilai total penjualan maksimum mencapai 1042.65, dapat diasumsikan bahwa transaksi dengan nilai lebih tinggi berkaitan dengan pembelian lebih banyak barang atau barang dengan harga lebih tinggi. Keragaman Penjualan (Standar Deviasi): Dengan standar deviasi yang relatif tinggi (245.89), data menunjukkan bahwa terdapat transaksi dengan total penjualan yang sangat bervariasi. Ini bisa menunjukkan bahwa sebagian besar transaksi memiliki total penjualan yang moderat, tetapi ada beberapa yang memiliki total penjualan

yang sangat tinggi atau sangat rendah. Distribusi Data: Berdasarkan statistik kuartil, sebagian besar transaksi (75%) memiliki total penjualan di bawah 471.35, sedangkan sebagian kecil transaksi memiliki total penjualan yang jauh lebih tinggi, bahkan ada yang mencapai lebih dari 1000.

3. Visualisasi Distribusi Total Penjualan Histogram:

Histogram menunjukkan bahwa distribusi total penjualan memiliki kecenderungan miring ke kiri, dengan sebagian besar transaksi memiliki total penjualan yang lebih rendah, sementara jumlah transaksi dengan total penjualan tinggi lebih sedikit. Dengan menambahkan Kernel Density Estimation (KDE), kita bisa melihat garis halus yang menunjukkan bahwa sebagian besar transaksi memiliki total penjualan di bawah rata-rata, dengan puncak distribusi berada di sekitar angka yang lebih rendah. Boxplot:

Boxplot menunjukkan adanya outlier atau transaksi dengan total penjualan yang sangat tinggi, yang terlihat sebagai titik yang berada jauh di luar whisker (batas normal). Median yang terletak di sekitar 253.85 lebih rendah dibandingkan dengan rata-rata 322.97, menunjukkan bahwa distribusi data agak miring ke kanan (positively skewed), dengan sebagian besar transaksi berada di bawah rata-rata dan hanya sedikit transaksi yang memiliki total penjualan jauh lebih tinggi.

4. Pengelompokan Berdasarkan Rentang Total Penjualan Rentang Penjualan: Data dibagi menjadi beberapa rentang penjualan untuk melihat distribusi transaksi dalam kategori-kategori tersebut:

0-50: 69 transaksi 51-100: 139 transaksi 101-200: 202 transaksi 201-500: 363 transaksi 501-1000: 218 transaksi >1000: 9 transaksi Sebagian besar transaksi (sekitar 80%) berada pada kategori 201-500 dan 501-1000, menunjukkan bahwa sebagian besar pembelian di supermarket ini berada pada kisaran harga menengah hingga tinggi. Hanya sedikit transaksi yang melibatkan penjualan sangat rendah (0-50) atau sangat tinggi (>1000).

3.4 4. Analisis Berdasarkan Tax 5%

[35]:
```
# Deskripsi statistik untuk tax 5%
tax_stats = df_supermarket['Tax 5%'].describe()
print(tax_stats)
```

```
count    1000.000000
mean       15.379369
std        11.708825
min         0.508500
25%         5.924875
50%        12.088000
75%        22.445250
max        49.650000
Name: Tax 5%, dtype: float64
```

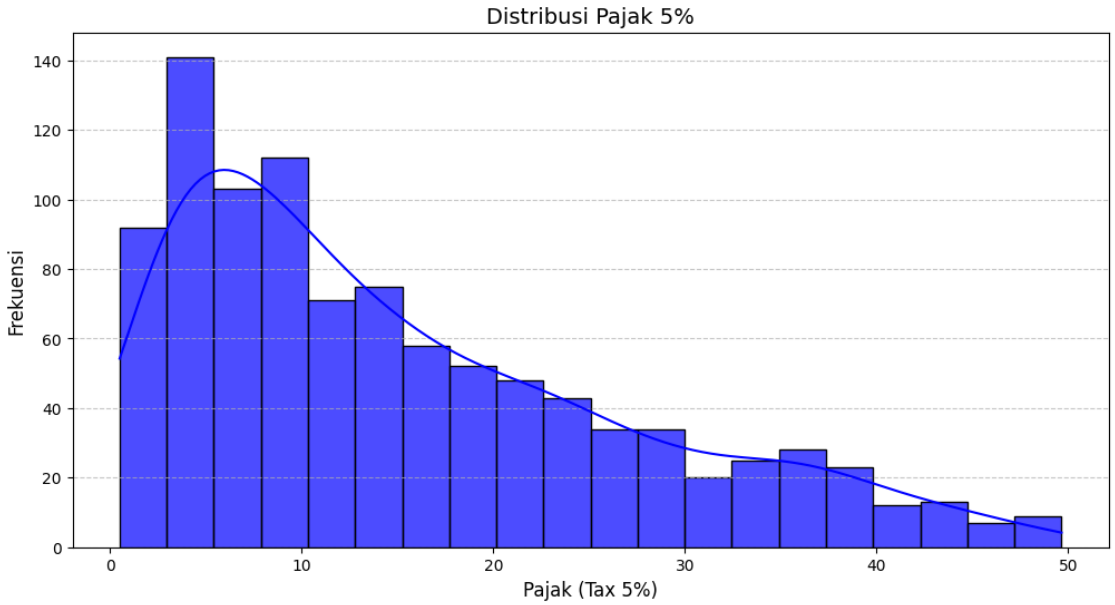
Hasilnya memberikan metrik penting, seperti:

- Count: Jumlah transaksi dengan pajak.
- Mean: Rata-rata pajak per transaksi.
- Std: Keragaman pajak (standar deviasi).

- Min: Pajak terendah.
- Max: Pajak tertinggi.

4.1 Visualisasi Distribusi Tax

[36]:
```
plt.figure(figsize=(12, 6))
sns.histplot(df_supermarket['Tax 5%'], bins=20, kde=True, color='blue', alpha=0.
↪7)
plt.title('Distribusi Pajak 5%', fontsize=14)
plt.xlabel('Pajak (Tax 5%)', fontsize=12)
plt.ylabel('Frekuensi', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



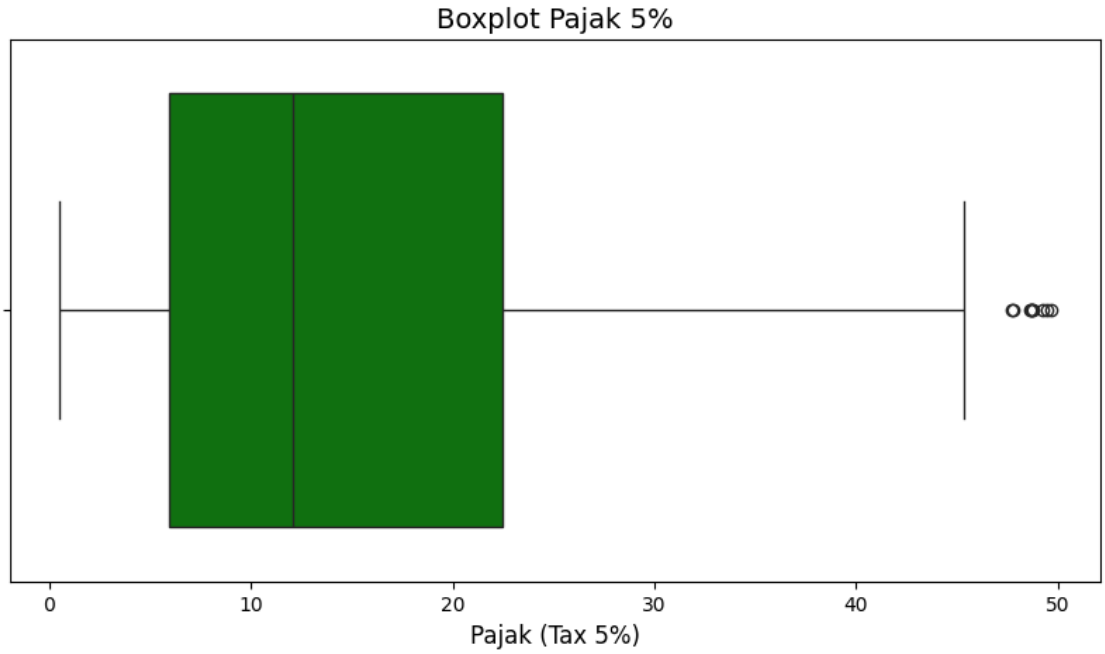
Penjelasan:

- Histogram: Menunjukkan frekuensi berbagai nilai pajak.
- KDE: Memberikan garis distribusi yang halus.

Analisis Visual:

- Nilai pajak yang sering muncul (modus).
- Pola distribusi pajak, apakah normal, miring, atau memiliki outlier.

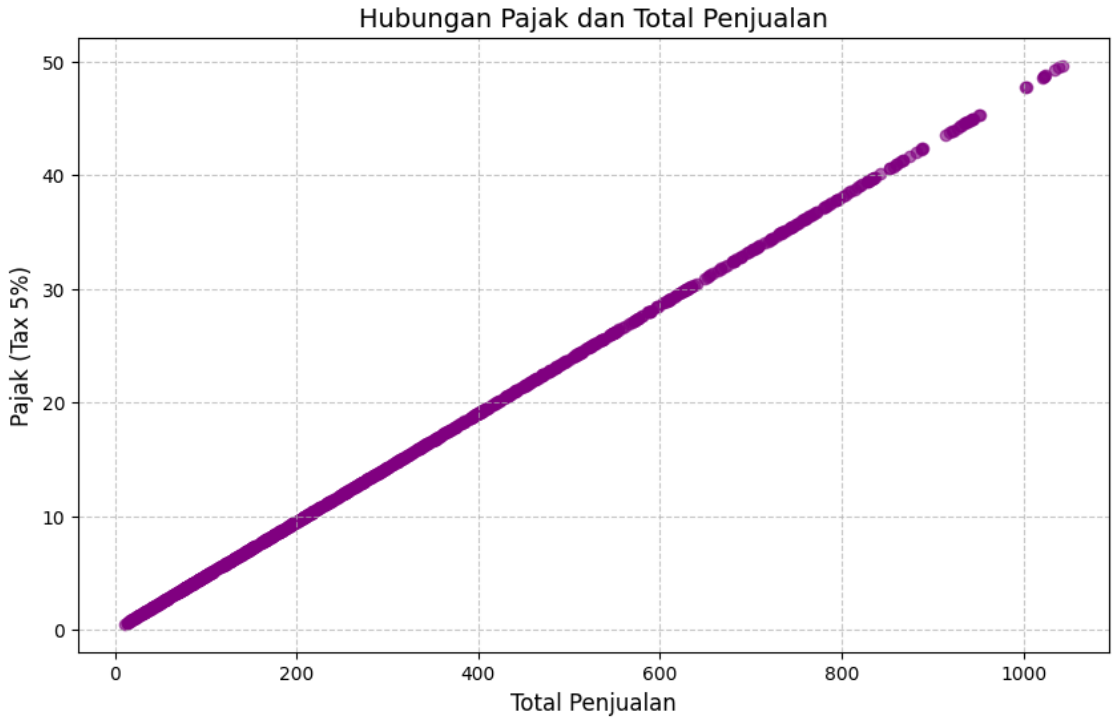
[37]:
```
plt.figure(figsize=(10, 5))
sns.boxplot(x=df_supermarket['Tax 5%'], color='green')
plt.title('Boxplot Pajak 5%', fontsize=14)
plt.xlabel('Pajak (Tax 5%)', fontsize=12)
plt.show()
```



Boxplot: Menampilkan rentang nilai pajak, termasuk outlier. Identifikasi transaksi dengan pajak sangat tinggi atau rendah.

4.2 Hubungan Pajak dengan Total Penjualan

[38]:
```
plt.figure(figsize=(10, 6))
plt.scatter(df_supermarket['Total'], df_supermarket['Tax 5%'], alpha=0.6,␣
 ↪color='purple')
plt.title('Hubungan Pajak dan Total Penjualan', fontsize=14)
plt.xlabel('Total Penjualan', fontsize=12)
plt.ylabel('Pajak (Tax 5%)', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

