PROJECT 2: AN ANALYSIS OF MOBILE SALES

1. Import Necessary Libraries

In [ ]:
```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error, r2_score
import warnings
warnings.filterwarnings('ignore')
import plotly.io as pio
pio.renderers.default = "notebook_connected"
```

2. Load Dataset

In [ ]:
```
df = pd.read_csv("/Users/apple/Downloads/Mobiles_Dataset.csv")
df.head(10)
```

Out[ ]:

	Product Name	Actual price	Discount price	Stars	Rating	Reviews	RAM (GB)	Storage (GB)	Display Size (inch)
0	Apple iPhone 15 (Green, 128 GB)	₹79,600	₹65,999	4.6	44,793 Ratings	2,402 Reviews	NIL	128	6.10
1	Apple iPhone 15 (Blue, 128 GB)	₹79,600	₹65,999	4.6	44,793 Ratings	2,402 Reviews	NIL	128	6.10

2	Apple iPhone 15 (Black, 128 GB)	₹79,600	₹65,999	4.6	44,793 Ratings	2,402 Reviews	NIL	128	6.10
3	OnePlus N20 SE (JADE WAVE, 128 GB)	₹19,999	₹11,489	4.0	1,005 Ratings	41 Reviews	4	128	6.56
4	OnePlus N20 SE (BLUE OASIS, 64 GB)	₹16,999	₹12,999	4.0	1,005 Ratings	41 Reviews	4	64	6.56
5	OnePlus 12R (Cool Blue, 128 GB)	₹39,999	₹38,989	4.5	4,278 Ratings	292 Reviews	8	128	6.78
6	SAMSUNG Galaxy F14 5G (GOAT Green, 128 GB)	₹17,490	₹10,990	4.2	45,538 Ratings	2,989 Reviews	4	128	6.60
7	CMF by Nothing Phone 1 (Blue, 128 GB)	₹19,999	₹15,999	4.4	8,057 Ratings	701 Reviews	6	128	6.67
8	CMF by Nothing Phone 1 (Blue, 128 GB)	₹21,999	₹17,999	4.3	2,355 Ratings	181 Reviews	8	128	6.67
9	vivo Y200e 5G (Black Diamond, 128 GB)	₹25,999	₹20,999	4.3	687 Ratings	36 Reviews	8	128	6.67

3. Explore Dataset

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 984 entries, 0 to 983
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Product Name        984 non-null    object 
 1   Actual price        984 non-null    object 
 2   Discount price      984 non-null    object 
 3   Stars               984 non-null    float64
 4   Rating              984 non-null    object 
 5   Reviews             984 non-null    object 
 6   RAM (GB)            984 non-null    object 
 7   Storage (GB)        984 non-null    object 
 8   Display Size (inch) 984 non-null    float64
 9   Camera              908 non-null    object 
 10  Description         984 non-null    object 
 11  Link                984 non-null    object 
dtypes: float64(2), object(10)
memory usage: 92.4+ KB
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
Product Name            0
Actual price            0
Discount price          0
Stars                   0
Rating                  0
Reviews                 0
RAM (GB)                0
Storage (GB)            0
Display Size (inch)     0
Camera                 76
Description             0
Link                    0
dtype: int64
```

4. Data Cleaning

In [ ]:
```
#Extract Brand Name from Product Name
df['Brand'] = df['Product Name'].str.extract(r'^(\w+)')
df['Brand'] = df['Brand'].astype(str).apply(lambda x: x.title())
df['Product Name'] = df['Product Name'].astype(str).apply(lambda x: x.tit

#Remove the money symbol in all rows of columns Actual price and Discount
df['Actual price ₹'] = df['Actual price'].str.replace('[₹,]','', regex = 
df['Discount price ₹'] = df['Discount price'].str.replace('[₹,]','', rege
df = df.drop(columns=['Actual price','Discount price'])

#Handle NIL values in Actual and Discount price columns
df['Actual price ₹'] = pd.to_numeric(df['Actual price ₹'], errors ='coerc
```

```
df['Actual price ₹'] = df['Actual price ₹'].replace('NIL', df['Actual pri
df['Actual price ₹'] = df['Actual price ₹'].fillna(df['Actual price ₹'].me
df['Discount price ₹'] = pd.to_numeric(df['Discount price ₹'], errors ='c
df['Discount price ₹'] = df['Discount price ₹'].replace('NIL', df['Discou
df['Discount price ₹'] = df['Discount price ₹'].fillna(df['Discount price

#Create Discount amount (%)
df['Discount amount (%)'] = round((df['Actual price ₹'] - df['Discount pr

#Remove Ratings and Reviews in two columns Ratings and Reviews
df['Number of Rating'] = df['Rating'].str.replace('[Ratings,]','', regex
df['Number of Reviews'] = df['Reviews'].str.replace('[Reviews,]','', rege
df = df.drop(columns=['Rating','Reviews'])

#Handle NIL values in the RAM (GB) (based on information of Description)
def extract_ram(description):
    extract_ram = re.search(r'(\d+)\s*(GB|MB)\s*RAM', description)
    ram = int(extract_ram.group(1)) if extract_ram else None
    return ram
df['RAM (GB)'] = df['Description'].apply(extract_ram)

#Handle NIL values in the Storage (GB) (based on information of Descripti
def extract_storage(description):
    extract_storage = re.search(r'(\d+)\s*(GB|MB)\s*(?:ROM|Internal|Stora
    storage = int(extract_storage.group(1)) if extract_storage else None
    return storage
df['Storage (GB)'] = df['Description'].apply(extract_storage)

#Replace | to + in the Camera column
df['Camera'] = df['Camera'].str.replace('|','+')
#Handle Null values in the Camera column
df['Camera'] = df['Camera'].apply(lambda x: 'Not Present' if pd.isna(x) o
df['Camera'] = df['Camera'].str.replace('0MP + 0MP','Not Present')

#Create Star Category and Price Category columns
df['Star Category'] = pd.cut(df['Stars'], bins = [0,3.4, 3.8, 4.2, 4.6, 5
df['Price Category'] = pd.cut(df['Actual price ₹'], bins = [0, 10000, 200

#Extract Main and Second Cameras
def extract_main_cam(camera):
    extract_main_camera = re.search(r'(\d+)MP\s*(\+)?\s*(\d+MP)?', camera
    main_camera = int(extract_main_camera.group(1)) if extract_main_camer
    return main_camera
df['Main Camera'] = df['Camera'].apply(extract_main_cam)

def extract_second_cam(camera):
    extract_second_camera = re.search(r'(\d+)MP\s*(\+)?\s*(\d+)(MP)', cam
    second_camera = int(extract_second_camera.group(3)) if extract_second
    return second_camera
df['Second Camera'] = df['Camera'].apply(extract_second_cam)

#Define the desired column order
desired_order = ['Product Name', 'Brand', 'Price Category', 'Actual price
```

```
df = df[desired_order]

#Drop the Description and Link columns
df = df.drop(columns =['Description', 'Link'])

#Duplicate the original data for further price analysis (still contain Ap
df_dup = df.copy()

#Handle the remaining null values in columns
print(df.isnull().sum())
df = df.dropna()

df_dup.head(10)
```

```
Product Name            0
Brand                   0
Price Category          0
Actual price ₹          0
Discount price ₹        0
Discount amount (%)     0
Stars                   0
Star Category           0
Number of Rating        0
Number of Reviews       0
RAM (GB)               55
Storage (GB)            4
Display Size (inch)     0
Camera                  0
Main Camera            90
Second Camera         388
dtype: int64
```

Out[ ]:

	Product Name	Brand	Price Category	Actual price ₹	Discount price ₹	Discount amount (%)	Stars	Star Category	N
0	Apple Iphone 15 (Green, 128 Gb)	Apple	Luxury	79600.0	65999.0	17.09	4.6	Good	
1	Apple Iphone 15 (Blue, 128 Gb)	Apple	Luxury	79600.0	65999.0	17.09	4.6	Good	
2	Apple Iphone 15 (Black, 128 Gb)	Apple	Luxury	79600.0	65999.0	17.09	4.6	Good	
3	Oneplus N20 Se (Jade Wave,	Oneplus	Mid	19999.0	11489.0	42.55	4.0	Fair	

	128 Gb)							
4	Oneplus N20 Se (Blue Oasis, 64 Gb)	Oneplus	Mid	16999.0	12999.0	23.53	4.0	Fair
5	Oneplus 12R (Cool Blue, 128 Gb)	Oneplus	Premium	39999.0	38989.0	2.53	4.5	Good
6	Samsung Galaxy F14 5G (Goat Green, 128 Gb)	Samsung	Mid	17490.0	10990.0	37.16	4.2	Fair
7	Cmf By Nothing Phone 1 (Blue, 128 Gb)	Cmf	Mid	19999.0	15999.0	20.00	4.4	Good
8	Cmf By Nothing Phone 1 (Blue, 128 Gb)	Cmf	High	21999.0	17999.0	18.18	4.3	Good
9	Vivo Y200E 5G (Black Diamond, 128 Gb)	Vivo	High	25999.0	20999.0	19.23	4.3	Good

In [ ]:

```
df['RAM (GB)'].unique()
df['Main Camera'].value_counts()
#Drop unreal RAM and Main Camera columns
df = df[df['RAM (GB)'] != 46875]
df_dup = df_dup[df_dup['RAM (GB)'] != 46875]
df = df[df['Main Camera'] != 108.0]
df_dup = df_dup[df_dup['Main Camera'] != 108.0]
#Print cleaned dataser
df.head(10)
```

Out[ ]:

	Product Name	Brand	Price Category	Actual price ₹	Discount price ₹	Discount amount (%)	Stars	Star Category	I
	Samsung Galaxy								

6	F14 5G (Goat Green, 128 Gb)	Samsung	Mid	17490.0	10990.0	37.16	4.2	Fair
7	Cmf By Nothing Phone 1 (Blue, 128 Gb)	Cmf	Mid	19999.0	15999.0	20.00	4.4	Good
8	Cmf By Nothing Phone 1 (Blue, 128 Gb)	Cmf	High	21999.0	17999.0	18.18	4.3	Good
9	Vivo Y200E 5G (Black Diamond, 128 Gb)	Vivo	High	25999.0	20999.0	19.23	4.3	Good
10	Vivo Y200E 5G (Black Diamond, 128 Gb)	Vivo	High	23999.0	19999.0	16.67	4.2	Fair
11	Oppo F25 Pro 5G (Ocean Blue, 128 Gb)	Oppo	High	28999.0	23999.0	17.24	4.3	Good
12	Motorola G85 5G (Urban Grey, 128 Gb)	Motorola	High	20999.0	17999.0	14.29	4.5	Good
13	Motorola G85 5G (Urban Grey, 128 Gb)	Motorola	High	20999.0	17999.0	14.29	4.5	Good
15	Motorola G64 5G (Ice Lilac, 256 Gb)	Motorola	Mid	19999.0	16999.0	15.00	4.2	Fair
17	Nothing Phone (2A) 5G	Nothing	High	25999.0	23999.0	7.69	4.4	Good

(Blue,
128 Gb)

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 527 entries, 6 to 977
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Product Name         527 non-null    object
 1   Brand                527 non-null    object
 2   Price Category       527 non-null    category
 3   Actual price ₹       527 non-null    float64
 4   Discount price ₹     527 non-null    float64
 5   Discount amount (%)  527 non-null    float64
 6   Stars                527 non-null    float64
 7   Star Category        527 non-null    category
 8   Number of Rating     527 non-null    object
 9   Number of Reviews    527 non-null    object
 10  RAM (GB)             527 non-null    float64
 11  Storage (GB)         527 non-null    float64
 12  Display Size (inch)  527 non-null    float64
 13  Camera               527 non-null    object
 14  Main Camera          527 non-null    float64
 15  Second Camera        527 non-null    float64
dtypes: category(2), float64(9), object(5)
memory usage: 63.2+ KB
```

In [ ]:
```
#Convert to category and int types in the cleaned dataset
df = df.astype({'Product Name': 'category', 'Brand':'category', 'Actual p
                 'Discount price ₹': 'int', 'Number of Rating': 'int','Nu
                 'Display Size (inch)': 'int', 'Main Camera': 'int', 'Sec
print(df.info())
print(df.describe(include= "all"))
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 527 entries, 6 to 977
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Product Name         527 non-null    category
 1   Brand                527 non-null    category
 2   Price Category       527 non-null    category
 3   Actual price ₹       527 non-null    int64
 4   Discount price ₹     527 non-null    int64
 5   Discount amount (%)  527 non-null    float64
 6   Stars                527 non-null    float64
 7   Star Category        527 non-null    category
 8   Number of Rating     527 non-null    int64
 9   Number of Reviews    527 non-null    int64
 10  RAM (GB)             527 non-null    int64
 11  Storage (GB)         527 non-null    int64
```

```
 12  Display Size (inch)  527 non-null    int64
 13  Camera               527 non-null    object
 14  Main Camera          527 non-null    int64
 15  Second Camera        527 non-null    int64
dtypes: category(4), float64(2), int64(9), object(1)
memory usage: 67.9+ KB
None
                                                    Product Name    Brand  \
count                                                        527      527
unique                                                       338       18
top     Oppo K12X 5G With 45W Supervooc Charger In-The...  Samsung
freq                                                          31       93
mean                                                         NaN      NaN
std                                                          NaN      NaN
min                                                          NaN      NaN
25%                                                          NaN      NaN
50%                                                          NaN      NaN
75%                                                          NaN      NaN
max                                                          NaN      NaN

       Price Category  Actual price ₹  Discount price ₹  Discount amount
(%)  \
count             527      527.000000        527.000000            527.0000
00
unique              5             NaN               NaN                   N
aN
top              High             NaN               NaN                   N
aN
freq              181             NaN               NaN                   N
aN
mean              NaN    28809.624288      25449.853890             10.1307
40
std               NaN    19153.496740      20853.271741             59.6662
18
min               NaN     1199.000000        899.000000           -600.6200
00
25%               NaN    18999.000000      14999.000000             13.7900
00
50%               NaN    25263.000000      20999.000000             18.1800
00
75%               NaN    32999.000000      31999.000000             25.0000
00
max               NaN   149999.000000     176999.000000             50.0000
00

            Stars Star Category  Number of Rating  Number of Reviews  \
count   527.00000           527        527.000000         527.000000
unique        NaN             4               NaN                NaN
top           NaN          Good               NaN                NaN
freq          NaN           313               NaN                NaN
mean      4.29203           NaN      21463.502846        1522.990512
std       0.16388           NaN      51286.624665        3104.517259
min       3.50000           NaN          4.000000           0.000000
```

```
25%        4.20000           NaN       904.500000         65.000000
50%        4.30000           NaN      5823.000000        454.000000
75%        4.40000           NaN     17216.000000       1510.000000
max        5.00000           NaN    429459.000000      23258.000000

          RAM (GB)  Storage (GB)  Display Size (inch)      Camera  \
count   527.000000    527.000000           527.000000         527
unique         NaN           NaN                  NaN          37
top            NaN           NaN                  NaN  50MP + 2MP
freq           NaN           NaN                  NaN         109
mean      8.707780    195.218216             5.806452         NaN
std       4.701953    105.790218             0.914873         NaN
min       2.000000      4.000000             1.000000         NaN
25%       8.000000    128.000000             6.000000         NaN
50%       8.000000    128.000000             6.000000         NaN
75%       8.000000    256.000000             6.000000         NaN
max      32.000000    512.000000             7.000000         NaN

        Main Camera  Second Camera
count    527.000000     527.000000
unique          NaN            NaN
top             NaN            NaN
freq            NaN            NaN
mean      40.277040       8.865275
std       24.105344      11.847048
min        2.000000       0.000000
25%       13.000000       2.000000
50%       50.000000       5.000000
75%       50.000000      10.000000
max      200.000000      64.000000
```

In [ ]:
```
df_dup.isnull().sum()
```

Out[ ]:
```
Product Name             0
Brand                    0
Price Category           0
Actual price ₹           0
Discount price ₹         0
Discount amount (%)      0
Stars                    0
Star Category            0
Number of Rating         0
Number of Reviews        0
RAM (GB)                55
Storage (GB)             4
Display Size (inch)      0
Camera                   0
Main Camera             90
Second Camera          368
dtype: int64
```

In [ ]:
```
#Convert to category and int types in the not cleaned dataset
df_dup = df_dup.astype({'Product Name': 'category', 'Brand':'category', '
                  'Discount price ₹': 'int', 'Number of Rating': 'int','Nu
```

```
                      'Display Size (inch)': 'int'})
df_dup.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 934 entries, 0 to 983
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Product Name          934 non-null    category
 1   Brand                 934 non-null    category
 2   Price Category        934 non-null    category
 3   Actual price ₹        934 non-null    int64
 4   Discount price ₹      934 non-null    int64
 5   Discount amount (%)   934 non-null    float64
 6   Stars                 934 non-null    float64
 7   Star Category         934 non-null    category
 8   Number of Rating      934 non-null    int64
 9   Number of Reviews     934 non-null    int64
 10  RAM (GB)              879 non-null    float64
 11  Storage (GB)          930 non-null    float64
 12  Display Size (inch)   934 non-null    int64
 13  Camera                934 non-null    object
 14  Main Camera           844 non-null    float64
 15  Second Camera         566 non-null    float64
dtypes: category(4), float64(6), int64(5), object(1)
memory usage: 122.0+ KB
```

