

Visualisasi Data dengan Python Matplotlib

Bagian II

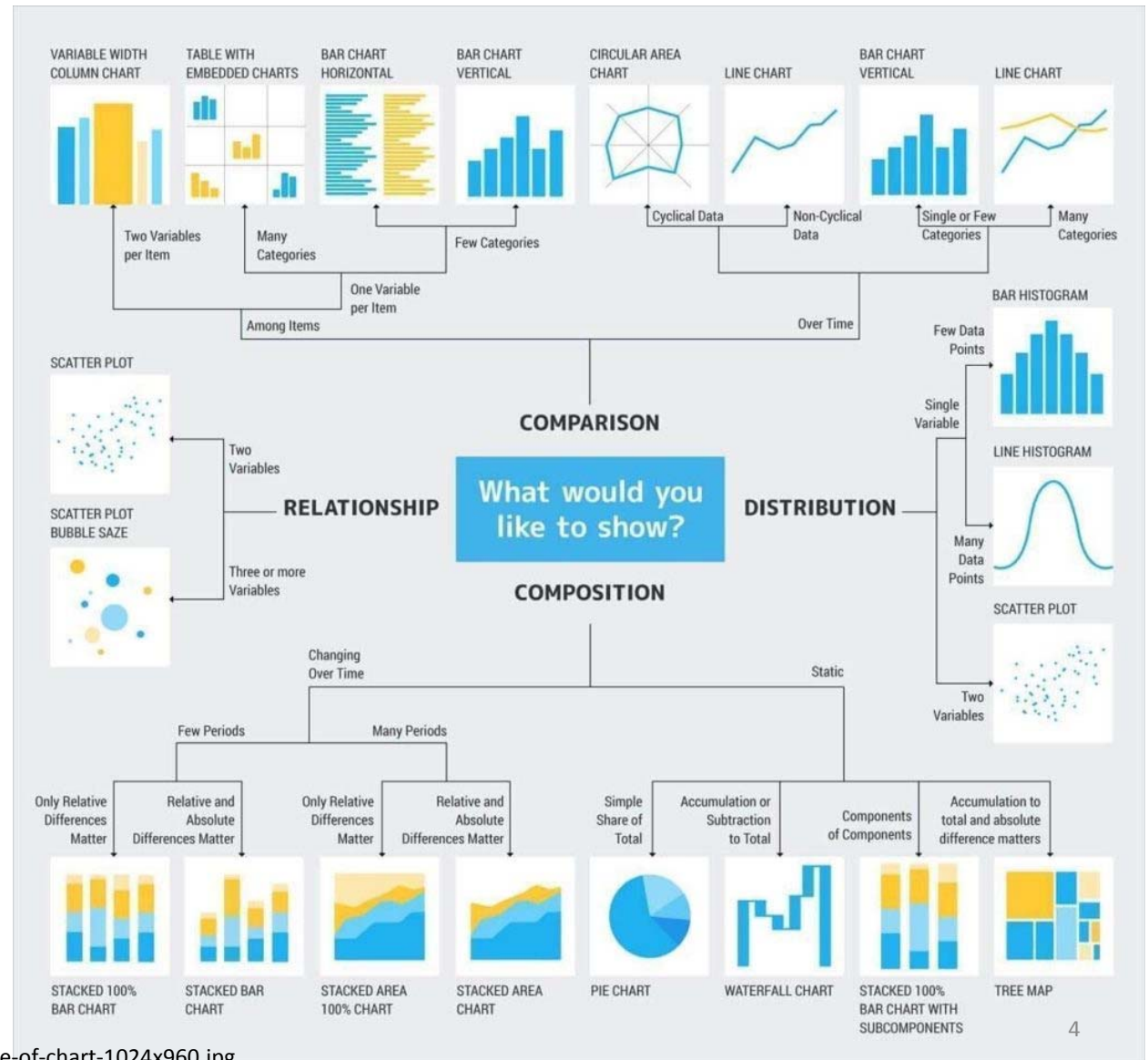
Topik

- Jenis-Jenis Visualisasi Data dan Tujuannya
 - Pendahuluan
 - Mengenal Jenis-Jenis Visualisasi Data dan Tujuannya
- Membuat Multi-line Chart dan Kustomisasi Lanjutan
 - Membuat Multi-Line Chart
 - Kustomisasi Legend
 - Kustomisasi Colormap
 - Membuat Line Chart GMV Breakdown by Top Provinces
 - Membuat Anotasi
- Membuat Pie-Chart dan Bar-Chart
 - Membuat Subset Data
 - Membuat Pie Chart
 - Membuat Bar Chart
 - Membuat Multi-Bar Chart
 - Membuat Stacked Chart

Mengenal Jenis-Jenis Visualisasi Data dan Tujuannya

- Python menyediakan berbagai jenis visualisasi data dimulai dari tampilan yang sederhana hingga yang kompleks.
- Setiap jenis visualisasi biasanya akan lebih tepat jika digunakan untuk suatu tujuan tertentu.
- Secara garis besar, ada empat tujuan visualisasi menurut Andrew Abela, yaitu:
 - Perbandingan/Komparasi (Comparison)
 - Distribusi (Distribution)
 - Komposisi (Composition)
 - Keterhubungan/Relasi (Relationship)

Mengenal Jenis- Jenis Visualisasi Data dan Tujuannya



Mengenal Jenis-Jenis Visualisasi Data dan Tujuannya

1. Perbandingan/Komparasi

- Digunakan untuk membandingkan sekumpulan nilai dengan nilai lainnya atau ingin melihat mana yang lebih besar, mana yang lebih tinggi, mana yang mengalami kenaikan, dsb.
- Untuk data yang berkaitan dengan **waktu**, umumnya divisualisasikan dengan **line-chart**.
- Jika nilai yang dibandingkan bersifat **kategorikal**, akan lebih cocok ditampilkan dengan **bar-chart**.

2. Distribusi

- Digunakan untuk melihat distribusi/persebaran data di suatu variabel.
- Visualisasi yang umum digunakan adalah **histogram**.
- Dari tampilan visualisai tersebut, bisa dilihat apakah datanya banyak terkumpul pada nilai-nilai yang kecil, berat di kanan, simetris, ataupun memiliki distribusi normal.
- Untuk melihat distribusi pada dua variabel, **scatterplot** bisa digunakan sebagai alternatifnya.

Mengenal Jenis-Jenis Visualisasi Data dan Tujuannya

3. Komposisi

- Digunakan untuk melihat komposisi dari suatu variabel jika di-breakdown terhadap suatu dimensi data.
- Visualisasi yang biasa digunakan adalah **stacked bar-chart** untuk jenis data **kategorikal** atau **stacked line-chart** untuk menampilkan data yang berkaitan dengan **waktu**.

4. Keterhubungan/Relasi

- Digunakan ketika ingin melihat keterhubungan antara suatu variabel dengan variabel lain.
- Misal, jika variabel A semakin tinggi, apakah variabel B juga semakin tinggi, atau justru lebih rendah, atau malah tidak ada keterhubungannya.
- Visualisasi yang biasa digunakan untuk tujuan ini adalah **scatter-plot**.

Membuat Multi-Line Chart

- Menggali secara lebih dalam, hal/alasan apa yang membuat nilai Gross Merchandise Value (GMV) meningkat.
- Sehingga informasi ini bisa diberikan ke Business Development sebagai pertimbangan dalam menentukan strategi bisnis ke depannya.

Membuat Multi-Line Chart

- Menggunakan dataset `retail_raw_reduced.csv`, lakukan breakdown berdasarkan dimensinya, misalnya berdasarkan kolom **brand** dan **order_month**.
- Dari hasil breakdown tersebut, dapat dilihat bagian mana yang paling berperan dalam kenaikan GMV tersebut.
- Dalam melakukan breakdown, gunakan fungsi **.unstack()** untuk menghitung jumlah seluruh nilai **gmv**.
- Terlebih dahulu ubah format tanggal supaya hanya menampilkan tahun dan bulan saja.

Kode untuk Membuat Multi-line Chart

Prerequisite I: Menyiapkan dataset

```
import pandas as pd
dataset = pd.read_csv('retail_raw_reduced.csv')
print('Ukuran dataset: %d baris dan %d kolom\n' % dataset.shape)
print('Lima data teratas:')
print(dataset.head())
```

Prerequisite II: Mengubah format tanggal supaya hanya menampilkan tahun dan bulan saja