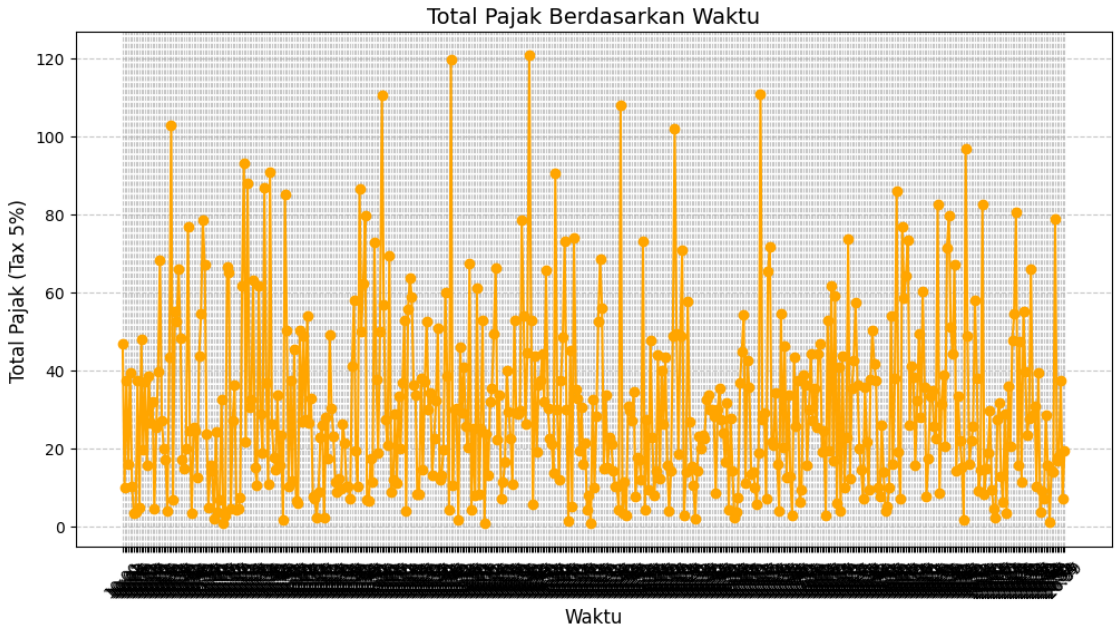


Analisis Visual:

- Hubungan antara total penjualan dan pajak seharusnya linier (Tax = Total * 0.05).
- Identifikasi transaksi dengan pajak tidak wajar (jika ada).

4.4 Hubungan Pajak dengan Waktu

[39]:
```
# Total pajak berdasarkan waktu transaksi
tax_time = df_supermarket.groupby('Time').agg(
    total_tax=('Tax 5%', 'sum')
).reset_index()

# Visualisasi
plt.figure(figsize=(12, 6))
plt.plot(tax_time['Time'].astype(str), tax_time['total_tax'], marker='o',␣
 ↪color='orange')
plt.title('Total Pajak Berdasarkan Waktu', fontsize=14)
plt.xlabel('Waktu', fontsize=12)
plt.ylabel('Total Pajak (Tax 5%)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```



Berdasarkan hasil deskripsi statistik untuk kolom Tax 5% (pajak 5%), berikut adalah analisisnya:

1. Statistik Deskriptif Pajak 5% Jumlah Data (Count): Terdapat 1000 transaksi yang memiliki nilai pajak 5%. Rata-rata (Mean): Rata-rata pajak untuk setiap transaksi adalah 15.38, yang menunjukkan bahwa pada umumnya pajak yang dikenakan dalam transaksi supermarket ini berada sekitar angka tersebut. Standar Deviasi (Std): Nilai standar deviasi sebesar 11.71 menunjukkan bahwa terdapat variasi yang cukup besar dalam nilai pajak antar transaksi. Artinya, ada beberapa transaksi dengan pajak yang jauh lebih tinggi atau lebih rendah dibandingkan rata-rata. Nilai Minimum (Min): Nilai pajak terendah adalah 0.51, yang mungkin terjadi pada transaksi dengan harga produk yang rendah atau transaksi yang melibatkan diskon atau promosi. Kuartil (25%, 50%, 75%): 25% dari transaksi memiliki pajak hingga 5.92. 50% (Median) dari transaksi memiliki pajak sekitar 12.09. 75% dari transaksi memiliki pajak hingga 22.45, yang menunjukkan bahwa 25% transaksi teratas memiliki pajak lebih tinggi dari angka tersebut. Nilai Maksimum (Max): Nilai pajak tertinggi adalah 49.65, yang mungkin terjadi pada transaksi dengan harga produk yang sangat tinggi atau banyak produk.

2. Analisis Pajak Berdasarkan Statistik Deskriptif Pajak Lebih Tinggi pada Pembelian Produk Mahal: Berdasarkan rentang nilai pajak, dapat diasumsikan bahwa transaksi dengan nilai pajak yang lebih tinggi berhubungan dengan pembelian produk dengan harga lebih tinggi. Hal ini terlihat pada nilai pajak maksimum yang mencapai 49.65. Keragaman Pajak (Standar Deviasi): Standar deviasi yang relatif tinggi (11.71) menunjukkan bahwa ada variasi yang besar dalam pajak yang dibebankan di setiap transaksi. Ini dapat disebabkan oleh adanya perbedaan harga unit produk yang dibeli, kuantitas, atau kemungkinan adanya promosi yang mempengaruhi pajak yang dikenakan.

3.5 5. Analisis Berdasarkan Quantity

[40]:
```
# Memastikan kolom Quantity ada dan dalam format numerik
df_supermarket['Quantity'] = pd.to_numeric(df_supermarket['Quantity'],␣
 ↪errors='coerce')

# Memeriksa statistik deskriptif kolom Quantity
quantity_stats = df_supermarket['Quantity'].describe()
print(quantity_stats)
```

```
count    1000.000000
mean        5.510000
std         2.923431
min         1.000000
25%         3.000000
50%         5.000000
75%         8.000000
max        10.000000
Name: Quantity, dtype: float64
```

- pd.to_numeric(): Memastikan data dalam kolom Quantity berupa angka.
- describe(): Memberikan statistik deskriptif seperti rata-rata, median, dan nilai minimum/maksimum.

5.1 Visualisasi Data

[41]:
```
# Mengelompokkan data berdasarkan rentang kuantitas
bins = [0, 5, 10, 15, 20, 50]  # Rentang kuantitas
labels = ['1-5', '6-10', '11-15', '16-20', '21-50']  # Label kategori
df_supermarket['Quantity_Group'] = pd.cut(df_supermarket['Quantity'],␣
 ↪bins=bins, labels=labels)

# Memeriksa beberapa baris untuk memastikan kolom baru terbentuk
show_df(df_supermarket)
```

	Invoice ID	Branch	Product line	Unit price	Quantity	Tax 5%	\
0	750-67-8428	A	Health and beauty	74.69	7	26.1415	
1	226-31-3081	C	Electronic accessories	15.28	5	3.8200	
2	631-41-3108	A	Home and lifestyle	46.33	7	16.2155	
3	123-19-1176	A	Health and beauty	58.22	8	23.2880	
4	373-73-7910	A	Sports and travel	86.31	7	30.2085	

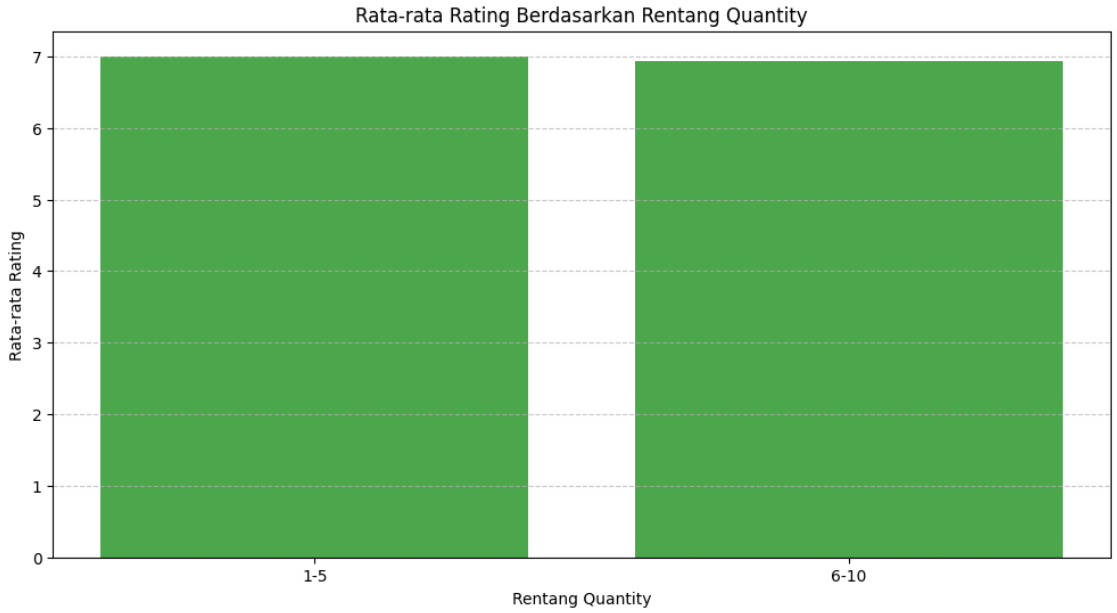
	Total	Date	Time	Payment	Rating	Day	Month	Year	\
0	548.9715	2019-01-05	13:08	Ewallet	9.1	5	1	2019	
1	80.2200	2019-03-08	10:29	Cash	9.6	8	3	2019	
2	340.5255	2019-03-03	13:23	Credit card	7.4	3	3	2019	
3	489.0480	2019-01-27	20:33	Ewallet	8.4	27	1	2019	
4	634.3785	2019-02-08	10:37	Ewallet	5.3	8	2	2019	

```
  Total_Range Quantity_Group
0    501-1000           6-10
1      51-100            1-5
2     201-500           6-10
3     201-500           6-10
4    501-1000           6-10


