Laptop Price Analysis Report

Introduction

This report presents a comprehensive analysis of the dataset containing specifications and prices of laptops. The aim is to identify key trends and factors influencing pricing.

Dataset Overview

The dataset includes attributes such as brand, processor type, RAM, storage, GPU, operating system, weight, and touchscreen capability, among others.

Key Observations

1. Brand Impact:

- Premium brands like Apple and Dell tend to have higher average prices.

- Budget brands such as Acer and Lenovo offer more affordable options.

2. Processor Type:

- Laptops with Intel Core i7 and AMD Ryzen 7 are significantly more expensive than those with Core i3 or Ryzen 3.

3. RAM and Storage:

- Price increases with higher RAM and SSD capacity.

- HDD-based laptops are generally cheaper.

4. GPU:

- Devices with dedicated GPUs (e.g., Nvidia GTX/RTX) command higher prices.

5. Other Factors:

- Laptops with touchscreen features and lighter weight tend to be priced higher.

Conclusion

The price of a laptop is primarily influenced by brand, processor, RAM, storage type, and GPU. Understanding these factors can guide buyers toward cost-effective choices based on their needs.

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("laptop_prices.csv",encoding="ISO-8859-1")
df.head()
```

Out[5]:

	Company	Product	TypeName	Inches	Ram	OS	Weight	Price_euros	Screen	ScreenW	...	RetinaDisplay	CPU_compa
0	Apple	MacBook Pro	Ultrabook	13.3	8	macOS	1.37	1339.69	Standard	2560	...	Yes	Int
1	Apple	Macbook Air	Ultrabook	13.3	8	macOS	1.34	898.94	Standard	1440	...	No	Int
2	HP	250 G6	Notebook	15.6	8	No OS	1.86	575.00	Full HD	1920	...	No	Int
3	Apple	MacBook Pro	Ultrabook	15.4	16	macOS	1.83	2537.45	Standard	2880	...	Yes	Int
4	Apple	MacBook Pro	Ultrabook	13.3	8	macOS	1.37	1803.60	Standard	2560	...	Yes	Int

5 rows × 23 columns

In [7]:
```
df.shape
```

Out[7]: (1275, 23)

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Company                0
Product                0
TypeName               0
Inches                 0
Ram                    0
OS                     0
Weight                 0
Price_euros            0
Screen                 0
ScreenW                0
ScreenH                0
Touchscreen            0
IPSpanel               0
RetinaDisplay          0
CPU_company            0
CPU_freq               0
CPU_model              0
PrimaryStorage         0
SecondaryStorage       0
PrimaryStorageType     0
SecondaryStorageType   0
GPU_company            0
GPU_model              0
dtype: int64
```

In [11]:
```
from sklearn.preprocessing import LabelEncoder

# Separate features and target
X = df.drop('Price_euros', axis=1)
y = df['Price_euros']

# Label encode categorical columns
categorical_columns = [
    'Company', 'Product', 'TypeName', 'OS', 'Screen',
    'Touchscreen', 'IPSpanel', 'RetinaDisplay', 'CPU_company',
    'CPU_model', 'PrimaryStorageType', 'SecondaryStorageType',
    'GPU_company', 'GPU_model'
]

label_encoders = {}
```

```
for col in categorical_columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
    label_encoders[col] = le  # Store encoder for potential inverse transform
```

In [13]:
```
X.head()
```

Out[13]:

	Company	Product	TypeName	Inches	Ram	OS	Weight	Screen	ScreenW	ScreenH	...	RetinaDisplay	CPU_company	CPU_
0	1	300	4	13.3	8	8	1.37	3	2560	1600	...	1	1	
1	1	301	4	13.3	8	8	1.34	3	1440	900	...	0	1	
2	7	50	3	15.6	8	4	1.86	1	1920	1080	...	0	1	
3	1	300	4	15.4	16	8	1.83	3	2880	1800	...	1	1	
4	1	300	4	13.3	8	8	1.37	3	2560	1600	...	1	1	

5 rows × 22 columns

In [15]:
```
y
```

Out[15]:
```
0       1339.69
1        898.94
2        575.00
3       2537.45
4       1803.60
         ...
1270     638.00
1271    1499.00
1272     229.00
1273     764.00
1274     369.00
Name: Price_euros, Length: 1275, dtype: float64
```

In [17]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[17]: ((1020, 22), (255, 22), (1020,), (255,))

In [19]:
```
from sklearn.linear_model import LinearRegression
le=LinearRegression()
le.fit(X_train,y_train)
```