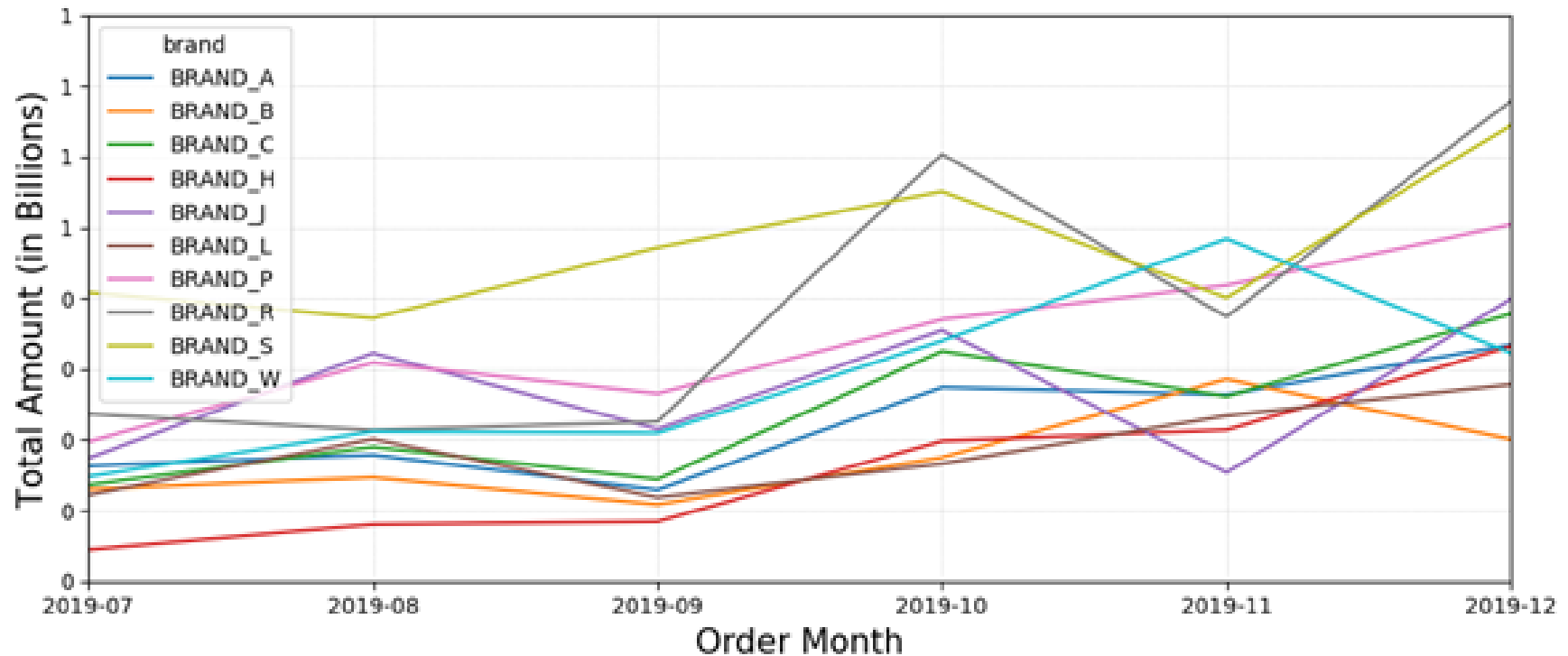
```
import datetime
dataset['order_month'] = dataset['order_date'].apply(lambda x:
datetime.datetime.strptime(x, "%Y-%m-%d").strftime('%Y-%m'))
print(dataset.head())
```

Kode untuk Membuat Multi-line Chart

- Breakdown berdasarkan dimensinya, misalnya berdasarkan kolom **brand** dan **order_month**.
- Gunakan fungsi **.unstack()** untuk menghitung jumlah seluruh nilai **gmv**.

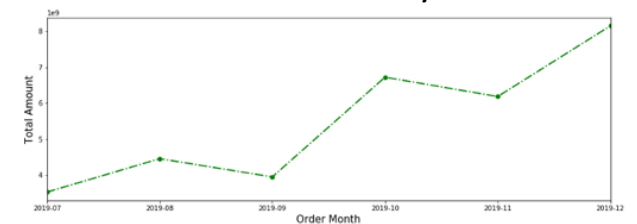
```
12 # Buat Multi-Line Chart
13 dataset.groupby(['order_month', 'brand'])['gmv'].sum().unstack().plot()
14 plt.title('Monthly GMV Year 2019 - Breakdown by
Brand', loc='center', pad=30, fontsize=20, color='blue')
15 plt.xlabel('Order Month', fontsize = 15)
16 plt.ylabel('Total Amount (in Billions)', fontsize = 15)
17 plt.grid(color='darkgray', linestyle=':', linewidth=0.5)
18 plt.ylim(ymin=0)
19 labels, locations = plt.yticks()
20 plt.yticks(labels, (labels/1000000000).astype(int))
21 plt.gcf().set_size_inches(10, 5)
22 plt.tight_layout()
23 plt.show()
```

Monthly GMV Year 2019 - Breakdown by Brand



- Dibandingkan dengan grafik sebelumnya, grafik ini menjadi lebih **beragam warna karena grafik ditampilkan dengan mengelompokkannya berdasarkan brandnya**.
- Sehingga, warna-warna tersebut secara otomatis akan mengikuti varian dari brand yang dikelompokkan.

Grafik sebelumnya



Kustomisasi Legend

- Grafik output memiliki komponen baru, yaitu legend.
- Karena brand berjumlah banyak dan letak secara default di tengah, nampak bahwa legend tersebut menutupi grafik.
- Supaya tidak menutupi grafik, digunakan function khusus untuk mengatur komponen legend, yaitu function **plt.legend()**.
- Beberapa parameter yang bisa ditambahkan pada function tersebut adalah:

1. **loc**: untuk menentukan posisi legend.

Berikut ini beberapa lokasi legend yang bisa definisikan:

- ***'upper left', 'upper right', 'lower left', 'lower right'***: berturut-turut untuk menempatkan legend di pojok atas kiri, atas kanan, bawah kiri, dan bawah kanan dari axes.
- ***'upper center', 'lower center', 'center left', 'center right'***: berturut-turut untuk meletakkan legend di tepi atas tengah, bawah tengah, tengah kiri, dan tengah kanan dari axes.
- ***'center'***: untuk menempatkan legend di tengah-tengah axes.

Kustomisasi Legend

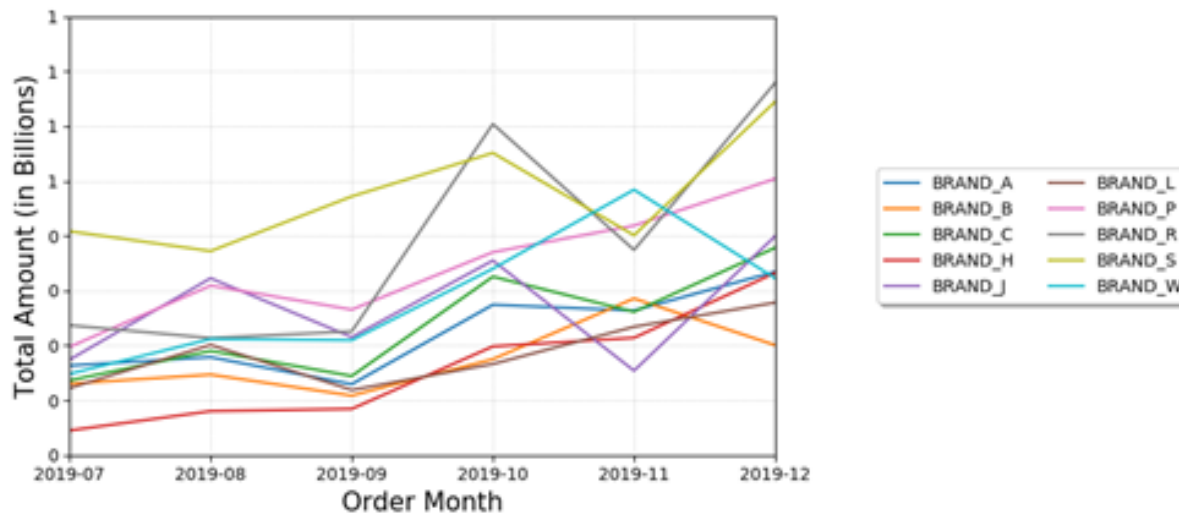
2. **'best'**: jika terjadi kebingungan dalam menentukan posisi yang terbaik, matplotlib dapat memilih salah satu dari sekian kemungkinan lokasi legend yang telah disebutkan di atas yang paling tidak overlap dengan isi grafik.
3. **bbox_to_anchor**: biasanya digunakan untuk melakukan adjust lokasi dari legend. Bisa berisi 2 angka yang menunjukkan koordinat x dan y, misalnya (1.6,0.5) yang berarti menggeser sejauh 1.6 ke kanan dan 0.5 ke atas. Selain itu, parameter ini juga bisa berisi 4 angka, dimana angka ketiga dan keempatnya menyatakan width (lebar) dan height (tinggi) dari legend.
4. **shadow**: jika diisi True, maka kotak legend akan memiliki bayangan.
5. **ncol**: jumlah kolom dari isi legend. Nilai defaultnya adalah 1.
6. **fontsize**: ukuran huruf pada legend.
7. **title**: untuk memberikan judul pada legend.
8. **title_fontsize**: ukuran huruf pada judul legend

Kustomisasi Legend

- Mengatur letak legend grafik di sebelah kanan dengan posisi di **1.6** sumbu-x dan **0.5** di sumbu-y.
- Supaya tidak terkesan terlalu penuh, set legend supaya memiliki **2** kolom.
- Menambahkan bayangan pada legend supaya terlihat hidup.

```
10 plt.legend(loc='right', bbox_to_anchor=(1.6, 0.5), shadow=True, ncol=2)
```