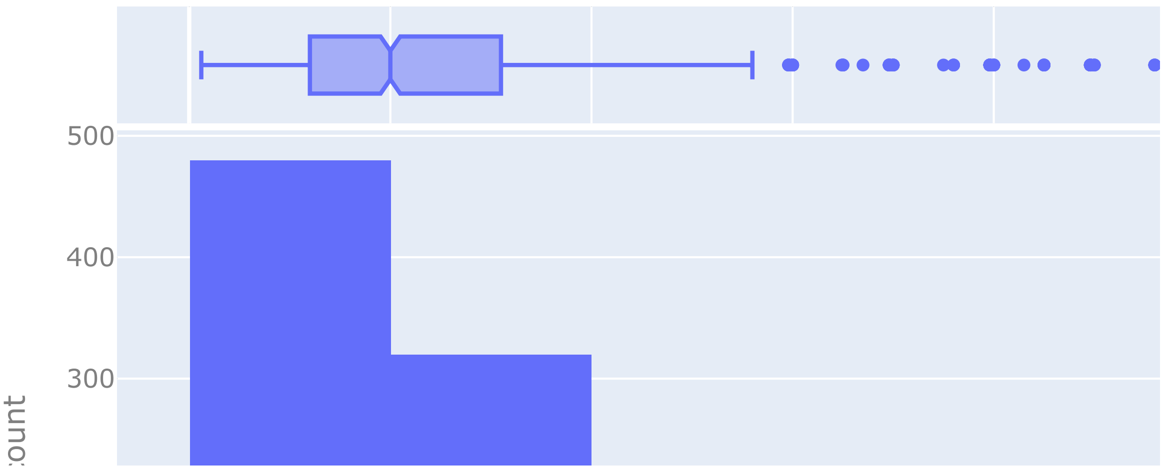
5. Data Visualization

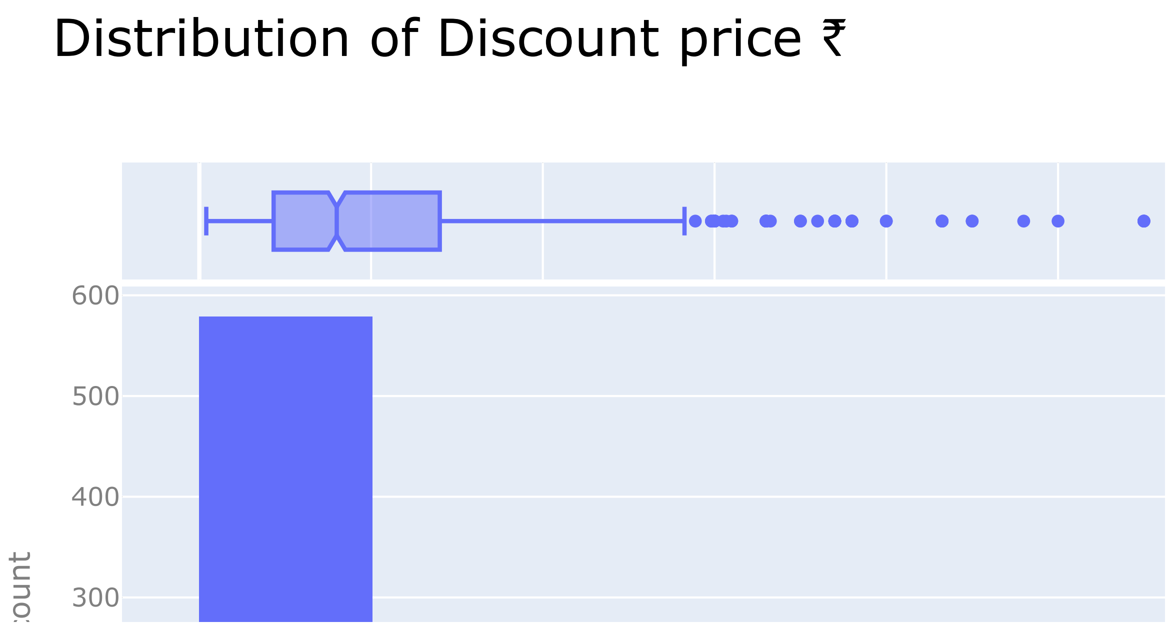
5.1 Overview

In [ ]:
```
# Create histograms for each numeric column
numeric_df = df_dup.select_dtypes(include =[float,int])
numeric_df = pd.DataFrame(numeric_df)
print(numeric_df)
numeric_df_columns = numeric_df.columns
for column in numeric_df.columns:
    fig = px.histogram(df_dup, x=column, title=f'Distribution of {column}
                       nbins=10, marginal='box')
    fig.update_layout(font_color="grey", font_size =12,
        title_font_color="black", title_font_size =24)
    fig.show()
```

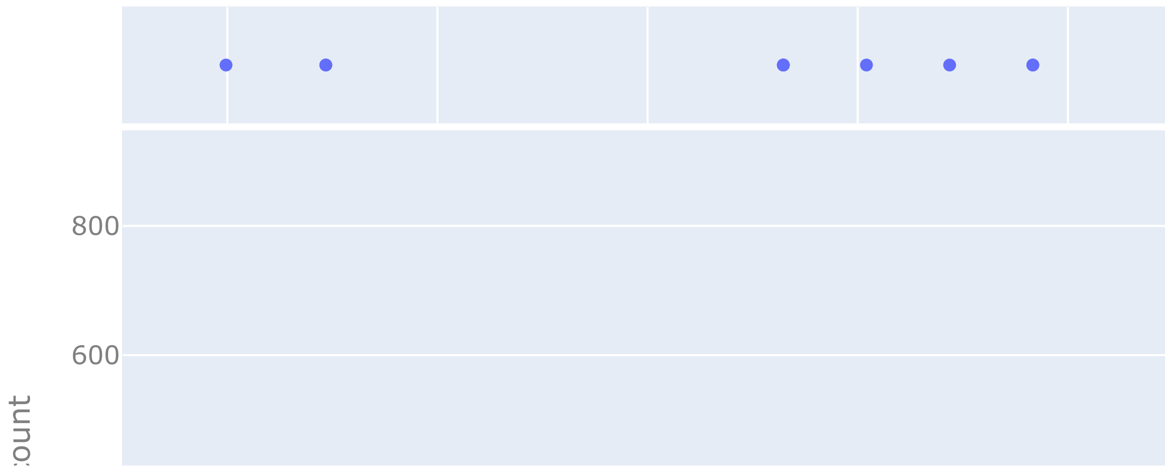
```
     Actual price ₹  Discount price ₹  Discount amount (%)  Stars  \
0             79600             65999                17.09    4.6   
1             79600             65999                17.09    4.6   
2             79600             65999                17.09    4.6   
3             19999             11489                42.55    4.0   
4             16999             12999                23.53    4.0   
..              ...               ...                  ...    ...   
979            1499               967                35.49    4.0   
980            1499               975                34.96    4.0   
981            1499               975                34.96    4.0   
982            1499               930                37.96    4.0   
983            1499               967                35.49    4.0   

     Number of Rating  Number of Reviews  RAM (GB)  Storage (GB)  \
0               44793               2402       NaN         128.0   
1               44793               2402       NaN         128.0   
2               44793               2402       NaN         128.0   
3                1005                 41       4.0         128.0   
4                1005                 41       4.0          64.0   
..                ...                ...       ...           ...   
979             11022                693      32.0          32.0   
980             11022                693      32.0          32.0   
981             11022                693      32.0          32.0   
982             11022                693      32.0          32.0   
983             11022                693      32.0          32.0   

     Display Size (inch)  Main Camera  Second Camera  
0                      6         48.0           12.0  
1                      6         48.0           12.0  
2                      6         48.0           12.0  
3                      6         50.0            NaN  
4                      6         50.0            NaN  
..                   ...          ...            ...  
979                    0          NaN            NaN  
980                    0          NaN            NaN  
981                    0          NaN            NaN  
982                    0          NaN            NaN  
983                    0          NaN            NaN  

[934 rows x 11 columns]
```

Distribution of Actual price ₹

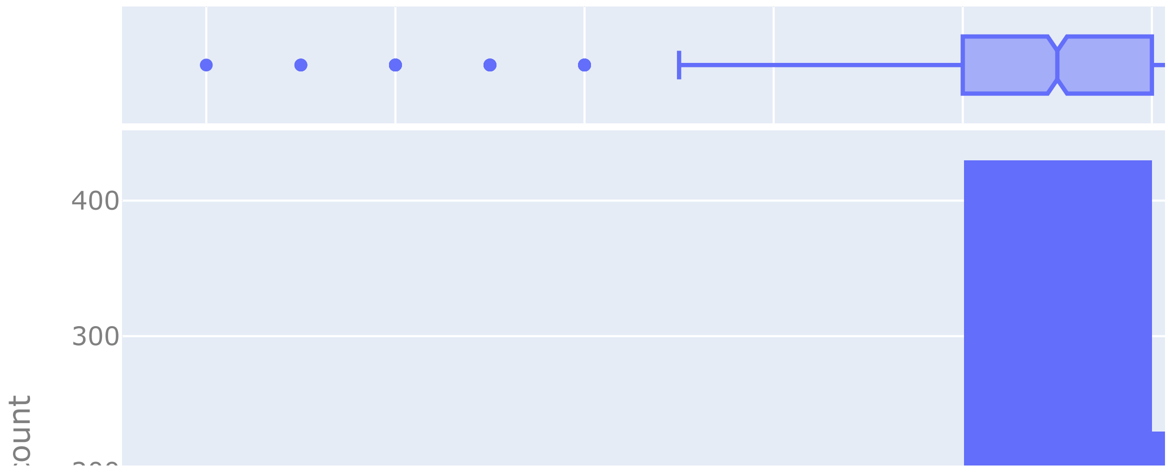



Distribution of Discount price ₹
600
500
400
300
ount

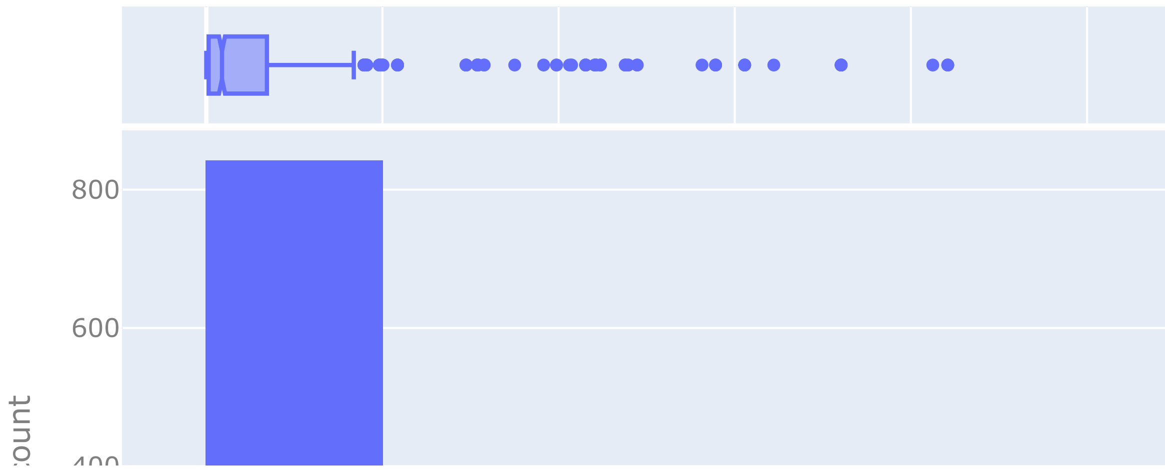
Distribution of Discount amount (%)

