

Laptop Prices Project

Welcome to the data analysis assignment on laptop prices! In this assignment, we will work with a dataset containing information about various laptops. The dataset includes several features detailing laptop specifications, such as brand, type, screen size, hardware details, and price. Through this analysis, you will gain hands-on experience in essential data analysis steps, including data cleaning, visualization, and exploratory data analysis (EDA).

Import Libraries, Loading the Dataset and Initial Exploration

- Load the dataset, display first few rows, check the structure of the dataset.
- Inspect the data types and missing values using `df.info()`
- Get basic statistics for numerical columns with `df.describe()`

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
warnings.warn("this will not show")
```

```
In [2]: data = pd.read_csv("laptop_data.csv")
data.head()
```

```
Out[2]:   Unnamed: 0  Company  TypeName  Inches  ScreenResolution  Cpu  Ram  N
```

0	0	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB
1	1	Apple	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8GB
2	2	HP	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB
3	3	Apple	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB

4	4	Apple	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB
---	---	-------	-----------	------	---------------------------------------	-------------------------	-----

```
In [3]: df = data.copy()
```

Data Cleaning:

Remove unnecessary columns:

- Drop the Unnamed: 0 column as it seems to be an index column.

```
In [4]: df.columns
```

```
Out[4]: Index(['Unnamed: 0', 'Company', 'TypeName', 'Inches', 'ScreenResolution',  
              'Cpu', 'Ram', 'Memory', 'Gpu', 'OpSys', 'Weight', 'Price'],  
             dtype='object')
```

```
In [5]: df.drop('Unnamed: 0', inplace = True, axis = 1)
```

Check for duplicates and missing values:

- Identify and remove any duplicate rows from the dataset.
- Check for missing values in the dataset.

```
In [6]: df.duplicated().sum()
```

```
Out[6]: np.int64(29)
```

```
In [7]: df = df.drop_duplicates()
```

```
In [8]: df.duplicated().sum()
```

```
Out[8]: np.int64(0)
```

Standardize data formats:

- Convert Weight to a numeric column (strip the "kg" and convert to float) etc. Apply similar operations to other required columns. You may consider adding new features.

- Verify the data types of each column and convert them to appropriate formats if necessary.

In [9]: `df.Weight.info()`

```
<class 'pandas.core.series.Series'>
Index: 1274 entries, 0 to 1273
Series name: Weight
Non-Null Count  Dtype
-----
1274 non-null   object
dtypes: object(1)
memory usage: 19.9+ KB
```

In [10]: `df.Weight.value_counts()`

```
Out[10]: Weight
2.2kg      116
2.1kg       58
2.4kg       42
2.5kg       38
2.3kg       37
...
3.52kg      1
2.21kg       1
2.191kg      1
2.34kg       1
4.0kg        1
Name: count, Length: 179, dtype: int64
```

In [11]: `df['Weight'] = df.Weight.str.replace('kg', '').astype('float')`

In [12]: `print(df.Weight.info())`
`print(df.Weight.value_counts())`

```
<class 'pandas.core.series.Series'>
Index: 1274 entries, 0 to 1273
Series name: Weight
Non-Null Count  Dtype
-----
1274 non-null   float64
dtypes: float64(1)
memory usage: 19.9 KB
None
Weight
2.200      119
2.100       58
2.000       45
2.400       42
2.500       38
...
0.990        1
2.591         1
2.210         1
2.191         1
2.340         1
Name: count, Length: 171, dtype: int64
```

```
In [13]: df['Ram'] = df.Ram.str.replace('GB', '').astype('int')
```

```
In [14]: print(df.Ram.info())
print(df.Ram.value_counts())
```

```
<class 'pandas.core.series.Series'>
Index: 1274 entries, 0 to 1273
Series name: Ram
Non-Null Count  Dtype
-----
1274 non-null   int64
dtypes: int64(1)
memory usage: 19.9 KB
None
Ram
8      613
4      366
16     198
6       35
12       25
32       17
2       16
24        3
64        1
Name: count, dtype: int64
```

```
In [15]: from skimpy import skim
skim(df)
```

skimpy summary

Data Summary		Data Types	
dataframe	Values	Column Type	Count
Number of rows	1274	string	7
Number of columns	11	float64	3
		int64	1

number

column_name	NA	NA %	mean	sd	p0
Inches	0	0	15.02	1.43	10.1
Ram	0	0	8.444	5.098	2
Weight	0	0	2.04	0.6694	0.69
Price	0	0	60500	37330	9271

string

column_name	NA	NA %	word
Company	0	0	
TypeName	0	0	
ScreenResolution	0	0	
Cpu	0	0	
Memory	0	0	
Gpu	0	0	
OpSys	0	0	

Clean categorical columns:

- Ensure consistent formatting in Company, TypeName, and OpSys columns (e.g., no leading/trailing spaces, proper case).
- Simplify OpSys categories if there are too many unique values (e.g., group similar OS types).

In [16]: `df.Company.value_counts()`

```
Out[16]: Company
Dell      291
Lenovo    289
HP        268
Asus      151
Acer      101
MSI       54
Toshiba   48
Apple     21
Samsung   9
Mediacom  7
Razer     7
Microsoft 6
Vero      4
Xiaomi    4
Chuwi     3
Fujitsu   3
Google    3
LG        3
Huawei     2
Name: count, dtype: int64
```

In [17]: `df.OpSys.value_counts()`

```
Out[17]: OpSys
Windows 10      1047
No OS           66
Linux           58
Windows 7       45
Chrome OS       27
macOS           13
Mac OS X        8
Windows 10 S    8
Android         2
Name: count, dtype: int64
```

Split compound columns:

- Parse ScreenResolution to extract key features such as:
- Resolution (e.g., Full HD, 4K)
- Touchscreen (Yes/No)
- Extract details from Cpu and Gpu columns, such as manufacturer

or core type.

In [18]: `df.Cpu.value_counts()`

```
Out[18]: Cpu
Intel Core i5 7200U 2.5GHz      190
Intel Core i7 7700HQ 2.8GHz     146
Intel Core i7 7500U 2.7GHz      132
Intel Core i7 8550U 1.8GHz       73
Intel Core i5 8250U 1.6GHz       72
...
Intel Core M 6Y54 1.1GHz        1
Samsung Cortex A72&A53 2.0GHz    1
AMD E-Series 9000 2.2GHz         1
Intel Core M 6Y30 0.9GHz         1
AMD A9-Series 9410 2.9GHz        1
Name: count, Length: 118, dtype: int64
```

```
In [19]: # Ekran çözünürlüğü ve dokunmatik ekran bilgilerini ayrıştırma
def parse_screen_resolution(screen_resolution):
    # Dokunmatik ekran var mı kontrolü
    touchscreen = 'Touchscreen' in screen_resolution or 'Touch'
    # Çözünürlük değerini alma
    resolution = screen_resolution.split()[-1] # "2560x1600" gibi
    return resolution, touchscreen

# Yeni sütunlar oluşturma
df[['Resolution', 'Touchscreen']] = df['ScreenResolution'].apply(
    lambda x: pd.Series(parse_screen_resolution(x))
)

# Dokunmatik ekranı True/False yerine Yes/No formatında ifade edelim
df['Touchscreen'] = df['Touchscreen'].apply(lambda x: 'Yes' if x else 'No')
```

```
In [20]: no_hd_data = df[~df['ScreenResolution'].str.contains('HD', case=False)]
no_hd_data.ScreenResolution.value_counts()
```

```
Out[20]: ScreenResolution
1366x768      262
1600x900       23
Touchscreen 1366x768    16
IPS Panel 1366x768       7
Touchscreen 2560x1440     7
IPS Panel Retina Display 2560x1600    6
IPS Panel Retina Display 2304x1440    6
Touchscreen 2256x1504     6
IPS Panel Touchscreen 2560x1440     5
IPS Panel Retina Display 2880x1800    4
1440x900       4
IPS Panel Touchscreen 1920x1200     4
IPS Panel 2560x1440       4
2560x1440       3
Touchscreen 2400x1600     3
1920x1080       3
IPS Panel Touchscreen 1366x768     3
IPS Panel Retina Display 2736x1824    1
IPS Panel Touchscreen 2400x1600     1
Name: count, dtype: int64
```

```
In [21]: def categorize_resolution(res):
        if '4K Ultra HD' in res:
            return '4K Ultra HD'
        elif 'Quad HD+' in res:
            return 'Quad HD+'
        elif 'Full HD' in res:
            return 'Full HD'
        elif 'HD' in res: # Sadece 'HD' varsa
            return 'HD'
        else:
            return None # Hiçbiri yoksa None (NaN)

        # Yeni sütun oluşturma
        df['HD_Category'] = df['ScreenResolution'].apply(categorize_reso
```

```
In [22]: df.HD_Category.isnull().sum()
```

```
Out[22]: np.int64(368)
```

```
In [23]: def fill_missing_hd(row, resolution):
        if row is not None: # Eğer zaten doluysa
            return row
        if '1366x768' in resolution or '1440x900' in resolution:
            return 'HD'
        elif '1920x1080' in resolution or '1920x1200' in resolution:
            return 'Full HD'
        elif '1600x900' in resolution:
            return 'HD Plus'
        elif '2560x1440' in resolution:
            return '2K Ultra HD'
        elif '3840x2160' in resolution or '2880x1800' in resolution:
            return '4K Ultra HD'
        elif '2256x1504' in resolution or '2304x1440' in resolution:
            return 'Quad HD'

        else:
            return None

        # Eksik değerleri doldurma
        df['HD_Category'] = df.apply(lambda x: fill_missing_hd(x['HD_Cat
```

```
In [24]: df['GHz'] = df['Cpu'].str.extract(r'(\d+\.\d*)\s?GHz')
```

```
In [25]: df.GHz.info()
```

```
<class 'pandas.core.series.Series'>
Index: 1274 entries, 0 to 1273
Series name: GHz
Non-Null Count  Dtype
-----
1274 non-null   object
dtypes: object(1)
memory usage: 19.9+ KB
```

```
In [26]: df.GHz.isnull().sum()
```

Out[26]: np.int64(0)

In [27]: `df['Cpu'] = df['Cpu'].str.replace(r'\d+(\.\d+)?GHz', '', regex=True)`

In [28]:

```
# Cpu'dan üretici ve çekirdek türü ayırma
def parse_cpu(cpu):
    if 'Intel' in cpu:
        manufacturer = 'Intel'
        core = cpu.split()[2] # Örneğin: Core i5
    elif 'AMD' in cpu:
        manufacturer = 'AMD'
        core = cpu.split()[1] # Örneğin: Ryzen 5
    else:
        manufacturer = 'Samsung'
        core = None
    return manufacturer, core