Monthly GMV Year 2019 - Breakdown by Brand



Didapatkan grafik dengan legend yang lebih rapi

Kustomisasi Colormap

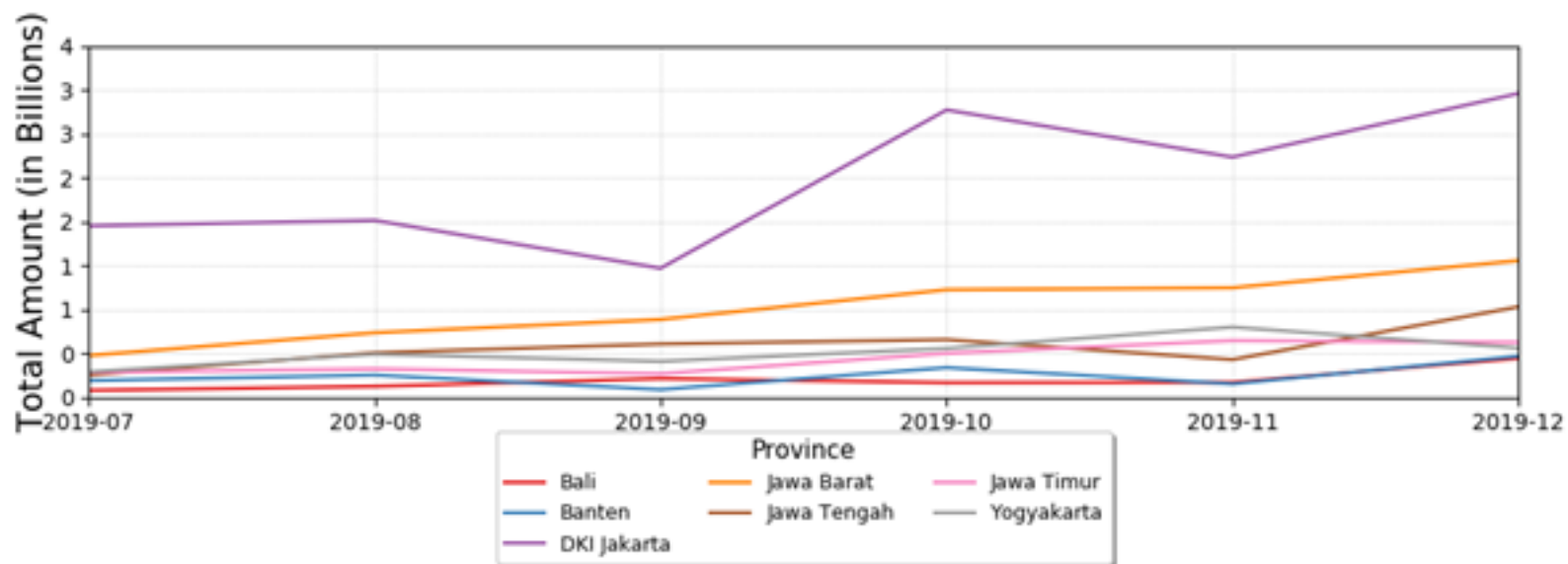
- Setelah melakukan breakdown berdasarkan brand, terlihat bahwa trend sepanjang Oktober-Desember 2019 ternyata mirip semua.
- Jadi, kemungkinan besar tidak ada faktor dari brand yang membuat nilai GMV naik.
- Sebagai alternative dalam mencari faktor yang membuat GMV naik, coba lakukan breakdown berdasarkan province.
- Di sini, kita hanya perlu mengganti kolom **brand** dengan **province** di **.groupby()**.
- Untuk legend-nya, lakukan sedikit perubahan dengan meletakkannya di **bawah tengah** dengan posisi koordinat **0.5** di sumbu-x dan **-0.5** sumbu-y.
- Supaya lebih detil, tambahkan judul '**Province**' pada legend tersebut.

```
1 import matplotlib.pyplot as plt
2 plt.clf()
3 dataset.groupby(['order_month', 'province'])['gmv'].sum().unstack().plot(cmap='Set1')
4 plt.title('Monthly GMV Year 2019 - Breakdown by Province', loc='center', pad=30, fontsize=20,
5 color='blue')
6 plt.xlabel('Order Month', fontsize = 15)
7 plt.ylabel('Total Amount (in Billions)', fontsize = 15)
8 plt.grid(color='darkgray', linestyle=':', linewidth=0.5)
9 plt.ylim(ymin=0)
10 labels, locations = plt.yticks()
11 plt.yticks(labels, (labels/1000000000).astype(int))
12 plt.legend(loc='lower center', bbox_to_anchor=(0.5, -0.5), shadow=True, ncol=3,
13 title='Province', fontsize=9, title_fontsize=11)
14 plt.gcf().set_size_inches(10, 5)
15 plt.tight_layout()
16 plt.show()
```

sebelumnya 'brand'

Kustomisasi Colormap

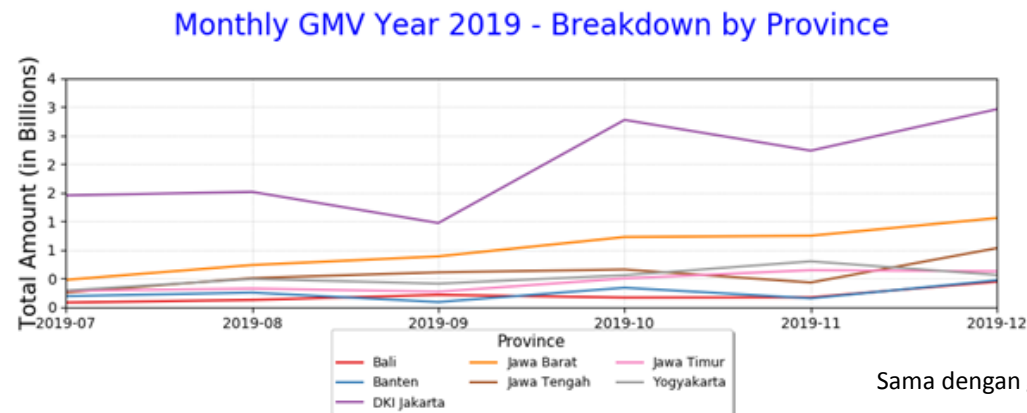
Monthly GMV Year 2019 - Breakdown by Province



- Tampak posisi legendnya berpindah tempat di bawah.
- Jika dibandingkan dengan output sebelumnya, legend di grafik ini sudah memiliki judul sehingga akan lebih jelas variabel apa yang digunakan untuk membandingkan data tersebut.

Kustomisasi Colormap

- Selain itu, perhatikan juga bahwa list warna pada grafik tersebut sudah berbeda, tidak default seperti pada waktu menggunakan **brand**.
- Mengapa bisa terjadi? Hal ini disebabkan karena adanya parameter **cmap** (color map) pada function **.plot()**.
- Parameter tersebut berfungsi untuk mendefinisikan sekelompok warna yang akan diberikan pada tiap line.
- Di sini yang digunakan adalah '**Set1**', yaitu satu set warna yang biasa digunakan untuk warna yang diskrit.
- Untuk mengetahui jenis-jenis color map lebih lanjut: <https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html>.

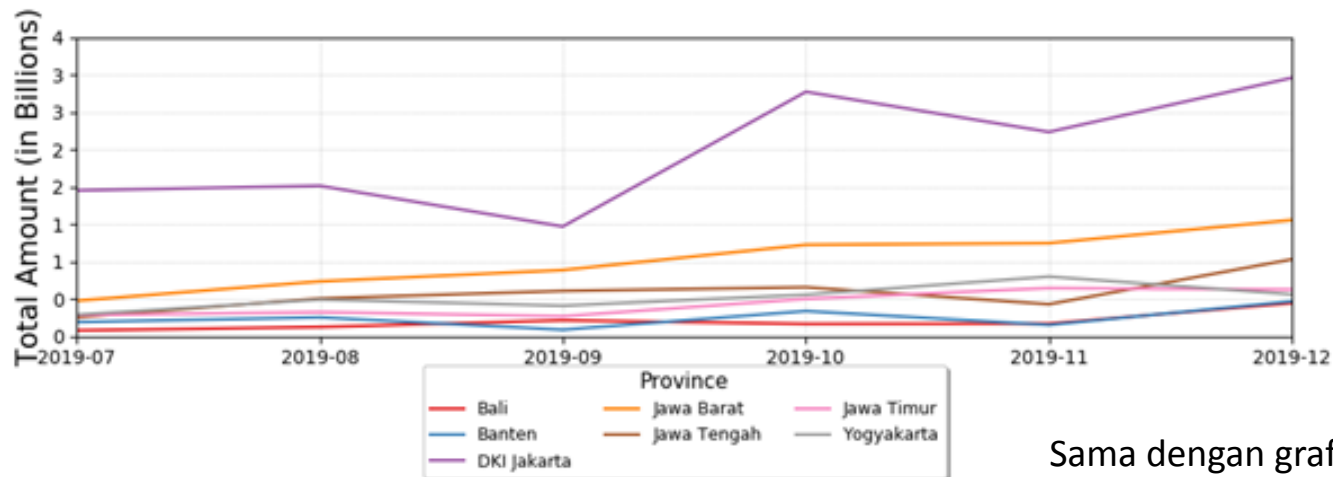


Sama dengan grafik di hal 16

Kustomisasi Colormap

- Dari grafik output, terlihat bahwa ada beberapa provinsi yang mendominasi kenaikan dari GMV, seperti DKI Jakarta dan Jawa Barat.
- Namun, karena jumlah provinsi yang begitu banyak, provinsi-provinsi lain yang ada di bawah jadi kurang terlihat jelas karena bercampur semua garisnya.
- Supaya terlihat lebih jelas, kita akan memperbaiki grafik ini dengan membuat *highlight* terhadap top 5 provincenya terlebih dahulu.

Monthly GMV Year 2019 - Breakdown by Province



Sama dengan grafik di hal 16 18

Membuat Line Chart GMV Breakdown by Top Provinces

- Dari grafik sebelumnya sudah terlihat cukup jelas bahwa kenaikan GMV dipengaruhi oleh provinsi.
- Oleh karena itu fokus diberikan terhadap provinsi-provinsi yang memiliki GMV besar, lakukan *highlight* untuk top 5 provinsi saja, sedangkan untuk provinsi lainnya kita akan menggabungkannya menjadi **'other'**.
- Langkah pertama, cek lima provinsi dengan GMV tertinggi terlebih dahulu dengan memasukkannya ke dalam variabel **top_provinces** seperti ini.