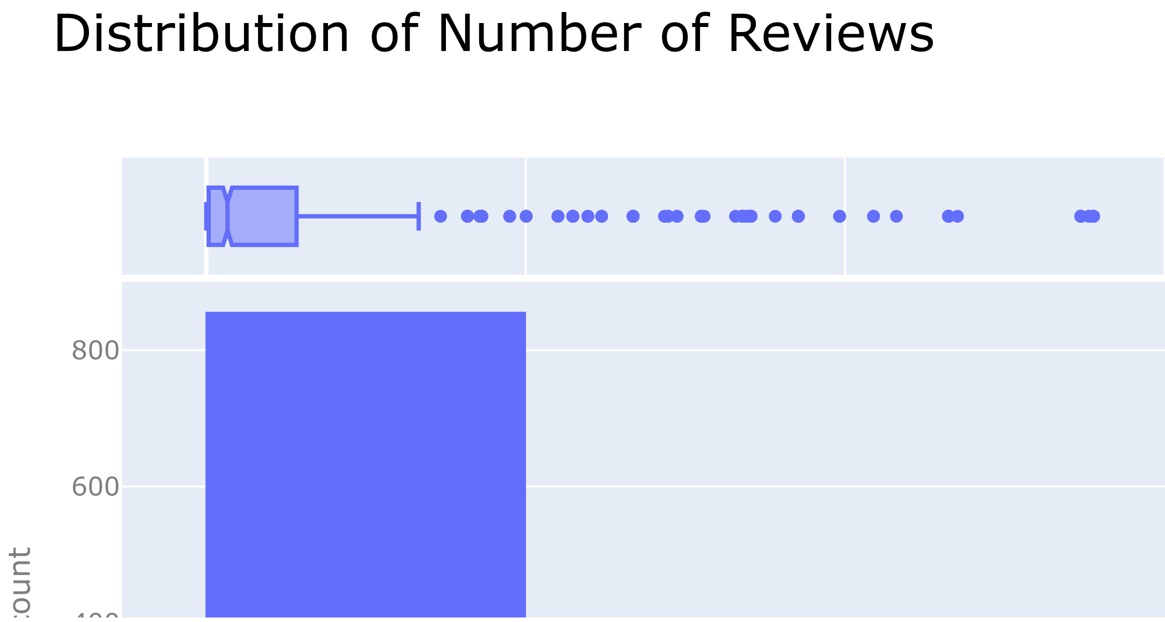


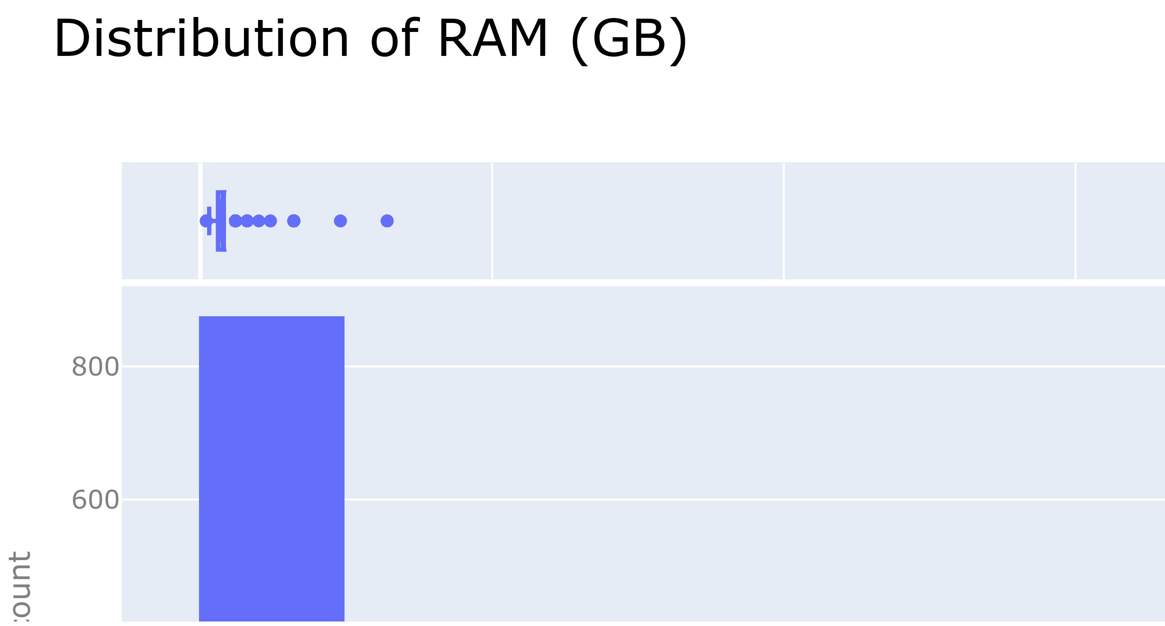
Distribution of Stars

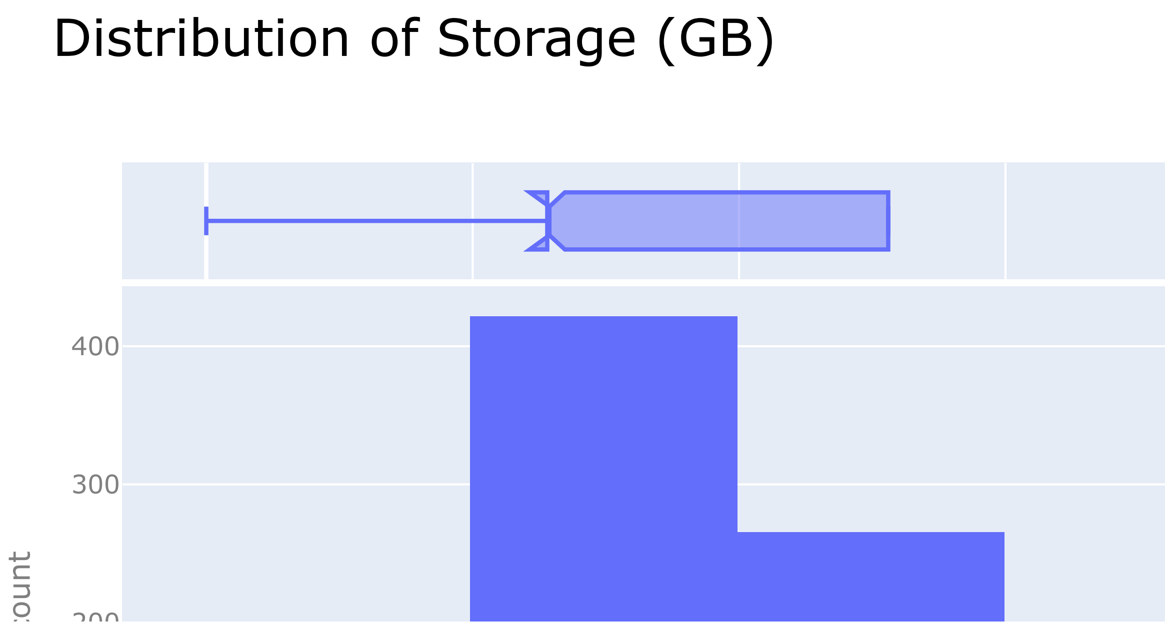


Distribution of Number of Rating

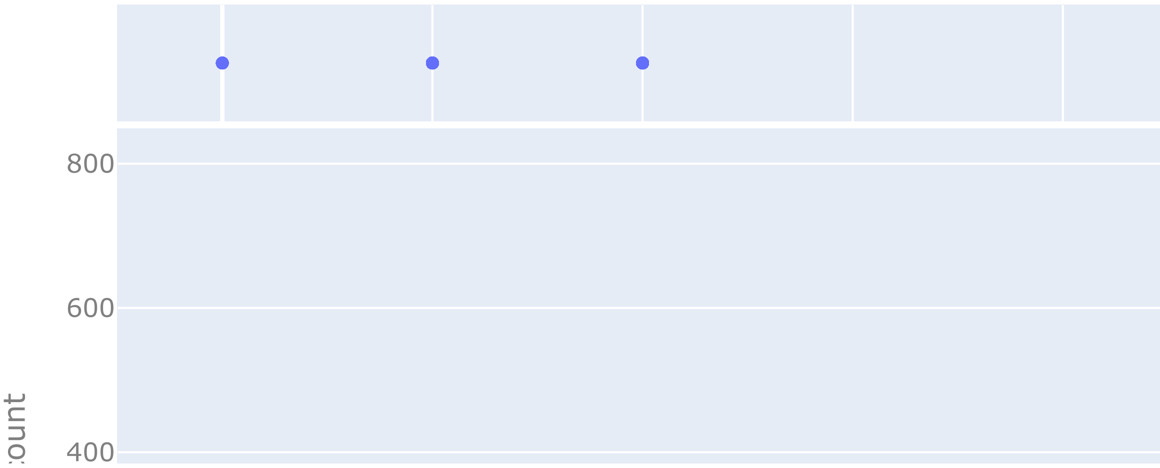



Distribution of Number of Reviews
800
600
count

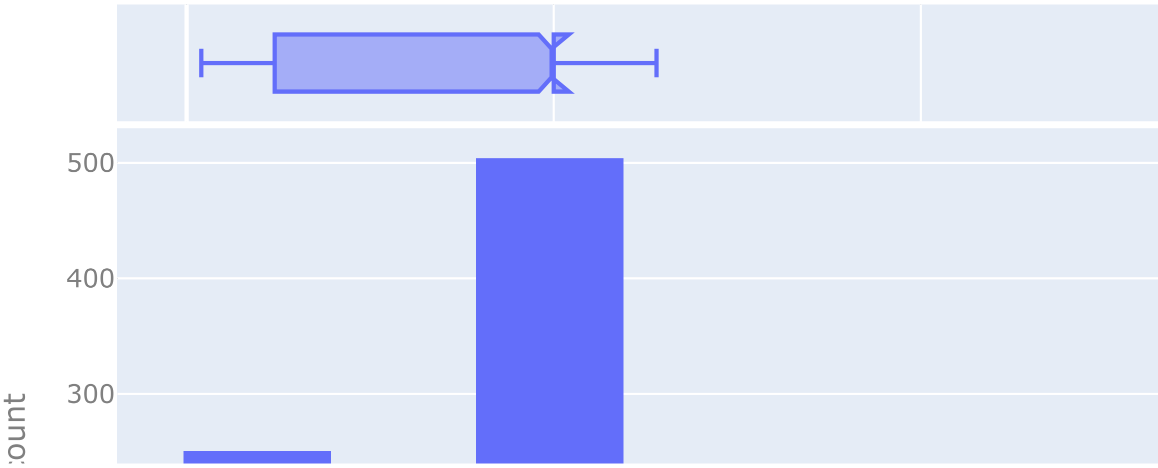

Distribution of RAM (GB)
800
600
count


Distribution of Storage (GB)
400
300
count

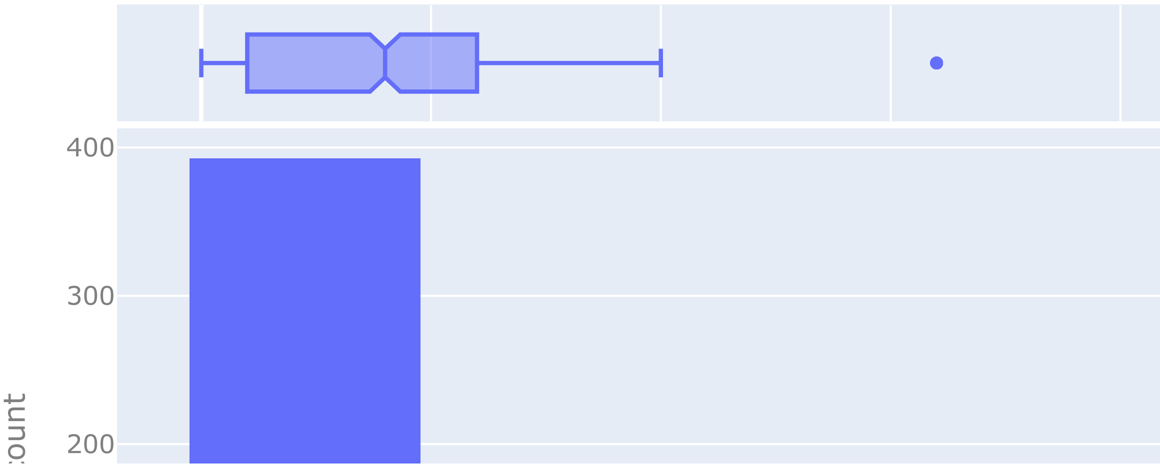
Distribution of Display Size (inch)



Distribution of Main Camera



Distribution of Second Camera



- Price Distribution:
    - The actual price chart shows that most mobile phones are clustered around the ₹20,000 to ₹40,000 range, with fewer phones in the premium (₹40,000–₹80,000) and luxury (above ₹80,000) ranges
    - Discount price distribution is heavily skewed toward lower prices, indicating aggressive discounting for mid-tier phones
- Discount Distribution:
    - Brands like Honor, Micromax, and Poco offer the highest discount percentages (up to 50%), while premium brands like Apple, Google, and Samsung offer minimal discounts
    - Mid-range phones see the highest average discount rates (around 25%), with high-end models offering much smaller discounts, or in some cases, none at all
- Star Distribtion:
    - The majority of phones have star ratings between 4.2 and 4.4, with a small number achieving ratings above 4.5. Very few phones have ratings below

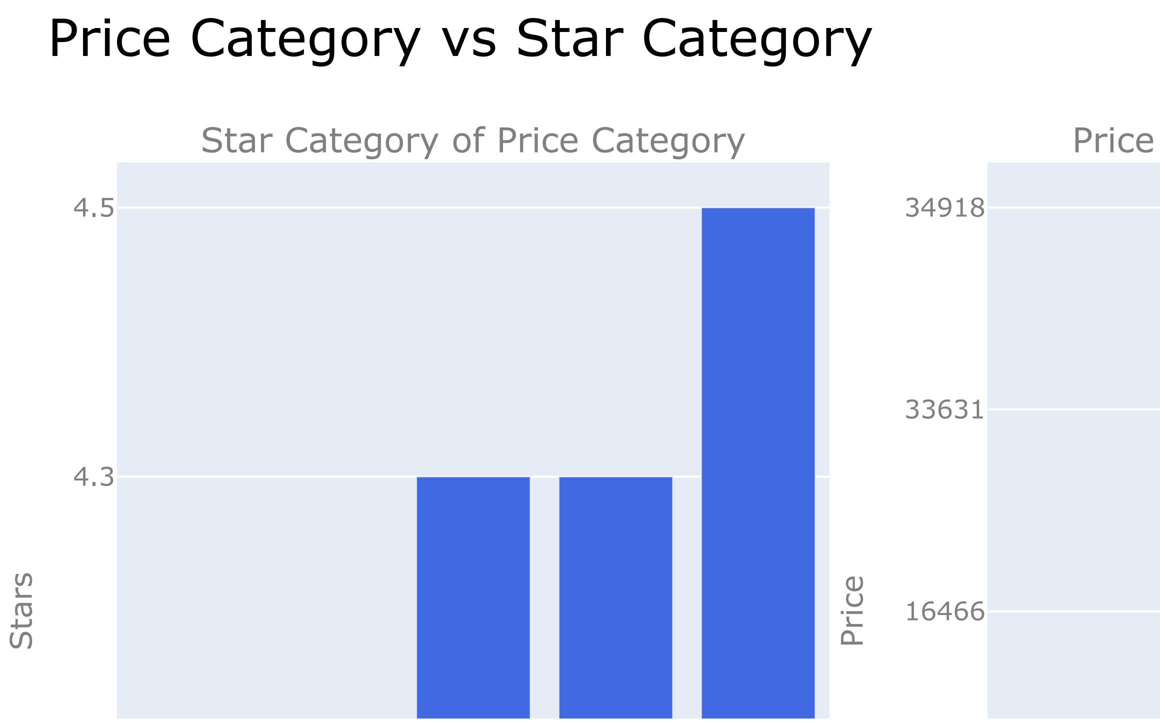
4.0, which suggests that most products are perceived as having decent quality by customers
    - Brands with high ratings (4.5 and above) include premium players like Apple, OnePlus, and Samsung, where consumer satisfaction tends to be higher
    - Brands with lower ratings (around 3.6 to 3.8) include lesser-known or budget brands like Vox, Karbonn, and Jio, indicating some level of dissatisfaction or unmet customer expectations
- Review Distribution:
    - Total reviews are heavily skewed towards well-known brands, with companies like Apple, Samsung, and Realme collecting the most reviews. Apple, for instance, has over 276,000 reviews, indicating a high level of customer engagement
    - Brands with fewer reviews include Vox, Karbonn, and Jio, which have under 100 reviews, reflecting their limited market reach or consumer engagement
    - Average reviews per product vary significantly, with premium brands typically garnering more reviews per product (e.g., Apple averages around 7,086 reviews per product) compared to budget brands (e.g., Vox with around 6 reviews per product)
- RAM Distribution:
    - The most common RAM configuration is 8GB, dominating the market with a significant share (around 361 entries). This is followed by 4GB and 12GB, with smaller shares
    - High-end configurations like 16GB or 32GB RAM are relatively rare and are typically found in premium devices
    - Low-end configurations like 2GB or 4GB RAM appear mostly in budget smartphones
- Storage Distribution:
    - 128GB storage is the most popular configuration, capturing a significant portion of the market (421 entries). This is followed by 256GB, with higher configurations like 512GB being less common and reserved for premium models
    - Smaller configurations like 32GB and 64GB are seen in lower-end devices, while 4GB storage is very rare and usually found in ultra-budget or legacy models
- Main Cam Distribution:
    - The most common main camera configuration is 50MP, particularly in mid-range and high-end devices. The rest of the configurations include 48MP, 12MP, and a few lower-end models featuring 2MP cameras
    - Higher-end models feature cameras in the range of 50MP, while budget phones stick to lower resolutions like 12MP or 2MP
- Second Cam Distribution:

- The most common second camera resolution is 2MP, especially in mid-range phones with dual-camera setups. The higher-end models have 12MP or better second cameras
- Lower-tier models either don't have second cameras or feature basic 2MP cameras as a secondary sensor

In [ ]:

```
#Calculate star category of price category
star_price = df_dup.groupby('Price Category')['Stars'].mean().round(1).so
star_price['Star Category'] = pd.cut(star_price['Stars'], bins=[0, 3.4, 3
print(star_price)

#Calculate price category of star category
price_star = df_dup.groupby('Star Category')['Actual price ₹'].mean().ast
price_star['Price Category'] = pd.cut(price_star['Actual price ₹'], bins
print(price_star)

categoryarray=[0,4.1,4.2,4.3,4.5]

#Visualize star category of price category
fig= make_subplots(rows=1, cols=2, subplot_titles=('Star Category of Pric

fig.add_bar(x=['Low', 'Mid'], y=[4.1, 4.2], marker_color='cornflowerblue'
fig.add_bar(x=['High', 'Premium','Luxury'], y=[4.3, 4.3,4.5], marker_colo
fig.update_xaxes(title_text='Price Category', type='category', row=1, col
fig.update_yaxes(title_text='Stars', type='category', categoryorder='arra

#Visualize price category of star category
#To see for example, if the rate is good, then customers are willing to p
fig.add_bar(x=['Poor', 'Not Preferred'], y=[1699, 7746], marker_color='ye
fig.add_bar(x=['Fair'], y=[16466], marker_color='olivedrab', name='Mid',
fig.add_bar(x=['Excellent', 'Good'], y=[33631,34918], marker_color='darko
fig.update_xaxes(title_text='Star Category', type='category', row=1, col=
fig.update_yaxes(title_text='Price', type='category', categoryorder='arra

fig.update_layout(title='Price Category vs Star Category',title_font_size
fig.show()
```

```
  Price Category  Stars Star Category
0            Low    4.1          Fair
1            Mid    4.2          Fair
2           High    4.3          Good
3        Premium    4.3          Good
4         Luxury    4.5          Good
    Star Category  Actual price ₹ Price Category
0            Poor            1699            Low
1   Not Preferred            7746            Low
2            Fair           16466            Mid
3       Excellent           33631        Premium
4            Good           34918        Premium
```



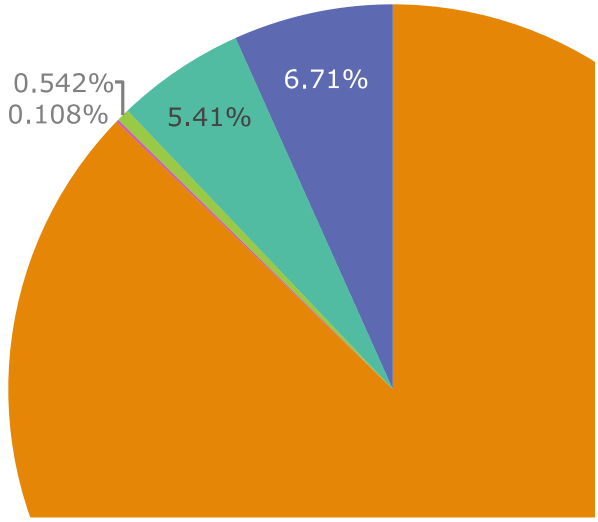
- Premium and Luxury Brands (first chart) tend to receive the highest ratings (above 4.3 stars), which shows that customers perceive these products as superior in quality, even though they are priced higher
- Low and Mid-range Price Categories (first chart) have comparatively lower star ratings, reflecting that price-sensitive customers might be more critical of product performance or experience
- The second chart shows that as star ratings increase, the price of the products also rises. Products with ratings categorized as "Excellent" and "Good" fall mostly into the higher price range (Premium category). This mean that if customers feel 'good', they are willing to pay high amount of money for a new phone.

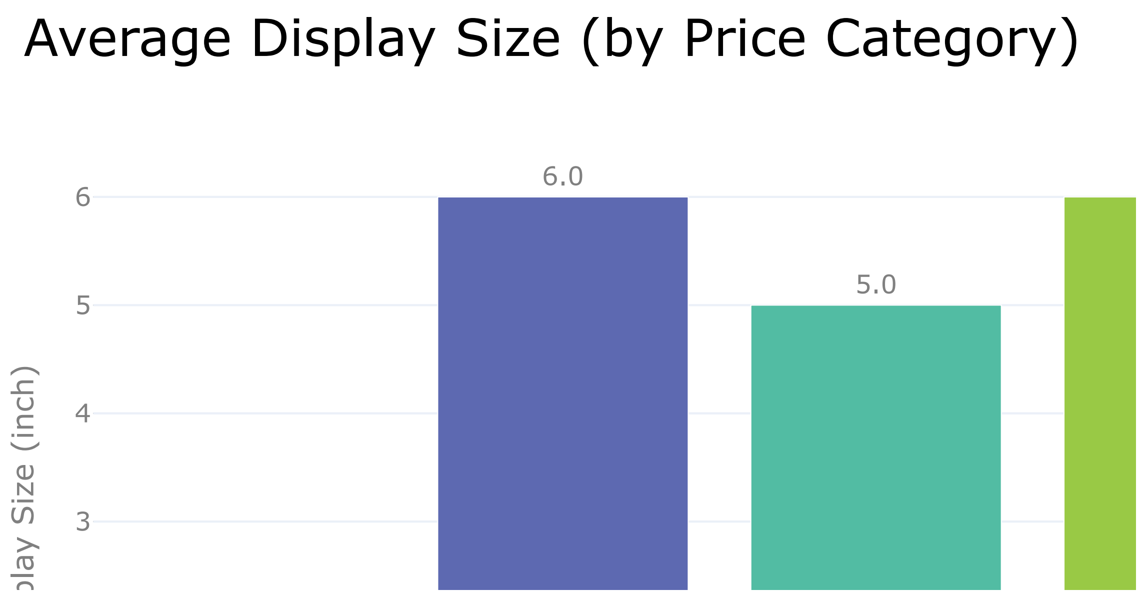
In [ ]:
```
#Visualize top 10 most common Display Size (inch)
```

```
most_common_display = df_dup['Display Size (inch)'].value_counts().reset_
most_common_display =most_common_display.rename(columns={'count':'Number 
most_common_display['Common Display Size Percentage']= (most_common_displ
most_common_display = pd.DataFrame(most_common_display)
most_common_display = most_common_display[most_common_display['Display Si
print(most_common_display)
display_pr_category = df_dup.groupby('Price Category')['Display Size (inc
print(display_pr_category)

fig1 = px.pie(most_common_display,values='Common Display Size Percentage'
             template="plotly_white", color_discrete_sequence=px.colors.q
fig1.update_layout(title_font_size =24, title_text ='Percentage of Displa
                  font_size=12,font_color="grey", height=500)

fig1.show()
#Visualize Average Display size of each Price Category
fig2 = px.bar(display_pr_category, x= 'Price Category', y='Display Size (
             color = 'Price Category', template="plotly_white", color_dis
             text = 'Display Size (inch)')
fig2.update_traces(textposition='outside', texttemplate='%{text:.2s}')
fig2.update_layout(height=500,
        font_color="grey", font_size =12,
        title_font_color="black", title_font_size =24)
fig2.show()
```

```
   Display Size (inch)  Number of Display Size (inch)  \
0                    6                            806
1                    1                             62
2                    2                             50
4                    7                              5
5                    5                              1

   Common Display Size Percentage
0                          152.94
1                           11.76
2                            9.49
4                            0.95
5                            0.19
  Price Category  Display Size (inch)
0            Low                    2
1            Mid                    6
2           High                    5
3        Premium                    6
4         Luxury                    6
```

Percentage of Display Size (inch)





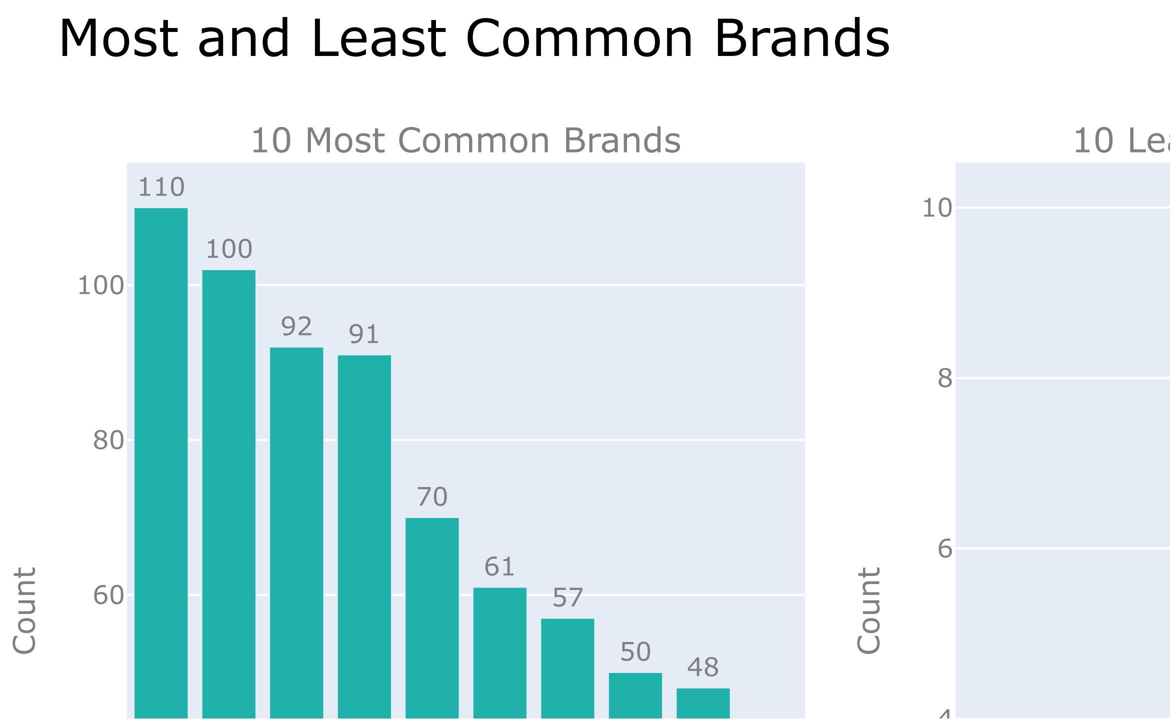
- At the present, the most common display size is 6 inch, followed by 1 and 2 inch. Meanwhile, the percentages of 7 and 5 inch phone are very small, illustrating that customers are leaning towards a 6 inch phone more than a very big one or a very small one.
- The mean display size is also consistent with price categories, in which customers choosing mid, premium, and luxury brands usally select 6.0 inch.

