```
from google.colab import files
uploaded = files.upload()
```

Choose Files flipkart_pro…0250405.csv

- **flipkart_products_20250405.csv**(text/csv) - 903362 bytes, last modified: 4/6/2025 - 100% done

```
Saving flipkart_products_20250405.csv to flipkart_products_20250405.csv
```

```
import pandas as pd

# Replace the filename if different
df = pd.read_csv('flipkart_products_20250405.csv')
df.head()
```

	Product Name	Price (₹)	Rating (★)	Number of Buyers	Total Sold	Available Stock	Main Category	Sub Category	Discount (%)	Seller	Return Policy	
0	Krishnamurthy-Devan Laboriosam Ultra Smartphon...	142247.04	3.2	7348	4812	364	Electronics	Smartphones	45	RetailNet	False	https://www.flipkart.com
1	Nanda-Mahal Dignissimos Lite Laptops 1	186922.43	4.1	2342	881	145	Electronics	Laptops	55	Flipkart Assured	False	https://www.flipkart.cor
2	Choudhury LLC Amet Plus Decor 15	11843.41	5.0	739	2580	206	Home	Decor	58	SuperComNet	True	https://www.flipkart.(

Next steps: Generate code with df View recommended plots New interactive sheet

```
# 1. Load the CSV
import pandas as pd
df = pd.read_csv('flipkart_products_20250405.csv')
print("Original Shape:", df.shape)
print(df.head())
```

```
Original Shape: (5000, 12)
                                        Product Name  Price (₹)  Rating (★)  \
0  Krishnamurthy-Devan Laboriosam Ultra Smartphon...  142247.04         3.2
1             Nanda-Mahal Dignissimos Lite Laptops 1  186922.43         4.1
2                   Choudhury LLC Amet Plus Decor 15   11843.41         5.0
3           Borah LLC Accusantium Lite Smartphones 9   10864.31         4.8
4               Murty Inc Placeat Pro Smartwatches 8   32950.41         4.5

   Number of Buyers  Total Sold  Available Stock Main Category Sub Category  \
0              7348        4812              364   Electronics  Smartphones
1              2342         881              145   Electronics      Laptops
2               739        2580              206          Home        Decor
3              1543        4562             1585   Electronics  Smartphones
4              7702        4925             1064   Electronics  Smartwatches

   Discount (%)            Seller  Return Policy  \
0            45         RetailNet          False
1            55  Flipkart Assured          False
2            58       SuperComNet           True
3             0       ElectroWorld          False
4            18         MobileHub          False

                                         Product URL
0  https://www.flipkart.com/Krishnamurthy-Devan-L...
1  https://www.flipkart.com/Nanda-Mahal-Dignissim...
2  https://www.flipkart.com/Choudhury-LLC-Amet-Pl...
3  https://www.flipkart.com/Borah-LLC-Accusantium...
4  https://www.flipkart.com/Murty-Inc-Placeat-Pro...
```

```
# Vertical Concat
df_part1 = df.iloc[:100]
df_part2 = df.iloc[100:200]
df_concat_vertical = pd.concat([df_part1, df_part2], axis=0)
print("Vertical Concat:", df_concat_vertical.shape)

# Horizontal Concat
```

```
df_concat_horizontal = pd.concat([df_part1.reset_index(drop=True), df_part2.reset_index(drop=True)], axis=1)
print("Horizontal Concat:", df_concat_horizontal.shape)
```

```
Vertical Concat: (200, 12)
Horizontal Concat: (100, 24)
```

```
df_appended = pd.concat([df_part1, df_part2], ignore_index=True)
print("Appended DataFrame:", df_appended.shape)
```

```
Appended DataFrame: (200, 12)
```

```
# Creating dummy second DataFrame to join on 'product_name'
# Assuming your column name is 'Product Name' instead of 'product_name'
df2 = df[['Product Name']].drop_duplicates().sample(100, random_state=42)
df2['category'] = ['Category ' + str(i%5) for i in range(100)]

# Inner Join
# Use the correct column name in the 'on' parameter for merge
df_inner = pd.merge(df, df2, on='Product Name', how='inner')
print("Inner Join:", df_inner.shape)

# Left Join
# Use the correct column name in the 'on' parameter for merge
df_left = pd.merge(df, df2, on='Product Name', how='left')
print("Left Join:", df_left.shape)
```

```
Inner Join: (100, 13)
Left Join: (5000, 13)
```

```
# Example: Group by rating and count
# Check if 'rating' column exists, if not, use actual column name
if 'rating' in df.columns:
    grouped = df.groupby('rating')['Product Name'].count().reset_index(name='product_count') # Changed 'product_name' to 'Product Name'