Output dari statement pada baris ke-7 berupa:

```
1 # Buat variabel untuk 5 propinsi dengan GMV tertinggi
2 top_provinces = (dataset.groupby('province')['gmv']
3                 .sum()
4                 .reset_index()
5                 .sort_values(by='gmv', ascending=False)
6                 .head(5))
7 print(top_provinces)
```

province	gmv
DKI Jakarta	14902763000
Jawa Barat	6128149000
Jawa Tengah	3485884000
Yogyakarta	3111241000
Jawa Timur	2649486000

Membuat Line Chart GMV Breakdown by Top Provinces

- Setelah itu, perlu dibuat kolom baru untuk memasukkan data-data dari provinsi mana yang masuk dalam variabel **top_provinces**.
- Jika ada yang tidak termasuk dalam variabel top_provinces, kita perlu mengganti namanya menjadi **'other'**.
- Manfaatkan function **.apply()** & **lambda** untuk membuat kolom ini

```
9 # Buat satu kolom lagi di dataset dengan nama province_top
10 dataset['province_top'] = dataset['province'].apply(lambda x: x if (x in
    top_provinces['province'].to_list()) else 'other')
```

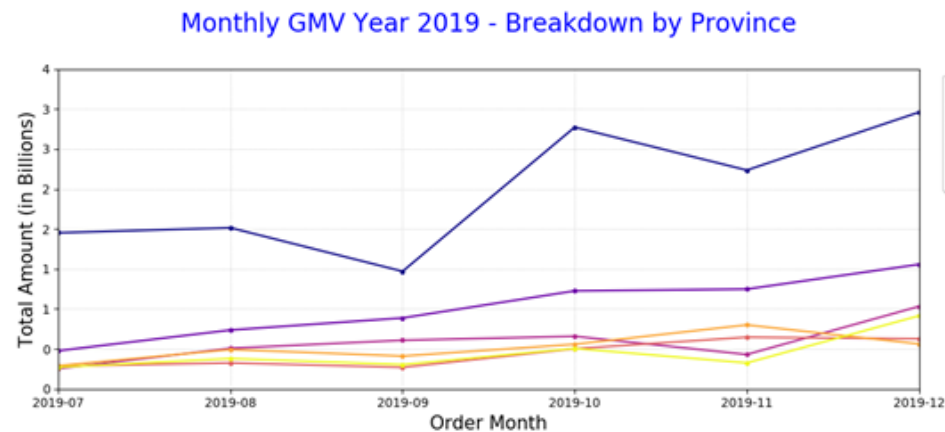
Membuat Line Chart GMV Breakdown by Top Provinces

- Setelah memiliki kolom baru, dilakukan update terhadap grafik.
- Legend ditempatkan di atas tengah dengan posisi **1. 1** di sumbu-x dan **1** di sumbu y.
- Untuk mengubah ukuran figure, bisa menggunakan function **plt.gcf().set_size_inches** yang diletakkan di akhir code sebelum plt.show().

```
12 # Plot multi-line chartnya
13 import matplotlib.pyplot as plt
14 dataset.groupby(['order_month', 'province_top'])['gmv'].sum().unstack().plot(marker='.',
    cmap='plasma')
15 plt.title('Monthly GMV Year 2019 - Breakdown by Province', loc='center', pad=30,
    fontsize=20, color='blue')
16 plt.xlabel('Order Month', fontsize = 15)
17 plt.ylabel('Total Amount (in Billions)', fontsize = 15)
18 plt.grid(color='darkgray', linestyle=':', linewidth=0.5)
19 plt.ylim(ymin=0)
20 labels, locations = plt.yticks()
21 plt.yticks(labels, (labels/1000000000).astype(int))
22 plt.legend(loc='upper center', bbox_to_anchor=(1.1, 1), shadow=True, ncol=1)
23 plt.gcf().set_size_inches(12, 5)
24 plt.tight_layout()
25 plt.show()
```

Membuat Line Chart GMV Breakdown by Top Provinces

- Setelah diubah ukuran figurenya, sekarang grafiknya sudah terlihat lebih memanjang.
- Begitu juga dengan data-data yang ditampilkan di grafik, sekarang sudah terlihat lebih rapi dan jelas setelah kita melakukan highlight terhadap top 5 province-nya.



Membuat Anotasi

- Setelah merapikan grafik, tambahkan **anotasi** supaya grafik lebih informatif.
- Anotasi mirip seperti teks, hanya saja ada suatu titik tertentu yang ditunjuk di dalamnya sebagai informasi tambahan.
- Selain teks, bisa menambahkan panah melalui anotasi tersebut.
- Untuk membuat anotasi, gunakan function **plt.annotate()**.
- Parameter pertama menunjukkan isi dari teks, sedangkan parameter kedua menunjukkan koordinat dari point yang akan dianotasi.
- Selain kedua parameter tersebut, ada beberapa parameter untuk anotasi yang bisa ditambahkan, yaitu:
 1. **xytext** : menentukan koordinat dari teks. Jika tidak diisi, maka teks akan diletakkan di koordinat point yang akan dianotasi, yaitu di parameter kedua.

Membuat Anotasi

2. **Arrowprops** : untuk menentukan konfigurasi dari panah yang ditambahkan di anotasi yang berupa dictionary. Beberapa hal yang bisa diset dari arrowprops ini yaitu:
- **arrowstyle**: untuk menentukan bentuk dari panah penunjuk. Isinya bisa berupa bentuk seperti ' $\rightarrow$ ', ' $\rightarrow|$ ', ' $\leftrightarrow$ ', ' $\leftarrow$ ' atau bisa juga berupa tulisan seperti 'fancy', 'simple', 'wedge', etc.
 - **connectionstyle**: untuk menentukan bentuk dari garis panahnya. Ada beberapa nilai yang bisa digunakan, misalnya:
 - 'arc' dan 'arc3' yang berupa garis lurus,
 - 'angle' untuk garis berbelok siku,
 - 'angle3' untuk garis berbelok lengkung, atau
 - 'bar' untuk garis yang berbelok siku sebanyak dua kali.
 - **color**: untuk menentukan warna dari panah

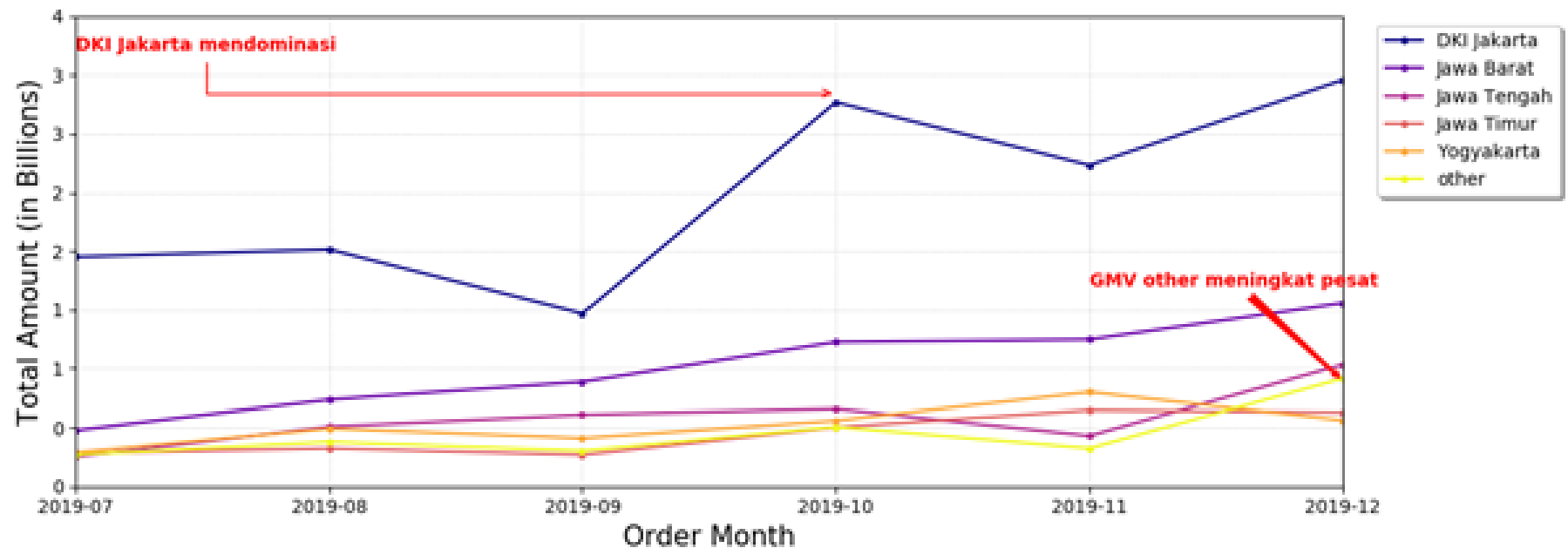
Membuat Anotasi

- Dengan menggunakan parameter-parameter tersebut, kita dapat menganotasi multi-line chart, yaitu **peningkatan GMV other** dan **jumlah GMV DKI Jakarta** yang mendominasi diantara top 5 province lainnya.
- Di sini, warna yang akan kita gunakan untuk anotasinya adalah warna **merah**, dengan posisi teksnya disesuaikan dengan letak nilai yang mau dianotasikan.

```
11 # Anotasi pertama
12 plt.annotate('GMV other meningkat pesat', xy=(5, 9000000000),
13             xytext=(4, 17000000000), weight='bold', color='red',
14             arrowprops=dict(arrowstyle='fancy',
15                             connectionstyle="arc3",
16                             color='red'))
17 # Anotasi kedua
18 plt.annotate('DKI Jakarta mendominasi', xy=(3, 33500000000),
19             xytext=(0, 37000000000), weight='bold', color='red',
20             arrowprops=dict(arrowstyle='->',
21                             connectionstyle="angle",
22                             color='red'))
```

Membuat Anotasi

Monthly GMV Year 2019 - Breakdown by Province



Membuat Subset Data

- Setelah melakukan analisis pada top 5 provinsi dengan GMV tertinggi, lakukan analisis lebih lanjut terhadap behavior transaksi di DKI Jakarta pada kurun waktu Oktober-Desember 2019 ini, mengingat trend-nya mirip dengan overall GMV.
- Cara yang dapat dilakukan adalah dengan mengecek dulu perbandingan GMV di kota-kota yang tersebar di DKI Jakarta. Hal ini dapat dilakukan dengan memfilter data-data dari provinsi DKI Jakarta selama bulan Oktober-Desember 2019. Hasil filter data ini selanjutnya akan disimpan sebagai dataframe bernama **dataset_dki_q4**.

```
1 dataset_dki_q4 = dataset[(dataset['province']=='DKI Jakarta') &
  (dataset['order_month'] >= '2019-10')]
2 print(dataset_dki_q4.head())
```