5.2 Brand Analysis

In [ ]:
```
#Calculate top 10 most common brand
most_common_brand = df_dup['Brand'].value_counts().sort_values(ascending
most_common_brand = most_common_brand.rename(columns={'count':'Most Commo
most_common_brand = most_common_brand.head(10)
#Calculate top 10 least common brand
least_common_brand = df_dup['Brand'].value_counts().sort_values(ascending
least_common_brand = least_common_brand.rename(columns={'count':'Least Co
least_common_brand = least_common_brand.head(10)
#Turn into dataframe
most_least_common_brand = pd.concat([most_common_brand, least_common_bran
```

```
print(most_least_common_brand)

#Visualise by drawing 2 bar charts side by side to compare
fig = make_subplots(rows=1, cols=2, subplot_titles = ('10 Most Common Bra
fig.add_bar(x=most_least_common_brand['Common Brand'], y=most_least_commo
fig.add_bar(x=most_least_common_brand['Not Common Brand'], y=most_least_c

fig.update_traces(textposition='outside', texttemplate='%{text:.2s}', row
fig.update_traces(textposition='outside', texttemplate='%{text:.1s}', row
fig.update_xaxes(title_text ='Brand', row=1, col=1)
fig.update_xaxes(title_text ='Brand', row=1, col=2)
fig.update_yaxes(title_text ='Count', row=1, col=1)
fig.update_yaxes(title_text ='Count', row=1, col=2)
fig.update_layout(title='Most and Least Common Brands', font_color="grey"
fig.show()
```

```
  Common Brand  Most Common Brand Count Not Common Brand  \
0       Realme                      110            Honor
1      Samsung                      102              Jio
2        Redmi                       92              Vox
3         Vivo                       91                I
4         Oppo                       70        Blackzone
5     Motorola                       61         Micromax
6         Poco                       57              Cmf
7      Infinix                       50           Xiaomi
8         Itel                       48          Karbonn
9        Apple                       39          Nothing

   Least Common Brand Count
0                         1
1                         2
2                         2
3                         3
4                         3
5                         4
6                         6
7                         8
8                         9
9                        10
```

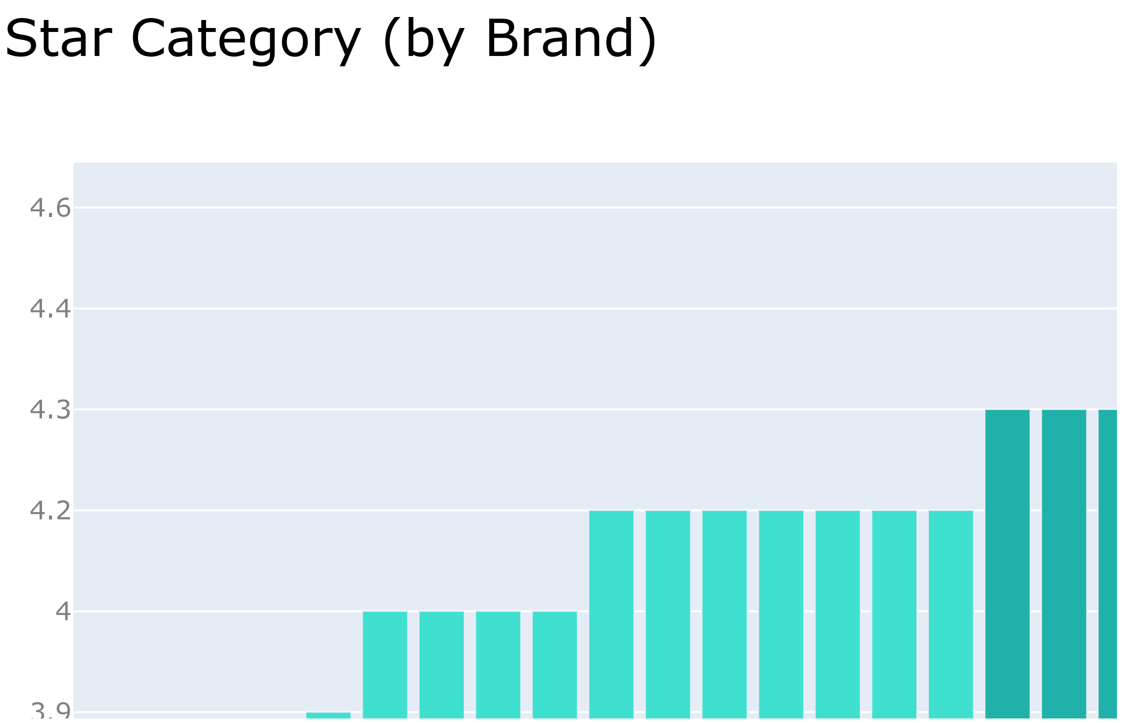


- Most Common Brands:
  - Realme leads the market with 110 counts, followed by Samsung (100), Redmi (92), and Vivo (91)
  - The top brands, especially Realme and Samsung, dominate significantly, with counts close to or above 100
- Least Common Brands:
  - Honor, Jio, and Vox appear at the bottom with only 1-2 counts
  - Xiaomi, despite being a recognized brand, is listed among the least common, potentially indicating a region-specific trend or particular time period

In [ ]:
```
#Calculate average star reviews of each brand and turn it into category
brand_star = df_dup.groupby('Brand')['Stars'].mean().round(1).sort_values
```

```
brand_star['Star Category'] = pd.cut(brand_star['Stars'], bins=[0, 3.4, 3
print(brand_star)

not_preferred_brand = brand_star[brand_star['Star Category']=='Not Prefer
fair_brand = brand_star[brand_star['Star Category']=='Fair']
good_brand = brand_star[brand_star['Star Category']=='Good']

#Visualize average star category of each brand
#To see which one is most rated
fig= go.Figure()
fig.add_bar(x=not_preferred_brand['Brand'], y=not_preferred_brand['Stars'
fig.add_bar(x=fair_brand['Brand'], y=fair_brand['Stars'], marker_color='t
fig.add_bar(x=good_brand['Brand'], y=good_brand['Stars'], marker_color='l

fig.update_xaxes(type='category')
fig.update_yaxes(type='category', categoryorder='array', categoryarray=[0
fig.update_layout(title='Star Category (by Brand)',title_font_size=24, ti
fig.show()
```

```
        Brand  Stars  Star Category
0         Vox    3.6  Not Preferred
1     Karbonn    3.7  Not Preferred
2         Jio    3.8  Not Preferred
3           I    3.8  Not Preferred
4       Nokia    3.9           Fair
5    Kechaoda    4.0           Fair
6        Itel    4.0           Fair
7    Micromax    4.0           Fair
8   Blackzone    4.0           Fair
9     Infinix    4.2           Fair
10      Honor    4.2           Fair
11     Google    4.2           Fair
12       Lava    4.2           Fair
13      Tecno    4.2           Fair
14       Poco    4.2           Fair
15      Redmi    4.2           Fair
16     Realme    4.3           Good
17   Motorola    4.3           Good
18    Samsung    4.3           Good
19       Oppo    4.3           Good
20       Iqoo    4.3           Good
21       Vivo    4.4           Good
22     Xiaomi    4.4           Good
23    Nothing    4.4           Good
24        Cmf    4.4           Good
25    Oneplus    4.4           Good
26      Apple    4.6           Good
```



- Highly Rated Brands:
  - Apple has the highest star rating, reaching close to 4.6, followed by OnePlus and Nothing, which are also highly rated above 4.5
  - Popular brands like Xiaomi, Vivo, and Samsung are also rated well, all above 4.2
- Low Rated Brands:
  - Vox, Karbonn, and Jio receive the lowest ratings, between 3.6 and 3.8, categorizing them as "Not Preferred."

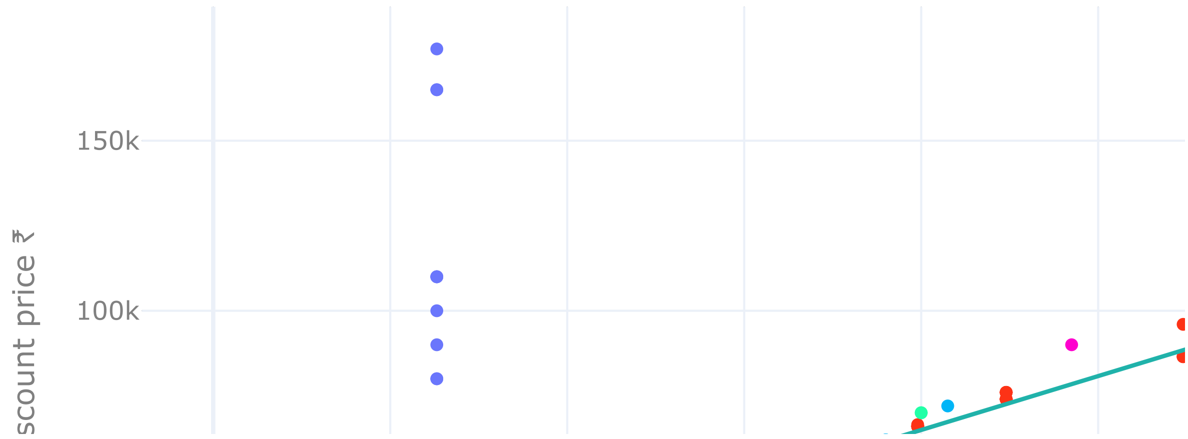
In [ ]:

```
#Calculate the mean of actual and discount price of each brand
average_actual_discount_price = df_dup.groupby('Brand')[['Actual price ₹'
average_actual_discount_price = pd.DataFrame(average_actual_discount_pric
print(average_actual_discount_price.head(10))
```

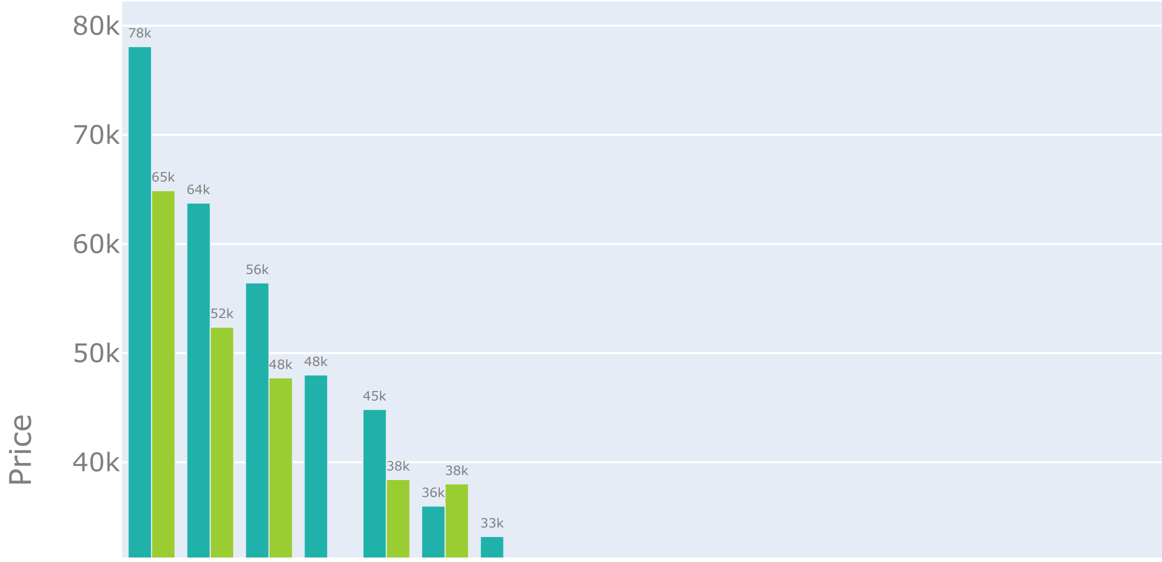
```
#Visualize the correlation of Actual & Discount Price (by Brand)
fig1 = px.scatter(df_dup, x= 'Actual price ₹', y='Discount price ₹', titl
                  color ='Brand', color_discrete_sequence=px.colors.quali
                  trendline="ols", trendline_scope="overall", trendline_c
fig1.update_layout(height=500, template="plotly_white",
        font_color="grey", font_size =12,
        title_font_color="black", title_font_size =24)
fig1.show()

#Visualise the mean of actual and discount price of each brand
fig2 = go.Figure(data=go.Bar(x=average_actual_discount_price['Brand'], y=
fig2.add_bar(x=average_actual_discount_price['Brand'], y=average_actual_d

fig2.update_traces(textposition='outside', texttemplate='%{text:.2s}')
fig2.update_xaxes(title_text ='Brand')
fig2.update_yaxes(title_text ='Price')
fig2.update_layout(font_color="grey", font_size =12,
                  title='Average Actual & Average Discount Price (by Bran
fig2.show()
```

	Brand	Actual price ₹	Discount price ₹
0	Apple	78084	64883
1	Xiaomi	63749	52374
2	Google	56434	47732
3	Honor	47999	25950
4	Oneplus	44838	38419
5	Samsung	35983	38022
6	Nothing	33199	27799
7	Vivo	28237	23924
8	Realme	25192	21138
9	Oppo	24988	19821

Correlation of Actual & Discount Price (by E



Average Actual & Average Discount Price (b



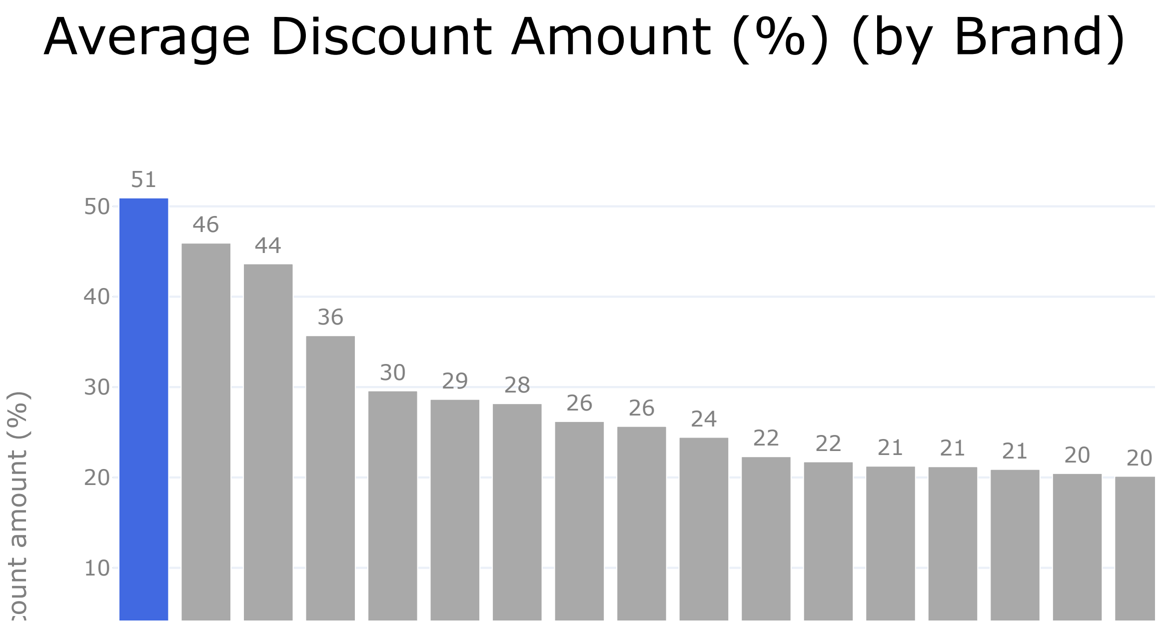
- Correlation of Actual & Discount Price (by Brand):
    - The chart shows a strong positive correlation between actual price and discount price. More expensive brands like Apple, Samsung, and OnePlus offer higher discounts in absolute terms, even though their percentage discounts may be relatively small
    - Brands like Realme, Oppo, and Vivo in the mid-range and low-price segments offer lower absolute discounts, but these are more impactful due to their lower price points
- Most Expensive Brands:
    - Apple stands out with an average price of 78k, followed by Xiaomi (65k) and Google (64k)
    - The more premium brands tend to maintain higher price points, such as