# Yeni sütunlar oluşturma
df[['Cpu_Manufacturer', 'Cpu_Core_Type']] = df['Cpu'].apply(
    lambda x: pd.Series(parse_cpu(x))
)
```

In [29]:

```
# Gpu'dan üretici türü ayırma
def parse_gpu(gpu):
    if 'Intel' in gpu:
        manufacturer = 'Intel'
    elif 'AMD' in gpu:
        manufacturer = 'AMD'
    elif 'ARM' in gpu:
        manufacturer = 'ARM'
    else:
        manufacturer = 'Nvidia'
    return manufacturer

# Yeni sütunlar oluşturma
df['Gpu_Manufacturer'] = df['Gpu'].apply(
    lambda x: pd.Series(parse_gpu(x))
)
```

Check for outliers in numerical columns:

- Identify outliers in columns like Inches, Weight, Ram, and Price using methods such as the IQR (Interquartile Range) or z-scores.
- Consider removing or capping the outliers if necessary to improve data quality.

In [30]:

```
# Numerik sütunlar
numerical_columns = ['Inches', 'Weight', 'Ram', 'Price']
# IQR yöntemiyle aykırı değerleri tespit etme fonksiyonu
def detect_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25) # 1. çeyrek
    Q3 = df[column].quantile(0.75) # 3. çeyrek
    IQR = Q3 - Q1 # Çeyrekler aralığı
```



```

IQR = Q3 - Q1 # Çeyrekler arası mesafe
# Alt ve üst eşik değerleri
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Aykırı değerlerin maskesi
outliers = (df[column] < lower_bound) | (df[column] > upper_bound)
return outliers
# Aykırı değerleri kontrol et ve sonuçları yazdır
for col in numerical_columns:
    outliers = detect_outliers_iqr(df, col)
    print(f"{col} sütununda {outliers.sum()} aykırı değer bulundu.")

```

Inches sütununda 37 aykırı değer bulundu.
 Weight sütununda 45 aykırı değer bulundu.
 Ram sütununda 219 aykırı değer bulundu.
 Price sütununda 28 aykırı değer bulundu.

In [31]:

```

from scipy.stats import zscore

# Z-skoru yöntemiyle aykırı değerleri tespit etme fonksiyonu
def detect_outliers_zscore(df, column, threshold=3):
    z_scores = zscore(df[column])
    outliers = (z_scores > threshold) | (z_scores < -threshold)
    return outliers
# Aykırı değerleri kontrol et ve sonuçları yazdır
for col in numerical_columns:
    outliers = detect_outliers_zscore(df, col)
    print(f"{col} sütununda {outliers.sum()} aykırı değer bulundu.")

```

Inches sütununda 4 aykırı değer bulundu.
 Weight sütununda 33 aykırı değer bulundu.
 Ram sütununda 21 aykırı değer bulundu.
 Price sütununda 12 aykırı değer bulundu.

In [32]:

```

import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import zscore

def detect_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = (df[column] < lower_bound) | (df[column] > upper_bound)
    return outliers

def detect_outliers_zscore(df, column, threshold=3):
    z_scores = zscore(df[column])
    outliers = (z_scores > threshold) | (z_scores < -threshold)
    return outliers

# Aykırı değerleri belirleme (örnek: Inches)
numerical_columns = ["Inches", "Weight", "Ram", "Price"]

# Create a figure with a specific size
fig, axes = plt.subplots(len(numerical_columns), 2, figsize=(15, 10))
fig.patch.set_facecolor('#f6f5f5')

# Loop through columns to create boxplots and histograms
for i, col in enumerate(numerical_columns):
    # Boxplot for outliers

```

```

sns.boxplot(data=df, x=col, ax=axes[i, 0], color='red')
axes[i, 0].set_title(f"Boxplot of {col}", fontsize=14, font
axes[i, 0].set_xlabel('')
axes[i, 0].set_ylabel('')

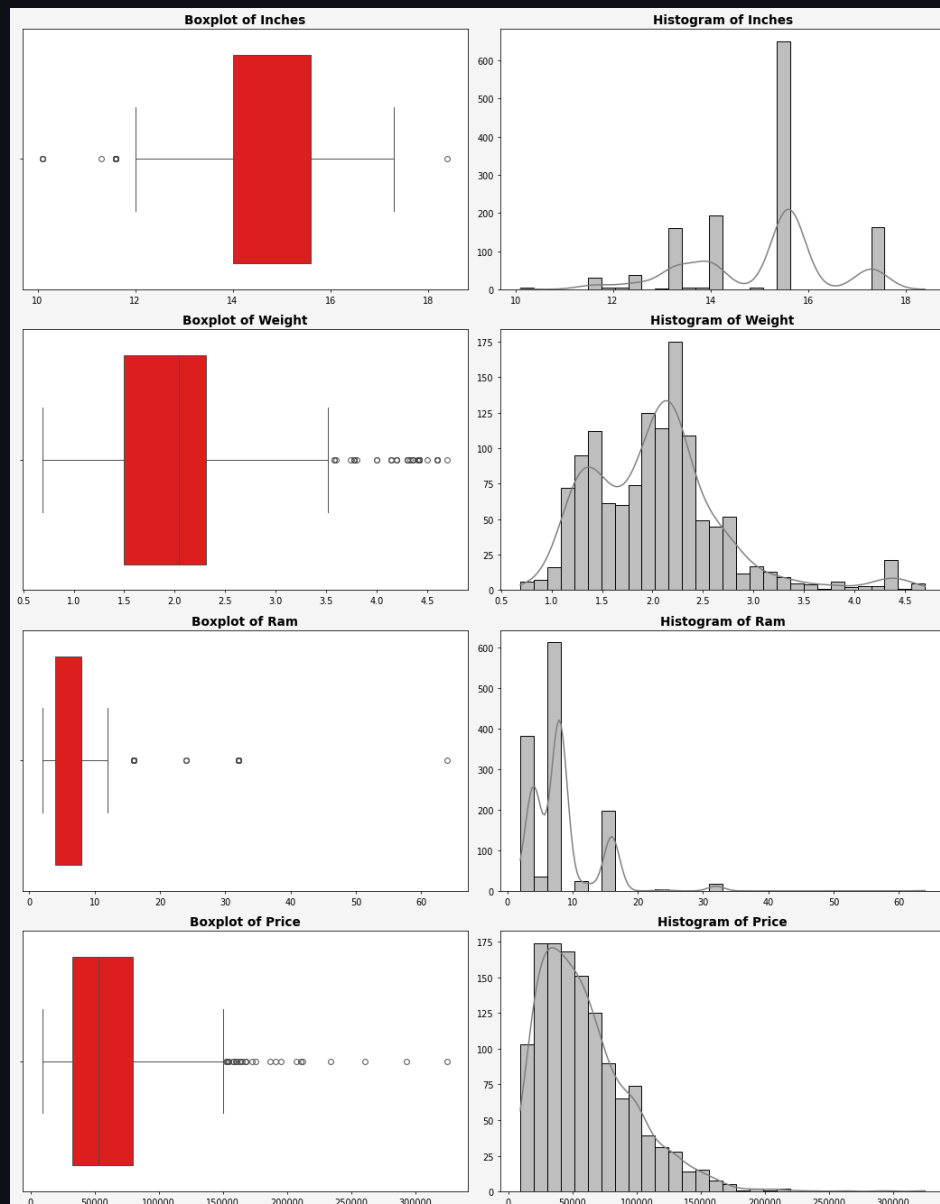
# Histogram for distribution with gray bars
sns.histplot(df[col], bins=30, kde=True, ax=axes[i, 1], col
axes[i, 1].set_title(f"Histogram of {col}", fontsize=14, fo
axes[i, 1].set_xlabel('')
axes[i, 1].set_ylabel('')

# Adjust layout to avoid overlap and ensure readability
plt.tight_layout(rect=[0, 0, 1, 0.96])

# Show the plot
plt.show()

# Aykırı değerlerin yüzdesini gösterme
for col in numerical_columns:
    iqr_outliers = detect_outliers_iqr(df, col)
    zscore_outliers = detect_outliers_zscore(df, col)
    print(f"{col} - IQR yöntemiyle aykırı değer yüzdesi: {iqr_o
    print(f"{col} - Z-score yöntemiyle aykırı değer yüzdesi: {z

```



Inches - IQR yöntemiyle aykırı değer yüzdesi: 2.90%
Inches - Z-score yöntemiyle aykırı değer yüzdesi: 0.31%

Weight - IQR yöntemiyle aykırı değer yüzdesi: 3.53%
Weight - Z-score yöntemiyle aykırı değer yüzdesi: 2.59%

Ram - IQR yöntemiyle aykırı değer yüzdesi: 17.19%
Ram - Z-score yöntemiyle aykırı değer yüzdesi: 1.65%

Price - IQR yöntemiyle aykırı değer yüzdesi: 2.20%
Price - Z-score yöntemiyle aykırı değer yüzdesi: 0.94%

- Yapılan analizde, RAM değişkeni en fazla aykırı değeri barındırmaktadır. 🇮🇹
- Diğer sürekli değişkenler (Inches, Weight, Price) ise yaklaşık olarak eşit sayıda aykırı değere sahiptir. ⚖️
- Aykırı değerleri tespit etmek için Z-skoru yöntemini kullandık ve bu sayede verideki aykırı değer sayısını düşürmüş olduk. 🔍📊

Analysis Goal

Distribution of Price:

- Create a histogram or box plot to visualize the price distribution.

In [34]:

```
# Create a subplot
fig, axes = plt.subplots(1, 2, figsize=(15,6), dpi=70)

# Set background color
fig.patch.set_facecolor('#f6f5f5')
axes[0].set_facecolor('#f6f5f5')
axes[1].set_facecolor('#f6f5f5')

# Histogram - Fiyat dağılımını gösterir
sns.histplot(df['Price'], bins=30, kde=True, ax=axes[0], color='black')
axes[0].set_title("Histogram of Price", fontsize=14, fontweight='bold')
axes[0].set_xlabel("Price", fontsize=12, color='black')
axes[0].set_ylabel("Frequency", fontsize=12, color='black')

# Boxplot - Fiyatın aykırı değerlerini vurgular
sns.boxplot(data=df, x='Price', ax=axes[1], color='black')
axes[1].set_title("Boxplot of Price", fontsize=14, fontweight='bold')
axes[1].set_xlabel("Price", fontsize=12, color='black')

# Remove grid lines and show plot
for ax in axes:
    ax.grid(False)
```

```
ax.set_facecolor('#f6f5f5')

# Title for both plots
fig.suptitle('Price Distribution and Outliers', fontsize=16,

# Adjust layout to avoid overlap
plt.tight_layout(rect=[0, 0, 1, 0.96])

plt.show()
```



- Grafiklerden de görülebileceği gibi price değişkenine ait dağılım sağa çarpıktır. 📊 ➡️
- Histogram ve kutu grafiği, fiyat değişkeninin çoğunlukla 50,000 TL altına yoğunlaştığını, ancak az sayıda çok yüksek fiyatlı gözlemle sağa çarpık bir dağılım gösterdiğini ortaya koyuyor. 💰 🇮🇹
- Aykırı değerler, 150,000 TL ve üzerindeki fiyatlarda yoğunlaşarak, veri setinde belirgin bir çeşitlilik ve dengesizlik olduğunu gösteriyor. ⚠️ 🛠️
- Bu durum, analiz sırasında medyan gibi aykırılıklardan daha az etkilenen metriklerin kullanılmasını veya aykırı değerlerin ayrı bir stratejiyle ele alınmasını gerektirebilir. 🧠 🔧

Company-wise Analysis:

- Plot the count of laptops for each company using a bar chart.
- Visualize the average price of laptops for each company.