Membuat Subset Data

- Setelah data difilter, terlihat bahwa data-data yang tersedia sekarang adalah untuk Provinsi DKI Jakarta saja sesuai yang diinginkan.

```
   order_id  order_date  customer_id  ... order_month      gmV province_top
0   1703458  2019-10-17         14004  ...   2019-10   7400000    DKI Jakarta
1   1706815  2019-10-24         17220  ...   2019-10   1208000    DKI Jakarta
2   1710718  2019-11-03         16518  ...   2019-11   8360000    DKI Jakarta
4   1702573  2019-10-16         15696  ...   2019-10   8950000    DKI Jakarta
6   1711399  2019-11-04         16791  ...   2019-11   1465000    DKI Jakarta
```

```
[5 rows x 12 columns]
```

Membuat Pie Chart

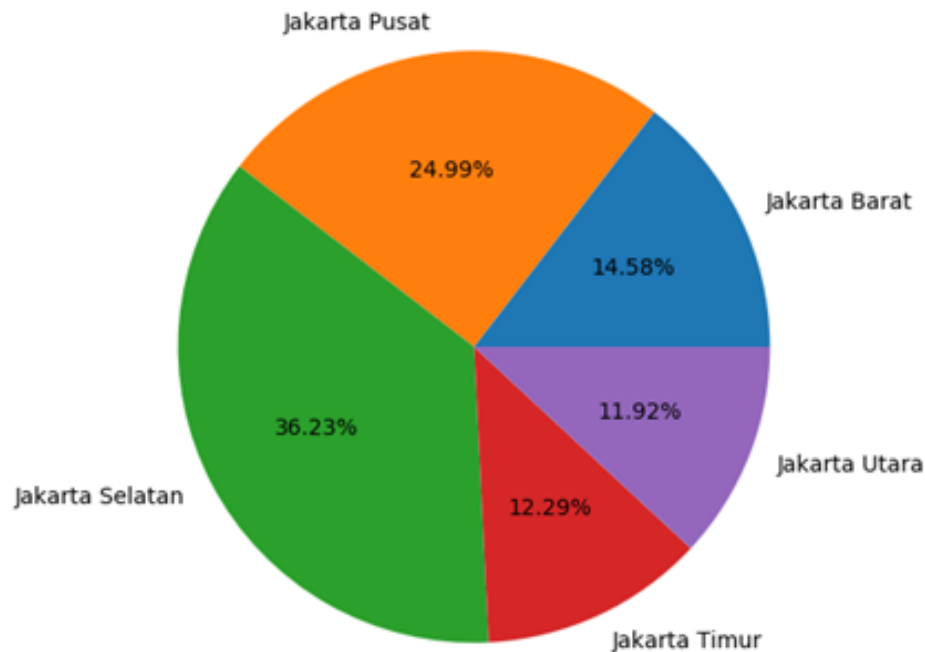
- Dataframe yang telah dibuat bisa disajikan dalam bentuk pie chart.
- Dilakukan dengan membuat dataframe agregat terlebih dahulu.
- Misal, kita akan membuat data agregat **gmv_per_city_dki_q4** untuk menghitung jumlah nilai GMV berdasarkan city.
- Kemudian, data agregat tersebut akan dimasukkan ke function **plt.pie()** untuk membuat pie chart-nya.
- Ada beberapa parameter yang bisa dimodifikasi untuk membuat pie chart, yaitu:
 - **labels:** array yang berisi label/tulisan yang menunjukkan masing-masing bagian pie.
 - **colors:** array yang berisi warna untuk masing-masing bagian pie.
 - **autopct:** format dari nilai persentasi yang ditampilkan, bisa berupa string atau function.
 - **shadow:** jika diisi True, maka pie chart yang kita buat tersebut akan memiliki bayangan. Nilai defaultnya adalah False.
 - **radius:** jari-jari dari pie-chart.

Membuat Pie Chart

- Selanjutnya, data tersebut dimasukkan dalam function plt.pie() dengan menjadikan kolom city sebagai labelnya.
- Sebelum membuat plot, atur ukuran figure terlebih dahulu.

```
1 import matplotlib.pyplot as plt
2 gmv_per_city_dki_q4 = dataset_dki_q4.groupby('city')['gmv'].sum().reset_index()
3 plt.figure(figsize=(6,6))
4 plt.pie(gmv_per_city_dki_q4['gmv'], labels =
  gmv_per_city_dki_q4['city'], autopct='%1.2f%%')
5 plt.title('GMV Contribution Per City - DKI Jakarta in Q4 2019', loc='center', pad=30,
  fontsize=15, color='blue')
6 plt.show()
```

GMV Contribution Per City - DKI Jakarta in Q4 2019



Membuat Pie Chart

Dari pie chart yang dihasilkan, terlihat bahwa kota yang mendominasi adalah Jakarta Selatan untuk nilai GMV-nya, yaitu sebesar 36.23%, disusul oleh Jakarta Pusat dan 3 kota lainnya di provinsi tersebut.

Membuat Bar Chart

- Menggunakan bar chart untuk membandingkan GMV dari tiap kota.
- Visualisasi seperti ini umumnya lebih disukai karena lebih mudah untuk melihat perbandingan antar kota.
- Syntax untuk membuat bar chart mirip dengan line chart, dilakukan dengan menentukan nilai untuk sumbu-x dan sumbu-y, lalu menambahkan parameter **kind='bar'** untuk menghasilkan bar chart.

```
1 import matplotlib.pyplot as plt
2 plt.clf()
3 dataset_dki_q4.groupby('city')
  ['gmv'].sum().sort_values(ascending=False).plot(kind='bar', color='green')
4 plt.title('GMV Per City - DKI Jakarta in Q4 2019',loc='center',pad=30, fontsize=15,
  color='blue')
5 plt.xlabel('City', fontsize = 15)
6 plt.ylabel('Total Amount (in Billions)', fontsize = 15)
7 plt.ylim(ymin=0)
8 labels, locations = plt.yticks()
9 plt.yticks(labels, (labels/1000000000).astype(int))
10 plt.xticks(rotation=0)
11 plt.show()
```

Membuat Bar Chart

- Perhatikan kode yang digunakan dalam membuat bar chart.
- Dalam membuat bar chart, tambahkan parameter rotation di **plt.xticks()** untuk memutar label di sumbu-x.
- Jika ingin menampilkan bar chart secara horizontal, ganti parameternya menjadi **kind='barh'**, dengan tidak lupa untuk juga menyesuaikan sumbu-x dan sumbu-y.



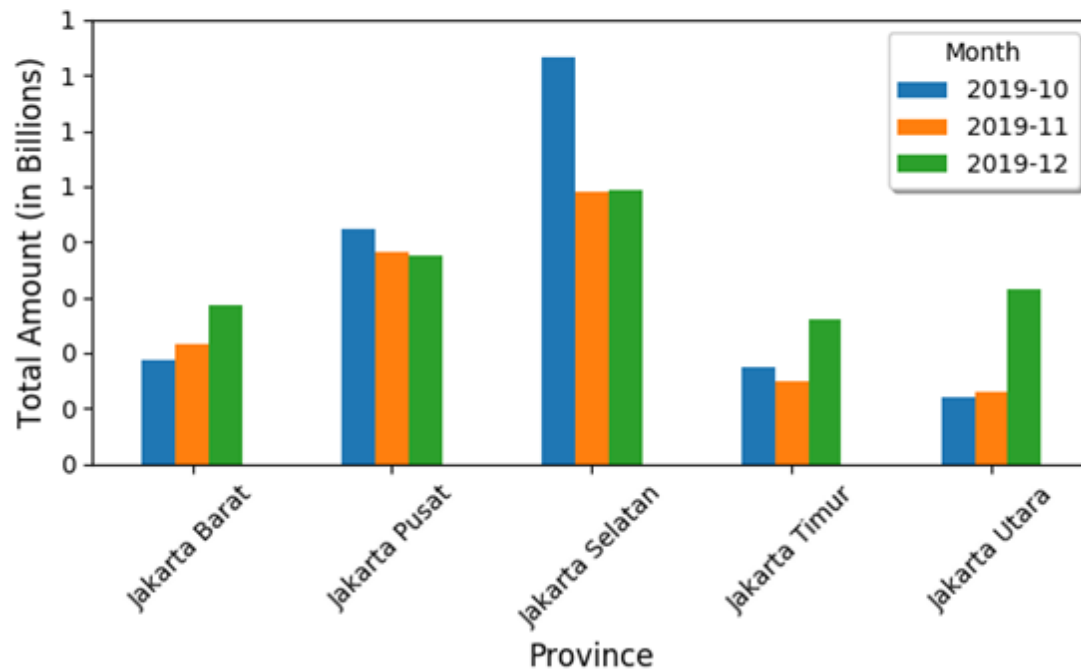
Membuat Multi-Bar Chart

- Dari output grafik sebelumnya, terlihat bahwa Jakarta Selatan dan Jakarta Pusat lumayan mendominasi untuk nilai GMV-nya.
- Tapi, kita masih belum mengetahui apakah ini konsisten di setiap bulannya.
- Untuk itu, kita perlu membuat multi-barchart-nya dulu untuk menganalisisnya.
- Untuk melihat kenaikan GMV dari masing-masing kota di setiap bulannya, kelompokkan dataset_dki_q4 berdasarkan **city** dan **order_month** nya.

```
1 import matplotlib.pyplot as plt
2 dataset_dki_q4.groupby(['city', 'order_month'])
  ['gmv'].sum().unstack().plot(kind='bar')
3 plt.title('GMV Per City, Breakdown by Month\nDKI Jakarta in Q4
  2019', loc='center', pad=30, fontsize=15, color='blue')
4 plt.xlabel('Province', fontsize = 12)
5 plt.ylabel('Total Amount (in Billions)', fontsize = 12)
6 plt.legend(bbox_to_anchor=(1, 1), shadow=True, title='Month')
7 plt.ylim(ymin=0)
8 labels, locations = plt.yticks()
9 plt.yticks(labels, (labels/1000000000).astype(int))
10 plt.xticks(rotation=45)
11 plt.tight_layout()
12 plt.show()
```