Jumlah baris: 1000
Jumlah kolom: 16
Kolom: ['Invoice ID', 'Branch', 'Product line', 'Unit price', 'Quantity', 'Tax
5%', 'Total', 'Date', 'Time', 'Payment', 'Rating', 'Day', 'Month', 'Year',
'Total_Range', 'Quantity_Group']
```

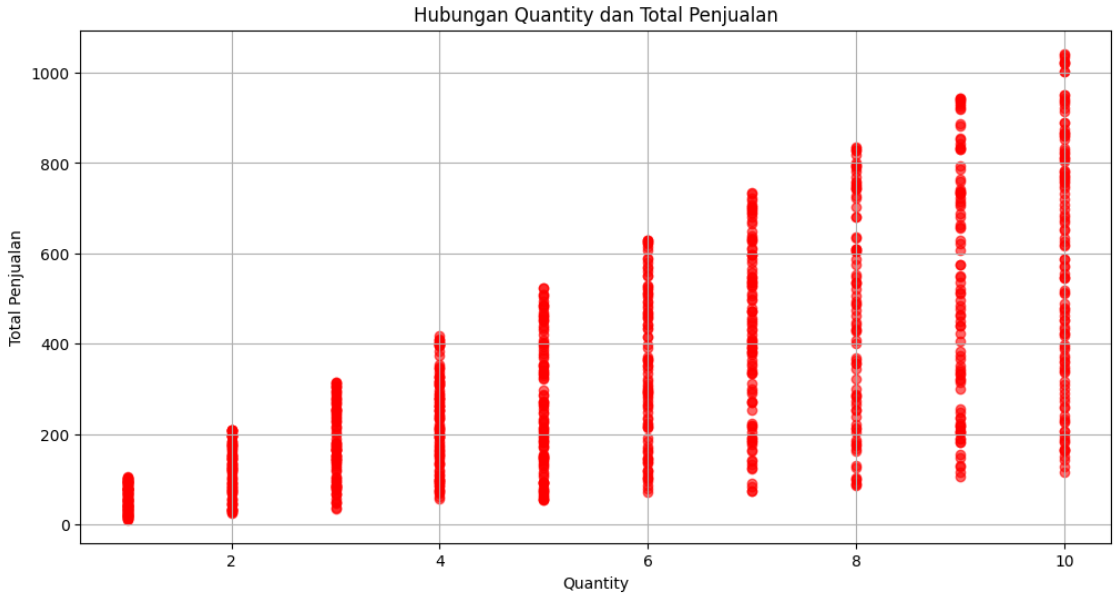
[42]:
```
# Menghitung rata-rata rating berdasarkan grup kuantitas
quantity_grouped = df_supermarket.groupby('Quantity_Group').agg(
    average_rating=('Rating', 'mean'),
    transaction_count=('Quantity', 'count')  # Jumlah transaksi
).reset_index()

# Memeriksa hasil pengelompokan
print(quantity_grouped)
```

```
  Quantity_Group  average_rating  transaction_count
0            1-5        7.002579                504
1           6-10        6.942339                496
2          11-15             NaN                  0
3          16-20             NaN                  0
4          21-50             NaN                  0

<ipython-input-42-abc2de9d19c8>:2: FutureWarning: The default of observed=False
is deprecated and will be changed to True in a future version of pandas. Pass
observed=False to retain current behavior or observed=True to adopt the future
default and silence this warning.
  quantity_grouped = df_supermarket.groupby('Quantity_Group').agg(
```

[43]:
```
# Grafik batang rata-rata rating berdasarkan rentang quantity
plt.figure(figsize=(12, 6))
plt.bar(quantity_grouped['Quantity_Group'], quantity_grouped['average_rating'],␣
 ↪color='g', alpha=0.7)
plt.title('Rata-rata Rating Berdasarkan Rentang Quantity')
plt.xlabel('Rentang Quantity')
plt.ylabel('Rata-rata Rating')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



[44]:
```
# Scatter plot antara quantity dan total penjualan
plt.figure(figsize=(12, 6))
plt.scatter(df_supermarket['Quantity'], df_supermarket['Total'], color='r',␣
 ↪alpha=0.6)
plt.title('Hubungan Quantity dan Total Penjualan')
plt.xlabel('Quantity')
plt.ylabel('Total Penjualan')
plt.grid(True)
plt.show()
```



Analisis:

1. Statistik Deskriptif Quantity Jumlah Data: Terdapat 1000 transaksi dalam dataset ini. Rata-rata (Mean): Rata-rata kuantitas per transaksi adalah 5.51, yang menunjukkan bahwa transaksi paling sering memiliki jumlah kuantitas produk sekitar 5 hingga 6 unit. Standar Deviasi (Std): Nilai standar deviasi sebesar 2.92 menunjukkan variasi yang cukup besar pada kuantitas produk yang dibeli dalam setiap transaksi. Nilai Minimum (Min): Kuantitas transaksi terendah adalah 1, yang berarti ada transaksi yang hanya membeli 1 produk. Kuartil (25%, 50%, 75%): 25% dari transaksi memiliki kuantitas hingga 3 produk. 50% (Median) dari transaksi memiliki kuantitas 5 produk. 75% dari transaksi memiliki kuantitas hingga 8 produk. Nilai Maksimum (Max): Kuantitas transaksi tertinggi adalah 10 produk.
2. Pengelompokan Berdasarkan Rentang Kuantitas Berdasarkan rentang kuantitas yang dibagi menjadi 5 kategori: 1-5: Memiliki 504 transaksi, yang merupakan kategori dengan jumlah transaksi terbanyak. Rata-rata rating untuk kategori ini adalah 7.00. 6-10: Memiliki 496 transaksi, yang hampir setara dengan kategori 1-5. Rata-rata rating untuk kategori ini sedikit lebih rendah, yaitu 6.94. 11-15, 16-20, dan 21-50: Tidak ada transaksi sama sekali dalam kategori ini. Hal ini menunjukkan bahwa hampir semua transaksi berada dalam rentang kuantitas 1-10 produk.
3. Analisis Rating Berdasarkan Kuantitas Rata-rata rating untuk kategori 1-5 sedikit lebih tinggi (7.00) dibandingkan dengan kategori 6-10 yang memiliki rata-rata rating 6.94. Hal ini bisa menunjukkan bahwa pelanggan yang membeli lebih sedikit produk mungkin lebih puas dengan pembelian mereka, meskipun perbedaannya tidak signifikan. Kehadiran transaksi yang lebih sedikit pada rentang kuantitas yang lebih tinggi (11-50 produk) menunjukkan bahwa transaksi dengan kuantitas besar sangat jarang terjadi, mungkin karena pembelian dalam jumlah besar tidak umum untuk jenis produk ini.

3.6 6. Analisis Berdasarkan Unit Price

[45]:
```
# Menentukan rentang unit price
bins_price = [0, 50, 100, 150, 200, 500]  # Rentang harga
labels_price = ['0-50', '51-100', '101-150', '151-200', '201-500']  # Label␣
 ↪kategori

# Membuat kolom Unit_Price_Group berdasarkan rentang harga
df_supermarket['Unit_Price_Group'] = pd.cut(df_supermarket['Unit price'],␣
 ↪bins=bins_price, labels=labels_price)

# Menampilkan beberapa baris untuk memeriksa kolom baru
show_df(df_supermarket)
```

	Invoice ID	Branch	Product line	Unit price	Quantity	Tax 5%	\
0	750-67-8428	A	Health and beauty	74.69	7	26.1415	
1	226-31-3081	C	Electronic accessories	15.28	5	3.8200	
2	631-41-3108	A	Home and lifestyle	46.33	7	16.2155	
3	123-19-1176	A	Health and beauty	58.22	8	23.2880	

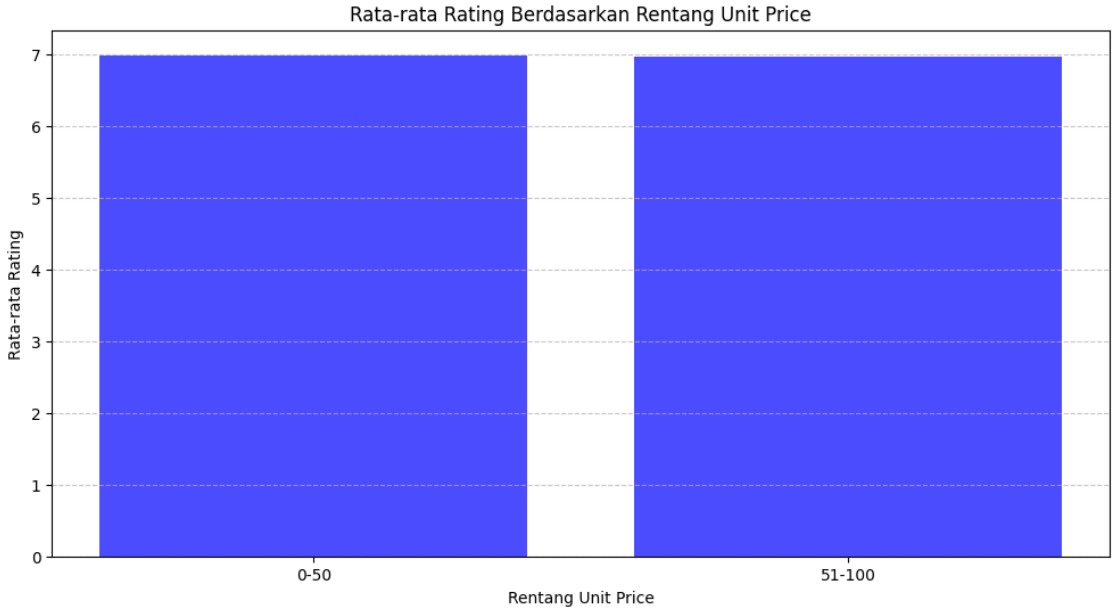
```
4  373-73-7910      A       Sports and travel       86.31         7  30.2085

      Total        Date   Time      Payment  Rating  Day  Month  Year  \
0  548.9715  2019-01-05  13:08      Ewallet     9.1    5      1  2019
1   80.2200  2019-03-08  10:29         Cash     9.6    8      3  2019
2  340.5255  2019-03-03  13:23  Credit card     7.4    3      3  2019
3  489.0480  2019-01-27  20:33      Ewallet     8.4   27      1  2019
4  634.3785  2019-02-08  10:37      Ewallet     5.3    8      2  2019

  Total_Range Quantity_Group Unit_Price_Group
0    501-1000           6-10           51-100
1      51-100            1-5             0-50
2     201-500           6-10             0-50
3     201-500           6-10           51-100
4    501-1000           6-10           51-100


Jumlah baris: 1000
Jumlah kolom: 17
Kolom: ['Invoice ID', 'Branch', 'Product line', 'Unit price', 'Quantity', 'Tax
5%', 'Total', 'Date', 'Time', 'Payment', 'Rating', 'Day', 'Month', 'Year',
'Total_Range', 'Quantity_Group', 'Unit_Price_Group']
```

[46]:
```
# Menghitung rata-rata rating dan jumlah transaksi per grup unit price
unit_price_grouped = df_supermarket.groupby('Unit_Price_Group').agg(
    average_rating=('Rating', 'mean'),  # Rata-rata rating
    transaction_count=('Quantity', 'count')  # Jumlah transaksi
).reset_index()

# Memeriksa hasil pengelompokan
show_df(unit_price_grouped)
```

```
  Unit_Price_Group  average_rating  transaction_count
0             0-50        6.983599                439
1           51-100        6.964171                561
2          101-150             NaN                  0
3          151-200             NaN                  0
4          201-500             NaN                  0


Jumlah baris: 5
Jumlah kolom: 3
Kolom: ['Unit_Price_Group', 'average_rating', 'transaction_count']

<ipython-input-46-d11de0ed3a01>:2: FutureWarning: The default of observed=False
is deprecated and will be changed to True in a future version of pandas. Pass
observed=False to retain current behavior or observed=True to adopt the future
default and silence this warning.
```

```
unit_price_grouped = df_supermarket.groupby('Unit_Price_Group').agg(
```