```
In [36]: company_counts = df['Company'].value_counts()
```

```
In [37]: # Create the plot
fig, ax = plt.subplots(figsize=(12, 6), dpi=90)
ax.set_facecolor('#f6f5f5')
fig.patch.set_facecolor('#f6f5f5')
```

```
# Set bar colors: First three bars will be red, others will
colors = ['#b20710', '#b20710', '#b20710'] + ['#221f1f'] *

# Plot bar chart with custom colors
company_counts.plot(kind='bar', color=colors, edgecolor='b

# Title and Labels
ax.set_title('Number of Laptops per Company', fontsize=18,
ax.set_xlabel('Company', fontsize=12, fontweight='bold', c
ax.set_ylabel('Number of Laptops', fontsize=12, fontweight

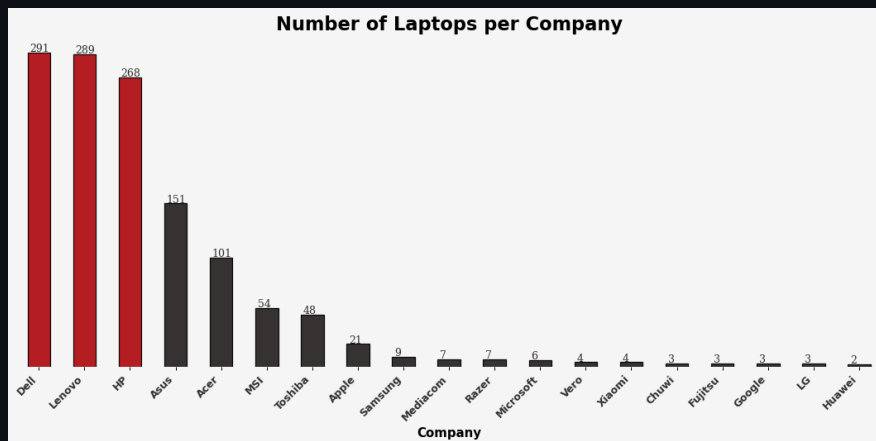
# Add text on top of each bar in a similar style as the pr
for i, value in enumerate(company_counts.values):
    x = i - 0.2
    y = value + 0.5 # Position text slightly above the bar
    ax.text(x, y, str(value), {'font': 'serif', 'weight':

# Rotate x-ticks and format them
ax.set_xticklabels(company_counts.index, rotation=45, ha='

# Remove grid lines and spines
ax.grid(axis='y', linestyle='--', alpha=0.7)
for loc in ['left', 'right', 'top', 'bottom']:
    ax.spines[loc].set_visible(False)

# Hide y-axis ticks
ax.axes.get_yaxis().set_visible(False)

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



- Grafik, farklı markaların ürün sayısını karşılaştırıyor. 📊
- Dell, Lenovo ve HP açık ara en yüksek ürün çeşitliliğine sahip markalar olup sırasıyla 291, 289 ve 268 ürünle lider konumdadır. 🏆 💻
- Asus ve Acer orta düzeyde bir temsil ile dikkat çekerken (sırasıyla 151 ve 101 ürün), Apple, Samsung, ve diğer markalar daha az ürün sunarak grafiğin alt sıralarında yer alıyor. 🍏 📱
- Özellikle Huawei, LG, ve benzeri markalar sınırlı sayıda ürünle

temsil ediliyor, bu da ürün portföylerinin daha dar olduğunu gösteriyor. 🛒 ▼

In [74]:

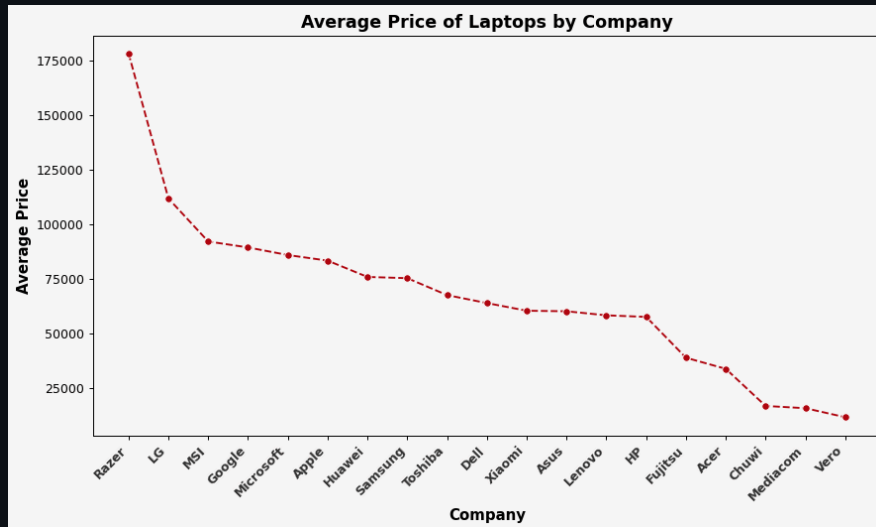
```
# Calculate average price per company
company_avg_price = df.groupby('Company')['Price'].mean()

# Create the plot
fig, ax = plt.subplots(figsize=(10, 6), dpi=90)
ax.set_facecolor('#f6f5f5')
fig.patch.set_facecolor('#f6f5f5')

# Line plot with customized style and red line color
sns.lineplot(x=company_avg_price.index, y=company_avg_price)

# Title and Labels
ax.set_title("Average Price of Laptops by Company", fontsize=14, fontweight='bold', color='red')
ax.set_xlabel("Company", fontsize=12, fontweight='bold', color='red')
ax.set_ylabel("Average Price", fontsize=12, fontweight='bold', color='red')
# Rotate x-ticks and format them
ax.set_xticklabels(company_avg_price.index, rotation=45, fontweight='bold', color='red')

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



- Grafik, şirketlere göre dizüstü bilgisayarların ortalama fiyatlarını göstermektedir. 💻💰
- Razer ve LG, yaklaşık 175,000 birimlik ortalama fiyatla en pahalı dizüstü bilgisayarları sunmaktadır. 🏆🏆
- MSI, Google, ve Microsoft da yüksek fiyatlı ürünleriyle öne çıkıyor. 🇮🇹🚀
- Buna karşılık, Vero, Mediacom, ve Chuwi, en düşük ortalama fiyatlara sahip markalardır ve ekonomik seçenekler sunmaktadır. 💰🛒
- Genel olarak, fiyatlar markadan markaya önemli ölçüde değişmekte, bazı markalar premium segmentte, bazıları ise daha uygun fiyatlı kategoride yer almaktadır. 🎯💡

Relationship between Screen Size and Price:

- Create a scatter plot showing the relationship between Inches and Price.

In [41]:

```
# Create the figure and gridspec layout
fig = plt.figure(figsize=(24, 8), dpi=90) # Wider figure
fig.patch.set_facecolor('#F5F6F6')
gs = fig.add_gridspec(4, 8) # Increased columns to 8 for
gs.update(wspace=0.4, hspace=0.3)

# Create subplots with modified positions - Left side (KDE)
ax1 = fig.add_subplot(gs[1, 0:3]) # Top KDE
ax2 = fig.add_subplot(gs[2:, 0:3]) # 2D KDE
ax3 = fig.add_subplot(gs[2:, 3]) # Price KDE

# Create subplot for scatter plot - Right side
ax4 = fig.add_subplot(gs[1:, 4:]) # Scatter plot

# List of axes for easy iteration
axes = [ax1, ax2, ax3, ax4]

# Style settings for KDE plots
for ax in [ax1, ax2, ax3]:
    ax.set_facecolor('#F5F6F6')
    ax.axes.get_yaxis().set_visible(False)
    ax.axes.get_xaxis().set_visible(False)
    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

# Restore necessary axes
ax2.axes.get_xaxis().set_visible(True)
ax2.axes.get_yaxis().set_visible(True)
ax1.axes.get_xaxis().set_visible(True)
ax1.spines['bottom'].set_visible(True)
ax3.axes.get_yaxis().set_visible(True)
ax1.spines['bottom'].set_visible(True)

# KDE Plots
sns.kdeplot(x='Inches', data=df, ax=ax1, shade=True, color='blue')
sns.kdeplot(y='Price', data=df, ax=ax3, shade=True, color='red')
sns.kdeplot(x='Inches', y='Price', data=df, ax=ax2, color='blue')

# Scatter plot
sns.scatterplot(data=df, x='Inches', y='Price', color='#B22222')
ax4.grid(linestyle='--', alpha=0.5)
ax4.set_facecolor('#F5F6F6')

# Labels and axis ticks for KDE plots
ax2.set_xlabel('Screen Size (Inches)', {'font': 'serif',
ax2.set_ylabel('Price', {'font': 'serif', 'size': 14, 'weight': 'bold'})
ax3.set_ylabel('')
ax1.set_xlabel('')
ax2.set_xticks(ticks=[])
ax2.set_yticks(ticks=[])

# X and Y ticks for KDE plots
```

```

ax1.set_xticks(ticks=np.arange(0, 20, 2))
ax1.set_xticklabels(np.arange(0, 20, 2), **{'font': 'serif'})
ax3.set_yticks(ticks=np.arange(0, 1500, 500))
ax3.set_yticklabels(np.arange(0, 1500, 500), **{'font': 'serif'})