else:
    # Replace 'actual_rating_column' with the correct column name from your DataFrame
    # Example: Group by rating and count
    # Check if 'rating' column exists, if not, use actual column name
    # Assuming 'Product Rating' is the correct column for ratings
    # Get the correct rating column name
    rating_column = df.columns[df.columns.str.contains('rating', case=False)].values[0]  # Use regex to find

    grouped = df.groupby(rating_column)['Product Name'].count().reset_index(name='product_count') # Changed 'product_name' to 'Product N
print(grouped.head())

# Average discounted_price per rating
# Check if 'Discounted Price' column exists, if not, find similar column using regex
# ----> BEGIN CHANGES <----
# Check columns for potential price columns using broader search
possible_price_columns = df.columns[df.columns.str.contains('price', case=False)]

# If potential price columns are found, use the first one
if len(possible_price_columns) > 0:
    discounted_price_column = possible_price_columns[0]
else:
    print(f"Available columns: {df.columns}")
    raise KeyError("Could not find a suitable column for discounted price. Please check your DataFrame.")
# ----> END CHANGES <----

avg_price = df.groupby(rating_column)[discounted_price_column].mean().reset_index()  # Changed 'discounted_price' to actual column name

print(avg_price.head())
print(grouped.head())

# Average discounted_price per rating
# Check if 'rating' column exists, if not, use actual column name
if 'rating' in df.columns:
    avg_price = df.groupby('rating')[discounted_price_column].mean().reset_index()  # Changed 'discounted_price' to actual column name
else:
    # Replace 'actual_rating_column' with the correct column name from your DataFrame
    avg_price = df.groupby(rating_column)[discounted_price_column].mean().reset_index()  # Use rating_column and actual column name here
print(avg_price.head())
```

```
   Rating (★)  product_count
0         3.0            127
1         3.1            245
2         3.2            275
```

```
3         3.3            250
4         3.4            236
   Rating (★)     Price (₹)
0         3.0  39189.134882
1         3.1  38398.475469
2         3.2  38785.663127
3         3.3  38136.953640
4         3.4  38746.147542
   Rating (★)  product_count
0         3.0            127
1         3.1            245
2         3.2            275
3         3.3            250
4         3.4            236
   Rating (★)     Price (₹)
0         3.0  39189.134882
1         3.1  38398.475469
2         3.2  38785.663127
3         3.3  38136.953640
4         3.4  38746.147542
```

```python
# Pivot table showing average discounted price by rating and availability
# Assuming 'Discounted Price' is the actual column name
# ----> BEGIN CHANGES <----
# Find the discounted price column (case-insensitive)
# Check if the assumed column name is present, if not search for a similar one using regex
if 'Discounted Price' in df.columns:
    discounted_price_column = 'Discounted Price'
else:
    possible_price_columns = df.columns[df.columns.str.contains('price', case=False)]
    if len(possible_price_columns) > 0:
        discounted_price_column = possible_price_columns[0]
    else:
        print(f"Available columns: {df.columns}")
        raise KeyError("Could not find a suitable column for discounted price. Please check your DataFrame.")

# Find the 'Product Rating' and 'availability' columns (case-insensitive)
rating_column = df.columns[df.columns.str.contains('rating', case=False)].values[0]  # Use regex to find

# ----> BEGIN CHANGES <----
# Check if a column containing 'availability' exists
availability_columns = df.columns[df.columns.str.contains('availability', case=False)]
if len(availability_columns) > 0:
    availability_column = availability_columns.values[0]
    pivot = df.pivot_table(values=discounted_price_column, index=rating_column, columns=availability_column, aggfunc='mean')
    print(pivot.head())
else:
    print("No column containing 'availability' found in the DataFrame. Skipping pivot table creation.")
# ----> END CHANGES <----
```

```
No column containing 'availability' found in the DataFrame. Skipping pivot table creation.
```