Out[19]:
```
LinearRegression()
```

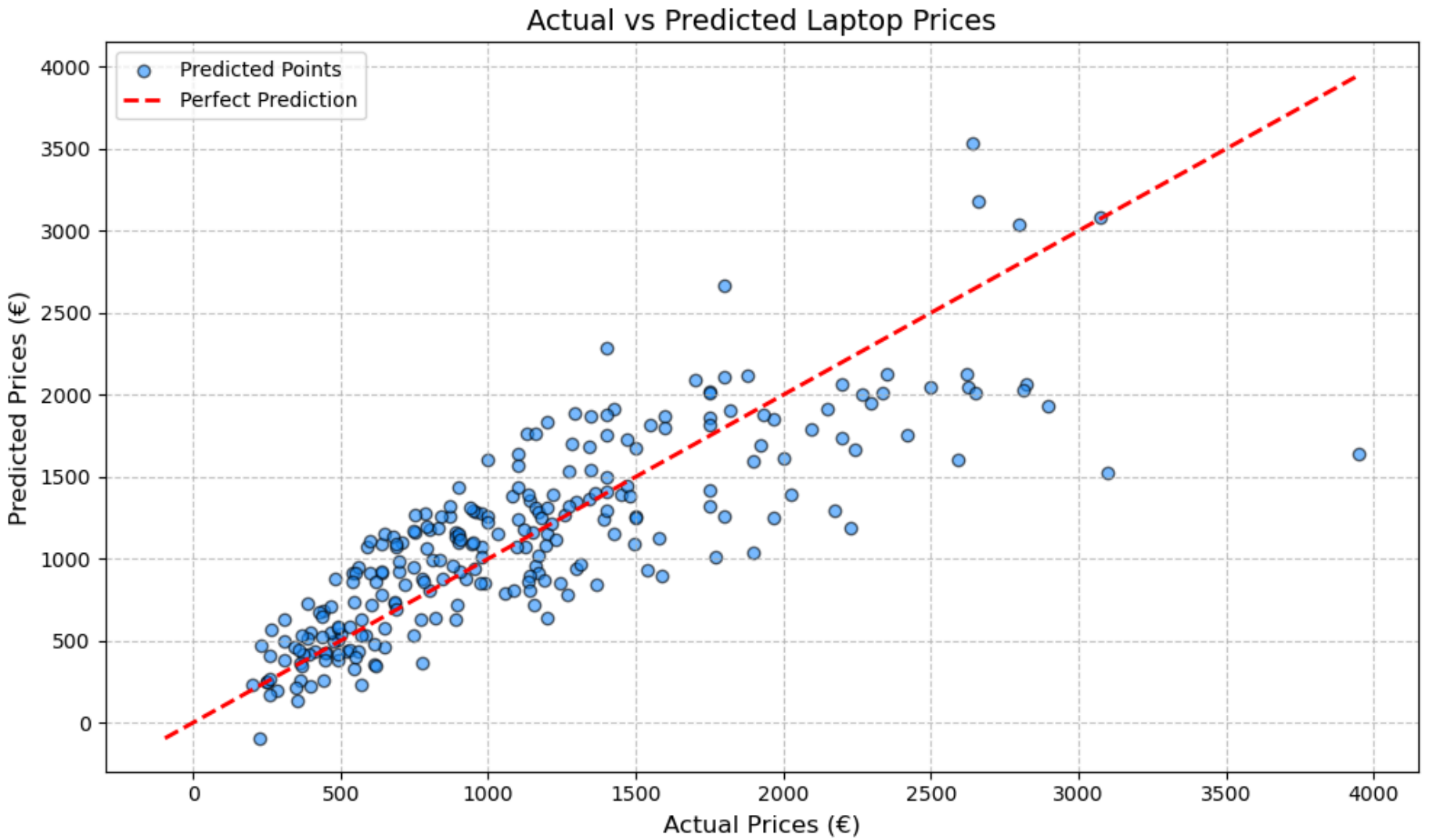
In [21]:
```
y_pred=le.predict(X_test)
```

In [23]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
mae=mean_absolute_error(y_pred,y_test)
mse=mean_squared_error(y_pred,y_test)
r2_score=r2_score(y_pred,y_test)
print('The mean absolute error is:',mae)
print('The mean squared error is:',mse)
print('The r2 score is:',r2_score)
```

```
The mean absolute error is: 281.79787366099055
The mean squared error is: 147774.93208603014
The r2 score is: 0.5704532857865052
```

In [25]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(10, 6))

# Scatter plot of actual vs predicted prices
plt.scatter(y_test, y_pred, color='dodgerblue', alpha=0.6, edgecolors='black', label='Predicted Points')

# Reference line for perfect prediction
min_val = min(min(y_test), min(y_pred))
max_val = max(max(y_test), max(y_pred))
plt.plot([min_val, max_val], [min_val, max_val], color='red', linestyle='--', linewidth=2, label='Perfect Predictio

# Add grid, labels, and title
plt.grid(True, linestyle='--', alpha=0.7)
```

```
plt.xlabel("Actual Prices (€)", fontsize=12)
plt.ylabel("Predicted Prices (€)", fontsize=12)
plt.title("Actual vs Predicted Laptop Prices", fontsize=14)
plt.legend()
plt.tight_layout()
plt.show()
```