Membuat Multi-Bar Chart

GMV Per City, Breakdown by Month
DKI Jakarta in Q4 2019



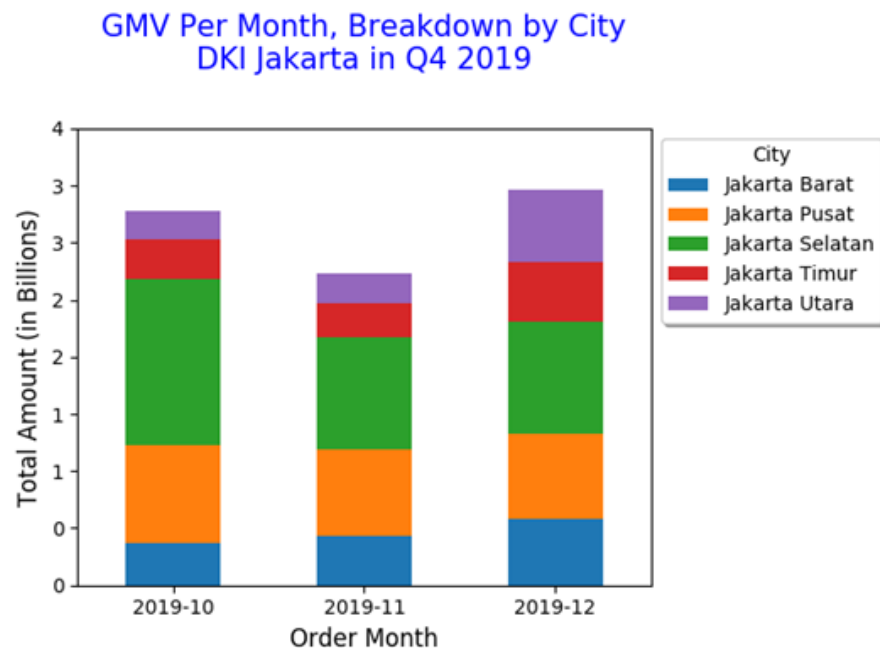
- Ternyata, setelah dibuat multi-bar chartnya, GMV di Jakarta Selatan mengalami kenaikan yang cukup pesat hanya pada saat bulan Oktober saja, sementara pada bulan November GMV ini mengalami penurunan yang drastis, lalu mengalami sedikit kenaikan pada bulan Desember.
- Untuk provinsi lainnya, seperti Jakarta Barat, Jakarta Timur, dan Jakarta Utara, nilai GMV-nya cenderung naik pada bulan Desember meskipun tidak sebanyak GMV yang ada di Jakarta Selatan.

Membuat Stacked Bar Chart

- Sebagai pembandingan, coba ubah multi-bar chart yang telah dibuat sebelumnya menjadi stacked bar chart.
- Ada yang sedikit berbeda dari kode yang akan digunakan, yaitu pengelompokkan dataset_dki_q4 berdasarkan **order_month**-nya dulu, baru berdasarkan **city**.
- Supaya lebih jelas perbandingan jumlah GMV nya, urutkan secara menurun.

```
1 import matplotlib.pyplot as plt
2 dataset_dki_q4.groupby(['order_month', 'city'])
  ['gmv'].sum().sort_values(ascending=False).unstack().plot(kind='bar', stacked=True)
3 plt.title('GMV Per Month, Breakdown by City\nDKI Jakarta in Q4
  2019', loc='center', pad=30, fontsize=15, color='blue')
4 plt.xlabel('Order Month', fontsize = 12)
5 plt.ylabel('Total Amount (in Billions)', fontsize = 12)
6 plt.legend(bbox_to_anchor=(1, 1), shadow=True, ncol=1, title='City')
7 plt.ylim(ymin=0)
8 labels, locations = plt.yticks()
9 plt.yticks(labels, (labels/1000000000).astype(int))
10 plt.xticks(rotation=0)
11 plt.tight_layout()
12 plt.show()
```

Membuat Stacked Bar Chart



- Kedua jenis bar chart yang dibuat sama-sama memberi informasi bahwa GMV mengalami penurunan dari bulan Oktober ke November, kemudian mengalami kenaikan yang pesat di bulan Desember.
- Berbeda dengan Jakarta Barat, GMV-nya justru semakin meningkat dari bulan Oktober hingga Desember.
- Sementara itu, di Jakarta Selatan, GMV-nya pada bulan Desember justru lebih kecil daripada GMV pada bulan Oktober.

Analisa

- Jika dilihat lebih lanjut, Jakarta Selatan adalah penyumbang GMV paling besar diantara provinsi-provinsi lainnya.
- Hal ini bisa menjadi pertimbangan bagi tim bisnis supaya bisa lebih “*meningkatkan*” lagi penjualan di provinsi tersebut.
- Dengan begitu, trend-nya bisa naik kembali seperti kota-kota lainnya.

Membuat Agregat Data Customer

- Jika dilihat lebih lanjut, Jakarta Barat memiliki potensi yang baik dalam penjualannya, dimana tiap bulannya selalu menunjukkan *growth* yang positif.
- Akan tetapi, kita belum mengetahui apakah ada pola yang menarik dari customer yang ada di DKI Jakarta.
- Oleh karena itu, coba lihat dulu summary untuk tiap customer tersebut.
- Selanjutnya, lihat seberapa banyak customer dengan customer_id tertentu melakukan transaksi pembelian.
- Untuk memperjelas, hitung total **quantity** untuk setiap transaksi.

```
1 data_per_customer = (dataset_dki_q4.groupby('customer_id')
2                       .agg({'order_id': 'nunique',
3                             'quantity': 'sum',
4                             'gmv': 'sum'})
5                       .reset_index()
6                       .rename(columns={'order_id': 'orders'}))
7 print(data_per_customer.sort_values(by='orders', ascending=False))
```

Membuat Agregat Data Customer

	customer_id	orders	quantity	gmv
0	12748	29	557	175286000
662	17841	26	270	144788000
207	14606	13	25	20714000
25	13089	8	166	93554000
462	16549	8	29	22714000
...	...	...	...	...
294	15154	1	4	2689000
291	15148	1	12	5400000
289	15121	1	8	2424000
288	15117	1	1	3494000
384	15975	1	42	25410000

711 rows × 4 columns

- Dari agregat data tersebut, didapatkan 711 customers yang bertransaksi di DKI Jakarta pada Q4 2020.
- Jika diurutkan dari jumlah order terbanyak, bisa dilihat bahwa **customer_id 12748** adalah customer yang paling banyak melakukan transaksi, yaitu sebanyak **29** transaksi dengan jumlah quantity mencapai **557** dan GMV lebih dari **175 juta**.
- Customer-customer seperti inilah yang harus tetap dijaga supaya semakin loyal dengan produk.

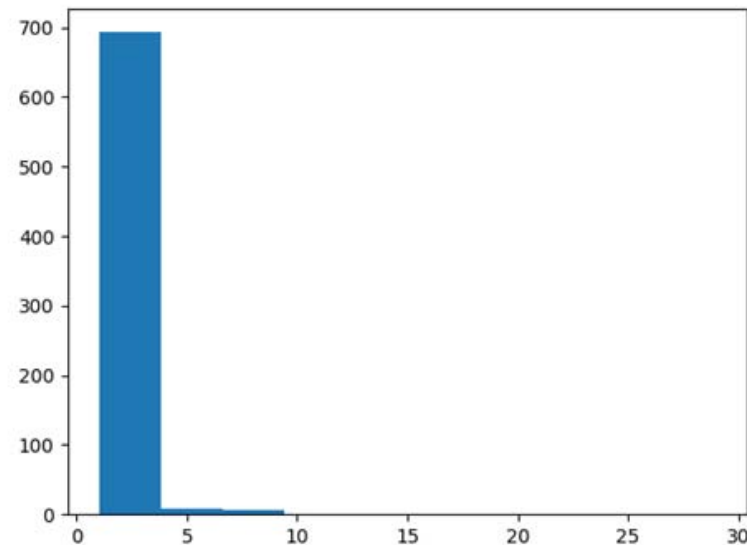
Membuat Histogram — Bagian 1

- Dari dataframe yang telah dibuat, dapat dilihat persebaran datanya secara sekaligus untuk masing-masing variabel menggunakan histogram.
- Untuk membuat histogram ini, gunakan function **plt.hist()** dengan memasukkan variabel yang ingin dicek.
- Parameter lainnya yang bisa ditambahkan dalam function tersebut:
 - **bins**: jumlah bin (kelompok nilai) yang diinginkan.
 - **range**: nilai minimum dan maksimum yang ingin ditampilkan.
 - **orientation**: menunjukkan arah 'horizontal' atau 'vertikal'.
 - **color**: warna bar di histogram.

Membuat Histogram — Bagian 1

- Dimulai dengan menggunakan variabel orders tanpa parameter apapun, akan dibuat sebuah tampilan plot sederhana untuk histogram pertama.

```
1 import matplotlib.pyplot as plt
2 plt.clf()
3 # Histogram pertama
4 plt.figure()
5 plt.hist(data_per_customer['orders'])
6 plt.show()
```



Membuat Histogram — Bagian 1

- Karena datanya terlalu berat ke kiri, dimana hampir semua customer memiliki jumlah transaksi kurang dari 10, perlu dilakukan set terhadap range-nya, yaitu dari 1 sampai 5.
- Setelah diatur range-nya, sekarang tampilan range-nya sudah tidak berat sebelah lagi.
- Jumlah customer yang melakukan transaksi kini terlihat lebih merata banyaknya untuk jumlah transaksi tertentu.