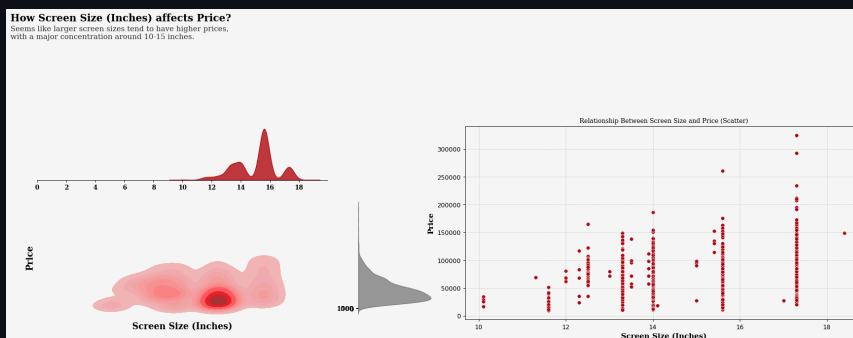
# Labels for scatter plot
ax4.set_title('Relationship Between Screen Size and Price',
              fontsize=14, fontweight='bold', font='serif')
ax4.set_xlabel('Screen Size (Inches)', fontsize=12, fontweight='bold', font='serif')
ax4.set_ylabel('Price', fontsize=12, fontweight='bold', font='serif')

# Add text annotations at the top
fig.text(0.1, 0.95, 'How Screen Size (Inches) affects Price?',
        {'font': 'serif', 'size': 16, 'weight': 'bold'})
fig.text(0.1, 0.90, 'Seems like larger screen sizes tend to have higher prices, with a major concentration around 10-15 inches.',
        {'font': 'serif', 'size': 12, 'weight': 'normal'})

# Adjust Layout
plt.tight_layout()
plt.subplots_adjust(top=0.85)

# Show the plot
plt.show()

```



- 10-15 inç ekran boyutu en popüler ve fiyat açısından çeşitlilik gösteren segmenttir. 📺 🛒
- Daha büyük ekran boyutları (16 inç ve üzeri), fiyatları önemli ölçüde artırmaktadır. 📈 💰
- Fiyat artışında ekran boyutu önemli bir etken olsa da, bazı istisnalar fiyatlandırmanın marka, donanım gibi diğer faktörlerden de etkilendiğini gösteriyor. 💡 ⚙️

RAM vs Price Analysis:

- Use a bar plot or scatter plot to analyze how Ram affects Price.

In [43]:

```

# Assuming 'ram_vs_price' is already defined as in your
ram_vs_price = df.groupby('Ram')['Price'].mean().sort_index()
# Sort the data by RAM values to ensure the bars are ordered
ram_vs_price = ram_vs_price.sort_index(ascending=False)

```



```
ram_vs_price = ram_vs_price.sort_index(ascending = False)

# Create the plot with the specific style
fig, ax = plt.subplots(figsize=(12, 6), dpi=90)
fig.patch.set_facecolor('#f6f5f5')
ax.set_facecolor('#f6f5f5')

# Set bar colors: First three bars will be black, others
colors = ['#b20710', '#b20710', '#b20710'] + ['#221f1f']

# Plot the bar chart
ram_vs_price.plot(kind='bar', color=colors, edgecolor='r')

# Title and Labels
ax.set_title('RAM and Average Price Relationship', fontst
ax.set_xlabel('RAM (GB)', fontsize=12, fontweight='bold')
ax.set_ylabel('Average Price', fontsize=12, fontweight=

# Add text on top of each bar
for i, value in enumerate(ram_vs_price.values):
    x = i - 0.2
    y = value + 0.5 # Position text slightly above the
    ax.text(x, y, str(round(value, 2)), {'font': 'serif'

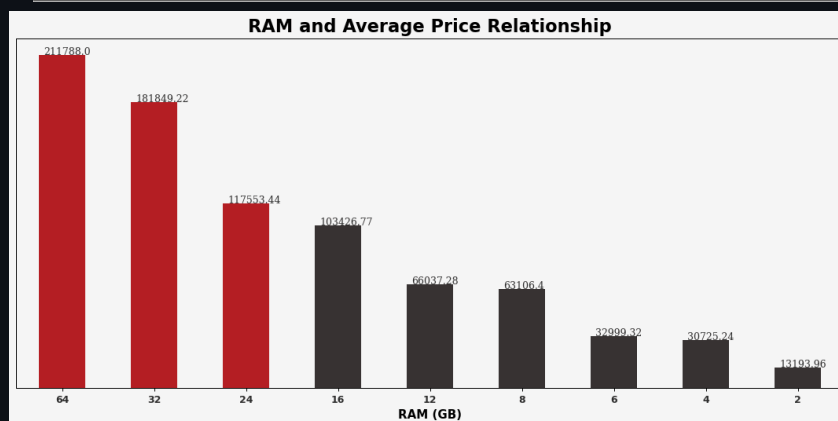
# Rotate x-ticks and format them
ax.set_xticklabels(ram_vs_price.index, rotation=0, fontst

# Remove grid lines and spines
ax.grid(axis='y', linestyle='--', alpha=0.7)

# Remove y-axis ticks
ax.axes.get_yaxis().set_visible(False)

# Adjust layout to prevent overlap
plt.tight_layout()

# Show plot
plt.show()
```



- Daha yüksek RAM kapasitesi, ortalama fiyatı önemli ölçüde artırmaktadır. 📦💡
- Özellikle 64 GB ve 32 GB RAM, üst düzey cihazlarda görülürken en yüksek fiyatlara sahiptir. 🖥️🚀
- 8 GB ve altı RAM kapasitesi, daha ekonomik fiyatlarla ilişkilidir ve giriş seviyesi kullanıcılara hitap eder. 🛒🔄

- Bu ilişki, RAM kapasitesinin cihazın fiyatını belirleyen önemli bir faktör olduğunu göstermektedir. 📊 ⚙️

Operating System Market Share:

- Create a pie chart or bar chart to show the distribution of laptops by OpSys.

In [45]:

```
os_counts = df['OpSys'].value_counts()

# Grafik boyutu ve düzen
fig, axs = plt.subplots(1, 2, figsize=(14, 6), dpi=90)
fig.patch.set_facecolor('#f6f5f5')

# Bar grafiği
colors = ['#221f1f'] + ['#b20710'] * (len(os_counts) - 1)
axs[0].bar(os_counts.index, os_counts.values, color=colors)
axs[0].set_facecolor('#f6f5f5')
axs[0].set_title('Operating System Distribution', fontweight='bold')
axs[0].set_xlabel('Operating System', fontsize=12, fontweight='bold')
axs[0].set_ylabel('Number of Laptops', fontsize=12, fontweight='bold')
axs[0].tick_params(axis='x', labelrotation=45, labelsizex=0)

# Çubuk üstü değer yazma
for i, value in enumerate(os_counts.values):
    axs[0].text(i, value + 0.5, str(value), ha='center', fontweight='bold')

# Pasta grafiği
largest_index = os_counts.idxmax() # En büyük dilimin indexi
colors_pie = ['#221f1f' if idx == largest_index else '#b20710' for i, idx in enumerate(os_counts.index)]

# Yüzdeleri düzgün yerleştirmek için `autopct` kullanımı
def autopct_format(pct):
    return f'{pct:.1f}%' if pct > 2 else '' # Küçük dilimleri gizle

wedges, texts, autotexts = axs[1].pie(
    os_counts,
    labels=None,
    startangle=140,
    colors=colors_pie,
    wedgeprops={'width': 0.3},
    autopct=autopct_format,
    textprops={'color': 'black', 'fontsize': 10}
)

# Yüzdeler daha düzgün yerleşsin diye hizalama
for autotext in autotexts:
    autotext.set_fontsize(10)
    autotext.set_weight('bold')
    autotext.set_rotation(60)

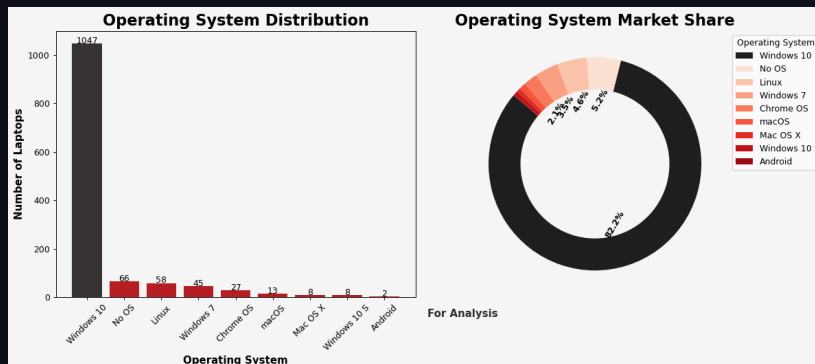
axs[1].set_facecolor('#f6f5f5')
axs[1].set_title('Operating System Market Share', fontweight='bold')

# Pasta grafiği için açıklama
axs[1].legend(labels=os_counts.index, title='Operating System Market Share')
```

```
fig.text(0.57, 0.15, 'For Analysis', ha='right', va='b')

# Yerleşimi sıkıştırma
plt.tight_layout()

# Grafik gösterimi
plt.show()
```



- Windows 10, tartışmasız bir şekilde piyasaya hakim ve en yaygın işletim sistemi. 🖥️ 👑
- Linux, No OS ve eski bir sürüm olan Windows 7, belli bir kullanıcı kitlesi tarafından hâlâ tercih edilse de, Windows 10'un çok gerisindeler. 🐧 📉
- macOS ve Android gibi alternatif işletim sistemleri oldukça düşük kullanım oranlarına sahip ve niş bir kitleye hitap ediyor. 🍏 🤖
- Genel tablo, Windows tabanlı sistemlerin dizüstü bilgisayar pazarındaki baskın konumunu açıkça göstermektedir. 🏆 🖥️

Weight Distribution:

- Plot a histogram to analyze the weight distribution of laptops.

In [47]:

```
# Create a figure with specific size and DPI
plt.figure(figsize=(14, 8), dpi=70)