In [27]:
```
from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor()
```

In [29]:
```
rfr.fit(X_train,y_train)
```

Out[29]:
```
RandomForestRegressor
RandomForestRegressor()
```

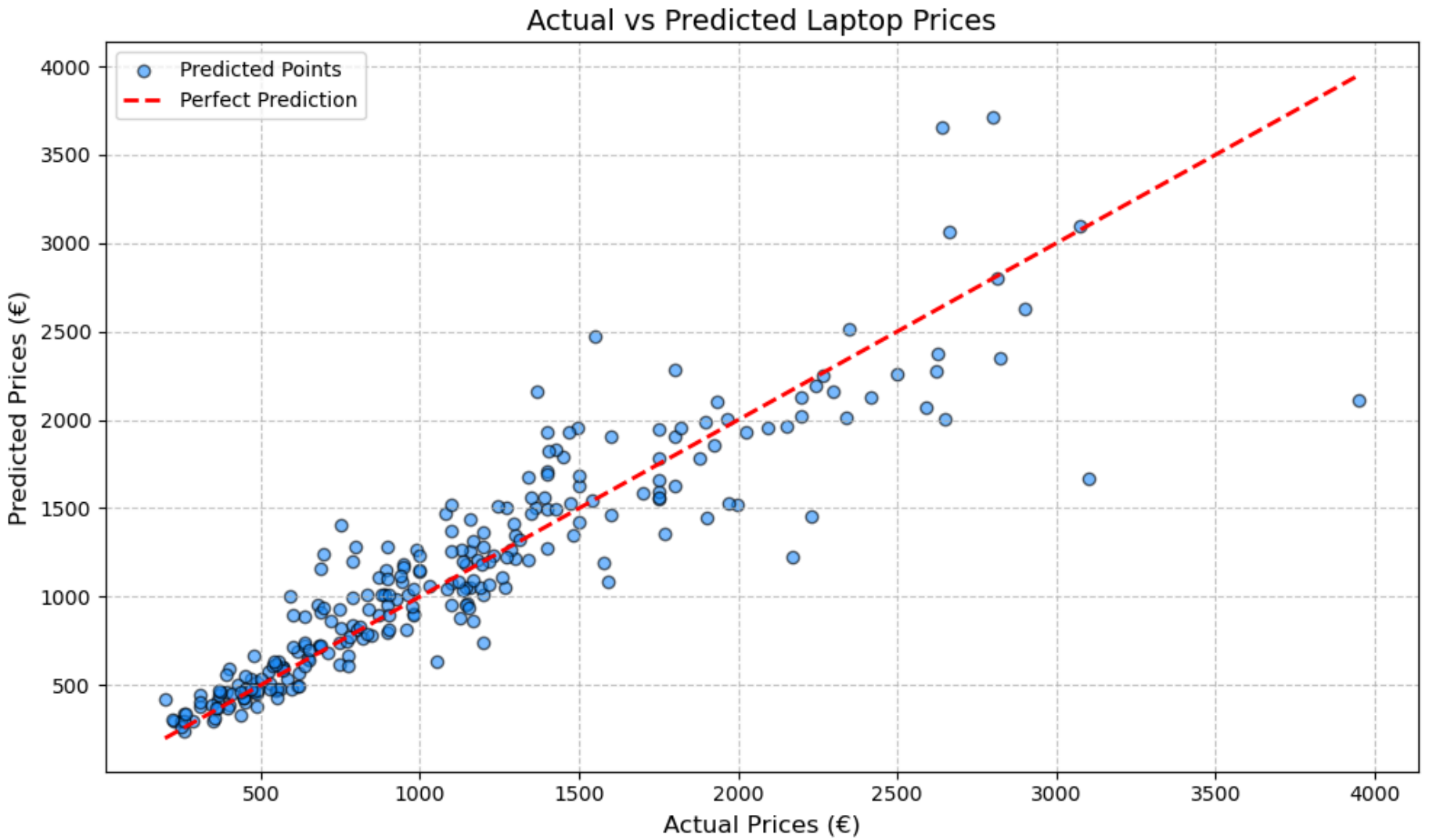
In [31]:
```
y_pred1=rfr.predict(X_test)
```