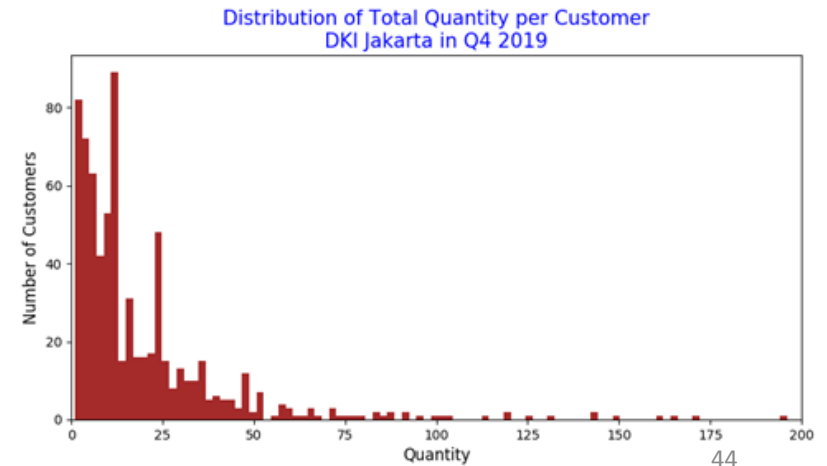
```
7 # Histogram kedua
8 plt.figure()
9 plt.hist(data_per_customer['orders'], range=(1,5))
10 plt.title('Distribution of Number of Orders per Customer\nDKI Jakarta in Q4
2019', fontsize=15, color='blue')
11 plt.xlabel('Number of Orders', fontsize = 12)
12 plt.ylabel('Number of Customers', fontsize = 12)
13 plt.show()
```



Membuat Histogram — Bagian 2

- Jika histogram bisa dibuat untuk menghitung jumlah transaksinya, ini histogram juga bisa dibuat untuk menghitung quantity-nya juga.
- Untuk menghitung nilai quantity-nya, lakukan set terhadap range-nya menjadi 200 untuk melihat banyaknya quantity dengan nilai maksimal 200.
- Untuk jumlah kelompok nilainya, kita akan bagi menjadi sebanyak 100 kelompok.
- Dari histogram yang dibuat ini, dapat dilihat bahwa persebaran nilai quantity paling banyak adalah quantity dengan range 0 hingga 25. Untuk quantity pada range selanjutnya cenderung memiliki persebaran nilai yang lebih sedikit

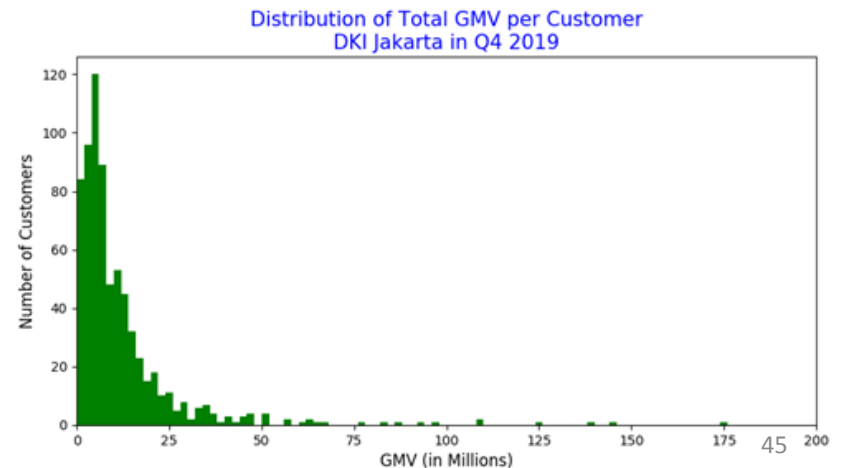
```
1 import matplotlib.pyplot as plt
2 plt.figure(figsize=(10,5))
3 plt.hist(data_per_customer['quantity'], bins=100, range=(1,200), color='brown')
4 plt.title('Distribution of Total Quantity per Customer\nDKI Jakarta in Q4
5 2019', fontsize=15, color='blue')
6 plt.xlabel('Quantity', fontsize = 12)
7 plt.ylabel('Number of Customers', fontsize = 12)
8 plt.xlim(xmin=0, xmax=200)
9 plt.show()
```



Membuat Histogram — Bagian 3

- Untuk menghitung persebaran nilai yang memiliki satuan yang tidak terlalu banyak, seperti quantity dan order, masukkan range nilai yang akan ditampilkan serta banyaknya kelompok nilai yang akan dispesifikasikan.
- Akan tetapi, berbeda halnya dengan GMV. Untuk melihat persebaran GMV, ticks label di sumbu-x harus dinormalisasi terlebih dahulu karena angka yang menunjukkan GMV biasanya bisa jadi terlalu panjang.
- Lakukan dengan cara menampilkan nilai GMV dalam jutaan dengan membagi tiap GMV-nya dalam satuan juta.
- Setelah grafiknya ditampilkan, ternyata variabel GMV ini juga memiliki distribusi yang mirip dengan kedua variabel sebelumnya, dimana grafiknya cenderung condong ke kiri.

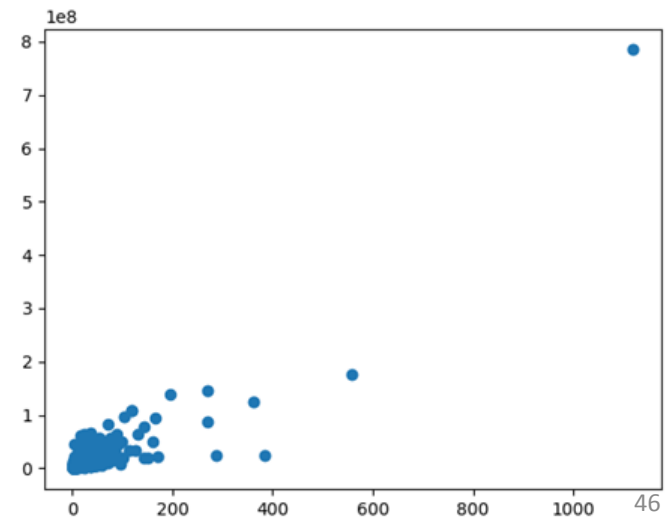
```
1 import matplotlib.pyplot as plt
2 plt.figure(figsize=(10,5))
3 plt.hist(data_per_customer['gmv'], bins=100, range=(1,200000000), color='green')
4 plt.title('Distribution of Total GMV per Customer\nDKI Jakarta in Q4
5 2019',fontsize=15, color='blue')
6 plt.xlabel('GMV (in Millions)', fontsize = 12)
7 plt.ylabel('Number of Customers',fontsize = 12)
8 plt.xlim(xmin=0,xmax=200000000)
9 labels, locations = plt.xticks()
10 plt.xticks(labels, (labels/1000000).astype(int))
11 plt.show()
```



Membuat Scatterplot

- Penyajian histogram tadi sebenarnya sudah cukup bagus untuk menampilkan persebaran nilai dari variabel tertentu.
- Akan tetapi, belum diketahui apakah variabel tersebut saling berkorelasi.
- Untuk itu, perlu dilihat dengan scatterplot.
- Untuk membuat scatterplot, gunakan function **plt.scatter()** dengan memasukkan variabel-variabel yang akan dibuat scatterplot.
- Misalnya, jika kita ingin menampilkan variabel quantity dan gmv, kita cukup memasukkan kedua variabel tersebut, kemudian menampilkan korelasi dari kedua variabel tersebut seperti pada kode berikut.

```
1 import matplotlib.pyplot as plt
2 plt.clf()
3 # Scatterplot pertama
4 plt.figure()
5 plt.scatter(data_per_customer['quantity'], data_per_customer['gmv'])
6 plt.show()
```



Membuat Scatterplot

- Dari scatterplot yang telah dibuat, bisa disimpulkan bahwa semakin besar quantity, maka GMV cenderung semakin tinggi.
- Akan tetapi, masih ada masalah lain dalam data tersebut, yaitu banyaknya data yang terkumpul di nilai-nilai kecil sehingga tidak begitu terlihat detailnya di area situ.

“Jika seorang customer telah membeli dengan banyak quantity, kemungkinan GMV dari customer tersebut juga akan semakin besar.”

Membuat Scatterplot

- Untuk itu, scatter plot tersebut perlu diperbaiki dengan mengubah ukuran figurenya menjadi 10 x 8, kemudian menentukan batasan nilai untuk sumbu-x dan sumbu-y nya supaya bisa menampilkan data-data yang tersebar di nilai-nilai yang kecil tersebut dengan lebih jelas.
- Sekarang, persebaran korelasinya sudah terlihat lebih jelas dari sebelumnya. Dari scatter plot yang dibuat, semakin jelas terlihat bahwa quantity dan GMV itu berkorelasi positif. Jika seorang customer telah membeli dengan banyak quantity, kemungkinan GMV dari customer tersebut juga akan semakin besar.

```
7 # Scatterplot kedua: perbaikan scatterplot pertama
8 plt.figure(figsize=(10,8))
9 plt.scatter(data_per_customer['quantity'],data_per_customer['gmv'], marker='+',
10 color='red')
11 plt.title('Correlation of Quantity and GMV per Customer\nDKI Jakarta in Q4
12 2019',fontsize=15, color='blue')
13 plt.xlabel('Quantity', fontsize = 12)
14 plt.ylabel('GMV (in Millions)',fontsize = 12)
15 plt.xlim(xmin=0,xmax=300)
16 plt.ylim(ymin=0,ymax=150000000)
17 labels, locations = plt.yticks()
18 plt.yticks(labels, (labels/1000000).astype(int))
19 plt.show()
```



Kesimpulan

- Setiap jenis grafik memiliki fungsinya sendiri-sendiri, tergantung dari jenis data yang ditampilkan.
- Dari grafik-grafik yang telah dipelajari, dapat disimpulkan bahwa fungsi dari masing-masing jenis grafik tersebut adalah:
 - Multi line-chart untuk melihat komparasi dari GMV dari berbagai provinsi, dari bulan ke bulan.
 - Pie chart atau multi bar-chart untuk melihat komposisi GMV per kota.
 - Histogram untuk melihat distribusi GMV atau quantity.
 - Scatterplot untuk memperlihatkan relasi antara quantity dan GMV.

Tugas 1. Menentukan brand top 5

- Dalam tugas ini, lakukan analisis untuk top brands di bulan Desember 2019.
- Cara:
 - membuat variabel baru bernama `dataset_top5brand_dec` yang berisi data penjualan bulan Desember 2019, dimana dataframe tersebut hanya memuat top 5 brand dengan quantity terjual terbanyak selama bulan Desember 2019.
 - Untuk visualisasi-visualisasi selanjutnya, hanya dataframe ini yang akan digunakan.