# Set background color
plt.gcf().patch.set_facecolor('#f6f5f5')

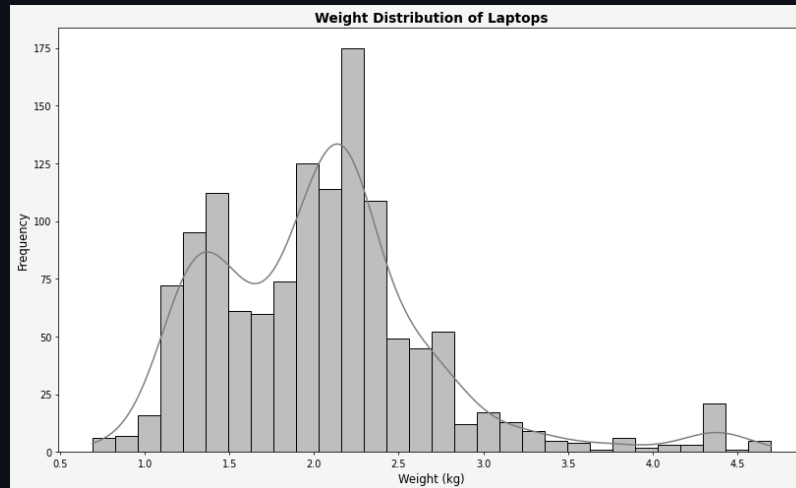
# Create histogram with KDE and customized appearance
sns.histplot(df['Weight'], bins=30, kde=True, color='black')

# Set title and axis labels with specific fonts and colors
plt.title('Weight Distribution of Laptops', fontsize=14, color='black')
plt.xlabel('Weight (kg)', fontsize=12, color='black')
```

```
plt.xlabel('Weight (kg)', fontweight='bold', color='black')
plt.ylabel('Frequency', fontsize=12, color='black')

# Remove grid
plt.grid(False)

# Show the plot
plt.show()
```



- Dizüstü bilgisayarların büyük bir çoğunluğu, taşınabilirlik açısından ideal olan 1.5 - 2.5 kg arasında yoğunlaşmaktadır. 🖥️ ⚖️
- Daha hafif veya daha ağır cihazlar, daha spesifik ihtiyaçlara yönelik üretilmiş olabilir. 🏠 📦
- Bu analiz, kullanıcıların genellikle orta ağırlık segmentindeki cihazları tercih ettiğini göstermektedir.



```
In [49]: df.Weight.mean()
```

```
Out[49]: np.float64(2.040400313971743)
```

```
In [50]: df.Weight.median()
```

```
Out[50]: np.float64(2.04)
```

```
In [51]: import numpy as np
from scipy.stats import skew
# Çarpıklık hesaplama
skewness = skew(df.Weight)

print(f"Çarpıklık (Skewness): {skewness}")
```

```
Çarpıklık (Skewness): 1.1496261365433234
```

Genel Dağılım:

- Histogram, dizüstü bilgisayarların çoğunlukla hangi ağırlık aralıklarında yoğunlaştığını gösterir. 🇩🇪
- Örneğin:
- Çoğu cihazın ağırlığı 1.5 kg ile 2.5 kg arasında yoğunlaşmış olabilir. 🖥️ ⚖️
- Aykırı Değerler:
- Aşırı hafif (örneğin, 1 kg'dan az) veya aşırı ağır (örneğin, 4 kg'dan fazla) cihazlar histogramda uç noktalarda gözlemlenebilir. 🏹 ▼
- Mean ve median değerleri göz önüne alındığında dağılım simetrik gibi gözükse de, skew fonksiyonuna göre sağa çarpık olduğu söylenebilir. ➡️ 📊

Price Correlation Analysis:

- Compute and visualize the correlation between numerical columns (Inches, Weight, Ram, etc.) and Price.

```
In [52]: correlation_matrix = df[['Inches', 'Weight', 'Ram',  
print("Korelasyon Matrisi:")  
print(correlation_matrix)
```

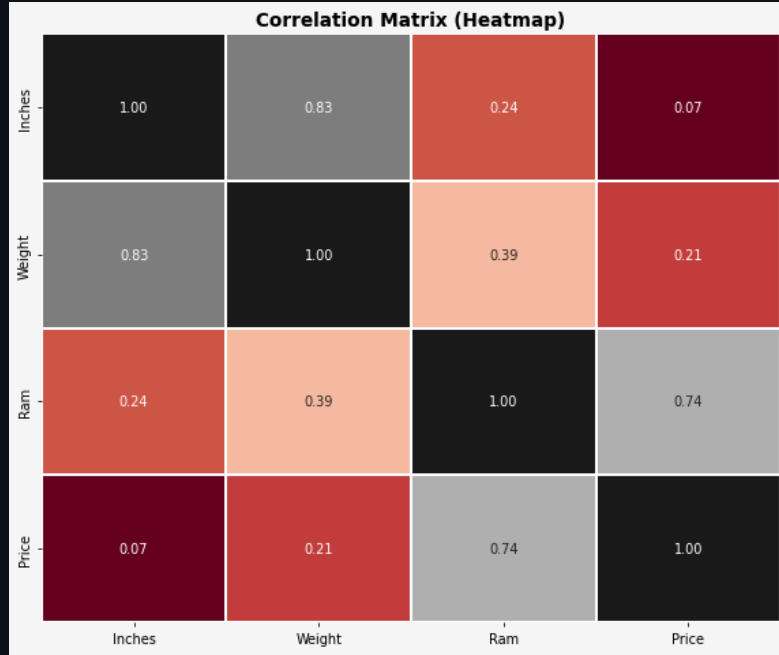
Korelasyon Matrisi:

	Inches	Weight	Ram	Price
Inches	1.000000	0.826634	0.241441	0.066990
Weight	0.826634	1.000000	0.389658	0.212192
Ram	0.241441	0.389658	1.000000	0.740106
Price	0.066990	0.212192	0.740106	1.000000

```
In [53]: # Create a figure with specific size and DPI  
plt.figure(figsize=(10, 8), dpi=70)  
  
# Set background color for the figure  
plt.gcf().patch.set_facecolor('#f6f5f5')  
  
# Create the heatmap for the correlation matrix  
sns.heatmap(correlation_matrix, annot=True, cmap='R',  
            linewidths=0.5, cbar=False)  
  
# Set title and labels with specific fonts and color  
plt.title('Correlation Matrix (Heatmap)', fontsize=
```

```
# Set the background color for the axes (plot area)
plt.gca().set_facecolor('#f6f5f5')

# Show the plot
plt.show()
```



Görsel Üzerindeki Değişkenler

Inches ile Diğer Değişkenler:

- "Inches" (muhtemelen ekran boyutu) ile "Weight" arasında 0.83 gibi güçlü bir pozitif korelasyon var. 📏 + 🏠
- Bu, daha büyük ekranlı cihazların daha ağır olma eğiliminde olduğunu gösterir.
- "Inches" ile diğer değişkenler arasında daha zayıf bir ilişki var (örneğin, "Price" ile 0.07 gibi düşük bir korelasyon). 📏 \$

Weight ile Diğer Değişkenler:

- "Weight" ile "Inches" arasında güçlü bir ilişki olduğu görülüyor (0.83). 📏 🏠
- "Weight" ile "Ram" ve "Price" arasında ise daha zayıf pozitif korelasyonlar var (sırasıyla 0.39 ve 0.21). 🏠 💰

Ram ile Diğer Değişkenler:

- "Ram" ile "Price" arasında güçlü bir pozitif korelasyon var (0.74). 📁 + 💰

Bu, daha yüksek RAM'e sahip cihazların genelde daha pahalı olduğunu gösterir.

- "Ram" ile diğer değişkenler arasında daha zayıf ilişkiler var.

Price ile Diğer Değişkenler:

- "Price" (fiyat) ile "Ram" arasında güçlü bir pozitif ilişki var (0.74). 💰 + 📈
- "Price" ile "Inches" ve "Weight" arasında oldukça zayıf korelasyonlar var (sırasıyla 0.07 ve 0.21). 📉

Sonuç

- Görsel, değişkenler arasında en güçlü ilişkinin "Ram" ve "Price" arasında olduğunu gösteriyor. 📈
- "Inches" ve "Weight" arasındaki pozitif ilişki de oldukça dikkat çekici. ⚖️
- Bazı değişkenler (örneğin, "Price" ve "Inches") arasında çok zayıf korelasyon var, bu da bu değişkenlerin birbirini çok fazla etkilemediğini gösterir. 💡

ScreenResolution Impact on Price:

- Analyze how different resolution types (e.g., Full HD, 4K) affect the price.

In [54]:

```
# Create a figure with specific size and DPI
plt.figure(figsize=(10, 6), dpi=70)

# Set background color for the figure
plt.gcf().patch.set_facecolor('#f6f5f5')

# Create the scatter plot with red points
sns.scatterplot(x='HD_Category', y='Price', data=d

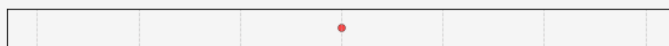
# Set title and axis labels with specific fonts and
plt.title('Screen Resolution Impact on Price', font
plt.xlabel('HD Category', fontsize=12, color='black')
plt.ylabel('Price', fontsize=12, color='black')

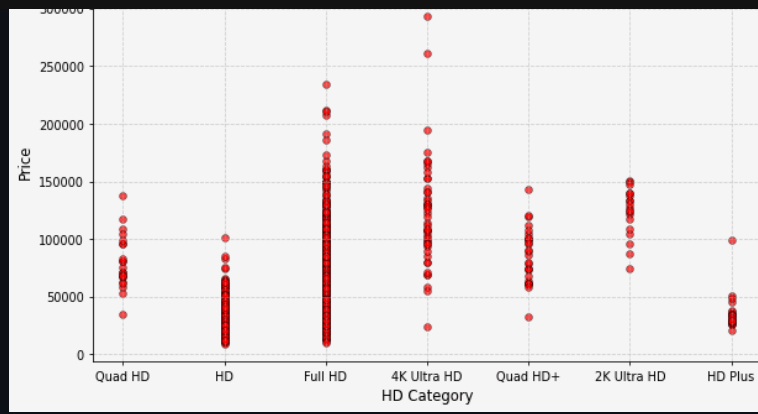
# Remove gridlines and set a soft grid style
plt.grid(linestyle='--', alpha=0.5)

# Set background color for the axes (plot area)
plt.gca().set_facecolor('#f6f5f5')

# Show the plot
plt.show()
```

Screen Resolution Impact on Price





Ekran Çözünürlükleri ve Fiyat İlişkisi

- 4K Ultra HD, genel olarak en yüksek fiyatlarla ilişkilendirilmiştir ve fiyatlar 300,000 birime kadar çıkmaktadır. 💎📺
- Bu, en pahalı ve premium cihaz segmentini temsil eder.
- Full HD, en sık kullanılan çözünürlük kategorisidir ve fiyatlar 50,000 - 100,000 aralığında yoğunlaşmıştır. 📊🔄
- Quad HD ve 2K Ultra HD, Full HD'den biraz daha yüksek fiyatlarla ilişkilendirilmiştir ancak 4K Ultra HD kadar yüksek değildir. 🔍💰
- HD ve HD Plus, fiyatların 50,000 birim altında yoğunlaştığı, ekonomik segmenti temsil eder. 💻💰

Top CPU/GPU Manufacturers:

- Extract and visualize the most common CPU and GPU manufacturers.

In [56]:

```
# CPU üreticilerinin verisini hazırlama
cpu_manufacturers = df['Cpu_Manufacturer'].value_counts()
cpu_manufacturers.columns = ['CPU_Manufacturer', 'Count']
cpu_manufacturers_top2 = cpu_manufacturers.head(2)

# Grafik boyutu ve düzen
fig, axes = plt.subplots(1, 2, figsize=(12, 6), dpi=100)
fig.patch.set_facecolor('#f6f5f5')

# 1. Grafik: Bar Plot
sns.barplot(x='CPU_Manufacturer', y='Count', data=cpu_manufacturers_top2)
```



```

axes[0].set_title('CPU Manufacturers', fontsize=12)
axes[0].set_xlabel('CPU Manufacturer', fontsize=12)
axes[0].set_ylabel('Count', fontsize=12, fontweight='bold')
axes[0].tick_params(axis='x', rotation=45, labels=True)

# Çubuk üzerindeki yazıları düzenleme
for p in axes[0].patches:
    axes[0].annotate(f'{p.get_height():.0f}',
                    (p.get_x() + p.get_width() / 2,
                     p.get_height() + 5),
                    ha='center', va='center',
                    fontsize=12, color='black',
                    xytext=(0, 9), textcoords='offsetpoints')

# 2. Grafik: Pie Chart (Top 2 CPU Üreticileri)
cpu_explode = [0.1 if i == cpu_manufacturers_top2[0] else 0]
axes[1].pie(cpu_manufacturers_top2['Count'], labels=cpu_manufacturers_top2['Manufacturer'],
            shadow=True, autopct='%1.1f%%', startangle=90,
            colors=['#8b0000', '#b20710'], textprops={'color': 'white'})

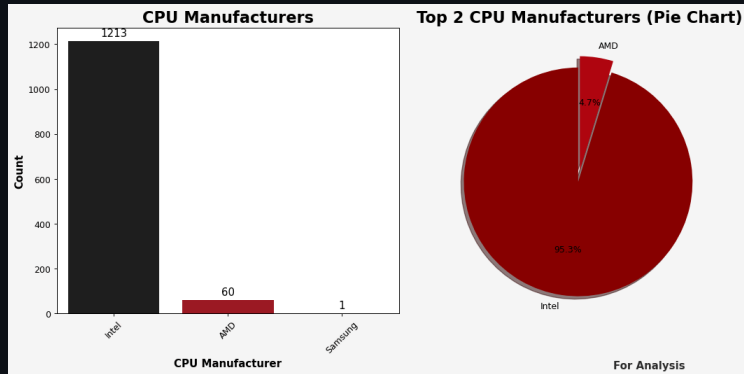
axes[1].set_title('Top 2 CPU Manufacturers (Pie Chart)')

# Sağ alt köşeye metin ekleme
fig.text(0.9, 0.02, 'For Analysis', ha='right', va='bottom',
        fontweight='bold', color='black', size=12)

# Grafik yerleşimini sıkıştırma
plt.tight_layout()

# Grafik gösterimi
plt.show()

```



İşlemci Üreticileri ve Pazar Payı

- Intel, dizüstü bilgisayar işlemci pazarına tamamen hakim 🏆, hem cihaz sayısında hem de pazar payında lider durumda. 🔧 💻
- AMD, alternatif bir üretici olarak varlığını sürdürüyor 🔄, ancak Intel ile kıyaslandığında oldukça sınırlı bir kullanım oranına sahip. ⚙️ 📊
- Samsung gibi diğer üreticiler, neredeyse ihmal edilebilir düzeyde pazar payına sahip. 📈 🇹🇷

In [58]:

```
# GPU üreticilerinin verisini hazırlama
gpu_manufacturers = df['Gpu_Manufacturer'].value_counts()
gpu_manufacturers.columns = ['GPU_Manufacturer', 'Count']
gpu_manufacturers_top3 = gpu_manufacturers.head(3)

# Grafik boyutu ve düzen
fig, axes = plt.subplots(1, 2, figsize=(14, 6), style='dark')
fig.patch.set_facecolor('#f6f5f5')

# Bar plot: İlk bar siyah olacak şekilde
sns.barplot(x='GPU_Manufacturer', y='Count', data=gpu_manufacturers_top3, style='dark')
axes[0].set_title('GPU Manufacturers', fontsize=12)
axes[0].set_xlabel('GPU_Manufacturer', fontsize=12)
axes[0].set_ylabel('Count', fontsize=12, fontweight='bold')
axes[0].tick_params(axis='x', rotation=45, labels=gpu_manufacturers_top3['GPU_Manufacturer'].values)

# Çubuk üzerindeki yazıları düzenleme
for p in axes[0].patches:
    axes[0].annotate(f'{p.get_height():.0f}',
                     (p.get_x() + p.get_width() / 2, p.get_height()),
                     ha='center', va='center',
                     fontsize=12, color='black',
                     xytext=(0, 9), textcoords='point')

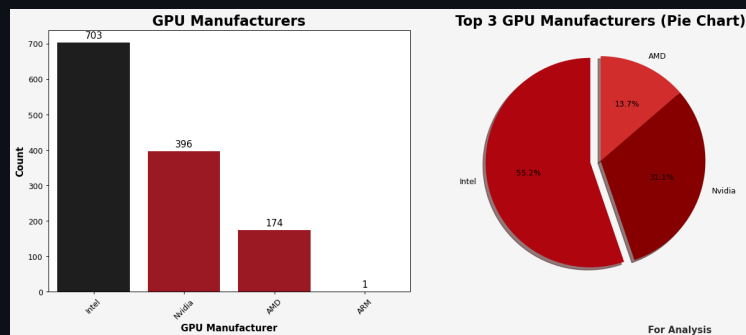
# Pie chart for top 3 GPU manufacturers
gpu_explode = [0.1 if i == gpu_manufacturers_top3['GPU_Manufacturer'].values[0] else 0]
axes[1].pie(gpu_manufacturers_top3['Count'], labels=gpu_manufacturers_top3['GPU_Manufacturer'].values,
            shadow=True, autopct='%1.1f%%', startangle=90, colors=['#b20710', '#8b0000', '#d32f2f'])

axes[1].set_title('Top 3 GPU Manufacturers (Pie Chart)', style='dark')

# Sağ alt köşeye metin ekleme
fig.text(0.9, 0.02, 'For Analysis', ha='right', va='bottom', style='dark')

# Grafik yerleşimini sıkıştırma
plt.tight_layout()

# Grafik gösterimi
plt.show()
```



GPU Üreticileri ve Pazar Payı

- Intel, GPU üretiminde lider 🏆 ve entegre grafik birimlerinin yaygınlığı sayesinde en yüksek pazar payına sahiptir. 💻 ⚡

- Nvidia, yüksek performans segmentinde güçlü

- Nvidia, yüksek performans segmentinde güçlü bir konumda 🚀 ve ciddi bir pazar payına sahiptir. 🎮💡
- AMD, daha küçük bir pazar payı ile alternatif bir seçenek olarak varlığını sürdürüyor. 🔧📊
- ARM, çok nadir kullanılan bir GPU üreticisi olarak karşımıza çıkıyor. 📺💡
- Bu grafik, Intel'in günlük kullanım ve genel ihtiyaçlar için baskın olduğunu, Nvidia'nın ise daha çok performans odaklı cihazlarda tercih edildiğini gösteriyor.

RAM Analysis:

- Identify the most common RAM configurations and their respective average prices.

In [60]:

```
# En yaygın 5 RAM kapasitesini belirleme
top_rams = df['Ram'].value_counts().head(5).index

# Hem frekans hem de ortalama fiyat bilgisi için
top_rams_data = df[df['Ram'].isin(top_rams)].groupby(
    frequency=('Ram', 'count'),
    avg_price=('Price', 'mean')
).reset_index().sort_values(by='frequency', ascending=False)

# Grafik boyutu ve düzen
fig, ax = plt.subplots(figsize=(12, 8), dpi=90)
fig.patch.set_facecolor('#f6f5f5')
ax.set_facecolor('#f6f5f5')

# Bar plot
bars = ax.bar(
    top_rams_data['Ram'],
    top_rams_data['frequency'],
    color='#b20710',
    edgecolor='none',
    alpha=0.9
)

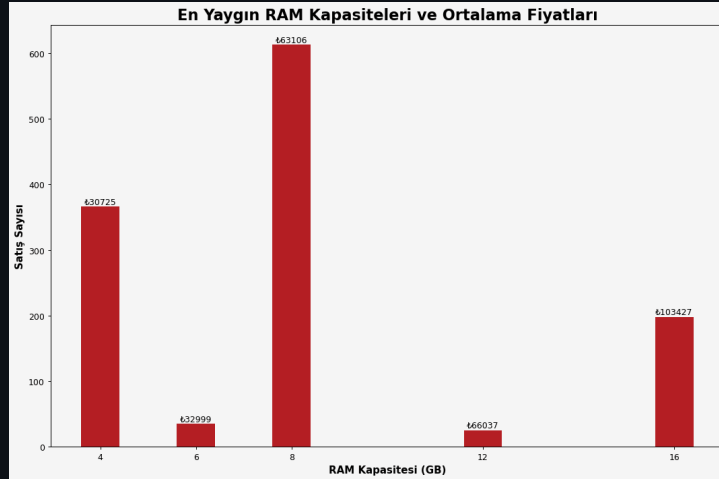
# Başlık ve eksen etiketleri
ax.set_title('En Yaygın RAM Kapasiteleri ve Ortalama Fiyatları')
ax.set_xlabel('RAM Kapasitesi (GB)', fontsize=12)
ax.set_ylabel('Satış Sayısı', fontsize=12, fontweight='bold')

# Çubuk üzerindeki yazıları düzenleme
for i, bar in enumerate(bars):
    yval = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2,
            yval,
            ha='center', va='bottom', fontsize=10)
```

```
# X ekseninde yalnızca belirli RAM kapasiteleri
ax.set_xticks(top_rams_data['Ram'])
ax.tick_params(axis='x', labelsiz=10, colors='