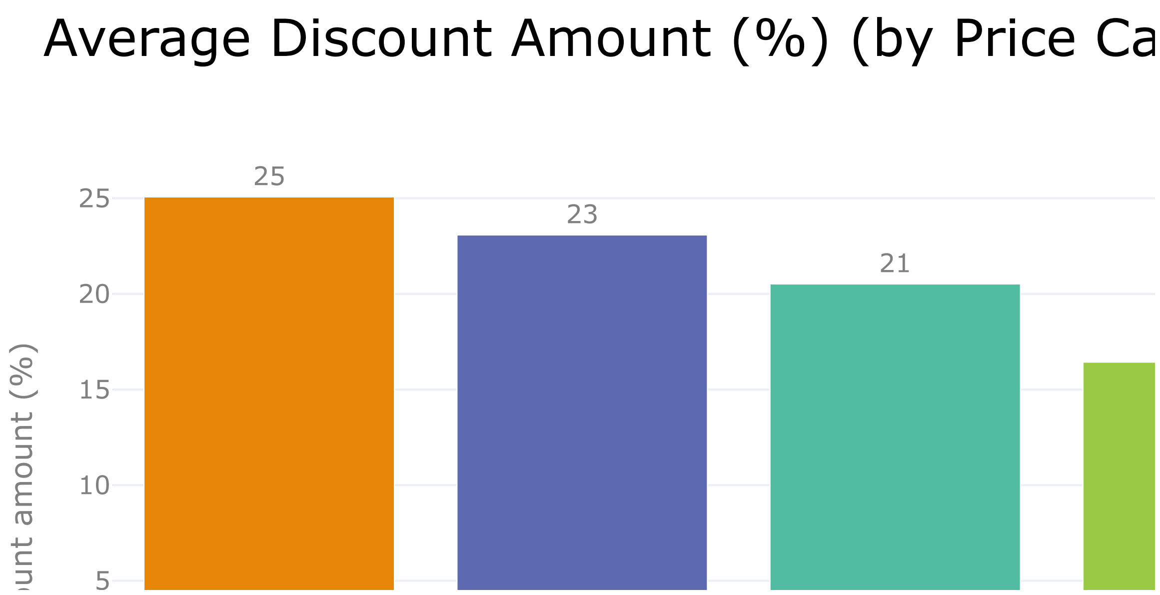
Honor (52k) and OnePlus (48k)

- Brands with Lower Prices:
    - Brands like Karbonn, Kechaoda, and Blackzone offer much lower prices, often in the 1-2k range
    - There's a clear divide in pricing between high-end brands (Apple, Google) and budget brands (Karbonn, Itel)

In [ ]:
```
#Calculate average discount % of brand and price category
average_discount_amount = df_dup.groupby('Brand')['Discount amount (%)'].
average_discount_amount = pd.DataFrame(average_discount_amount)
print(average_discount_amount.head(10))

avg_discount_price_category = df_dup.groupby('Price Category')['Discount
avg_discount_price_category = pd.DataFrame(avg_discount_price_category)
print(avg_discount_price_category)

#Visualize Average Discount Amount (%) of each Brand
average_discount_amount['color'] = "darkgray"
average_discount_amount['color'][0] = "royalblue"
average_discount_amount['color'][26] = "crimson"

fig1 = px.bar(average_discount_amount, x= 'Brand', y='Discount amount (%)
             color = 'color', template="plotly_white", color_discrete_seq
             text = 'Discount amount (%)')
fig1.update_traces(textposition='outside', texttemplate='%{text:.2s}')
fig1.update_layout(showlegend=False,
        font_color="grey", font_size =10,
        title_font_color="black", title_font_size =24)
fig1.show()

#Visualize Average Discount Amount (%) of each Price Category
fig2 = px.bar(avg_discount_price_category, x= 'Price Category', y='Discou
             color = 'Price Category', template="plotly_white", color_dis
             text = 'Discount amount (%)')
fig2.update_traces(textposition='outside', texttemplate='%{text:.2s}')
fig2.update_layout(height=500,
        font_color="grey", font_size =12,
        title_font_color="black", title_font_size =24)
fig2.show()
```

```
        Brand  Discount amount (%)
0         Vox                50.94
1       Honor                45.94
2           I                43.65
3    Micromax                35.70
4    Kechaoda                29.59
5   Blackzone                28.65
6       Nokia                28.18
7     Karbonn                26.20
8        Poco                25.66
9       Redmi                24.45
  Price Category  Discount amount (%)
0            Mid                25.07
1            Low                23.08
2         Luxury                20.52
3        Premium                16.43
4           High                -7.09
```





In the first chart:

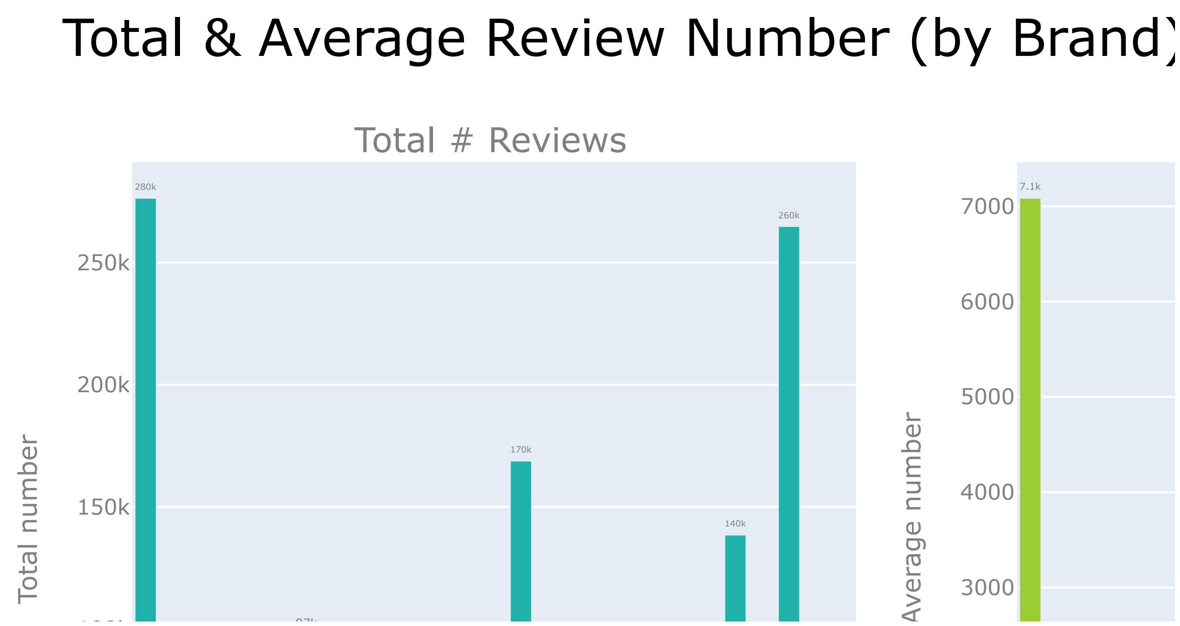
- Highest Discounts:
  - Vox offers the highest discount at 51%, followed by Honor (46%) and "I" (44%)
  - These brands are likely using heavy discounting to drive sales, which aligns with the previous data showing lower ratings and market share
- Negative Discount (Price Increase):
  - Interestingly, Samsung shows a negative discount (-22%), suggesting that their prices may have increased rather than decreased
  - Google also shows a very low discount of 3.7%, which is uncommon for the high-end brand category

In the second chart:

- Products in the mid price category usually are discounted more than the other price categories
- Meanwhile, those in the high price category are negatively discounted, with 7.1%

In [ ]:
```
#Compare Total and average number of reviews of each Brand
total_mean_brand_reviews = df_dup.groupby('Brand')['Number of Reviews'].a
total_mean_brand_reviews = total_mean_brand_reviews.rename(columns={'Tota
total_mean_brand_reviews['Average Number of Reviews']= total_mean_brand_r
print(total_mean_brand_reviews)

fig = make_subplots(rows=1, cols=2, subplot_titles = ('Total # Reviews',
fig.add_bar(x=total_mean_brand_reviews['Brand'], y=total_mean_brand_revie
fig.add_bar(x=total_mean_brand_reviews['Brand'], y=total_mean_brand_revie

fig.update_traces(textfont_size=12, textposition='outside', texttemplate=
fig.update_xaxes(title_text='Brand', row=1,col=1)
fig.update_xaxes(title_text='Brand', row=1,col=2)
fig.update_yaxes(title_text='Total number', row=1,col=1)
fig.update_yaxes(title_text='Average number', row=1,col=2)
fig.update_layout(title="Total & Average Review Number (by Brand)", title
                  font_size=10,font_color="grey",
                  xaxis_tickangle=90)

fig.show()
```

```
        Brand  Total Number of Reviews  Average Number of Reviews
0       Apple                   276380                       7086
1   Blackzone                     1569                        523
2         Cmf                     2646                        441
3      Google                    18180                       1212
4       Honor                       63                         63
5           I                       38                         12
6     Infinix                    97189                       1943
7        Iqoo                     1814                         90
8        Itel                     4431                         92
9         Jio                      136                         68
10    Karbonn                      591                         65
11   Kechaoda                    24046                       1202
12       Lava                     7714                        241
13   Micromax                     8466                       2116
14   Motorola                   168786                       2766
15      Nokia                    57754                       2062
16    Nothing                    29980                       2998
17    Oneplus                    22499                        661
18       Oppo                     8700                        124
19       Poco                    91352                       1602
20     Realme                    41753                        379
21      Redmi                    85851                        933
22    Samsung                   138499                       1357
23      Tecno                      590                         34
24       Vivo                   264831                       2910
25        Vox                       12                          6
26     Xiaomi                      444                         55
```



Brands like Apple, Vivo, Motorola, and Samsung receive more reviews than the other brands, showing that these brands are more common than the other ones.

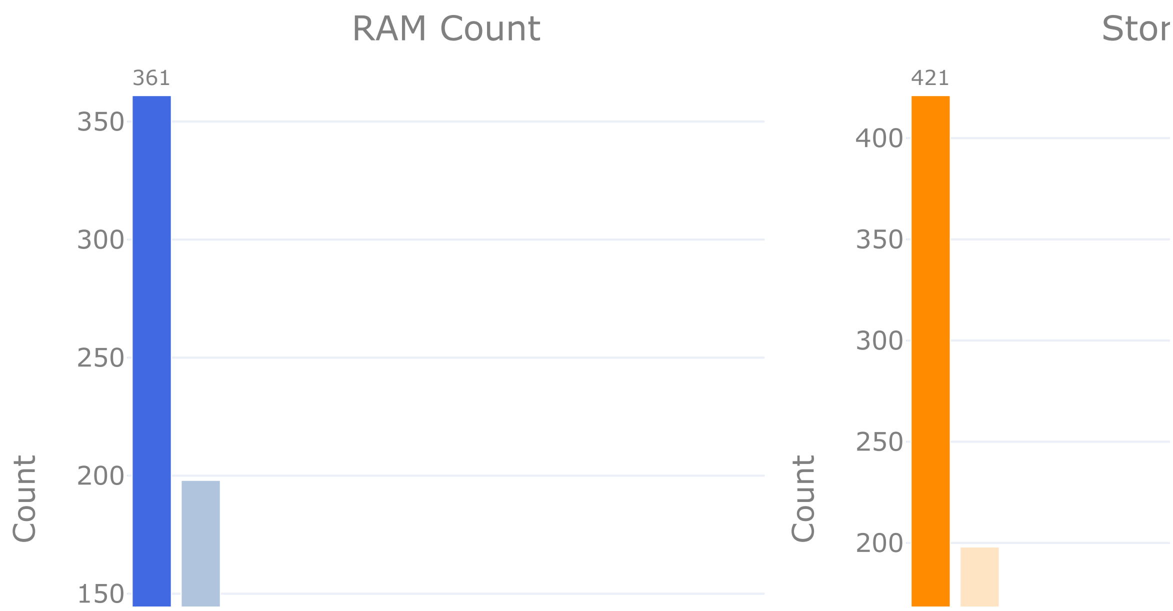
5.3 RAM and Storage Analysis

In [ ]:
```
#Most Common RAM and Storage (GB)
common_ram= df_dup['RAM (GB)'].value_counts().reset_index().astype(int)
common_storage= df_dup['Storage (GB)'].value_counts().reset_index().astyp
print(common_ram)
print(common_storage)

highest_ram = common_ram[common_ram['RAM (GB)']==8]
other_ram = common_ram[common_ram['RAM (GB)']!=8]

highest_storage = common_storage[common_storage['Storage (GB)']==128]
other_storage = common_storage[common_storage['Storage (GB)']!=128]

#Visualise by drawing 2 bar charts side by side to compare
fig = make_subplots(rows=1, cols=2, subplot_titles = ('RAM Count', 'Stora
fig.add_bar(x=highest_ram['RAM (GB)'], y=highest_ram['count'], textpositi
```

```
             marker_color='royalblue', text=highest_ram['count'], name='Mo
fig.add_bar(x=other_ram['RAM (GB)'], y=other_ram['count'], marker_color='
fig.add_bar(x=highest_storage['Storage (GB)'], y=highest_storage['count']
             marker_color='darkorange', text=highest_storage['count'], nam
fig.add_bar(x=other_storage['Storage (GB)'], y=other_ram['count'], marker

fig.update_xaxes(title_text ='RAM (GB)', type='category', row=1, col=1)
fig.update_xaxes(title_text ='Storage (GB)', type='category', row=1, col=
fig.update_yaxes(title_text ='Count', row=1, col=1)
fig.update_yaxes(title_text ='Count', row=1, col=2)
fig.update_layout(template='plotly_white', title='Most Common RAM & Stora
fig.show()
print('The most common RAM is 8GB, while he most common storage is 128GB'
```

```
    RAM (GB)  count
0          8    361
1          4    198
2         12    116
3          6    102
4         32     61
5          3     13
6         16     12
7          2      6
8         64      4
9         24      2
10        48      2
11       500      1
12        20      1
    Storage (GB)  count
0            128    421
1            256    265
2             64     81
3             32     65
4            512     44
5              4     25
6              3      7
7              0      7
8             16      6
9             24      5
10             5      2
11            48      1
12            20      1
```

Most Common RAM & Storage




```
The most common RAM is 8GB, while he most common storage is 128GB
```

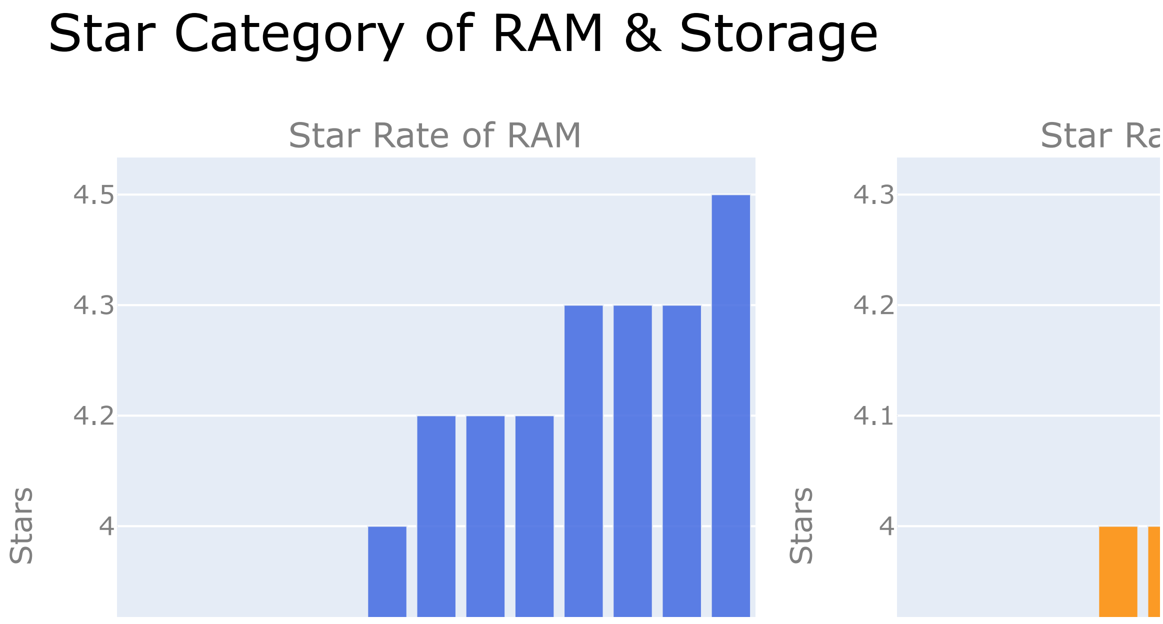
- RAM Count Analysis:
  - 8 GB RAM is the most common among the devices, with a count of 361, indicating it's a popular choice for many consumers.
  - 4 GB RAM and 12 GB RAM are also significant but much less common than 8 GB.
  - Larger RAM sizes like 32 GB and 64 GB are far less common, possibly indicating that they are either higher-end options or less in demand.
- Storage Count Analysis:
  - 128 GB storage is the most common, with a count of 421, suggesting it is the preferred option for many buyers.

- 256 GB and 64 GB storage options are also fairly popular but do not match the prevalence of 128 GB.
- Higher storage capacities like 512 GB are much less common, possibly due to higher price points or being niche products.