In [33]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
mae1=mean_absolute_error(y_pred1,y_test)
mse1=mean_squared_error(y_pred1,y_test)
r2_score1=r2_score(y_pred1,y_test)
print('The mean absolute error is:',mae1)
print('The mean squared error is:',mse1)
print('The r2 score is:',r2_score1)
```

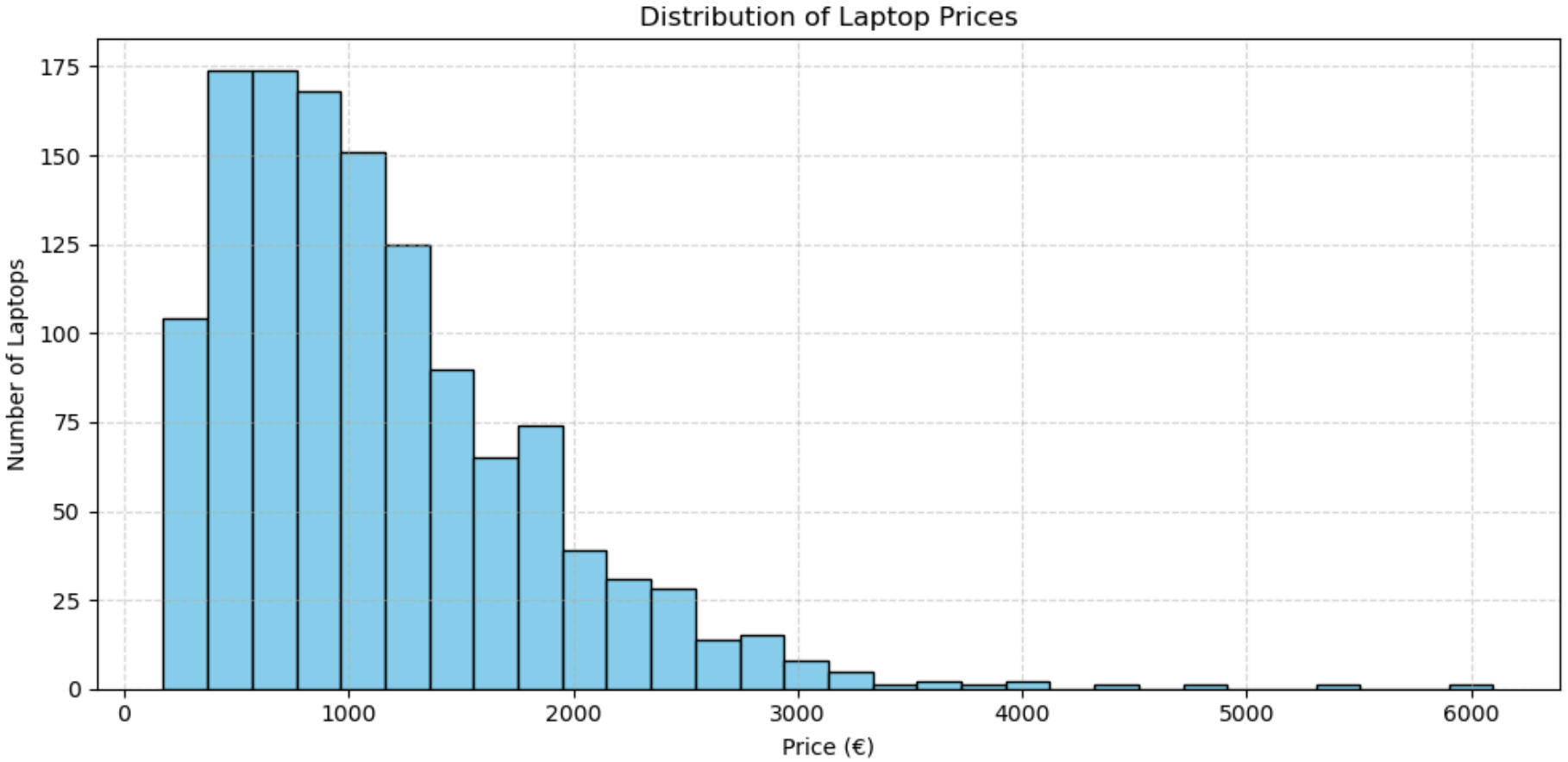
```
The mean absolute error is: 178.10724626143792
The mean squared error is: 80139.10304219375
The r2 score is: 0.7991965927408159
```

In [35]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(10, 6))

# Scatter plot of actual vs predicted prices
plt.scatter(y_test, y_pred1, color='dodgerblue', alpha=0.6, edgecolors='black', label='Predicted Points')

# Reference line for perfect prediction
min_val = min(min(y_test), min(y_pred1))
max_val = max(max(y_test), max(y_pred1))
plt.plot([min_val, max_val], [min_val, max_val], color='red', linestyle='--', linewidth=2, label='Perfect Predictio

# Add grid, labels, and title
plt.grid(True, linestyle='--', alpha=0.7)
```

```
plt.xlabel("Actual Prices (€)", fontsize=12)
plt.ylabel("Predicted Prices (€)", fontsize=12)
plt.title("Actual vs Predicted Laptop Prices", fontsize=14)
plt.legend()
plt.tight_layout()
plt.show()
```