[47]:
```
# Membuat grafik batang untuk rata-rata rating per rentang unit price
plt.figure(figsize=(12, 6))
plt.bar(unit_price_grouped['Unit_Price_Group'],␣
 ↪unit_price_grouped['average_rating'], color='b', alpha=0.7)
plt.title('Rata-rata Rating Berdasarkan Rentang Unit Price')
plt.xlabel('Rentang Unit Price')
plt.ylabel('Rata-rata Rating')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



1. Rentang Unit Price yang Paling Banyak Dipesan Rentang harga 0-50 memiliki jumlah transaksi terbanyak dengan 439 transaksi. Namun, rata-rata ratingnya adalah 6.98, yang sedikit lebih rendah dibandingkan dengan rentang harga 51-100. Rentang harga 51-100 memiliki 561 transaksi, yang sedikit lebih tinggi daripada rentang harga 0-50. Rata-rata ratingnya adalah 6.96, yang hampir setara dengan rentang harga 0-50.

2. Rentang Unit Price yang Tidak Ada Transaksinya Rentang harga 101-150, 151-200, dan 201-500 tidak memiliki transaksi sama sekali (jumlah transaksi = 0), sehingga tidak ada data rating untuk kelompok harga ini. Hal ini mungkin menunjukkan bahwa produk dalam rentang harga tersebut tidak tersedia atau tidak diminati oleh pelanggan di supermarket ini.

3. Analisis Rating Rata-rata rating untuk kedua rentang harga yang memiliki transaksi (0-50 dan 51-100) tidak jauh berbeda, namun cukup rendah (sekitar 6-7). Ini bisa menunjukkan bahwa meskipun produk dalam rentang harga ini banyak terjual, kepuasan pelanggan secara keseluruhan masih bisa ditingkatkan.

4. Rekomendasi Produk dengan harga lebih rendah (0-50) lebih banyak diminati, meskipun

ratingnya agak rendah. Untuk meningkatkan kepuasan pelanggan, mungkin supermarket bisa memperbaiki kualitas produk atau memberikan promosi untuk meningkatkan rating. Produk dengan harga menengah (51-100) juga banyak dibeli, namun dengan rating yang mirip dengan produk harga rendah. Hal ini menunjukkan bahwa peningkatan kualitas atau strategi pemasaran bisa bermanfaat di rentang harga ini. Supermarket harus mengevaluasi produk dalam rentang harga 100 ke atas, karena tidak ada transaksi yang terjadi. Ini bisa jadi indikasi bahwa produk tersebut perlu disesuaikan harganya atau dipromosikan dengan lebih efektif.

4 Korelasi

[48]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

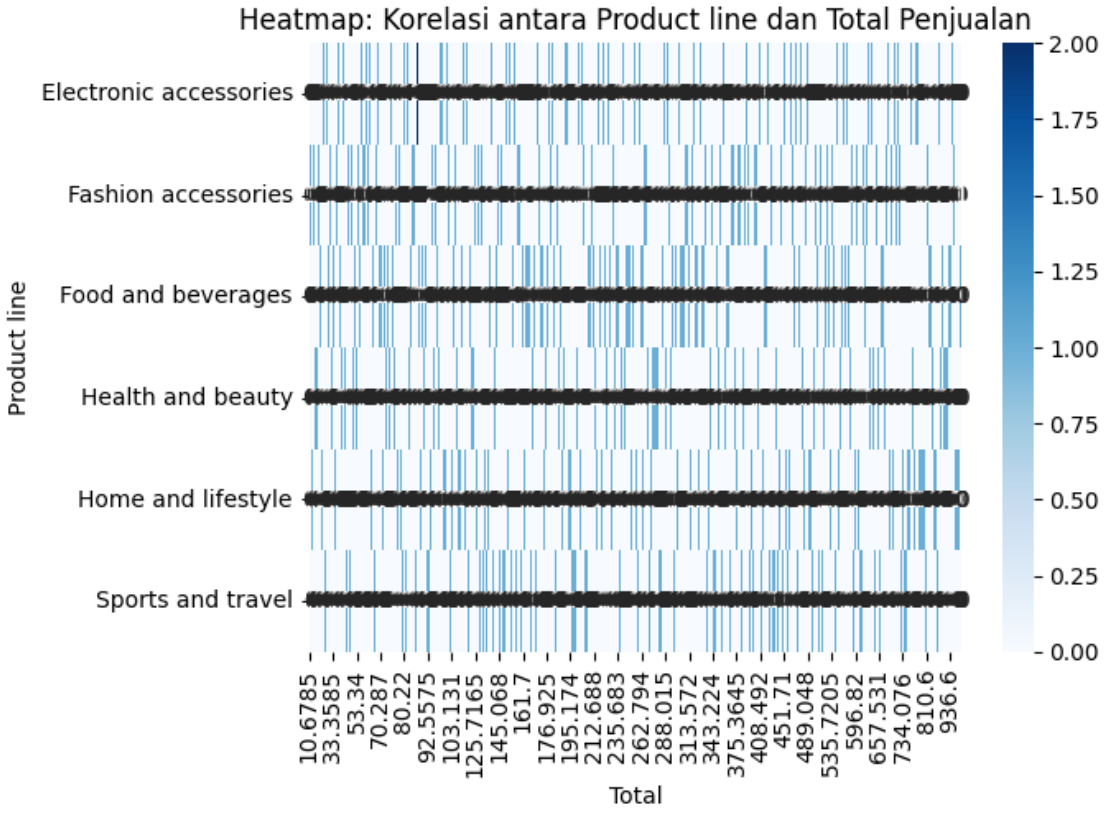
4.1 Korelasi antara Kategori Produk (Product Line) dan Total Penjualan

[49]:
```
# Tabel kontingensi antara Product line dan Total Penjualan
contingency_table = pd.crosstab(df_supermarket['Product line'],␣
 ↪df_supermarket['Total'])

# Uji Chi-Square
chi2_stat, p_value, dof, expected = stats.chi2_contingency(contingency_table)

# Menampilkan hasil uji Chi-Square
print(f"Hasil Uji Chi-Square antara Product line dan Total Penjualan:")
print(f"Chi2 Stat: {chi2_stat:.2f}")
print(f"P-Value: {p_value:.4f}")
print(f"Degree of Freedom: {dof}")
print(f"Expected Frequencies: \n{expected}")
```

```
Hasil Uji Chi-Square antara Product line dan Total Penjualan:
Chi2 Stat: 4958.28
P-Value: 0.4443
Degree of Freedom: 4945
Expected Frequencies:
[[0.17  0.17  0.17  … 0.17  0.17  0.17 ]
 [0.178 0.178 0.178 … 0.178 0.178 0.178]
 [0.174 0.174 0.174 … 0.174 0.174 0.174]
 [0.152 0.152 0.152 … 0.152 0.152 0.152]
 [0.16  0.16  0.16  … 0.16  0.16  0.16 ]
 [0.166 0.166 0.166 … 0.166 0.166 0.166]]
```

[50]:
```
# Visualisasi dengan heatmap
sns.heatmap(contingency_table, annot=True, cmap='Blues', fmt='g')
```

```
plt.title("Heatmap: Korelasi antara Product line dan Total Penjualan")
plt.show()

# Interpretasi hasil
if p_value < 0.05:
    print("Terdapat hubungan yang signifikan antara 'Product line' dan 'Total Penjualan'.")
else:
    print("Tidak terdapat hubungan yang signifikan antara 'Product line' dan 'Total Penjualan'.")
```




```
Tidak terdapat hubungan yang signifikan antara 'Product line' dan 'Total Penjualan'.
```

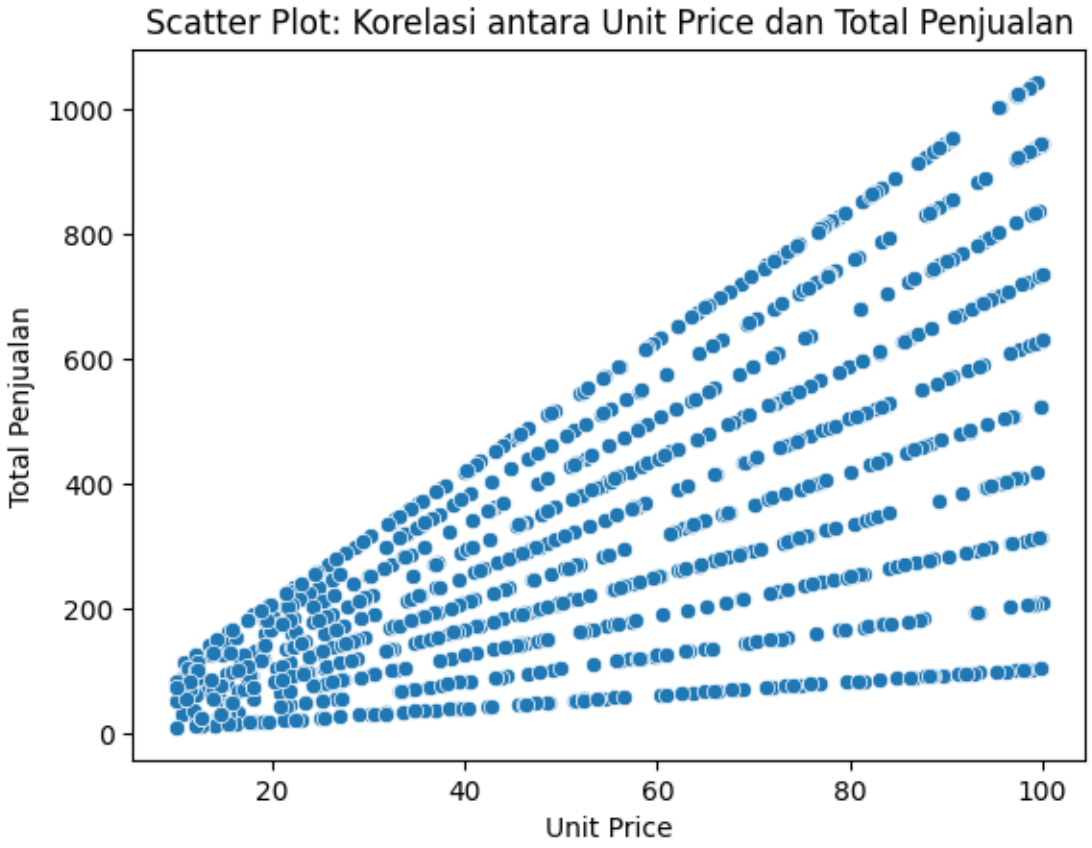
1. Nilai p-value (0.4443) lebih besar dari 0.05 (tingkat signifikansi yang umum digunakan). Ini menunjukkan bahwa tidak ada hubungan yang signifikan antara variabel Product line dan Total Penjualan.
2. Nilai Chi-Square statistik (4958.28) besar, tetapi dengan derajat kebebasan (4945) yang hampir sama besar, hal ini konsisten dengan p-value tinggi.
3. Frekuensi harapan (expected) merata dan kecil, mengindikasikan distribusi yang cenderung acak.

Berdasarkan hasil uji Chi-Square tidak terdapat hubungan yang signifikan antara Product line dan Total Penjualan. Hal ini berarti bahwa variasi penjualan tidak dipengaruhi oleh kategori produk (Product line).

4.2 Korelasi antara Unit price (Harga Produk) dan Total Penjualan

[51]:
```
# Menghitung korelasi Pearson antara Unit price dan Total Penjualan
correlation_pearson = df_supermarket[['Unit price', 'Total']].
 ↪corr(method='pearson').iloc[0, 1]
print(f"Korelasi Pearson antara Unit Price dan Total Penjualan:␣
 ↪{correlation_pearson:.2f}")

# Visualisasi dengan scatter plot
sns.scatterplot(data=df_supermarket, x='Unit price', y='Total')
plt.title("Scatter Plot: Korelasi antara Unit Price dan Total Penjualan")
plt.xlabel("Unit Price")
plt.ylabel("Total Penjualan")
plt.show()
```