In [ ]:
```
#Calculate average star reviews of each RAM and Storage
ram_star = df_dup.groupby('RAM (GB)')['Stars'].mean().round(1).sort_value
storage_star = df_dup.groupby('Storage (GB)')['Stars'].mean().round(1).so
ram_star['Star Category'] = pd.cut(brand_star['Stars'], bins=[0, 3.4, 3.8
storage_star['Star Category'] = pd.cut(brand_star['Stars'], bins=[0, 3.4,
print(ram_star)
print(storage_star)

not_preferred_ram = ram_star[ram_star['Star Category']=='Not Preferred']
not_preferred_storage = storage_star[storage_star['Star Category']=='Not
fair_ram = ram_star[brand_star['Star Category']=='Fair']
fair_storage = storage_star[brand_star['Star Category']=='Fair']


#Visualize average star category of RAM and Storage
#To see which one is most preferred
fig=make_subplots(rows=1,cols=2, subplot_titles=('Star Rate of RAM', 'Sta
fig.add_bar(x=not_preferred_ram['RAM (GB)'], y=not_preferred_ram['Stars']
fig.add_bar(x=fair_ram['RAM (GB)'], y=fair_ram['Stars'], marker_color='ro
fig.add_bar(x=not_preferred_storage['Storage (GB)'], y=not_preferred_stor
fig.add_bar(x=fair_storage['Storage (GB)'], y=fair_storage['Stars'], mark

fig.update_xaxes(title_text='RAM (GB)', type='category', row=1,col=1)
fig.update_yaxes(title_text='Stars', type='category', categoryorder='arra
fig.update_yaxes(title_text='Stars', type='category', categoryorder='arra
fig.update_xaxes(title_text='Storage (GB)', type='category',row=1,col=2)
fig.update_layout(title='Star Category of RAM & Storage',title_font_size=
fig.show()
```

	RAM (GB)	Stars	Star Category
0	20.0	3.6	Not Preferred
1	48.0	3.8	Not Preferred
2	64.0	3.8	Not Preferred
3	2.0	3.9	Not Preferred
4	500.0	3.9	Fair
5	32.0	4.0	Fair
6	3.0	4.2	Fair
7	4.0	4.2	Fair
8	24.0	4.2	Fair
9	6.0	4.3	Fair
10	8.0	4.3	Fair
11	12.0	4.3	Fair
12	16.0	4.5	Fair

	Storage (GB)	Stars	Star Category
0	20.0	3.6	Not Preferred
1	48.0	3.7	Not Preferred
2	0.0	3.9	Not Preferred
3	4.0	3.9	Not Preferred
4	16.0	4.0	Fair
5	32.0	4.0	Fair
6	3.0	4.1	Fair
7	24.0	4.1	Fair
8	5.0	4.2	Fair
9	64.0	4.2	Fair
10	128.0	4.3	Fair
11	256.0	4.3	Fair
12	512.0	4.3	Fair



- Star Rate of RAM:
    - Devices with 8 GB, 12 GB, and 16 GB RAM have higher average ratings (around 4.2 to 4.4 stars), suggesting that these configurations meet customer expectations better
    - 2 GB, 20 GB, 48 GB, and 64 GB RAM options have lower ratings, possibly due to performance limitations or being less balanced for typical usage
- Star Rate of Storage:
    - Storage sizes like 64 GB, 128 GB, 256 GB, and 512 GB tend to have higher ratings (above 4 stars).
    - Smaller storage options like 3 GB, 4 GB, and 5 GB have lower ratings, likely due to limited capacity for modern app and media needs

In [ ]:
```
#Calculate actual and discount price of ram and storage
#To see if RAM and Storage affect price
ram_avgprice = df_dup.groupby('RAM (GB)')['Actual price ₹'].mean().sort_v
```

```
ram_avg_dis_price = df_dup.groupby('RAM (GB)')['Discount price ₹'].mean()
ram_avg_dis_price = ram_avg_dis_price.drop(columns='RAM (GB)')
ram_avg_price = pd.concat([ram_avgprice, ram_avg_dis_price], axis=1)

storage_avgprice = df_dup.groupby('Storage (GB)')['Actual price ₹'].mean(
storage_avgprice = storage_avgprice[storage_avgprice!=0]
storage_avg_dis_price = df_dup.groupby('Storage (GB)')['Discount price ₹'
storage_avg_dis_price = storage_avg_dis_price.drop(columns='Storage (GB)'
storage_avg_dis_price = storage_avg_dis_price[storage_avg_dis_price!=0]
storage_avg_price = pd.concat([storage_avgprice, storage_avg_dis_price],

print(ram_avg_price)
print(storage_avg_price)

#Visualise by drawing 2 bar charts side by side to compare
fig = make_subplots(rows=1, cols=2, subplot_titles = ('Price of RAM', 'Pr
fig.add_bar(x=ram_avg_price['RAM (GB)'], y=ram_avg_price['Actual price ₹'
fig.add_bar(x=ram_avg_price['RAM (GB)'], y=ram_avg_price['Discount price 
fig.add_bar(x=storage_avg_price['Storage (GB)'], y=storage_avg_price['Act
fig.add_bar(x=storage_avg_price['Storage (GB)'], y=storage_avg_price['Dis

fig.update_traces(textposition='outside', texttemplate='%{text:.2s}')
fig.update_xaxes(title_text ='RAM (GB)', type='category', tickfont=dict(s
fig.update_xaxes(title_text ='Storage (GB)', type='category', tickfont=di
fig.update_yaxes(title_text ='Price', row=1, col=1)
fig.update_yaxes(title_text ='Price', row=1, col=2)
fig.update_layout(title='Average Actual & Discount price of RAM & Storage
                  font_color="grey", font_size =10, title_font_color="bla
fig.show()
```

	RAM (GB)	Actual price ₹	Discount price ₹
0	16	72187	63742
1	12	42566	43137
2	8	28979	24187
3	6	17871	13276
4	4	11587	8361
5	3	9845	7512
6	2	8099	5874
7	500	3499	3490
8	32	2881	2087
9	48	2649	1610
10	64	2332	1529
11	20	1999	1450
12	24	1899	1199

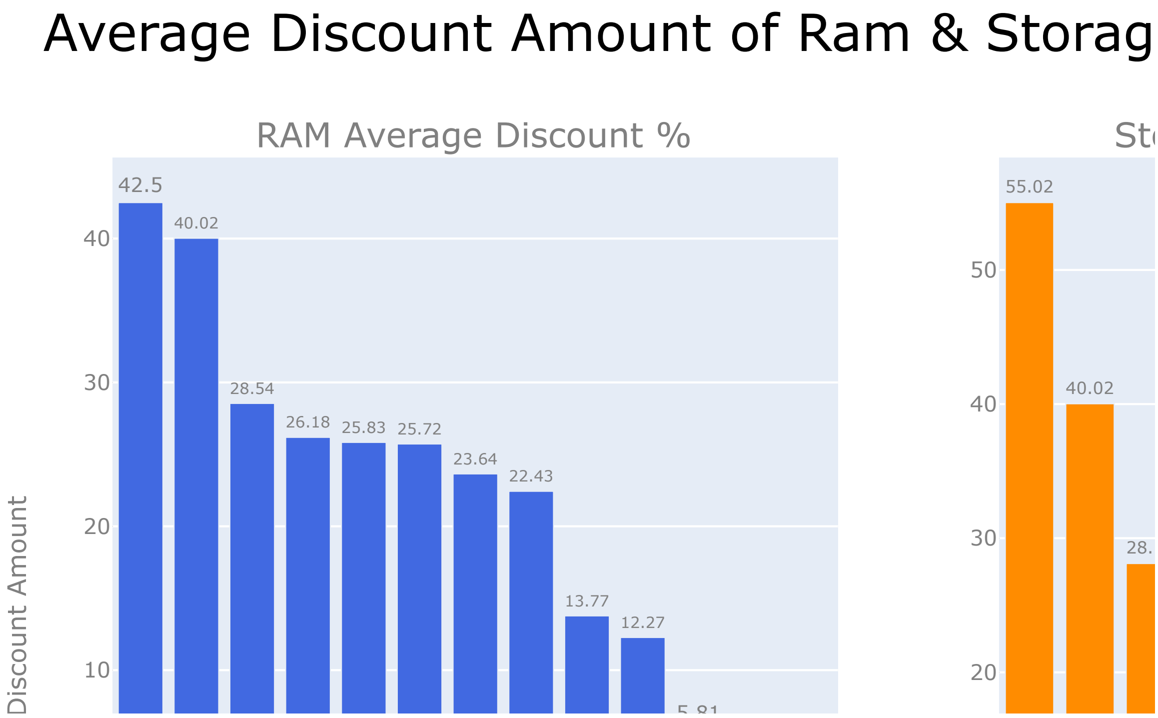
	Storage (GB)	Actual price ₹	Discount price ₹
0	512.0	58361	54385
1	256.0	34826	32494
2	128.0	25573	20001
3	64.0	10861	8125
4	32.0	3660	2848
5	4.0	3650	2216
6	16.0	3365	1559
7	48.0	2599	1429
8	20.0	1999	1199
9	5.0	1699	1169
10	24.0	1539	1156
11	NaN	1528	1147
12	3.0	1277	985

High RAM and Storage configurations (16GB RAM, 512GB storage) are mostly seen in high-end phones, while 2GB or 4GB RAM is seen in low-end phones.

In [ ]:
```
#Calculate average discount % of ram and storage
ram_discount_amount = df_dup.groupby('RAM (GB)')['Discount amount (%)'].m
storage_discount_amount = df_dup.groupby('Storage (GB)')['Discount amount
storage_discount_amount = storage_discount_amount[storage_discount_amount
print(ram_discount_amount)
print(storage_discount_amount)

#Visualize average discount % of ram and storage
fig = make_subplots(rows=1, cols=2, subplot_titles=('RAM Average Discount
fig.add_bar(x=ram_discount_amount['RAM (GB)'], y=ram_discount_amount['Dis
fig.add_bar(x=storage_discount_amount['Storage (GB)'], y=storage_discount

fig.update_traces(textposition='outside', texttemplate='%{text:.2f%}')
```

```
fig.update_xaxes(title_text ='RAM (GB)', type='category', row=1, col=1)
fig.update_xaxes(title_text ='Storage (GB)', type='category', row=1, col=
fig.update_yaxes(title_text ='Discount Amount', row=1, col=1)
fig.update_yaxes(title_text ='Discount Amount', row=1, col=1)
fig.update_layout(font_color="grey", font_size =10, title='Average Discou
fig.show()
```

```
    RAM (GB)  Discount amount (%)
0       48.0                42.50
1       20.0                40.02
2       32.0                28.54
3        4.0                26.18
4        2.0                25.83
5        6.0                25.72
6       24.0                23.64
7        3.0                22.43
8        8.0                13.77
9       64.0                12.27
10      16.0                 5.81
11     500.0                 0.26
12      12.0               -13.62
    Storage (GB)  Discount amount (%)
0           48.0                55.02
1           20.0                40.02
2           32.0                28.11
3            4.0                27.30
4           24.0                25.33
5           64.0                24.09
6            3.0                23.33
8          128.0                21.85
9           16.0                19.12
10           5.0                15.89
11         256.0                 1.81
12         512.0                -5.17
```

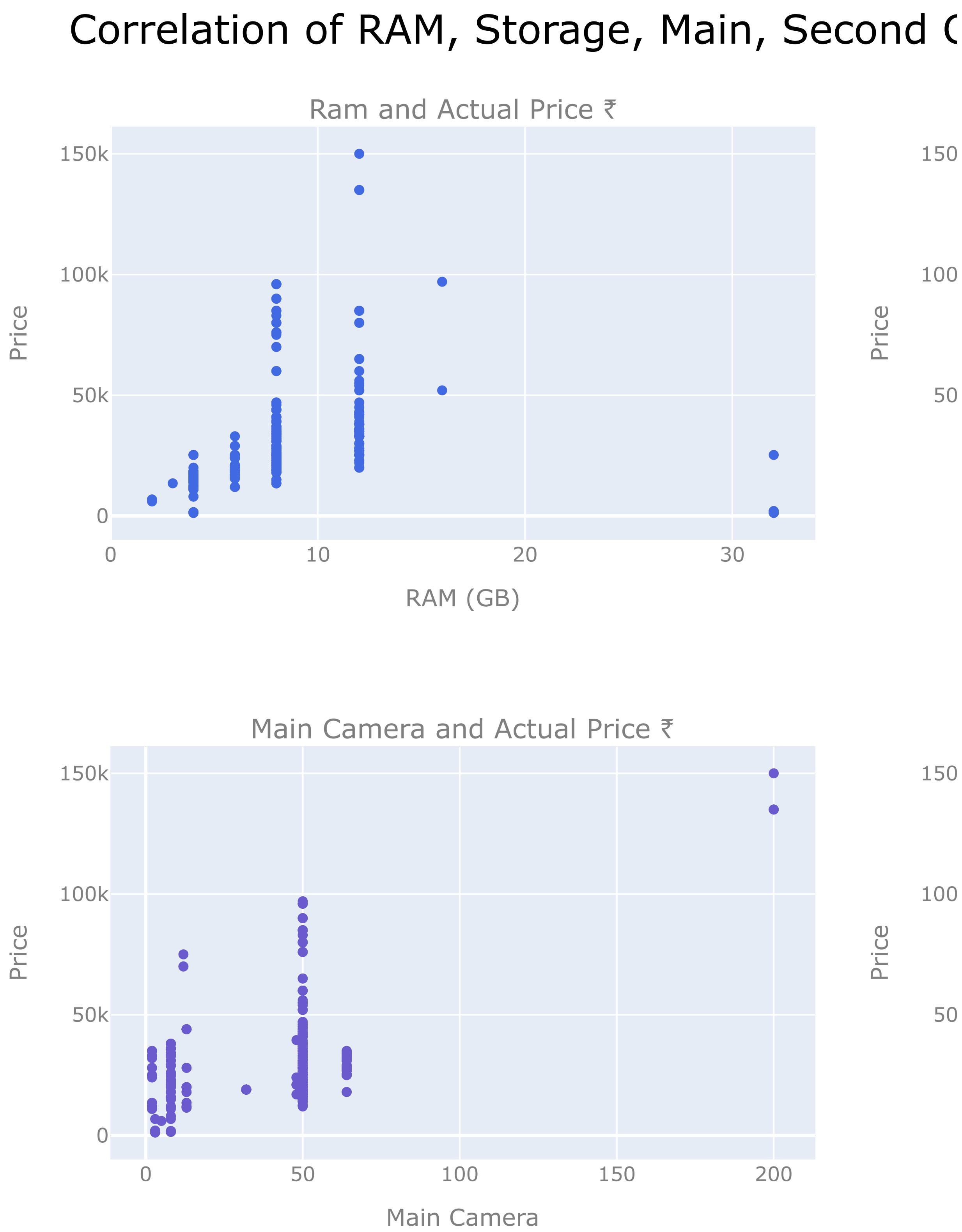


- Regarding RAM, while 48 and 20 GB RAM have high discount amount to boost sales, the opposite is seen in 12 GB RAM
- Regarding Storage, while the discount amount trend for 48 and 20 GB Storage is the same as RAM, that of 512 GB reiceives negative discount

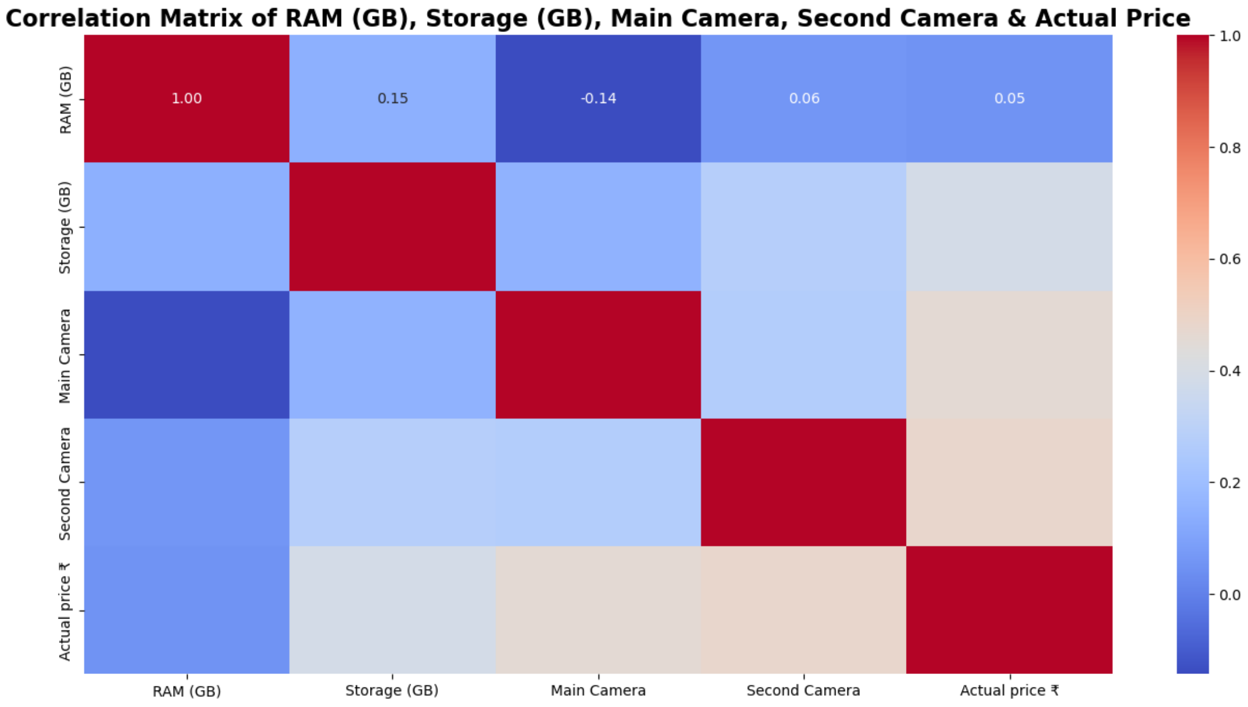
In [ ]:

```
#Visualize scatter plot of Price with RAM, Storage, Main, Second Cam
selected_columns = ['RAM (GB)', 'Storage (GB)', 'Main Camera', 'Second Ca
selected_columns1 = ['RAM (GB)', 'Storage (GB)', 'Main Camera', 'Second C
rows, cols = 2, 2
fig = make_subplots(rows=rows, cols=cols,
                    subplot_titles=('Ram and Actual Price ₹', 'Storage an
                    vertical_spacing=0.2)

for i, column in enumerate(selected_columns):
    row = i // cols + 1
```

```
    col = i % cols + 1
    fig.add_scatter(x=df[column], y=df['Actual price ₹'], mode='markers',
    fig.update_xaxes(title_text=column, row=row, col=col)
    fig.update_yaxes(title_text='Price', row=row)

fig.update_traces(marker_color= 'royalblue', row=1,col=1)
fig.update_traces(marker_color= 'darkorange', row=1,col=2)
fig.update_traces(marker_color= 'slateblue', row=2,col=1)
fig.update_traces(marker_color= 'plum', row=2,col=2)
fig.update_layout(title='Correlation of RAM, Storage, Main, Second Camera
fig.show()
```



