Meringkas Data: Statistik Ringkasan

Setelah bisa memilih dan mengurutkan data, langkah selanjutnya adalah meringkasnya. Bayangkan Anda memiliki ribuan baris data penjualan, Anda tentu tidak ingin melihatnya satu per satu. Anda ingin tahu, "Berapa rata-rata penjualan?" atau "Berapa harga termahal?". Di sinilah statistik ringkasan berperan.

Metode Statistik Umum

Pandas menyediakan banyak metode yang mudah digunakan untuk menghitung statistik pada sebuah kolom (Series). Metode ini biasanya mengabaikan nilai yang hilang (*missing values*) secara otomatis.

Contoh-contoh utama:

- `.mean()`: Menghitung nilai rata-rata.
- `.median()`: Menemukan nilai tengah.
- `.min()`: Menemukan nilai terkecil.
- `.max()`: Menemukan nilai terbesar.
- `.sum()`: Menjumlahkan semua nilai.
- `.std()`: Menghitung standar deviasi (ukuran sebaran data).

Contoh Penggunaan:

```
# Menghitung rata-rata harga satuan dari data penjualan kita
rata_rata_harga = df_penjualan['Harga Satuan'].mean()
print(rata_rata_harga)
```

Metode `.agg()` (Aggregate)

Terkadang, Anda ingin menerapkan fungsi ringkasan Anda sendiri atau beberapa fungsi sekaligus. Di sinilah metode `.agg()` sangat berguna. Anda bisa memberikan nama fungsi (sebagai string) atau fungsi yang Anda definisikan sendiri.

Contoh: Menerapkan beberapa fungsi sekaligus.

```
# Menghitung nilai minimum dan maksimum dari harga satuan
df_penjualan['Harga Satuan'].agg(['min', 'max'])
```

Latihan Singkat

Gunakan `df_penjualan` dari proyek kita sebelumnya.

1. Hitung **total pendapatan** dari semua transaksi. (Gunakan `.sum()` pada kolom yang sesuai).
2. Temukan **harga satuan termurah** dan **termahal** dari semua produk yang terjual dalam satu perintah. (Gunakan `.agg()`).

Meringkas Data: Menghitung Nilai

Selain statistik seperti rata-rata atau jumlah, seringkali kita ingin tahu: "Ada berapa banyak data untuk setiap kategori?". Misalnya, "Ada berapa banyak transaksi untuk setiap produk?" atau "Ada berapa banyak mobil untuk setiap merek?".

Menghindari Perhitungan Ganda

Seperti yang dijelaskan di materi Anda, jika kita hanya ingin menghitung jumlah anjing unik, kita tidak bisa langsung menghitung semua baris di data kunjungan dokter hewan, karena satu anjing bisa berkunjung berkali-kali. Kita harus menghapus data duplikat terlebih dahulu.

Metode utamanya adalah **`.drop_duplicates()`**.

Contoh: Menghapus baris duplikat berdasarkan satu kolom.

```
# Hanya akan menyimpan baris pertama untuk setiap nama anjing
yang unik
anjing_unik = df_kunjungan.drop_duplicates(subset='nama')
```

Anda juga bisa memberikan daftar kolom pada `subset` untuk mencari duplikat berdasarkan kombinasi beberapa kolom.

Menghitung Nilai dengan `.value_counts()`

Setelah kita memiliki data yang unik, metode yang paling umum digunakan untuk menghitung frekuensi setiap nilai dalam sebuah kolom adalah **`.value_counts()`**.

Metode ini sangat berguna dan memiliki beberapa argumen penting:

- `sort=True`: (Default) Mengurutkan hasil dari yang paling banyak ke yang paling sedikit.
- `normalize=True`: Menampilkan hasil sebagai proporsi atau persentase, bukan jumlah mentah.

Contoh Penggunaan:

```
# Menghitung ada berapa banyak anjing untuk setiap ras
jumlah_per_ras = anjing_unik['ras'].value_counts()
print(jumlah_per_ras)

# Menghitung proporsi anjing untuk setiap ras
proporsi_per_ras =
anjing_unik['ras'].value_counts(normalize=True)
print(proporsi_per_ras)
```

Latihan Singkat

Gunakan `df_penjualan` kita.

1. Hitung ada **berapa banyak transaksi** yang terjadi untuk setiap **'Produk'**.
2. Tampilkan hasil di atas dalam urutan dari produk yang paling sering muncul ke yang paling jarang.