```
Korelasi Pearson antara Unit Price dan Total Penjualan: 0.63
```



Nilai Korelasi Pearson (r): 0.63. Nilai 0.63 menunjukkan korelasi positif yang cukup kuat antara Unit Price dan Total Penjualan. Artinya, ketika Unit Price meningkat, maka Total Penjualan cenderung ikut meningkat secara linear.

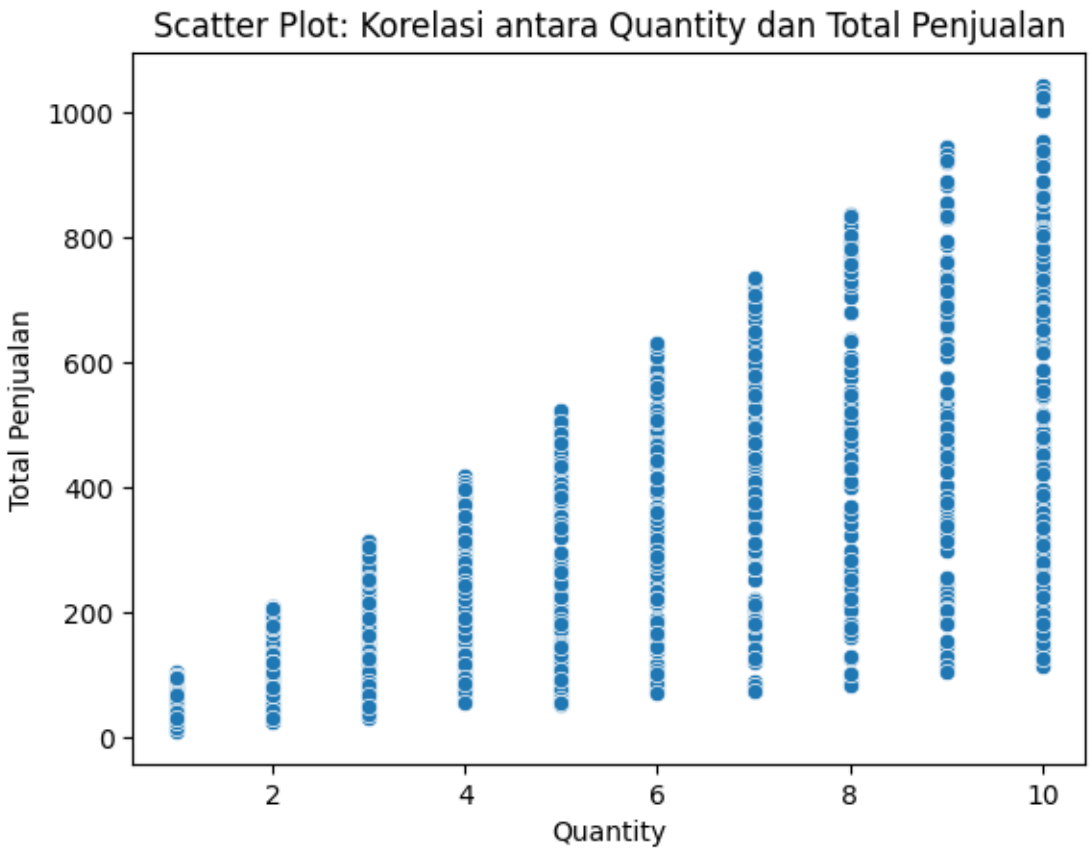
Interpretasi Scatter Plot: 1. Titik-titik data membentuk pola diagonal ke atas, menunjukkan bahwa kenaikan Unit Price sejalan dengan kenaikan Total Penjualan. 2. Terdapat penyebaran data yang relatif lebar, namun tren utama tetap menunjukkan hubungan positif. Meskipun ada variabilitas, pola hubungan masih konsisten. 3. Banyak titik data terkonsentrasi di bagian Unit Price yang lebih rendah (antara 10-40), tetapi Total Penjualan bervariasi.

Berdasarkan hasil Korelasi Pearson , terdapat hubungan positif yang cukup kuat antara Unit Price dan Total Penjualan. Jadi, semakin tinggi harga satuan produk maka semakin besar pula nilai total penjualannya. Kenaikan harga produk kemungkinan memengaruhi peningkatan total penjualan, mungkin karena produk-produk dengan harga lebih tinggi memiliki nilai jual lebih besar.

4.3 Korelasi antara Quantity (Jumlah Pembelian) dan Total Penjualan

[52]:
```
# Menghitung korelasi Pearson antara Quantity dan Total Penjualan
correlation_pearson = df_supermarket[['Quantity', 'Total']].
 ↪corr(method='pearson').iloc[0, 1]
print(f"Korelasi Pearson antara Quantity dan Total Penjualan:␣
 ↪{correlation_pearson:.2f}")

# Visualisasi dengan scatter plot
sns.scatterplot(data=df_supermarket, x='Quantity', y='Total')
plt.title("Scatter Plot: Korelasi antara Quantity dan Total Penjualan")
plt.xlabel("Quantity")
plt.ylabel("Total Penjualan")
plt.show()
```

```
Korelasi Pearson antara Quantity dan Total Penjualan: 0.71
```



Berdasarkan analisis scatter plot dan nilai korelasi Pearson:

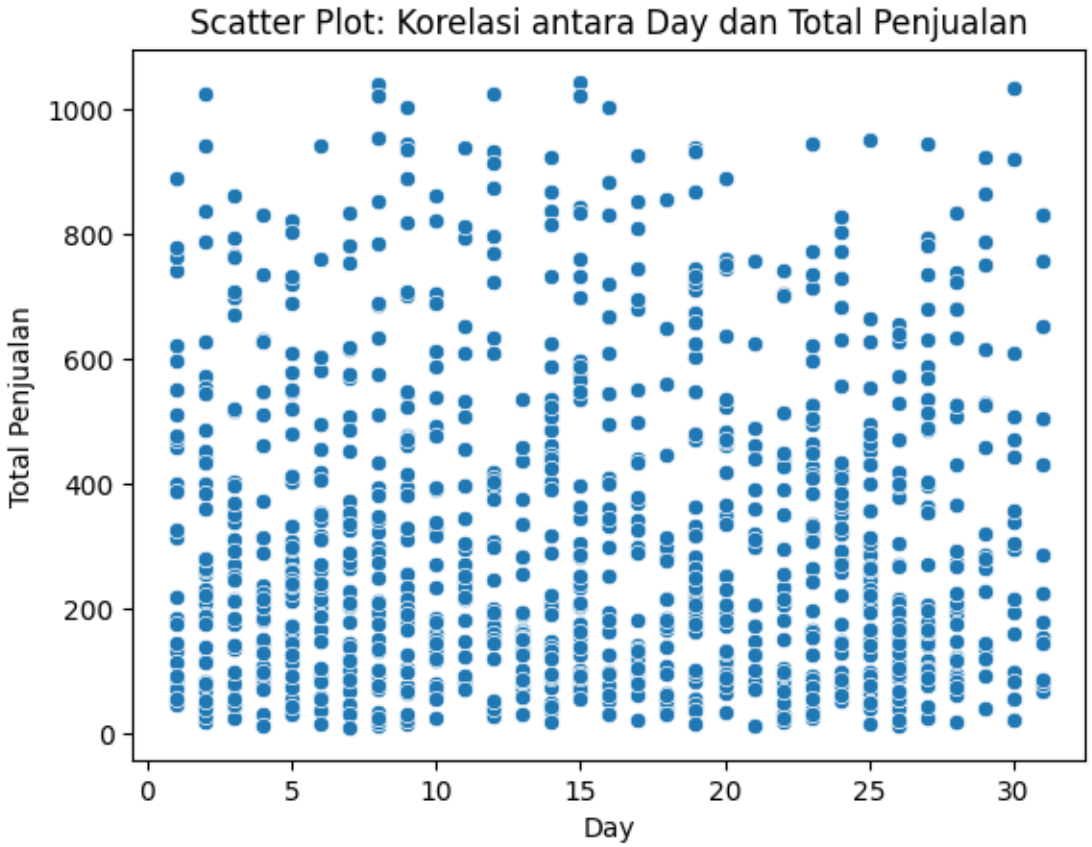
1. Hubungan Linear Positif: Quantity memiliki hubungan linear positif yang kuat dengan Total Penjualan.
2. Korelasi Kuat: Nilai korelasi Pearson 0.71 mengindikasikan bahwa kenaikan Quantity berkontribusi signifikan terhadap peningkatan Total Penjualan.
3. Implikasi Bisnis: Jika supermarket ingin meningkatkan Total Penjualan, salah satu caranya adalah dengan meningkatkan Quantity barang yang dijual.

4.4 Korelasi antara Day dan Total Penjualan

[53]:
```
# Menghitung korelasi Pearson antara Day dan Total Penjualan
correlation_pearson = df_supermarket[['Day', 'Total']].corr(method='pearson').
  ↪iloc[0, 1]
print(f"Korelasi Pearson antara Day dan Total Penjualan: {correlation_pearson:.
  ↪2f}")

# Visualisasi dengan scatter plot
sns.scatterplot(data=df_supermarket, x='Day', y='Total')
plt.title("Scatter Plot: Korelasi antara Day dan Total Penjualan")
```

```
plt.xlabel("Day")
plt.ylabel("Total Penjualan")
plt.show()
```

```
Korelasi Pearson antara Day dan Total Penjualan: -0.00
```



Berdasarkan analisis scatter plot dan nilai korelasi Pearson:

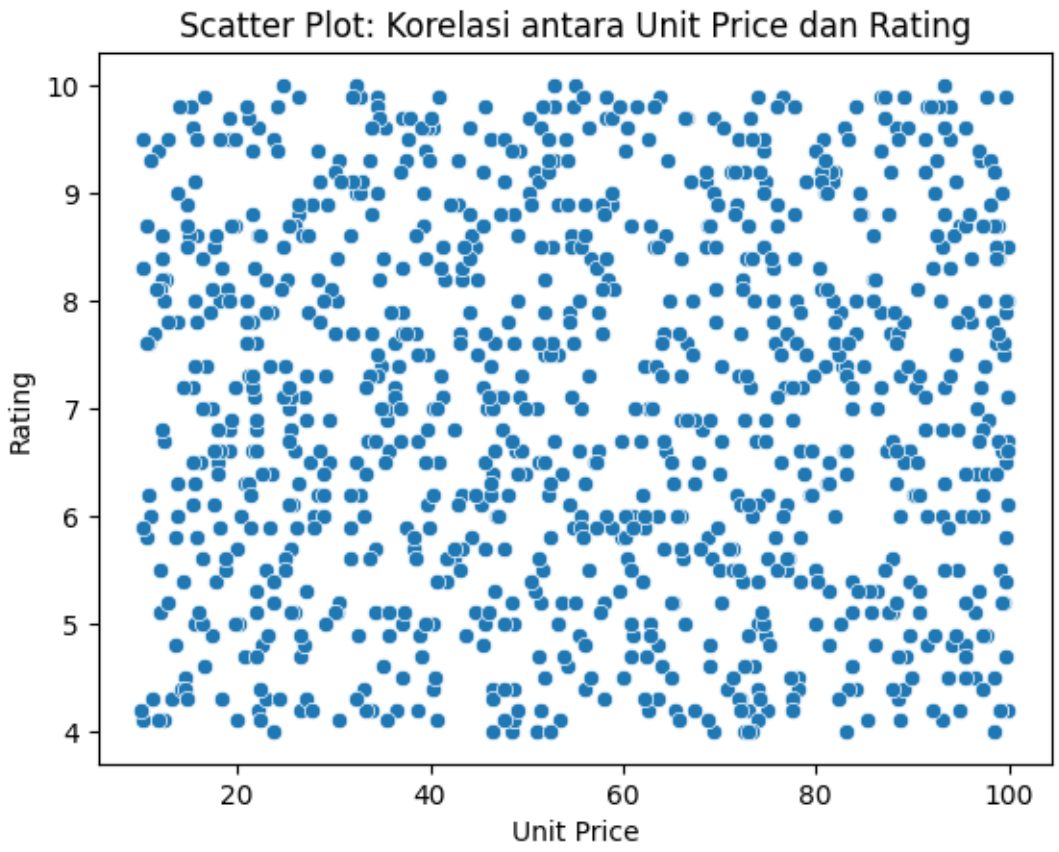
1. Tidak Ada Hubungan Linear: Korelasi sebesar -0.00 menunjukkan bahwa Day tidak memiliki hubungan linear dengan Total Penjualan.
2. Distribusi Acak: Total Penjualan cenderung bervariasi secara acak di seluruh hari dalam satu bulan.

4.5 Korelasi antara Unit Price dan Rating

[54]:
```
# Menghitung korelasi Pearson antara Unit price dan Rating
correlation_pearson = df_supermarket[['Unit price', 'Rating']].
  ↪corr(method='pearson').iloc[0, 1]
print(f"Korelasi Pearson antara Unit Price dan Rating: {correlation_pearson:.
  ↪2f}")
```