# Yerleşimi sıkıştırma
plt.tight_layout()

# Grafik gösterimi
plt.show()
```



RAM Kapasiteleri ve Fiyat Segmentleri

- 8 GB RAM, en popüler kapasite olup 📁 fiyat-performans açısından öne çıkmaktadır. 🔥💰
- 16 GB RAM, premium segmentte yer alırken daha yüksek fiyatlarla ilişkilidir. 💎💻
- 4 GB RAM, uygun fiyatlı cihazlar için en çok tercih edilen giriş seviyesi seçenektir. 💛📉
- Satış miktarı ile fiyat arasında genelde ters orantı vardır; yüksek fiyatlı RAM kapasiteleri daha az satılmaktadır. 📊⬇️

Memory Type and Price Relationship:

- Analyze the impact of memory type (e.g., HDD, SSD, Hybrid) on the price.

In [62]:

```
df[df['Memory'] == '8GB SSD']
```

Out[62]:

Company	TypeName	Inches	ScreenResolution
ASUS	15.6"	15.6	IPS Panel Full HD

950 HP Workstation 15.6 1920x1080

◀ ▶

In [63]: `df.iloc[950, df.columns.get_loc('Memory')] = '256GB SSD'`

◀ ▶

In [64]: `df.iloc[950]`

Out[64]:

Company	HP
Processor	Intel Core i7 6820HQ
Type	Workstation
Screen Size (Inches)	15.6
Screen Resolution	IPS Panel Full HD 1920x1080
CPU	Intel Core i7 6820HQ
RAM	8
Memory	256GB SSD
GPU	Nvidia Quadro M1000M
Operating System	Windows 10
Weight (kg)	2.3
Price	119826.7
Resolution	1920x1080
Touchscreen	No
HD Category	Full HD
Refresh Rate (Hz)	60
CPU Manufacturer	Intel
CPU Core Type	i7
GPU Manufacturer	Nvidia

Name: 950, dtype: object

In [65]:

```
# Kategorilere göre ortalama fiyat hesaplama (Price)
avg_prices = df.groupby('Memory')['Price'].mean()

# En popüler 10 bellek türünü seçme
avg_prices = avg_prices.sort_values('Price', ascending=False)

# Fiyata göre sıralama (büyükten küçüğe)
avg_prices = avg_prices.sort_values('Price', ascending=False)

# Grafik oluşturma
fig, ax = plt.subplots(figsize=(14, 8), dpi=68)

# Arka plan renkleri
fig.patch.set_facecolor('#f6f5f5')
```

```

ax.set_facecolor('#f6f5f5')

# Kırmızı çubuklar
ax.barh(y=avg_prices['Memory'],
        width=avg_prices['Price'],
        height=0.15,
        color='#b20710')

# Kırmızı noktalar (daireler)
ax.scatter(y=avg_prices['Memory'],
           x=avg_prices['Price'],
           s=avg_prices['Price']*0.005, # Nokta büyüklüğü
           c='#b20710')

# Merkez çizgisi
ax.axvline(x=0, ymin=0, ymax=1,
           linewidth=0.8, linestyle='--', color='black')

# Etiketleri ekleme
for idx, (memory_type, price) in enumerate(zip(avg_prices['Memory'], avg_prices['Price'])):
    # Kategori ismi
    ax.text(-500, idx, memory_type,
           horizontalalignment='right',
           fontsize=12, fontfamily='serif', fontweight='bold',
           color='black', alpha=0.9)

    # Fiyat değeri
    ax.text(price + 4000, idx, f"${price:,.2f}",
           horizontalalignment='left',
           fontsize=10, fontfamily='serif', fontweight='normal',
           color='black', alpha=0.8)

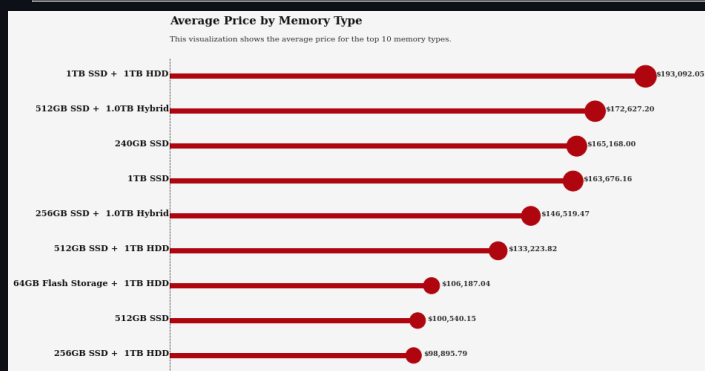
# Çerçeve ve eksenleri kaldırma
for loc in ['left', 'right', 'top', 'bottom']:
    ax.spines[loc].set_visible(False)
ax.axes.get_xaxis().set_visible(False)
ax.axes.get_yaxis().set_visible(False)

# X eksen sınırları
max_price = max(avg_prices['Price'])
ax.set_xlim([0, max_price + 5000])

# Başlık ve alt başlık
ax.text(0, len(avg_prices) + 0.5, 'Average Price by Memory Type',
       fontsize=16, fontfamily='serif', fontweight='bold',
       color='black', alpha=0.9)
ax.text(0, len(avg_prices), 'This visualization shows the average price for the top 10 memory types.',
       fontsize=11, fontfamily='serif', color='black', alpha=0.8)

plt.tight_layout()
plt.show()

```



Depolama Türleri ve Fiyat İlişkisi

- SSD ve HDD kombinasyonları, özellikle yüksek depolama kapasiteleri ile en pahalı seçenekleri oluşturuyor. 📁💡
- Tek bir SSD türü (örneğin, 1TB SSD), daha ekonomik seçeneklere kıyasla yüksek fiyatlara sahip. 💻💰
- 512GB SSD + 2TB HDD, fiyat-performans açısından daha uygun bir seçenek sunuyor. ⚖️✅
- Yüksek kapasiteli hibrit depolama türleri, fiyatları önemli ölçüde artırmaktadır. 📊💡
- Bu analiz, depolama kapasitesi ve türlerinin fiyatlar üzerinde önemli bir etkisi olduğunu açıkça göstermektedir. 💡📊

High-end Laptop Analysis:

- Define a "high-end" laptop (e.g., based on Price or specs) and analyze their characteristics.

In [67]:

```
# Define a "high-end" Laptop
high_end_price_threshold = 80000 # You can define this threshold based on your needs
high_end_laptops = df[df['Price'] > high_end_price_threshold]

# Characteristics of high-end Laptops
high_end_laptops_characteristics = high_end_laptops[['Screen Size', 'RAM', 'Storage', 'GPU', 'CPU']]

# Analyze frequency of each characteristic
high_end_analysis = high_end_laptops_characteristics.groupby('Characteristic').count().reset_index()

# Example: Get the most common characteristics
common_touchscreen = high_end_laptops['Touchscreen'].value_counts().index[0]
common_hd_category = high_end_laptops['HD Category'].value_counts().index[0]
common_cpu_manufacturer = high_end_laptops['CPU Manufacturer'].value_counts().index[0]
common_gpu_manufacturer = high_end_laptops['GPU Manufacturer'].value_counts().index[0]
common_ram = high_end_laptops['Ram'].value_counts().index[0]

# Display the analysis results
#print("High-End Laptop Characteristics Analysis Results")
#print("Touchscreen Distribution: ", common_touchscreen)
#print("HD Category Distribution: ", common_hd_category)
#print("CPU Manufacturer Distribution: ", common_cpu_manufacturer)
#print("GPU Manufacturer Distribution: ", common_gpu_manufacturer)
#print("RAM Distribution: ", common_ram)
```

```
# Show high-end Laptops data
high_end_laptops.head()
```

Out[67]:

	Company	Type	Name	Inches	ScreenResolution
--	---------	------	------	--------	------------------

3	Apple	Ultrabook		15.4	IPS Panel Reti Displ 2880x18
4	Apple	Ultrabook		13.3	IPS Panel Reti Displ 2560x16
6	Apple	Ultrabook		15.4	IPS Panel Reti Displ 2880x18
12	Apple	Ultrabook		15.4	IPS Panel Reti Displ 2880x18
15	Apple	Ultrabook		13.3	IPS Panel Reti Displ 2560x16

In [68]:

```
# Grafik boyutu ve düzen
fig, axs = plt.subplots(2, 2, figsize=(18, 18))
fig.patch.set_facecolor('#f6f5f5')

# Veriler ve başlıklar
data_and_titles = [
    (common_touchscreen, "Touchscreen Distribüsyonu"),
    (common_hd_category, "HD Category Distribüsyonu"),
    (common_cpu_manufacturer, "CPU Manufacturer"),
    (common_gpu_manufacturer, "GPU Manufacturer")
]

# Grafik çizimi
for ax, (data, title) in zip(axs.flat, data_and_titles):
    ax.bar(data.index, data.values, color='#f6f5f5')
    ax.set_facecolor('#f6f5f5')
    ax.set_title(title, fontsize=18, fontweight='bold')
    ax.set_xlabel(data.index.name if data.index.name else 'Index',
                  fontsize=12, fontweight='bold')
    ax.set_ylabel('Count', fontsize=12, fontweight='bold')
    ax.tick_params(axis='x', labelrotation=45)

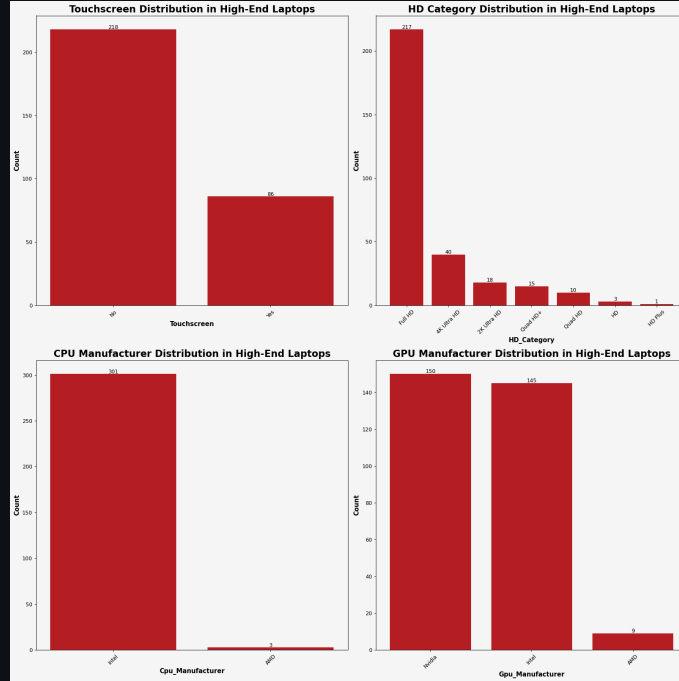
    # Çubuk üstü değer yazma
    for i, value in enumerate(data.values):
        ax.text(i, value + 0.5, str(value),
                fontweight='bold',
                fontfamily='serif')

# Boş eksenleri kaldır
if len(data_and_titles) < len(axs.flat):
    for ax in axs.flat[len(data_and_titles):]:
        ax.axis('off')

# Yerleşimi sıkıştırma
plt.tight_layout()
```



```
# Grafik gösterimi
plt.show()
```



Teknolojik Özellikler ve Pazar Trendleri

- Dokunmatik ekran: Daha çok standart ekran tercih edilmektedir. Yüksek segmentte dokunmatik ekran kullanımı nispeten azdır.
- Ekran çözünürlüğü: Full HD en yaygın çözünürlük olsa da, 4K Ultra HD premium özellikli cihazlarda oldukça popülerdir.
- CPU: Intel, yüksek segment cihazlarda tamamen hakimdir. AMD'nin varlığı oldukça sınırlıdır.
- GPU: NVIDIA, yüksek performanslı grafik birimlerinde liderken, Intel entegre grafiklerde güçlü bir alternatiftir.