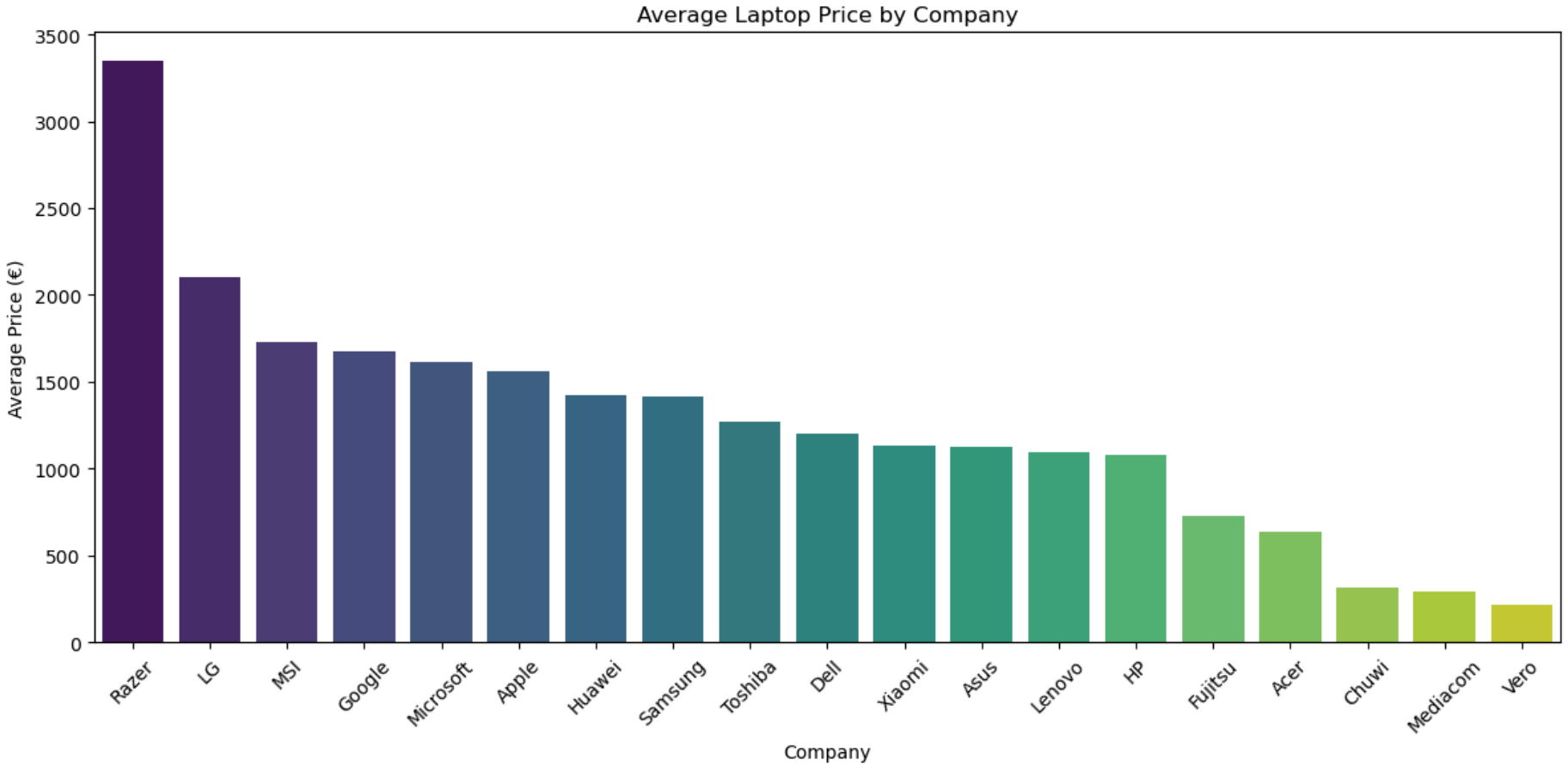


In [37]:
```
#Distribution of Laptop Prices
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
plt.hist(df['Price_euros'], bins=30, color='skyblue', edgecolor='black')
plt.title('Distribution of Laptop Prices')
plt.xlabel('Price (€)')
plt.ylabel('Number of Laptops')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



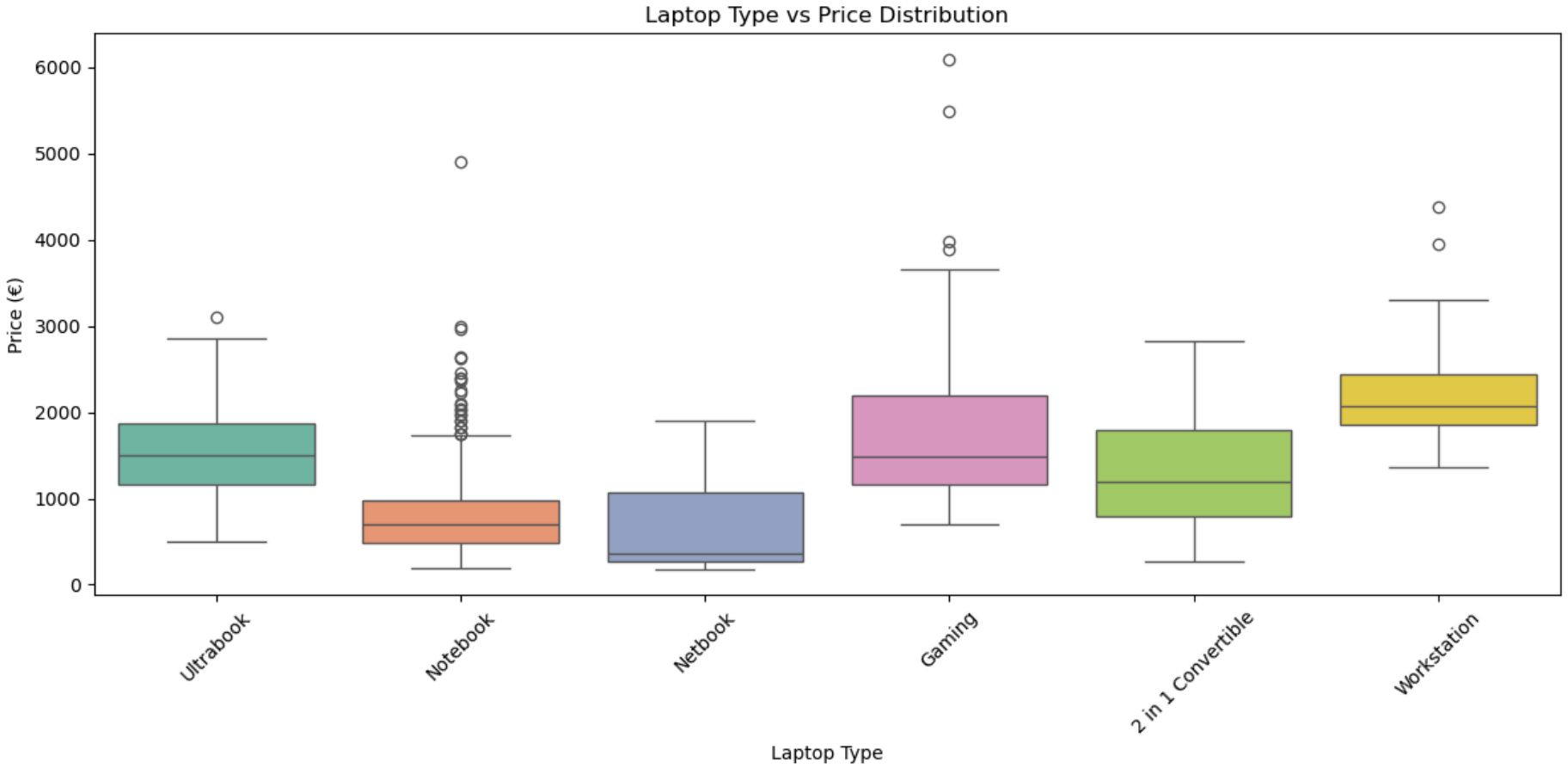
In [39]:
```
#Average Price of Laptop by company
import seaborn as sns