```
# Visualisasi dengan scatter plot
sns.scatterplot(data=df_supermarket, x='Unit price', y='Rating')
plt.title("Scatter Plot: Korelasi antara Unit Price dan Rating")
plt.xlabel("Unit Price")
plt.ylabel("Rating")
plt.show()
```

```
Korelasi Pearson antara Unit Price dan Rating: -0.01
```



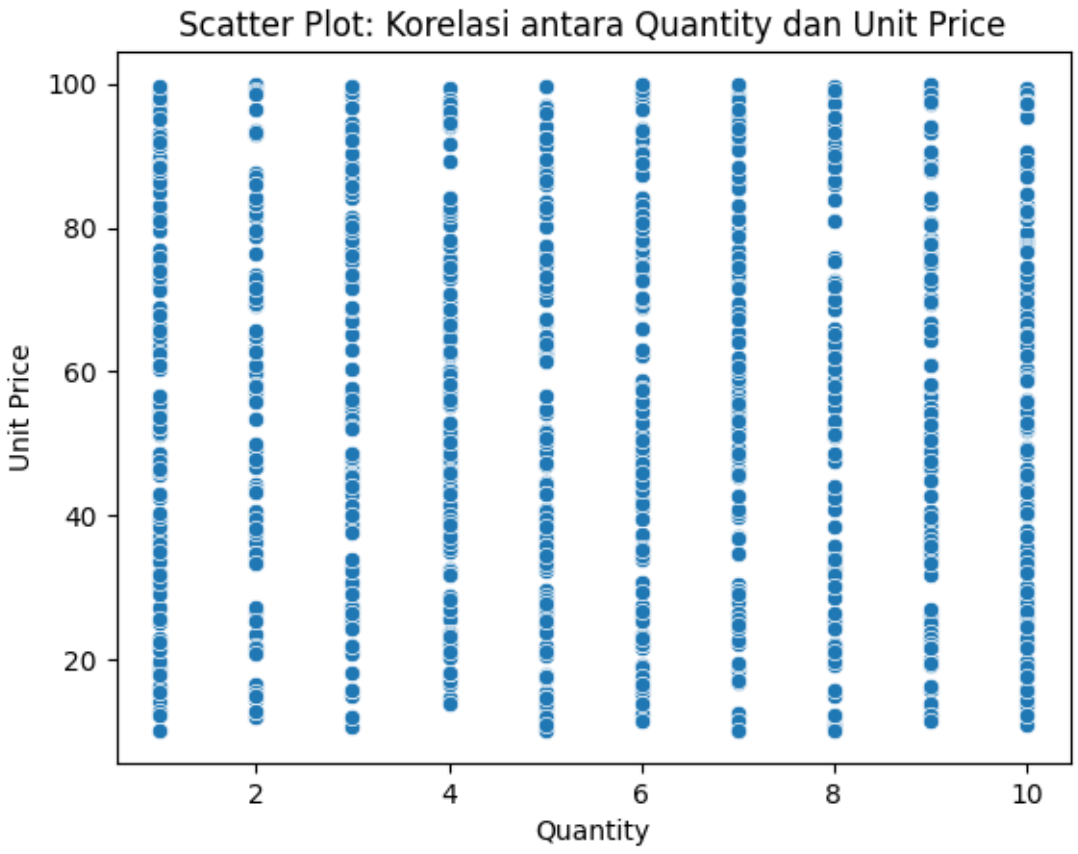
Berdasarkan analisis scatter plot dan nilai korelasi Pearson:

1. Tidak Ada Hubungan Linear: Korelasi sebesar -0.01 menunjukkan bahwa Unit Price tidak memengaruhi Rating secara linear.
2. Distribusi Acak: Rating tersebar secara acak di seluruh rentang harga, artinya faktor lain mungkin memengaruhi penilaian produk.
3. Harga produk (Unit Price) bukanlah faktor signifikan dalam menentukan Rating pelanggan.

4.6 Korelasi antara Quantity dan Unit Price

[55]:
```
# Menghitung korelasi Pearson antara Quantity dan Unit Price
correlation_pearson = df_supermarket[['Quantity', 'Unit price']].
 ↪corr(method='pearson').iloc[0, 1]
print(f"Korelasi Pearson antara Quantity dan Unit Price: {correlation_pearson:.
 ↪2f}")

# Visualisasi dengan scatter plot
sns.scatterplot(data=df_supermarket, x='Quantity', y='Unit price')
plt.title("Scatter Plot: Korelasi antara Quantity dan Unit Price")
plt.xlabel("Quantity")
plt.ylabel("Unit Price")
plt.show()
```

```
Korelasi Pearson antara Quantity dan Unit Price: 0.01
```



Berdasarkan scatter plot dan nilai korelasi Pearson, dapat disimpulkan bahwa Quantity dan Unit Price tidak memiliki hubungan yang kuat atau signifikan. Peningkatan atau penurunan Quantity tidak memengaruhi Unit Price secara langsung.

5 Analisis Popularitas

[56]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

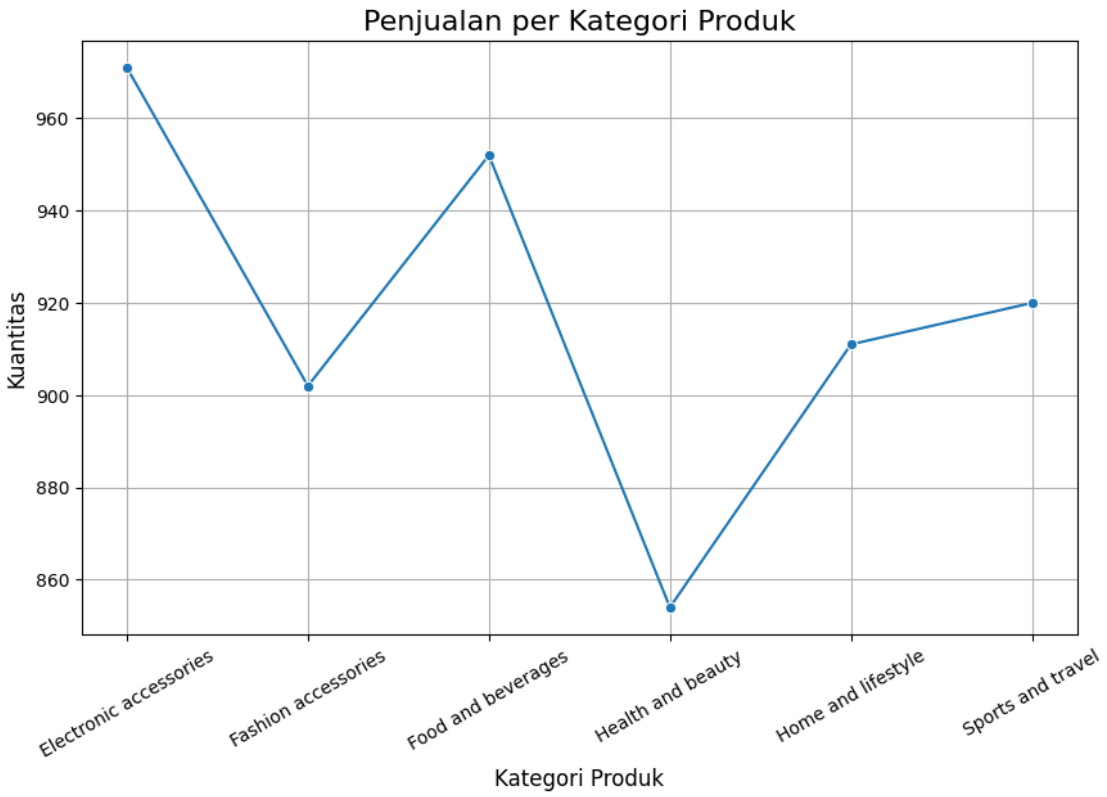
5.1 Produk dengan penjualan terbanyak

[57]:
```
# Penjualan terbanyak berdasarkan kuantitas produk yang terjual
popularity = df_supermarket.groupby('Product line').agg({
    'Quantity' : 'sum', # Total unit terjual
})

# Menampilkan hasil analisis
print("Penjualan produk terbanyak: ")
print(popularity)

# Mengurutkan berdasarkan jumlah unit terjual
popularity_sort = popularity.sort_values(by='Quantity', ascending=False)

# Menampilkan hasil yang sudah diurutkan
print("\nPenjualan produk terbanyak setelah diurutkan:")
print(popularity_sort)
```

```
Penjualan produk terbanyak:
                         Quantity
Product line
Electronic accessories        971
Fashion accessories           902
Food and beverages            952
Health and beauty             854
Home and lifestyle            911
Sports and travel             920

Penjualan produk terbanyak setelah diurutkan:
                         Quantity
Product line
Electronic accessories        971
Food and beverages            952
Sports and travel             920
Home and lifestyle            911
Fashion accessories           902
Health and beauty             854
```

[58]:
```
# Visualisasi penjualan per kategori
plt.figure(figsize=(10,6))
```

```
sns.lineplot(data=popularity.reset_index(), x='Product line', y='Quantity',␣
 ↪marker='o')
plt.title('Penjualan per Kategori Produk', fontsize=16)
plt.xlabel('Kategori Produk', fontsize=12)
plt.ylabel('Kuantitas', fontsize=12)
plt.xticks(rotation=30)
plt.grid()
plt.show()
```



Hasil analisis: 1. Produk electronic accessories menjadi kategori paling banyak dibeli (sangat diminati) oleh pelanggan. 2.Produk health and beauty menjadi kategori penjualan terendah (kurang diminati), dengan rentang selisih penjualan yang cukup jauh dari kategori yang lain.

5.2 Rata-rata rating per kategori produk

5.2.1 Metode Bayesian Average

Bayesian Average sering digunakan untuk mendapatkan nilai rating yang lebih adil dan akurat, terutama saat beberapa produk memiliki jumlah rating yang sedikit.

Kelebihan Bayesian: 1. Bias Data dengan Jumlah Rating Sedikit

Produk dengan jumlah rating yang kecil sering memiliki rating tinggi atau rendah yang ekstrem. Bayesian memastikan produk tersebut tidak langsung mendominasi peringkat.

2. Penilaian Lebih Stabil

Bayesian memasukkan informasi prior (rating rata-rata seluruh produk) sebagai dasar penilaian, sehingga nilai rata-rata lebih stabil.

Rumus Bayesian Average:

6 Bayesian Average

$$+ \quad / +$$

Keterangan:

$$= \text{Rata-rata rating produk}$$

$$= \text{Jumlah rating untuk produk tersebut}$$

$$= \text{Rata-rata rating keseluruhan}$$

$$= \text{Threshold jumlah minimum rating yang dianggap stabil}$$

[59]:
```python
# Menghitung rata-rata rating setiap produk menggunakan Bayesian Average
# 1. Rata-rata rating keseluruhan
C = df_supermarket['Rating'].mean()

print(f"C = {C}")