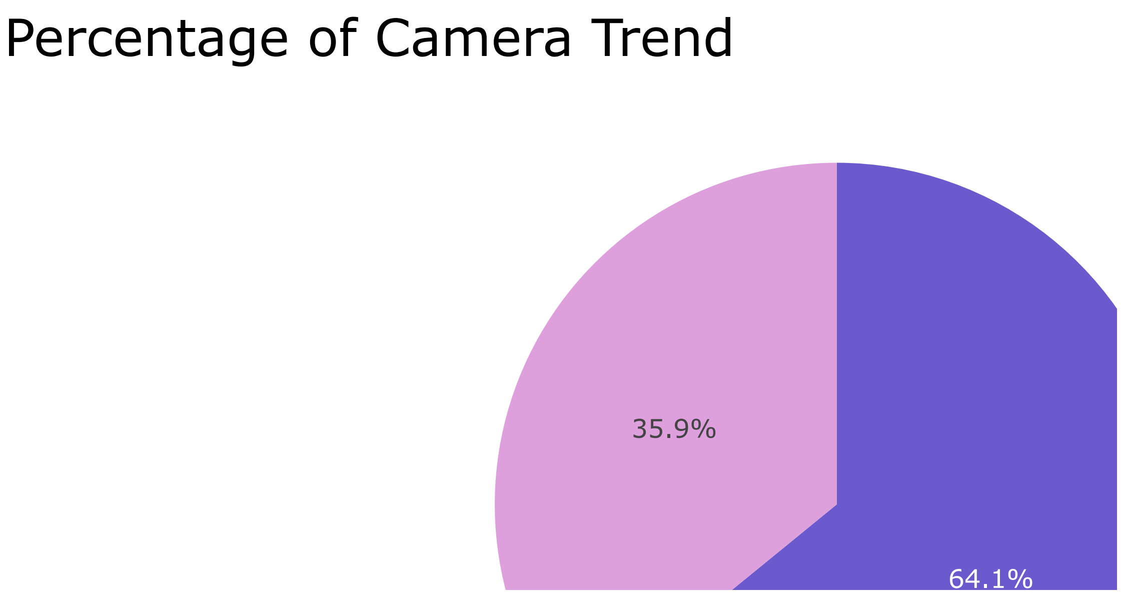
In [ ]:
```
#Visualize correlation matrix by heatmap (RAM (GB), Storage (GB), Main Ca
plt.figure(figsize=(16,8))
sns.heatmap(df[selected_columns1].corr(), annot = True, cmap='coolwarm',
plt.title('Correlation Matrix of RAM (GB), Storage (GB), Main Camera, Sec
plt.show()
```



- RAM vs Price:
  - There is a positive correlation between RAM size and the price of the mobile phone. As the RAM increases, the price tends to increase, especially in the higher RAM segments (10GB and above).
  - Phones with around 8–12GB RAM show varying price points, indicating a wider price range for mid-to-high RAM phones.
- Storage vs Price:
  - Mobile phones with larger storage capacity tend to have higher prices. Phones with 128GB, 256GB, and 512GB storage are clustered around higher price points.
  - Even within the same storage size (e.g., 128GB), there seems to be a large variation in price, possibly due to differences in other features like camera quality, brand, or performance.
- Main Camera vs Price:
  - Phones with higher main camera resolution (around 50 MP and 200 MP) generally fall in the higher price category.
  - There's a large cluster of phones around 12-64 MP for the main camera resolution, indicating this range is common for most phones, but price variations exist.
- Second Camera vs Price:
  - Similar to the main camera, phones with higher second camera resolution (10–50 MP) tend to be more expensive.
  - Phones with dual cameras having higher resolutions on the second camera also seem to push the price upwards.

5.4 Camera Analysis

In [ ]:
```
#Handle null values in the Camera columns from the original (df_dup) one
df_cleaned_cam = df_dup.dropna(subset=['Main Camera'])
df_cleaned_cam['Second Camera'] = df_cleaned_cam['Second Camera'].replace
#Calculate percentage having one or two cameras
have_second_cam = df_cleaned_cam['Second Camera'].dropna()
percentage_two_cam = (len(have_second_cam)/len(df_cleaned_cam['Second Cam
percentage_two_cam = round(percentage_two_cam,2)
percentage_one_cam = 100 - percentage_two_cam
percentage_one_cam = round(percentage_one_cam,2)

value = [64.1,35.9]
name = ['One Camera', 'Two Cameras']
colors=['slateblue', 'plum']

fig = go.Figure(data=go.Pie(values=value, labels=name, marker_colors=colo
fig.update_layout(title_font_size =24, title_text ='Percentage of Camera
                  font_size=12,font_color="grey", height=500, template="p

fig.show()
print(f'Percentage of Phones having two cameras: {percentage_two_cam}%')
print(f'Percentage of Phones having only one cameras: {percentage_one_cam
```



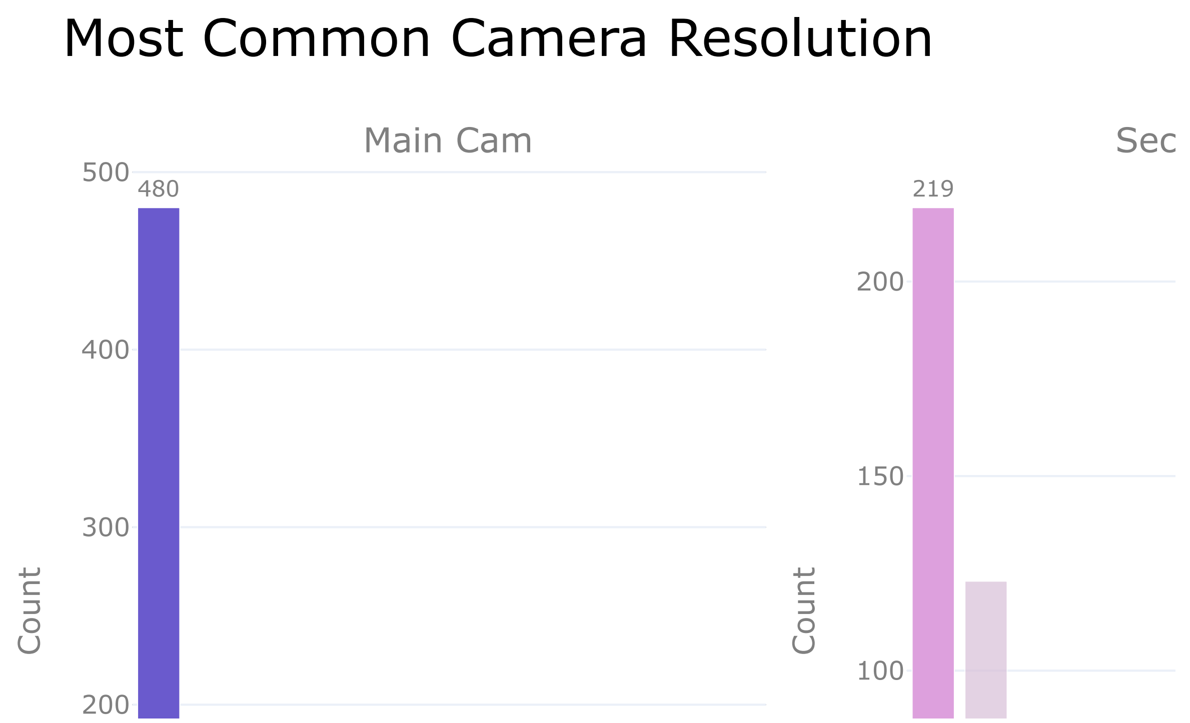

```
Percentage of Phones having two cameras: 64.1%
Percentage of Phones having only one cameras: 35.9%
```

The phones having only main camera is still more popular than those having two cameras. However, following the development of technology, it is likely that phones having two cammera will be common in the upcomming years.

In [ ]:
```
#Most Common Camera Resolution
common_main_camera= df_dup['Main Camera'].value_counts().reset_index()
common_second_camera= df_dup['Second Camera'].value_counts().reset_index(
common_second_camera = common_second_camera[common_second_camera['Second
print(common_main_camera)
print(common_second_camera)

highest_main_cam = common_main_camera[common_main_camera['Main Camera']==
other_main_cam = common_main_camera[common_main_camera['Main Camera']!=50

highest_second_cam = common_second_camera[common_second_camera['Second Ca
other_second_cam = common_second_camera[common_second_camera['Second Came

#Visualise by drawing 2 bar charts side by side to compare
fig = make_subplots(rows=1, cols=2, subplot_titles = ('Main Cam', 'Second
fig.add_bar(x=highest_main_cam['Main Camera'], y=highest_main_cam['count'
```

```
              marker_color='slateblue', text=highest_main_cam['count'], nam
fig.add_bar(x=other_main_cam['Main Camera'], y=other_main_cam['count'], m
fig.add_bar(x=highest_second_cam['Second Camera'], y=highest_second_cam['
              marker_color='plum', text=highest_second_cam['count'], name='
fig.add_bar(x=other_second_cam['Second Camera'], y=other_second_cam['coun

fig.update_xaxes(title_text ='Main Cam Resolution', type='category', row=
fig.update_xaxes(title_text ='Second Cam Resolution', type='category', ro
fig.update_yaxes(title_text ='Count', row=1, col=1)
fig.update_yaxes(title_text ='Count', row=1, col=2)
fig.update_layout(template='plotly_white', title='Most Common Camera Reso
fig.show()
print('The most common main camera resolution is 50MP, while he most comm
```

```
    Main Camera  count
0          50.0    480
1           8.0    107
2           3.0     54
3          64.0     51
4          32.0     35
5          13.0     34
6           2.0     26
7          12.0     26
8          48.0     23
9         200.0      5
10         16.0      2
11          5.0      1
    Second Camera  count
0             2.0    219
1             8.0    123
2            12.0     71
3            50.0     27
4            13.0     27
6             5.0     25
7            16.0     17
8            10.0     12
9            32.0     10
10           20.0      6
11           48.0      3
12           64.0      1
```



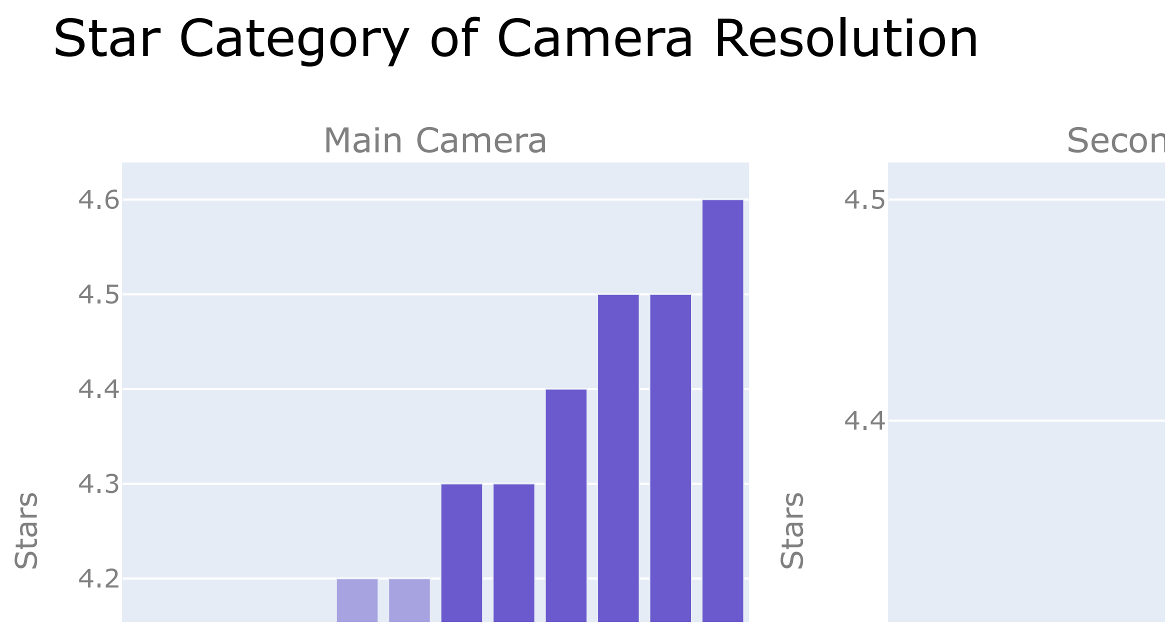

```
The most common main camera resolution is 50MP, while he most common secon
d camera resolution is 2MP
```

- Main Cam Analysis:
    - 50MP main cam is the most common among the devices, with a count of 480, indicating it's a common choice for many consumers
    - This is followed by 8 and 3MP resolution, although the figures are much lower than 50 MP
    - High resolution like 200MP and Low resolution like 48MP are far less common, possibly indicating that they are either higher-end options or less in demand
- Second Cam Analysis:
    - 2MP is the most common, with a count of 219, suggesting it is the preferred

option for many buyers
- This is followed by 8 and 12MP resolution, although the figures are not as significant as that of 2 MP
- Higher second cam resolution like 48 and 64MP are much less common, possibly due to higher price points or being niche products

In [ ]:
```
#Calculate average star category of main and second cameras
main_star = df_dup.groupby('Main Camera')['Stars'].mean().round(1).sort_v
second_star = df_dup.groupby('Second Camera')['Stars'].mean().round(1).so
second_star = second_star[second_star['Second Camera']!=0]
main_star['Star Category'] = pd.cut(main_star['Stars'], bins=[0, 3.4, 3.8
second_star['Star Category'] = pd.cut(second_star['Stars'], bins=[0, 3.4,
print(main_star)
print(second_star)

not_preferred_main = main_star[main_star['Star Category']=='Not Preferred
fair_main = main_star[main_star['Star Category']=='Fair']
good_main = main_star[main_star['Star Category']=='Good']

fair_second = second_star[second_star['Star Category']=='Fair']
good_second = second_star[second_star['Star Category']=='Good']

#Visualize average star category of main and second cameras
#To see which cam resolution is most preferred
fig=make_subplots(rows=1,cols=2, subplot_titles=('Main Camera', 'Second C
fig.add_bar(x=not_preferred_main['Main Camera'], y=not_preferred_main['St
fig.add_bar(x=fair_main['Main Camera'], y=fair_main['Stars'], marker_colo
fig.add_bar(x=good_main['Main Camera'], y=good_main['Stars'], marker_colo
fig.add_bar(x=fair_second['Second Camera'], y=fair_second['Stars'], marke
fig.add_bar(x=good_second['Second Camera'], y=good_second['Stars'], marke

fig.update_xaxes(title_text='Main camera resolution', type='category', ro
fig.update_yaxes(title_text='Stars', type='category', categoryorder='arra
fig.update_yaxes(title_text='Stars', type='category', categoryorder='arra
fig.update_xaxes(title_text='Second camera resolution', type='category',r
fig.update_layout(title='Star Category of Camera Resolution',title_font_s
fig.show()
```

```
    Main Camera  Stars  Star Category
0           5.0    3.6  Not Preferred
1           3.0    4.0           Fair
2           2.0    4.1           Fair
3          13.0    4.1           Fair
4           8.0    4.2           Fair
5          16.0    4.2           Fair
6          50.0    4.3           Good
7          64.0    4.3           Good
8          32.0    4.4           Good
9          48.0    4.5           Good
10        200.0    4.5           Good
11         12.0    4.6           Good
    Second Camera  Stars Star Category
1             5.0    4.2          Fair
2            16.0    4.2          Fair
3            20.0    4.2          Fair
4             2.0    4.3          Good
5             8.0    4.3          Good
6            13.0    4.3          Good
7            32.0    4.4          Good
8            48.0    4.4          Good
9            50.0    4.4          Good
10           10.0    4.5          Good
11           12.0    4.5          Good
12           64.0    4.5          Good
```



- Main Camera Resolution vs Stars Rating:
  - Phones with main camera resolutions of 32MP and above (except for 12MP) have higher user ratings (above 4.3 stars).
  - Lower camera resolutions (below 16 MP) tend to receive lower ratings, suggesting that camera quality heavily influences user satisfaction.
- Second Camera Resolution vs Stars Rating:
  - Similarly, higher second camera resolutions correspond with higher user ratings, with a peak around 4.5 stars.
  - Lower-resolution second cameras (5, 16, and 20MP) receive comparatively just fair ratings.