Meringkas Data: Statistik Berkelompok

Sejauh ini kita telah meringkas seluruh kolom. Namun, kekuatan sebenarnya dari analisis data adalah saat kita bisa **membandingkan ringkasan antar grup**. Misalnya, alih-alih bertanya "Berapa rata-rata penjualan?", kita bertanya "Berapa rata-rata penjualan **untuk setiap kategori produk**?".

Metode `.groupby()`

Inilah gunanya metode `.groupby()`. Proses ini biasanya terdiri dari tiga langkah:

1. **Group by:** Kelompokkan DataFrame berdasarkan satu atau lebih kolom.
2. **Select:** Pilih kolom yang ingin Anda ringkas.
3. **Aggregate:** Terapkan fungsi ringkasan (seperti `.mean()`, `.sum()`, `.count()`).

Contoh Penggunaan:

```
# Menghitung rata-rata berat anjing untuk setiap warna bulu
rata_rata_berat_per_warna =
dogs.groupby('color')['weight_kg'].mean()
print(rata_rata_berat_per_warna)
```

Beberapa Ringkasan Sekaligus

Sama seperti sebelumnya, Anda bisa menggunakan `.agg()` untuk menerapkan beberapa fungsi ringkasan sekaligus pada data yang sudah dikelompokkan.

Contoh:

```
# Menghitung nilai min, max, dan jumlah berat anjing untuk
setiap warna
dogs.groupby('color')['weight_kg'].agg(['min', 'max', 'sum'])
```

Mengelompokkan Berdasarkan Beberapa Kolom

Anda juga bisa mengelompokkan data berdasarkan kombinasi beberapa kolom dengan memberikan sebuah list nama kolom ke `.groupby()`.

Contoh:

```
# Menghitung rata-rata berat anjing untuk setiap kombinasi warna dan ras
dogs.groupby(['color', 'breed'])['weight_kg'].mean()
```

Latihan Singkat

Gunakan `df_penjualan` kita.

1. Hitung **total pendapatan** (`.sum()`) untuk **setiap 'Kategori'** produk.
2. Hitung **rata-rata 'Jumlah Terjual'** (`.mean()`) untuk **setiap 'Produk'**.

Meringkas Data: Pivot Tables

Pivot table adalah cara lain untuk membuat statistik ringkasan berkelompok. Banyak orang, terutama yang terbiasa dengan Excel, merasa pivot table lebih intuitif untuk data yang perlu diringkas dalam format tabel silang (dua dimensi).

Pada dasarnya, `.pivot_table()` adalah alternatif yang lebih fleksibel dari `.groupby()`.

Struktur Dasar Pivot Table

Metode `.pivot_table()` memiliki beberapa argumen utama:

- `values`: Kolom yang ingin Anda ringkas (dihitung statistiknya).
- `index`: Kolom yang akan menjadi baris dari pivot table Anda.
- `columns`: Kolom yang akan menjadi kolom dari pivot table Anda.
- `aggfunc`: Fungsi agregasi yang ingin Anda gunakan (default-nya adalah `'mean'`).
- `fill_value`: Nilai untuk mengisi sel yang kosong (jika ada kombinasi baris/kolom yang tidak memiliki data).
- `margins=True`: Menambahkan baris dan kolom "All" yang berisi total ringkasan.

Contoh Penggunaan

Mari kita ubah contoh `.groupby()` kita menjadi `.pivot_table()`.

Contoh 1: Menggantikan `.groupby()` sederhana

```
# Cara groupby
dogs.groupby('color')['weight_kg'].mean()

# Cara pivot table
dogs.pivot_table(values='weight_kg', index='color')
```

Keduanya akan menghasilkan output yang sama, hanya formatnya sedikit berbeda.

Contoh 2: Pivot dengan Kolom Ini adalah kekuatan utama pivot table. Misalnya, kita ingin melihat rata-rata berat anjing, dikelompokkan berdasarkan `color` (baris) dan `breed` (kolom).

```
dogs.pivot_table(values='weight_kg', index='color',
columns='breed')
```

Hasilnya akan menjadi tabel di mana setiap sel adalah rata-rata berat untuk kombinasi warna dan ras tertentu. Jika ada sel yang kosong (misalnya, tidak ada anjing Chihuahua berwarna putih), nilainya akan `NaN`.

Latihan Singkat

Gunakan `df_penjualan` kita.

1. Buatlah sebuah **pivot table** yang menunjukkan **rata-rata** '**Jumlah Terjual**' untuk setiap '**Kategori**' produk. Jadikan 'Kategori' sebagai `index`.
2. Buat pivot table yang lebih kompleks untuk melihat **total** '**Total Pendapatan**'. Jadikan '**Kategori**' sebagai `index` dan '**Produk**' sebagai `columns`.