plt.figure(figsize=(12, 6))
avg_price_by_company = df.groupby('Company')['Price_euros'].mean().sort_values(ascending=False)
sns.barplot(x=avg_price_by_company.index, y=avg_price_by_company.values, palette='viridis')
plt.xticks(rotation=45)
plt.title('Average Laptop Price by Company')
plt.ylabel('Average Price (€)')
plt.xlabel('Company')
plt.tight_layout()
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_6688/629283051.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=avg_price_by_company.index, y=avg_price_by_company.values, palette='viridis')
```



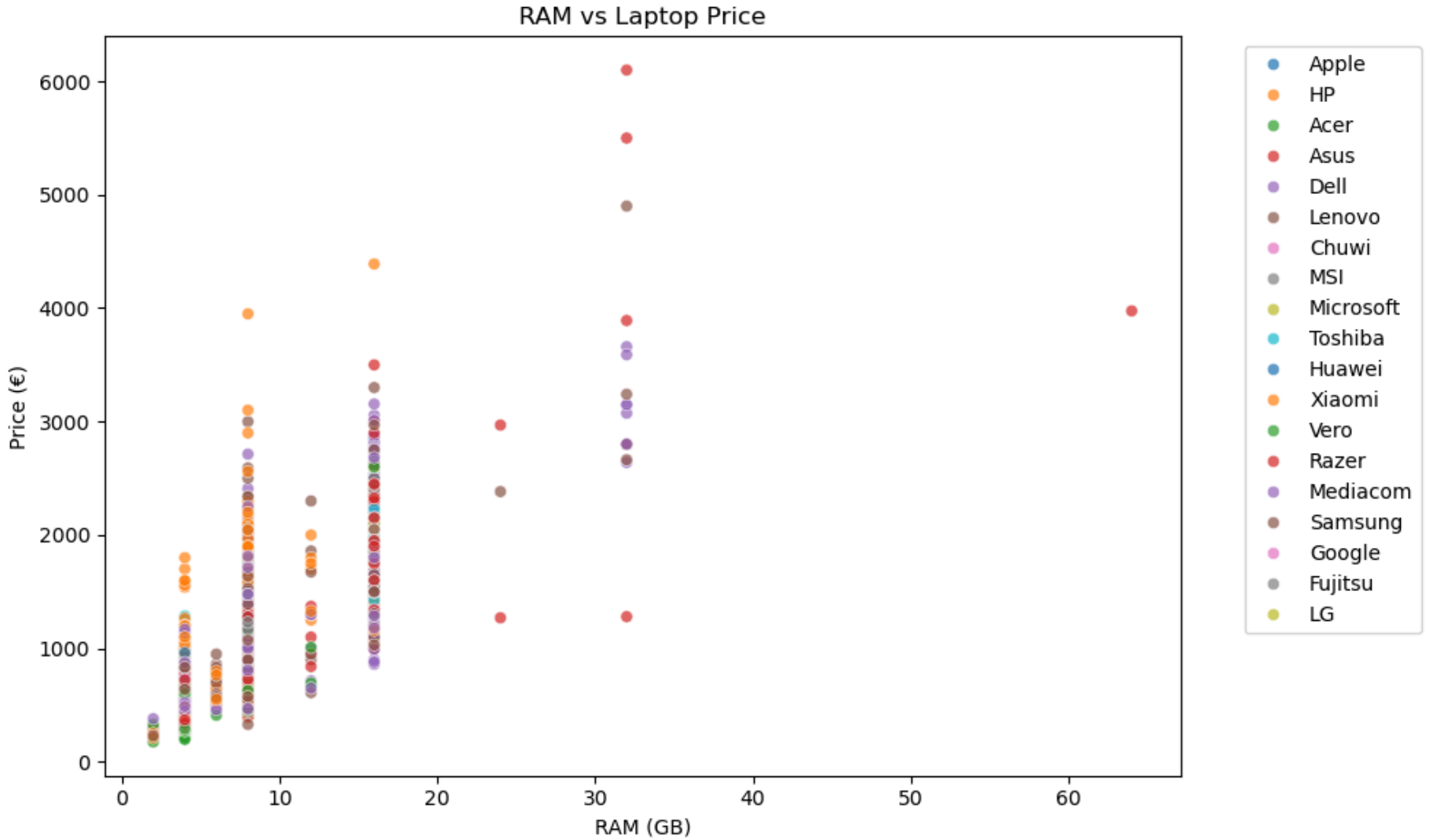
In [37]:
```
#Laptop price Vs Price Distribution
plt.figure(figsize=(12, 6))