Lengkapi kode berikut ini:

```
#mengambil informasi top 5 brands berdasarkan quantity
top_brands = (dataset[
].groupby('brand')['
'].sum()
.reset_index()
.sort_values(by='quantity',ascending=False)
.head(5))

#membuat dataframe baru, filter hanya di bulan Desember 2019 dan hanya
top 5 brands
dataset_top5brand_dec = dataset[(
) &
(dataset['brand'].isin(top_brands['brand'].to_list()))]
# print top brands
print(top_brands)
```

Hasil running:

Di sini, brand yang termasuk dalam `top_brands` adalah **BRAND_S**, **BRAND_P**, **BRAND_R**, **BRAND_C**, dan **BRAND_A**.

	brand	quantity
8	BRAND_S	2197
6	BRAND_P	2194
7	BRAND_R	1508
2	BRAND_C	1365
0	BRAND_A	1315

Tugas 2: Multi-line chart daily quantity untuk brand top 5

- Selanjutnya, lakukan analisis terhadap top brands di bulan Desember 2019.
- Buatlah visualisasi multi-line chart untuk menampilkan banyaknya quantity yang terjual dalam harian dengan melakukan breakdown per brandnya.
- Pada satu titik dimana lonjakan quantity paling besar terjadi, tandai dengan anotasi.
- Itu artinya, akan terlihat 1 tanggal di grafik tersebut dimana ada salah satu brand yang mengalami lonjakan (quantity lebih tinggi dari tanggal-tanggal lain).

Lengkapi kode berikut ini:

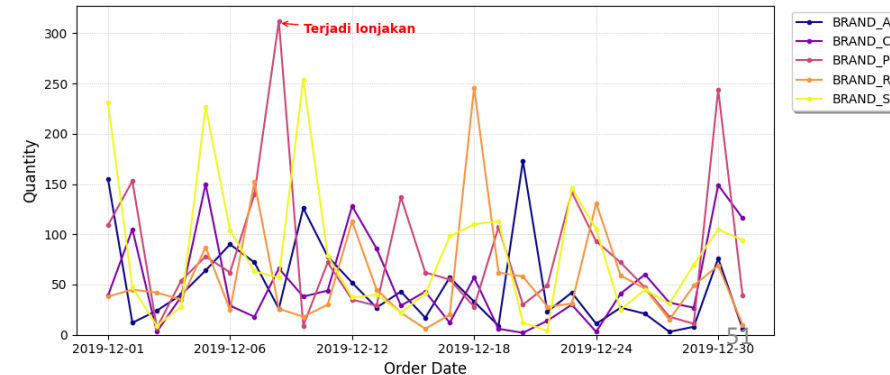
```
import matplotlib.pyplot as plt
dataset_top5brand_dec.groupby(['', '']).sum()
.unstack().plot(marker='.', cmap='plasma')
plt.title('Daily Sold Quantity Dec 2019 - Breakdown by Brands', loc='center', pad=30, fontsize=15, color='blue')
plt.xlabel('', fontsize = 12)
plt.ylabel('', fontsize = 12)
plt.grid(color='darkgray', linestyle=':', linewidth=0.5)
plt.ylim(ymin=0)
plt.legend(loc='upper center', bbox_to_anchor=(1.1, 1), shadow=True, ncol=1)
plt.annotate('Terjadi lonjakan', xy=(7, 310), xytext=(8, 300),
            weight='bold', color='red',
            arrowprops=dict(arrowstyle='->',
                            connectionstyle="arc3",
                            color='red'))

plt.gcf().set_size_inches(10, 5)
plt.tight_layout()
plt.show()
```

Hasil running:

Di sini dapat dilihat bahwa nilai quantity dari **brand_P** melonjak pesat setelah tanggal 6 Desember, yaitu hingga mencapai **300 lebih**.

Daily Sold Quantity Dec 2019 - Breakdown by Brands



Tugas 3: Kuantitas penjualan brand top 5 selama Desember 2019

- Setelah dilakukan analisis quantity terhadap brand-nya, lakukan analisis lebih lanjut pada jumlah product untuk masing-masing brand yang laku selama bulan Desember 2019.
- Untuk visualisasi, gunakan barchart dengan mengurutkan nilai unik dari **product_id** secara menurun.

Lengkapi kode berikut ini:

```
import matplotlib.pyplot as plt

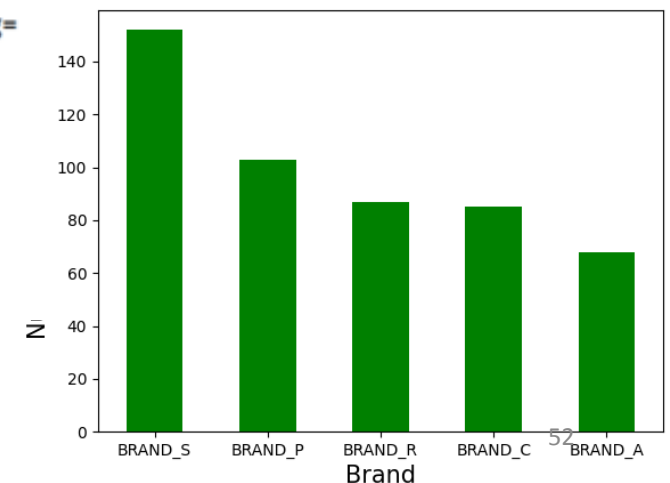
plt.clf()
dataset_top5brand_dec.groupby('')[ ''].unique().sort_values(ascending=
False).plot(kind='bar', color='green')
plt.title(' ',loc='center',pad=30,
fontsize=15, color='blue')
plt.xlabel(' ', fontsize = 15)
plt.ylabel(' ',fontsize = 15)
plt.ylim(ymin=0)
plt.xticks(rotation=0)
plt.show()
```

Hasil running:

Di sini, brand dengan jumlah produk yang paling banyak adalah **BRAND_S**, dengan jumlah produk mencapai **140 lebih**.

Untuk jumlah produk paling sedikit ditempati oleh **BRAND_A** dengan banyaknya produk mencapai **60 lebih**.

Number of Sold Products per Brand, December 2019



Tugas 4: Penjualan produk diatas 100 dan dibawah 100 selama Desember 2019

- Dari sekian banyaknya produk yang telah dianalisis, ada hal menarik yang masih menjadi pertanyaan kita, yaitu adakah pola yang menarik dari banyaknya produk yang terjual sebanyak 100 ke atas dan yang terjual kurang dari nilai tersebut?
- Untuk mengamati pola tersebut, perlu dibuat visualisasinya dalam bentuk stacked chart untuk melakukan breakdown terhadap bar chart yang dihasilkan di Tugas 3, yaitu untuk produk yang terjual ≥ 100 dan < 100 di bulan Desember 2019.
- Untuk jumlah produk per **brand**-nya, urutkan brand secara menurun dengan brand di sebelah kiri adalah brand dengan product yang lebih banyak.

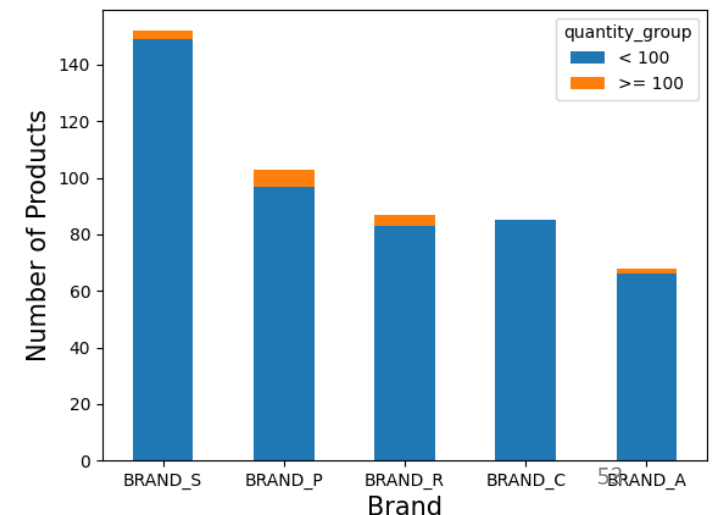
Lengkapi kode berikut ini:

```
import matplotlib.pyplot as plt
#membuat dataframe baru, untuk agregat jumlah quantity terjual per product
dataset_top5brand_dec_per_product =
dataset_top5brand_dec.groupby(['brand', 'quantity_group'])['product_id'].sum().reset_index()
#beri kolom baru untuk menandai product yang terjual >= 100 dan <100
dataset_top5brand_dec_per_product['quantity_group'] =
dataset_top5brand_dec_per_product['quantity'].apply(lambda x: '>= 100' if x >= 100 else '< 100')
dataset_top5brand_dec_per_product.sort_values('quantity', ascending=False, inplace=True)
#membuat referensi pengurutan brand berdasarkan banyaknya semua product
s_sort =
dataset_top5brand_dec_per_product.groupby('brand')['product_id'].nunique().sort_values(ascending=False)
#plot stacked barchart
dataset_top5brand_dec_per_product.groupby(['brand', 'quantity_group'])['product_id'].nunique().
reindex(index=s_sort.index, level='brand').unstack().plot(kind='bar', stacked=True)
plt.title('Number of Sold Products per Brand, December 2019', loc='center', pad=30, fontsize=15,
color='blue')
plt.xlabel('Brand', fontsize = 15)
plt.ylabel('Number of Products', fontsize = 15)
plt.ylim(ymin=0)
plt.xticks(rotation=0)
plt.show()
```

Hasil running:

Ternyata, produk dengan jumlah penjualan **kurang dari 100** lebih **mendominasi** dibandingkan dengan yang total penjualannya mencapai 100 ke atas.

Number of Sold Products per Brand, December 2019



Referensi

- “Data Analyst Career Track Series” from DQLAB about “Data Visualization With Python Matplotlib For Beginner — Part 2”.
- <https://academy.dqlab.id/main/package/practice/165>