# 2. Menentukan threshold m (jumlah minimum rating dianggap stabil)
# 2a. Menghitung jumlah rating per produk
rating_count = df_supermarket.groupby('Product line')['Rating'].count()

# 2b. Statistik distribusi jumlah rating
mean_count = rating_count.mean()
median_count = rating_count.median()
quantile_25 = rating_count.quantile(0.25)  # Kuartil 25%
quantile_75 = rating_count.quantile(0.75)  # Kuartil 75%

# 2c. Menampilkan distribusi jumlah rating
print(f"\nRata-rata jumlah rating: {mean_count}")
print(f"Median jumlah rating: {median_count}")
print(f"Kuartil 25%: {quantile_25}")
print(f"Kuartil 75%: {quantile_75}")

# 2d. Menentukan threshold (m)
m = int(median_count)  # karena threshold bisa diambil dari median
print(f"\nThreshold (m) yang digunakan: {m}\n")
```

```
C = 6.9727

Rata-rata jumlah rating: 166.66666666666666
Median jumlah rating: 168.0
```

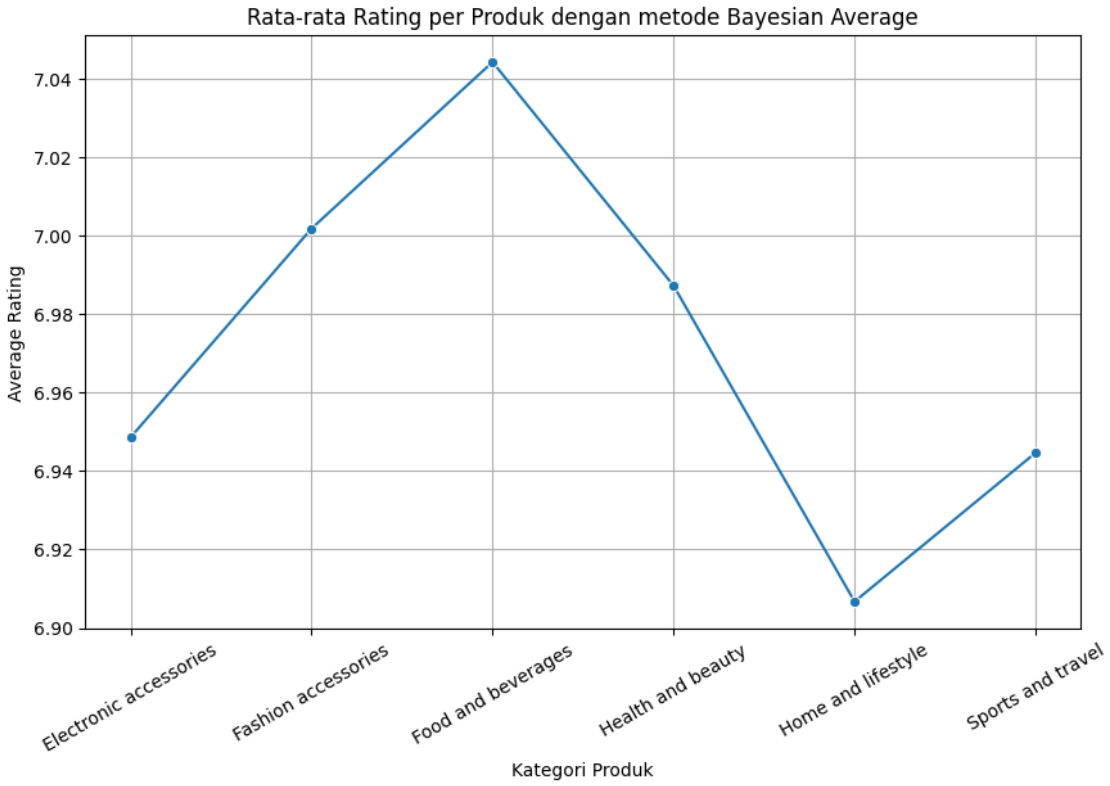
```
Kuartil 25%: 161.5
Kuartil 75%: 173.0

Threshold (m) yang digunakan: 168
```

[60]:
```python
# 3. Hitung Bayesian Average untuk setiap Product line
bayesian_rating = df_supermarket.groupby('Product line').agg(
    total_rating=('Rating', 'sum'),
    count_rating=('Rating', 'count')
).reset_index()

print(bayesian_rating)

# Formula Bayesian Average
bayesian_rating['Bayesian Average'] = (
    (bayesian_rating['count_rating'] * bayesian_rating['total_rating'] /␣
 ↪bayesian_rating['count_rating']) + (m * C)
) / (bayesian_rating['count_rating'] + m)

# 4. Menampilkan hasil
print("\nRata-rata Rating dengan Pendekatan Bayesian:")
print(bayesian_rating[['Product line', 'Bayesian Average']])

# 5. Mengurutkan hasil berdasarkan Bayesian Average
bayesian_sort = bayesian_rating.sort_values(by='Bayesian Average',␣
 ↪ascending=False)

print("\nRata-rata Rating setelah diurutkan:")
print(bayesian_sort[['Product line', 'Bayesian Average']])
```

```
             Product line  total_rating  count_rating
0  Electronic accessories        1177.2           170
1     Fashion accessories        1251.2           178
2      Food and beverages        1237.7           174
3       Health and beauty        1064.5           152
4      Home and lifestyle        1094.0           160
5       Sports and travel        1148.1           166

Rata-rata Rating dengan Pendekatan Bayesian:
             Product line  Bayesian Average
0  Electronic accessories          6.948561
1     Fashion accessories          7.001773
2      Food and beverages          7.044192
3       Health and beauty          6.987230
4      Home and lifestyle          6.906749
5       Sports and travel          6.944651
```

```
Rata-rata Rating setelah diurutkan:
             Product line  Bayesian Average
2      Food and beverages          7.044192
1     Fashion accessories          7.001773
3       Health and beauty          6.987230
0  Electronic accessories          6.948561
5       Sports and travel          6.944651
4      Home and lifestyle          6.906749
```

[61]:
```
# Visualisasi rata-rata rating
plt.figure(figsize=(10, 6))
sns.lineplot(data=bayesian_rating, x='Product line', y='Bayesian Average',␣
↪marker='o')
plt.title('Rata-rata Rating per Produk dengan metode Bayesian Average')
plt.xlabel('Kategori Produk')
plt.ylabel('Average Rating')
plt.xticks(rotation=30)
plt.grid()
plt.show()
```



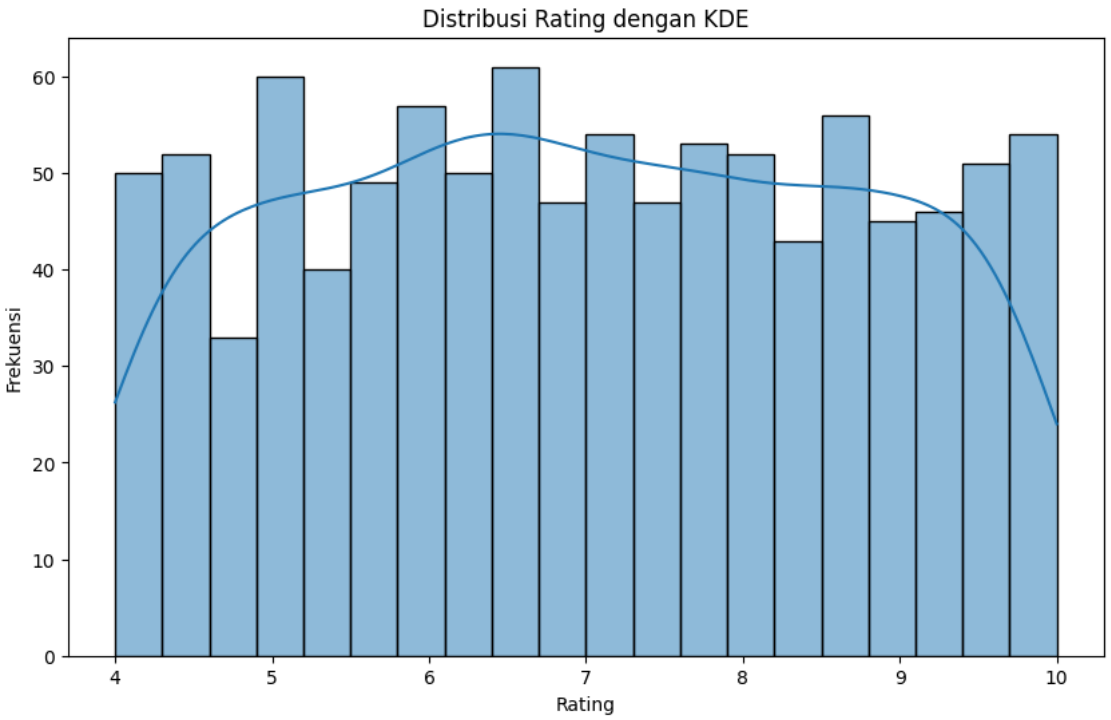
Hasil analisis: 1. Produk food and beverages menjadi kategori dengan rata-rata rating paling tinggi, diikuti dengan produk fashion accessories dan health and beauty. 2. Produk home and

lifestyle mendapat rata-rata rating terendah dari pelanggan.

Hasil analisis hubungan antara penjualan terbanyak dan rata-rata rating: 1. Electronic accessories menjadi penjualan terbanyak dengan 971 penjualan tetapi menempati posisi keempat untuk urutan rating pelanggan dengan rata-rata 6,948. Hal ini menunjukkan bahwa pelanggan sangat berminat dengan produk electronic accessories tapi kualitas produk perlu ditingkatkan. 2. Produk Fashion accessories dan Health and Beauty memiliki rating tinggi dengan rata-rata 7,001 dan 6,987. Namun, dari kuantitas penjualan, kedua jenis produk ini berada di posisi dua terendah dengan penjualan sebanyak 902 dan 854. Sehingga promosi untuk kedua produk tersebut perlu lebih difokuskan untuk menarik minat pelanggan.

6.0.1 Distribusi rating dengan Kernel Density Estimation

[62]:
```
# Visualisasi distribusi rating menggunakaan histogram dengan KDE
plt.figure(figsize=(10, 6))
sns.histplot(df_supermarket['Rating'], bins=20, kde=True)
plt.title('Distribusi Rating dengan KDE')
plt.xlabel('Rating')
plt.ylabel('Frekuensi')
plt.show()
```



Hasil analisis: 1. Bentuk Distribusi

- Kurva KDE menunjukkan distribusi relatif merata di antara nilai rating 4 hingga 10.

- Puncak distribusi berada di sekitar nilai 6 hingga 7, yang berarti sebagian besar rating cenderung terkumpul di nilai tersebut.
- Tidak ada nilai rating yang mendominasi secara signifikan, tetapi terdapat tren kecil yang menunjukkan sedikit peningkatan frekuensi rating di sekitar nilai 6 hingga 7.

2. Nilai Ekstrim

- Rating 4 dan Rating 10 memiliki frekuensi yang sedikit lebih rendah dibanding nilai di tengah-tengah. Namun, frekuensi untuk nilai ekstrim ini tidak sepenuhnya terabaikan, artinya masih ada pengguna yang memberi rating rendah (4) atau rating sempurna (10).

3. Analisis Tren Umum

- Rating sedang (6-7) paling sering muncul, menunjukkan sebagian besar penilaian berkisar pada penilaian netral atau cukup positif.
- Kurva KDE menunjukkan distribusi yang mendekati seragam, tidak terlalu condong ke sisi kiri (rating rendah) maupun kanan (rating tinggi). Ini menunjukkan kualitas produk yang dinilai cukup konsisten.

4. Kualitas Data

- KDE yang mulus menunjukkan distribusi data yang cukup stabil, tanpa adanya lonjakan atau gap yang signifikan antara rating. Ini berarti tidak ada outlier mencolok yang mendistorsi distribusi rating.

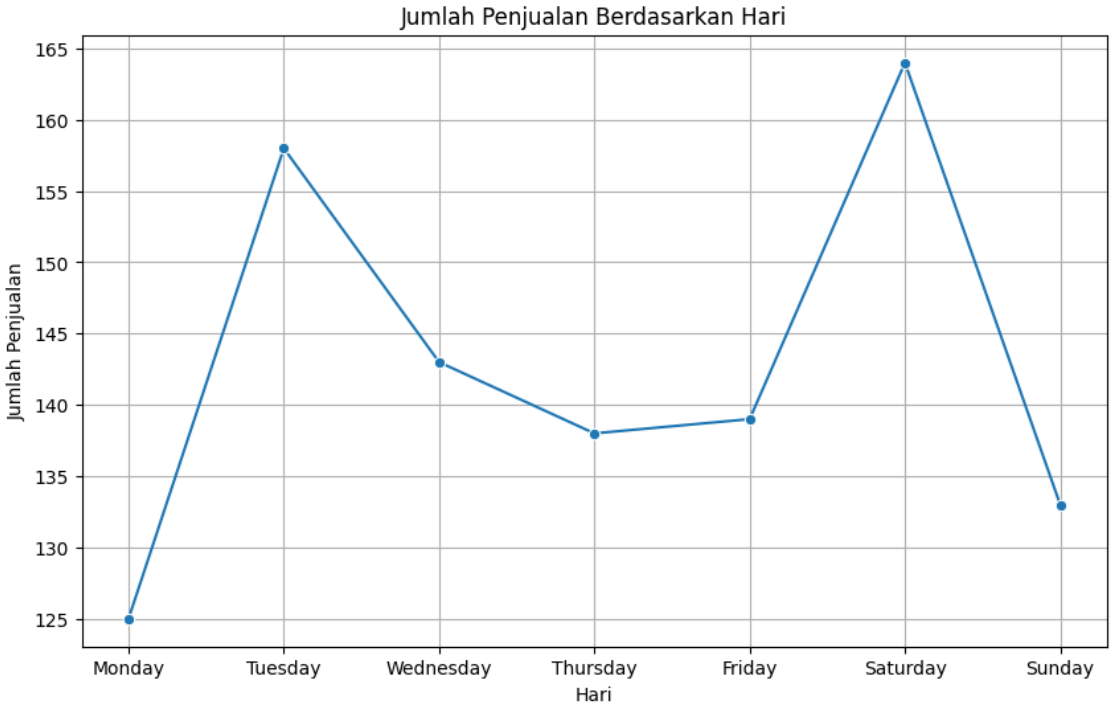
7 Analisis waktu penjualan

7.0.1 Tren penjualan berdasarkan hari

[63]:
```
# Konversi kolom 'Date' ke format datetime
df_supermarket['Date'] = pd.to_datetime(df_supermarket['Date'])
df_supermarket['Day'] = df_supermarket['Date'].dt.day_name()
df_supermarket['Hour'] = pd.to_datetime(df_supermarket['Time'], format='%H:%M').
 ↪dt.hour