sns.boxplot(x='TypeName', y='Price_euros', data=df, palette='Set2')
plt.xticks(rotation=45)
plt.title('Laptop Type vs Price Distribution')
plt.xlabel('Laptop Type')
plt.ylabel('Price (€)')
plt.tight_layout()
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_6036/859585806.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='TypeName', y='Price_euros', data=df, palette='Set2')
```



In [41]:
```
#RAM Vs Price
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Ram', y='Price_euros', data=df, hue='Company', palette='tab10', alpha=0.7)
plt.title('RAM vs Laptop Price')
plt.xlabel('RAM (GB)')
plt.ylabel('Price (€)')
```

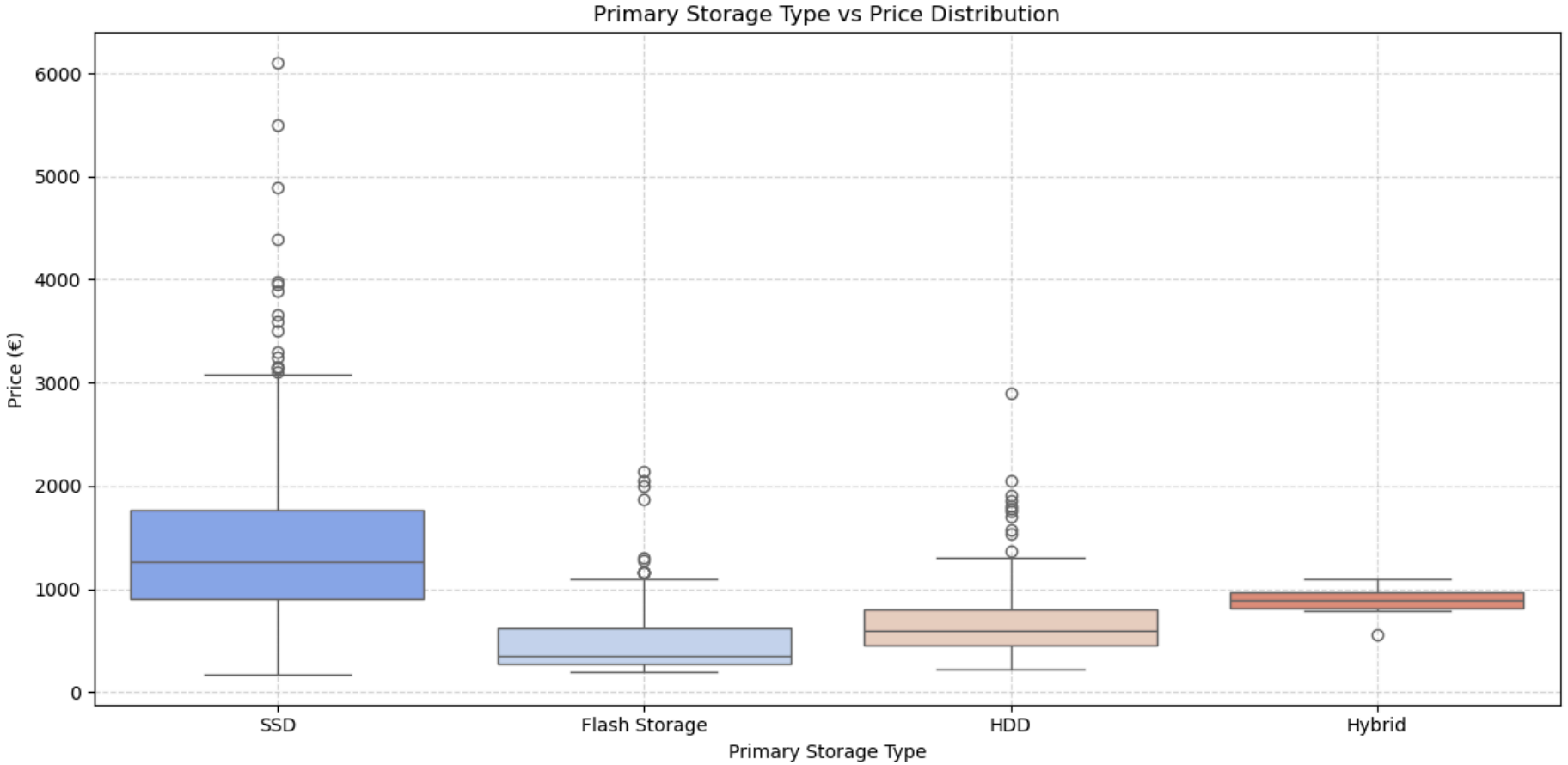
```
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