```
In [69]: df.Price.describe()
```

```
Out[69]: count    1274.000000
         mean     60503.185074
         std      37333.222977
         min       9270.720000
        25%      32495.605200
        50%      52693.920000
        75%      79773.480000
         max     324954.720000
         Name: Price, dtype: float64
```

TypeName Analysis:

- Explore the distribution of TypeName and its relationship with price (e.g., Gaming vs Ultrabook).

In [70]:

```
# Kategorilere göre ortalama fiyat hesaplama
avg_prices2 = df.groupby('TypeName')['Price'].mean()

# Fiyata göre sıralama (büyükten küçüğe)
avg_prices2 = avg_prices2.sort_values('Price', ascending=False)

# Grafik oluşturma
fig, ax = plt.subplots(figsize=(14, 8), dpi=100)

# Arka plan renkleri
fig.patch.set_facecolor('#f6f5f5')
ax.set_facecolor('#f6f5f5')

# Kırmızı çubuklar
ax.barh(y=avg_prices2['TypeName'],
        width=avg_prices2['Price'],
        height=0.15,
        color='#b20710')

# Kırmızı noktalar (daireler)
# Nokta boyutunu fiyatla orantılı yapma (0.005)
ax.scatter(y=avg_prices2['TypeName'],
           x=avg_prices2['Price'],
           s=avg_prices2['Price']*0.005, #
           c='#b20710')

# Merkez çizgisi
ax.axvline(x=0, ymin=0, ymax=1,
           linewidth=0.8, linestyle='--', color='black')

# Etiketleri ekleme
for idx, (type_name, price) in enumerate(zip(avg_prices2['TypeName'], avg_prices2['Price'])):
    # Kategori ismi
    ax.text(-500, idx, type_name,
           horizontalalignment='right',
           fontsize=12, fontfamily='serif',
           color='black', alpha=0.9)

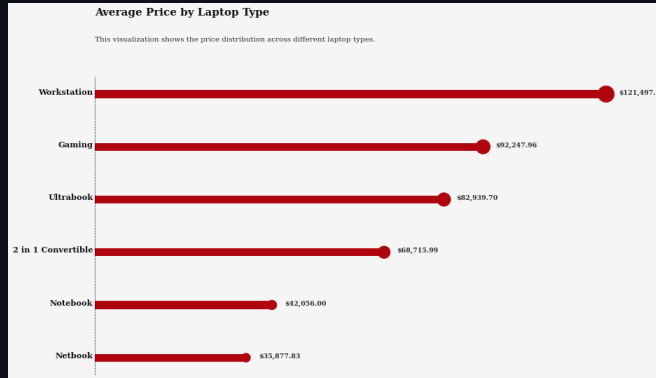
    # Fiyat değeri
    ax.text(price + 3000, idx, f"${price:,.0f}",
           horizontalalignment='left',
           fontsize=10, fontfamily='serif',
           color='black', alpha=0.8)

# Çerçeve ve eksenleri kaldırma
for loc in ['left', 'right', 'top', 'bottom']:
    ax.spines[loc].set_visible(False)
ax.axes.get_xaxis().set_visible(False)
ax.axes.get_yaxis().set_visible(False)









# X eksenini sınırları
max_price = max(avg_prices2['Price'])
ax.set_xlim([0, max_price + 5000])
```

```
# Başlık ve alt başlık
ax.text(0, len(avg_prices2) + 0.5, 'Average Price by Laptop Type',
       fontsize=16, fontfamily='serif', fontcolor='black', alpha=0.9)
ax.text(0, len(avg_prices2), 'This visualization shows the price distribution across different laptop types.',
       fontsize=11, fontfamily='serif', fontcolor='black', alpha=0.9)

plt.tight_layout()
plt.show()
```



Cihaz Türlerine Göre Fiyatlandırma ve Performans

- Workstation cihazları, profesyonel işler için sundukları üst düzey performans nedeniyle en pahalı kategoriyi oluşturuyor.  
- Gaming laptoplar, oyun performansına odaklanarak yüksek fiyatlı kategoride yer alıyor.  
- Netbook ve notebook, uygun fiyatlarıyla temel kullanıcı ihtiyaçlarına hitap ediyor.  
- Fiyatlar, cihazların sunduğu performans, taşınabilirlik ve kullanım amacına göre belirgin şekilde farklılık gösteriyor.  

Feel free to include any additional analyses.

In [72]:

```
from wordcloud import WordCloud
text_data = ' '.join(df['Company'].astype(str))
            ' '.join(df['TypeName'].astype(str))
            ' '.join(df['Cpu'].astype(str))
            ' '.join(df['Gpu'].astype(str))
            ' '.join(df['OpSys'].astype(str))

wordcloud = WordCloud(width=800, height=400)
```

◀ ▶



```
company_counts = df['Company'].value_counts()
```

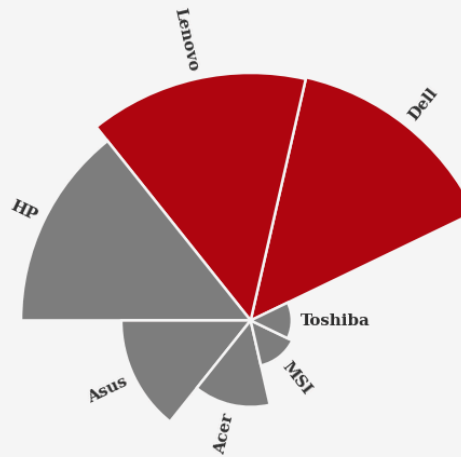
```
alignment = "left"
if angle >= np.pi / 2 and angle < 3 *
    alignment = "right"
    rotation = rotation + 180
ax.text(
    x=angle,
    y=lowerLimit + bar.get_height() +
    s=label,
    ha=alignment,
    va='center',
    rotation=rotation,
    rotation_mode="anchor",
    fontdict={'family': 'serif', 'size'
    alpha=0.8
)

# Başlık ve açıklama
fig.text(0.25, 1.05, 'Company Distribution
fig.text(0.25, 0.975, 'Top 6 companies bas

# Grafik gösterimi
fig.show()
```

Company Distribution - Top 6 Companies

Top 6 companies based on the count in the dataset.



Conclusions

Sonuçlar:

- Dokunmatik Ekran Kullanımı:
- Dokunmatik ekranlı cihazlar, yüksek segment dizüstü bilgisayar pazarında daha az tercih edilmektedir. Bu durum, performans odaklı kullanıcıların işlemci

gucu, GPU yetenekleri ve ekran kalitesi gibi özelliklere öncelik verdiğini göstermektedir. 🖥️

Ekran Çözünürlüğü:

- Full HD, maliyet ve kalite dengesini sağladığı için en yaygın kullanılan çözünürlüktür. 4K Ultra HD ise premium segmentte, yüksek detay ve renk doğruluğu gerektiren kullanıcılar için öne çıkmaktadır. ✨ 🖥️

CPU ve GPU Tercihleri:

- Intel işlemciler, yüksek performanslı güvenilir yapısıyla pazara hakimdir. ⚙️
- NVIDIA GPU'lar, oyun, içerik üretimi ve profesyonel iş yükleri için lider konumdadır. 🎮 📁
- Intel entegre GPU'lar ise hafif iş yükleri için hala tercih edilmektedir. 🔋
- AMD'nin varlığı ise sınırlıdır. ⚡

Farklı Kullanım Amaçları:

- Yüksek segment dizüstü bilgisayarlar, oyun oynama, profesyonel işler ve günlük kullanım gibi çok çeşitli ihtiyaçları karşılamak için farklı özelliklere sahiptir. 🎮 🖥️

🔧 Öneriler:

- Hedefe Yönelik Ürün Geliştirme:
- Çoğu kullanıcıyı hedefleyen Full HD ve premium kullanıcıları hedefleyen 4K Ultra HD ekran seçeneklerini geliştirin. Özellikle yüksek yenileme hızı ve geniş renk gamı gibi özellikler ekleyin. 🌈 🛠️
- Dokunmatik ekranlı ve dokunmatik olmayan modeller arasında denge sağlayarak, performans odaklı kategoriyi güçlendirin. 🖱️ 🖥️

AMD Ürünlerini Genişletme:

- AMD işlemci ve GPU seçeneklerini artırarak rekabetçi alternatifler sunun

ve AMD'nin oyun ve profesyonel segmentte artan popülaritesinden faydalanın. 🔥 💻

GPU İnovasyonu:

- NVIDIA ile iş birliğini güçlendirerek oyun ve içerik üretimi modellerini daha da geliştirin. Aynı zamanda, entegre GPU çözümleriyle batarya ömrünü optimize edin. 🟢 🔧

Kişiselleştirilebilir Seçenekler:

- Kullanıcı tabanını genişletmek için RAM, depolama ve GPU gibi özelleştirilebilir konfigürasyonlar sunun. 🔧 📁

Pazarlama Stratejileri:

- Intel CPU ve NVIDIA GPU üstünlüğünü vurgulayan kampanyalar düzenleyerek performans odaklı kullanıcıları hedefleyin. 🔊
- 4K Ultra HD dizüstü bilgisayarları, içerik üreticileri ve hassasiyet gerektiren profesyonel işler için pazarlayın. 💻 💡

Dokunmatik Ekran Özelliklerini Artırma:

- Yüksek segment kategorisinde daha fazla 2'si 1 arada dönüşebilir modeller sunarak esneklik arayan kullanıcıları hedefleyin. 🔄 📁
-