# Analisis penjualan per hari
sales_per_day = df_supermarket['Day'].value_counts().reset_index()
sales_per_day.columns = ['Day', 'Sales']

# Mengurutkan berdasarkan urutan hari
order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',␣
 ↪'Sunday']
sales_per_day['Day'] = pd.Categorical(sales_per_day['Day'], categories=order,␣
 ↪ordered=True)
sales_per_day = sales_per_day.sort_values('Day')

print("\nPenjualan Berdasarkan Hari:")
print(sales_per_day)
```

```
Penjualan Berdasarkan Hari:
         Day  Sales
6     Monday    125
1    Tuesday    158
2  Wednesday    143
4   Thursday    138
3     Friday    139
0   Saturday    164
5     Sunday    133
```

[64]:
```
# Visualisasi penjualan per hari
plt.figure(figsize=(10, 6))
sns.lineplot(data=sales_per_day, x='Day', y='Sales', marker='o')
plt.title('Jumlah Penjualan Berdasarkan Hari')
plt.xlabel('Hari')
plt.ylabel('Jumlah Penjualan')
plt.grid()
plt.show()
```



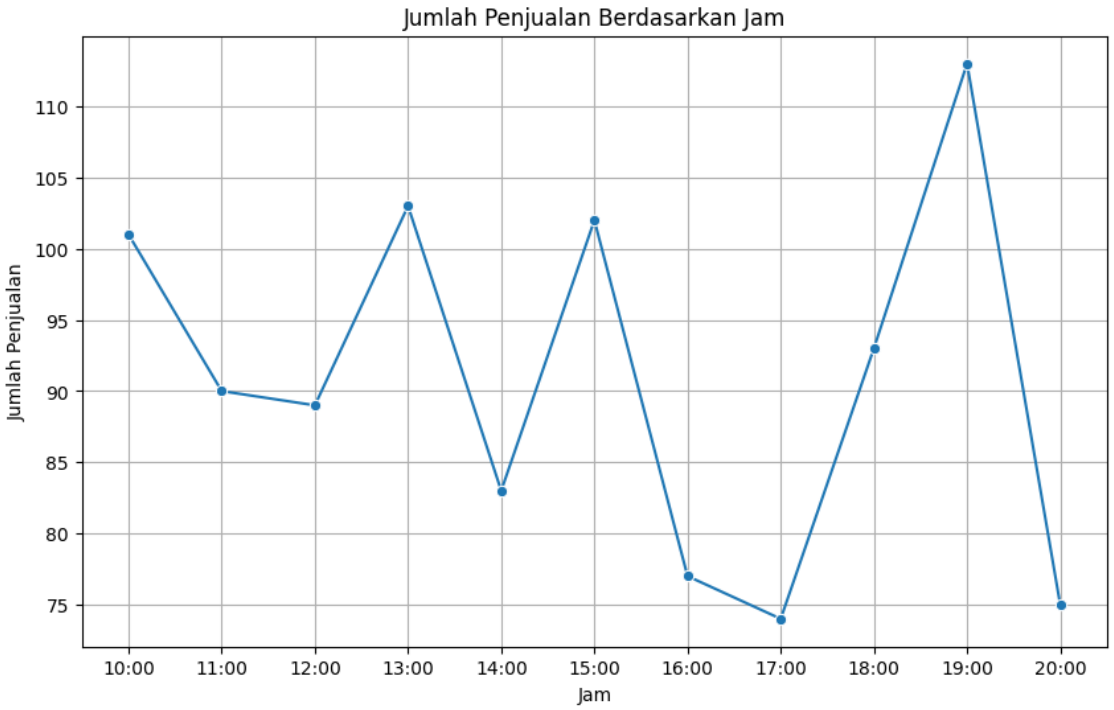
Hasil analisis: 1. Penjualan tertinggi terjadi pada hari Sabtu, menunjukkan bahwa akhir pekan (terutama Sabtu) adalah waktu dengan aktivitas pembelian paling banyak. 2. Penjualan rendah terjadi pada hari Minggu dan Senin, hal ini menunjukkan aktivitas penjualan lebih sedikit pada hari tersebut. 3. Secara keseluruhan, ada pola peningkatan penjualan mendekati akhir pekan, terutama dari hari Kamis ke Sabtu, dan ada pola penurunan dari hari Selasa ke Kamis.

7.0.2 Tren penjualan berdasarkan jam operasional

[65]:
```
# Konversi kolom 'Date' ke format datetime
df_supermarket['Date'] = pd.to_datetime(df_supermarket['Date'])
df_supermarket['Day'] = df_supermarket['Date'].dt.day_name()
df_supermarket['Hour'] = pd.to_datetime(df_supermarket['Time'], format='%H:%M').
 ↪dt.hour

# Analisis penjualan per jam
df_supermarket['Hour'] = df_supermarket['Hour'].apply(lambda x: f"{x:02}:00")
sales_per_hour = df_supermarket['Hour'].value_counts().reset_index()
sales_per_hour.columns = ['Hour', 'Sales']
sales_per_hour = sales_per_hour.sort_values('Hour')

print("\nPenjualan Berdasarkan Jam:")
print(sales_per_hour)
```

```
Penjualan Berdasarkan Jam:
     Hour  Sales
3   10:00    101
5   11:00     90
6   12:00     89
1   13:00    103
7   14:00     83
2   15:00    102
8   16:00     77
10  17:00     74
4   18:00     93
0   19:00    113
9   20:00     75
```

[66]:
```
# Visualisasi penjualan per jam operasional
plt.figure(figsize=(10, 6))
sns.lineplot(data=sales_per_hour, x='Hour', y='Sales', marker='o')
plt.title('Jumlah Penjualan Berdasarkan Jam')
plt.xlabel('Jam')
plt.ylabel('Jumlah Penjualan')
plt.grid()
plt.show()
```



Hasil analisis: 1. Penjualan tertinggi erjadi pada pukul 19:00. Ini menunjukkan bahwa jam 7 malam adalah waktu puncak aktivitas penjualan. 2. Penjualan terendah terjadi pada rentang pukul 16:00–17:00. Ini menunjukkan penurunan tajam di sore hari sebelum akhirnya meningkat kembali. 3. Pola fluktuasi secara keseluruhan:

- Penjualan cenderung menurun pada rentang pukul 10:00 hingga 12:00
- Peningkatan signifikan terjadi pada pukul 13:00 dan 15:00.
- Penjualan mengalami penurunan tajam setelah pukul 15:00 hingga pukul 17:00.
- Penjualan kembali naik drastis mulai pukul 18:00 dan mencapai puncaknya pada pukul 19:00

- Penjualan menurun kembali pada pukul 20:00

8 Export to PDF

[67]:
```
!apt-get install pandoc
!apt-get update
!apt-get install -y texlive-xetex texlive-fonts-recommended texlive-latex-extra
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
pandoc is already the newest version (2.9.2.1-3ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 54 not upgraded.
Hit:1 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease
```

```
Hit:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64
InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://r2u.stat.illinois.edu/ubuntu jammy InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:8 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:9 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy
InRelease
Hit:10 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Reading package lists… Done
W: Skipping acquire of configured file 'main/source/Sources' as repository
'https://r2u.stat.illinois.edu/ubuntu jammy InRelease' does not seem to provide
it (sources.list entry misspelt?)
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
texlive-fonts-recommended is already the newest version (2021.20220204-1).
texlive-latex-extra is already the newest version (2021.20220204-1).
texlive-xetex is already the newest version (2021.20220204-1).
0 upgraded, 0 newly installed, 0 to remove and 54 not upgraded.
```

[68]:
```
from google.colab import drive
drive.mount('/content/drive')

# Copy the notebook file from Google Drive to the current working directory
!cp "/content/drive/My Drive/Colab Notebooks/Statprob24_FP.ipynb" ./

# Convert the notebook to a PDF
!jupyter nbconvert --to PDF "./Statprob24_FP.ipynb"
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
[NbConvertApp] Converting notebook ./Statprob24_FP.ipynb to PDF
[NbConvertApp] Support files will be in Statprob24_FP_files/
[NbConvertApp] Making directory ./Statprob24_FP_files
[NbConvertApp] Writing 143174 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 507403 bytes to Statprob24_FP.pdf
```