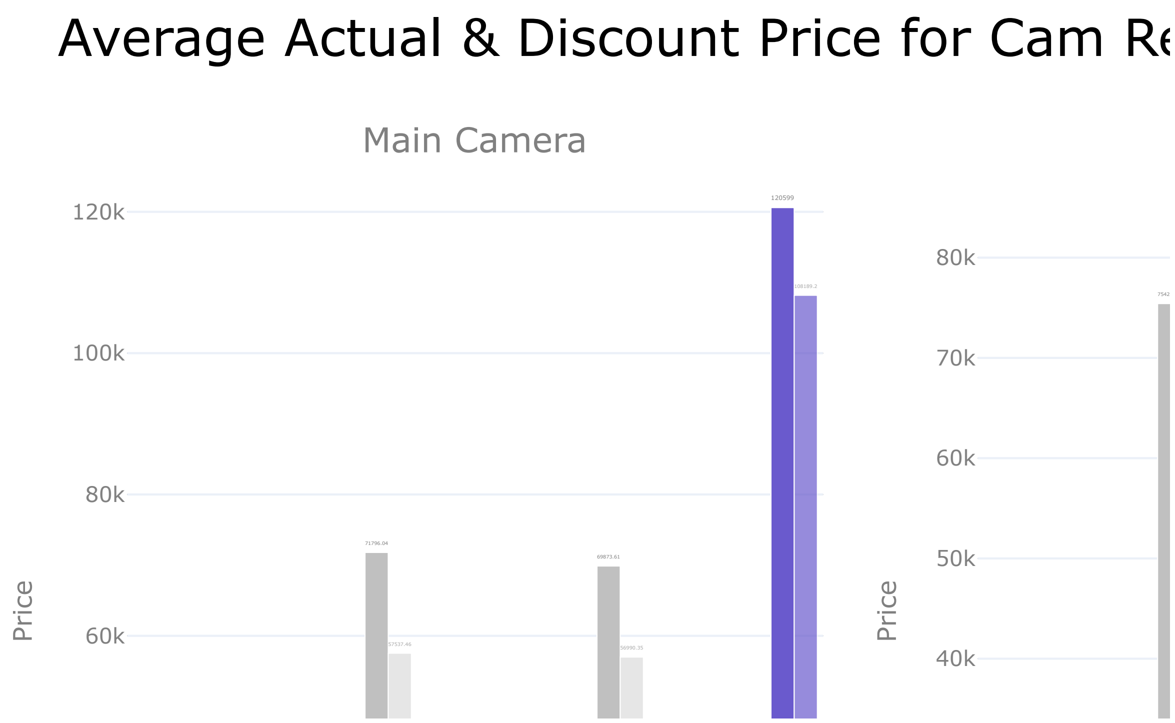
In [ ]:
```
#Calculate average actual price for main and second cam
#To see if the resolution of each cam affect the price or not
main_cam_price = df_dup.groupby('Main Camera')[['Actual price ₹','Discoun
main_cam_price = main_cam_price.rename(columns={'Actual price ₹':'Main ca
second_cam_price = df_dup.groupby('Second Camera')[['Actual price ₹','Dis
second_cam_price = second_cam_price.rename(columns={'Actual price ₹':'Sec
```

```
second_cam_price = second_cam_price[second_cam_price['Second Camera']!=0]
avg_main_second_cam_price = pd.concat([main_cam_price, second_cam_price],
print(avg_main_second_cam_price)

#Visualize average actual price for main and second cam
colors_main_actual = ['silver',] * len(main_cam_price)
colors_main_actual[11] = 'slateblue'

colors_main_discount = ['gainsboro',] * len(main_cam_price)
colors_main_discount[11] = 'slateblue'

colors_second_actual = ['silver',] *  len(avg_main_second_cam_price)
colors_second_actual[10] = 'plum'

colors_second_discount = ['gainsboro',] *  len(avg_main_second_cam_price)
colors_second_discount[4] = 'plum'

fig = make_subplots(rows=1, cols=2, subplot_titles=('Main Camera','Second
fig.add_bar(x=avg_main_second_cam_price['Main Camera'], y=avg_main_second
fig.add_bar(x=avg_main_second_cam_price['Main Camera'], y=avg_main_second
fig.add_bar(x=avg_main_second_cam_price['Second Camera'], y=avg_main_seco
fig.add_bar(x=avg_main_second_cam_price['Second Camera'], y=avg_main_seco

fig.update_traces(textposition='outside', texttemplate='%{text:.2f%}')
fig.update_xaxes(title_text ='Main cam resolution', type='category', row=
fig.update_xaxes(title_text ='Second cam resolution', type='category', ro
fig.update_yaxes(title_text ='Price', row=1, col=1)
fig.update_yaxes(title_text ='Price', row=1, col=2)
fig.update_layout(font_color="grey", template='plotly_white', font_size =
fig.show()

print('For the main camera resolution, the one having the highest mean ac
print('For the second camera resolution, the one having the highest mean
```

	Main Camera	Main cam actual price	Main cam discount price
0	2.0	17646.38	12822.96
1	3.0	2933.31	1651.70
2	5.0	5999.00	4799.00
3	8.0	19820.16	15918.54
4	12.0	71796.04	57537.46
5	13.0	18153.82	14593.85
6	16.0	9999.00	6999.00
7	32.0	17970.43	15124.71
8	48.0	69873.61	56990.35
9	50.0	27679.40	24501.21
10	64.0	36039.31	31076.25
11	200.0	120599.00	108189.20
12	NaN	NaN	NaN

\

	Second Camera	Second cam actual price	Second cam discount price
0	NaN	NaN	NaN
1	2.0	22604.41	17908.11
2	5.0	19949.20	16335.20
3	8.0	29057.28	25025.64
4	10.0	75420.33	69832.33
5	12.0	65242.65	64231.39
6	13.0	32833.37	30554.56
7	16.0	21116.65	16646.06
8	20.0	35999.00	29999.00
9	32.0	33199.00	27799.00
10	48.0	84993.00	47999.00
11	50.0	55256.93	48777.44
12	64.0	59999.00	54999.00



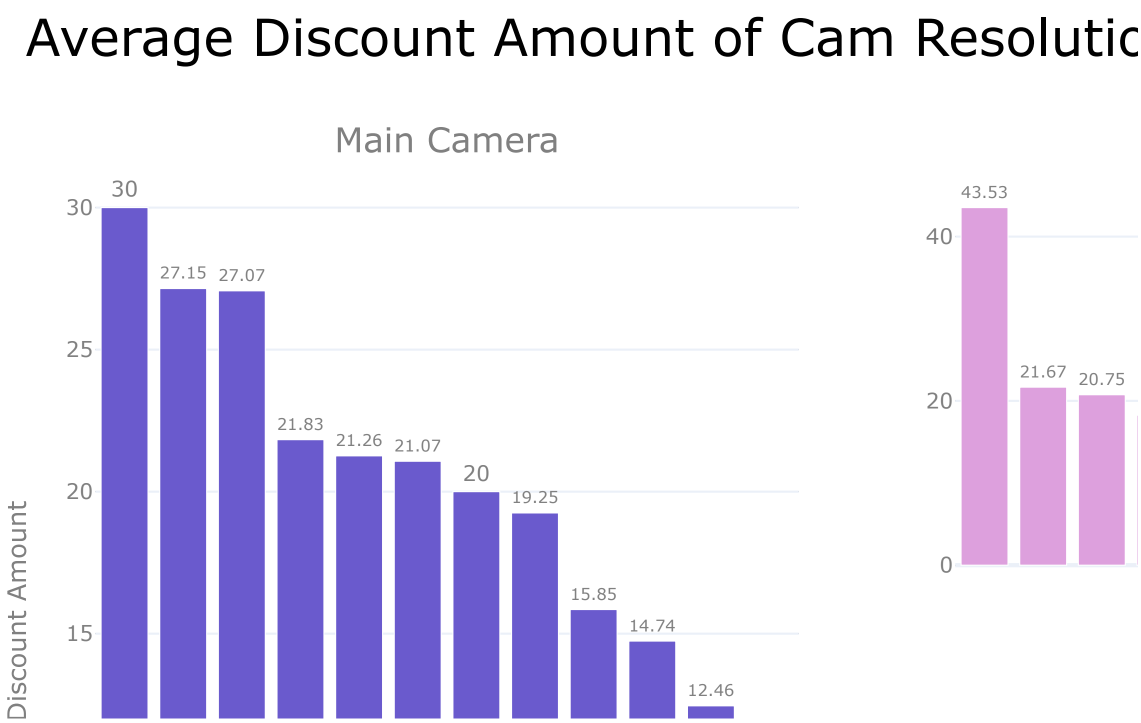

```
For the main camera resolution, the one having the highest mean actual and
discount price is 200MP
For the second camera resolution, the one having the highest mean actual p
rice is 48MP, while that of discount price is 10MP
```

- Main Camera Price Comparison:
  - Phones with 200 MP cameras have the highest average actual and discount prices, indicating they are likely flagship devices with cutting-edge features.
  - Phones with mid-range camera resolutions (50 MP, 64 MP) also have higher prices, but there are budget options available with 12 MP and 32 MP cameras.
- Second Camera Price Comparison:
  - For second cameras, phones with 48 MP have high actual prices, while phones with 10 MP second cameras see the highest discounts.

- Phones with mid-range second cameras (around 12 MP) tend to be more affordable.

In [ ]:
```
#Calculate average discount % of main and second cam resolution
main_cam_discount_amount = df_dup.groupby('Main Camera')['Discount amount
second_cam_discount_amount = df_dup.groupby('Second Camera')['Discount am
second_cam_discount_amount =second_cam_discount_amount[second_cam_discoun
print(main_cam_discount_amount)
print(second_cam_discount_amount)

#Visualize average discount % of main and second cam resolution
fig = make_subplots(rows=1, cols=2, subplot_titles=('Main Camera','Second
fig.add_bar(x=main_cam_discount_amount['Main Camera'], y=main_cam_discoun
fig.add_bar(x=second_cam_discount_amount['Second Camera'], y=second_cam_d

fig.update_traces(textposition='outside', texttemplate='%{text:.2f%}')
fig.update_xaxes(title_text ='Main cam resolution', type='category', row=
fig.update_xaxes(title_text ='Second cam resolution', type='category', ro
fig.update_yaxes(title_text ='Discount Amount', row=1, col=1)
fig.update_yaxes(title_text ='Discount Amount', row=1, col=1)
fig.update_layout(template='plotly_white', font_color="grey", font_size =
fig.show()
```

```
    Main Camera  Discount amount (%)
0          16.0                30.00
1           2.0                27.15
2           3.0                27.07
3           8.0                21.83
4          13.0                21.26
5          48.0                21.07
6           5.0                20.00
7          12.0                19.25
8          32.0                15.85
9         200.0                14.74
10         64.0                12.46
11         50.0                10.24
    Second Camera  Discount amount (%)
0            48.0                43.53
2             2.0                21.67
3            16.0                20.75
4             5.0                18.26
5            20.0                16.70
6            50.0                15.45
7             8.0                14.56
8            32.0                12.69
9            64.0                 8.33
10           13.0                 4.76
11           12.0               -30.88
12           10.0               -54.75
```



- Main Camera generally has a higher discount amount compared to the Second Camera, regardless of resolution. The highest discount amount for the Main Camera is observed at 16MP resolution, with an average discount of 30, while the lowest is at 50MP resolution, with an average discount of 10.24.
- Second Camera shows a decreasing trend in discount amount as the resolution increases. The highest discount amount for the Second Camera is observed at 48MP resolution, with an average discount of 43.53, while the lowest is at 10 resolution, with an average discount of -54.75

6. Feature Engineering

In [ ]:
```
df= df.drop(columns=['Camera', 'Product Name'])
print(df.info())
#Encode categorical data
categorical_col = ['Brand', 'Price Category', 'Star Category']
le = LabelEncoder()
df['Brand'] = le.fit_transform(df['Brand'])
df['Price Category'] = le.fit_transform(df['Price Category'])
df['Star Category'] = le.fit_transform(df['Star Category'])

#Scale data before modelling
x = df.drop('Actual price ₹', axis =1)
y = df['Actual price ₹']
scaler_x= StandardScaler()
x = pd.DataFrame(scaler_x.fit_transform(x), columns= x.columns)
scaler_y= StandardScaler()
y = scaler_y.fit_transform(y.values.reshape(-1, 1))
print(f'Shape of x: {x.shape}')
print(f'Shape of y: {y.shape}')
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 527 entries, 6 to 977
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Brand                527 non-null    category
 1   Price Category       527 non-null    category
 2   Actual price ₹       527 non-null    int64
 3   Discount price ₹     527 non-null    int64
 4   Discount amount (%)  527 non-null    float64
 5   Stars                527 non-null    float64
 6   Star Category        527 non-null    category
 7   Number of Rating     527 non-null    int64
 8   Number of Reviews    527 non-null    int64
 9   RAM (GB)             527 non-null    int64
 10  Storage (GB)         527 non-null    int64
 11  Display Size (inch)  527 non-null    int64
 12  Main Camera          527 non-null    int64
 13  Second Camera        527 non-null    int64
dtypes: category(3), float64(2), int64(9)
memory usage: 52.0 KB
None
Shape of x: (527, 13)
Shape of y: (527, 1)
```

7. Model Selection and Evaluation

In [ ]:
```
#Split data into the train and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,

#Linear Regression
lr_model = LinearRegression()
lr_model.fit(x_train, y_train)
```

```
y_pred_lr = lr_model.predict(x_test)
#Evaluate linear regression model using mse and r2
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)
print(f'MSE Linear Regression:{mse_lr}, and R² Score:{r2_lr}')


# Decision Tree Regressor
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(x_train, y_train)
y_pred_dt = dt_model.predict(x_test)
#Evaluate tree regressor using mse and r2
mse_dt = mean_squared_error(y_test, y_pred_dt)
r2_dt = r2_score(y_test, y_pred_dt)
print(f'MSE Decision Tree Regressor: {mse_dt}, and R² Score:{r2_dt}')

# Random Forest Regressor
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(x_train, y_train)
y_pred_rf = rf_model.predict(x_test)
#Evaluate forest regressor using mse and r2
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
print(f'MSE Random Forest Regressor: {mse_rf}, and R² Score:{r2_rf}')
```

```
MSE Linear Regression:0.036260548750932355, and R² Score:0.960862698574505
3
MSE Decision Tree Regressor: 0.008711829657860276, and R² Score:0.99059701
20411243
MSE Random Forest Regressor: 0.008055835100765942, and R² Score:0.99130505
03251236
```

Comment:

- Lower MSE values indicate better model performance, so both the Decision Tree and Random Forest regressors are performing significantly better than the Linear Regression model
- An R^2 score close to 1 indicates that the model explains a high proportion of the variance in the dependent variable. The Random Forest Regressor has an R^2 score of approximately 0.9913, which is excellent, suggesting that it explains over 99% of the variance

Overall, both the Decision Tree and Random Forest regressors demonstrate strong performance with low MSE values and high R^2 scores. The Random Forest Regressor, in particular, shows the best performance among the three. In short, the Random Forest model would be the best choice based on these metrics.

8. Business Insights and Recommendations

8.1 Comprehensive Business Insights from Mobile Sales Data

1. Brand-Specific Strategies and Insights

- Premium Brands: Apple, Google, and Samsung:
    - These brands maintain high price points with limited discounts, relying on a strategy focused on premium features, brand loyalty, and superior product quality.
    - Their high star ratings (above 4.3) reflect strong customer satisfaction and a focus on the customer experience.
    - This approach helps preserve their premium brand image and ensures profitability without engaging in aggressive price competition.
- Recommendation:
    - Continue focusing on innovation and introducing exclusive features that set them apart.
    - Utilize limited-time discounts during major shopping events like Black Friday or festive sales to create urgency and boost sales without compromising their premium image.
    - Offering extended warranties or premium service packages could further enhance customer loyalty and justify their pricing strategy, especially for high-end models.
- Mid-Range Brands: Realme, Oppo, and Vivo:
    - These brands focus on the mid-range market, balancing price and performance with moderate discount strategies.
    - They maintain good star ratings, which suggests customer satisfaction is generally positive, though not as high as the premium segment.
    - Their market position makes them vulnerable to competition from both premium brands, which offer better features at higher prices, and budget brands, which appeal to price-sensitive customers.
- Recommendation:
    - Enhance brand differentiation by highlighting unique features or user-friendly innovations that resonate with the target audience.
    - Continue offering moderate discounts but avoid over-reliance on price cuts, as it could erode perceived value.
    - Strengthen after-sales service and customer engagement programs to build loyalty and differentiate from both premium and budget competitors.
- Budget Brands: Karbonn, Kechaoda, and Vox:
    - These brands compete primarily on low price points, often accompanied by

hefty discounts to attract price-sensitive customers.
  - However, they tend to have lower star ratings, indicating potential quality issues or gaps in customer satisfaction.
  - This can limit their sales potential and result in weaker brand loyalty, as customers may prioritize savings but become dissatisfied with product quality over time.
- Recommendation:
  - Focus on improving product quality and addressing common customer complaints, as even small improvements could positively impact star ratings.
  - Emphasize value-for-money features in marketing campaigns, such as battery life or display size, which are appealing at lower price points.
  - Consider bundling devices with accessories or basic service packages to create a sense of added value, even at low price points.

2. Market Segmentation by RAM and Storage Preferences

- Most Common RAM & Storage:
  - 8 GB RAM is the most popular configuration, with a count of 361, indicating a strong preference for balanced performance.
  - For storage, 128 GB is the leading choice, with 421 units, suggesting it hits the right balance between capacity and affordability for most consumers.
  - Lesser-used options include higher RAM configurations (e.g., 32 GB and 64 GB) and higher storage capacities like 512 GB, which are typically reserved for more premium models.
- Customer Satisfaction Trends:
  - Devices with 8 to 16 GB RAM tend to receive higher star ratings (between 4.2 and 4.4 stars), suggesting that these specifications are more in line with customer needs for multitasking and performance.
  - For storage, ratings above 4 stars are associated with devices offering 64 GB and higher, indicating that sufficient storage is a key factor in customer satisfaction.

3. Market Segmentation by Main and Second Cam Preferences

- Main Camera:
  - 50MP main cameras are standard in mid-range devices, offering excellent image quality for the price. Higher-end models should focus on more advanced camera features.
  - Recommendation: Mid-range brands should continue using 50MP cameras but emphasize software improvements (AI, night mode). Premium brands

  - should market multi-camera setups and advanced features like optical zoom and image stabilization.
- Second Camera:
  - 2MP second cameras are common but not particularly valued, while premium devices offer better secondary camera options for versatile photography.
  - Recommendation: For mid-range brands, upgrading to 8MP+ secondary cameras could enhance the photography experience, while premium brands should promote the multi-camera versatility for content creators.

4. Overall Market Dynamics and Consumer Preferences

- Demand Concentration: The highest demand is in the mid-range market with 8 GB RAM and 128 GB storage devices, balancing affordability with adequate performance for most users.
- Star Ratings as a Critical Factor: Higher star ratings correlate with configurations that balance performance and capacity. This indicates that customer satisfaction is closely tied to both hardware specifications and overall user experience.
- Segmentation Based on Price Sensitivity: Premium brands rely on brand prestige and innovative features, while mid-range brands focus on affordability with a fair balance of quality. Budget brands compete primarily on price but struggle with perceived quality.

8.2 Strategic Recommendations

1. For Premium Brands:

- Focus on Exclusive Offerings: Continue emphasizing features like advanced cameras, proprietary software enhancements, and high-end design.
- Limited-Time Promotions: Implement strategic discounts during peak shopping seasons to spur demand without diluting the premium brand image.
- Enhance Customer Support: Providing options like AppleCare, Samsung Premium Care, or Google Preferred Care can help maintain loyalty and justify higher prices.

2. For Mid-Range Brands:

- Differentiate on Value: Highlight unique features like high-refresh-rate displays or fast-charging capabilities that set them apart from both budget and premium competitors.
- Loyalty Programs: Implement customer loyalty programs that encourage repeat purchases and positive reviews, boosting word-of-mouth.
- Balance Discounts with Quality: Continue offering discounts but ensure product

quality remains high to prevent a decline in ratings.

3. For Budget Brands:

- Quality Focused Improvements: Invest in quality control to address issues that lead to lower ratings, such as battery life or build quality.
- Emphasize Affordability in Marketing: Highlight affordability and essential features that meet basic needs to appeal to cost-conscious buyers.
- Offer Bundled Value: Providing affordable accessories or simple warranties can create a perception of added value, making budget devices more attractive.

Conclusion: The mobile market is characterized by distinct segments, each with unique strategies and consumer needs. Premium brands should focus on maintaining their high-quality image while using strategic promotions. Mid-range brands need to balance competitive pricing with quality offerings to maintain their position, while budget brands should aim for improved quality and value bundling to overcome challenges with customer satisfaction.

By tailoring strategies to these insights, brands can better align their offerings with customer expectations, improving sales performance and maintaining market relevance.