In [77]:
```
#Storage type vs Price
plt.figure(figsize=(12, 6))
sns.boxplot(x='PrimaryStorageType', y='Price_euros', data=df, palette='coolwarm')
plt.title('Primary Storage Type vs Price Distribution')
```

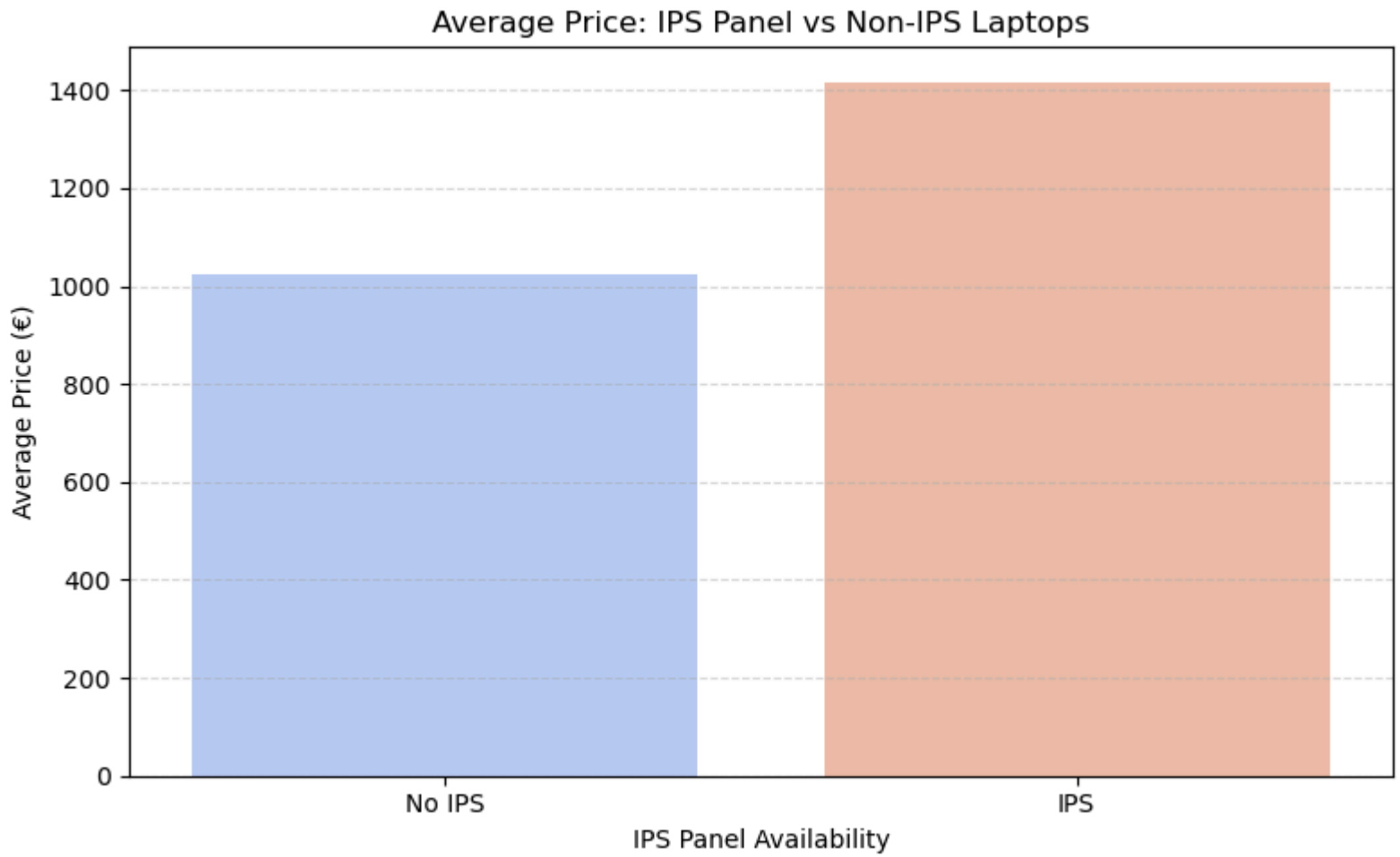
```
plt.xlabel('Primary Storage Type')
plt.ylabel('Price (€)')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_1258/609379220.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='PrimaryStorageType', y='Price_euros', data=df, palette='coolwarm')
```



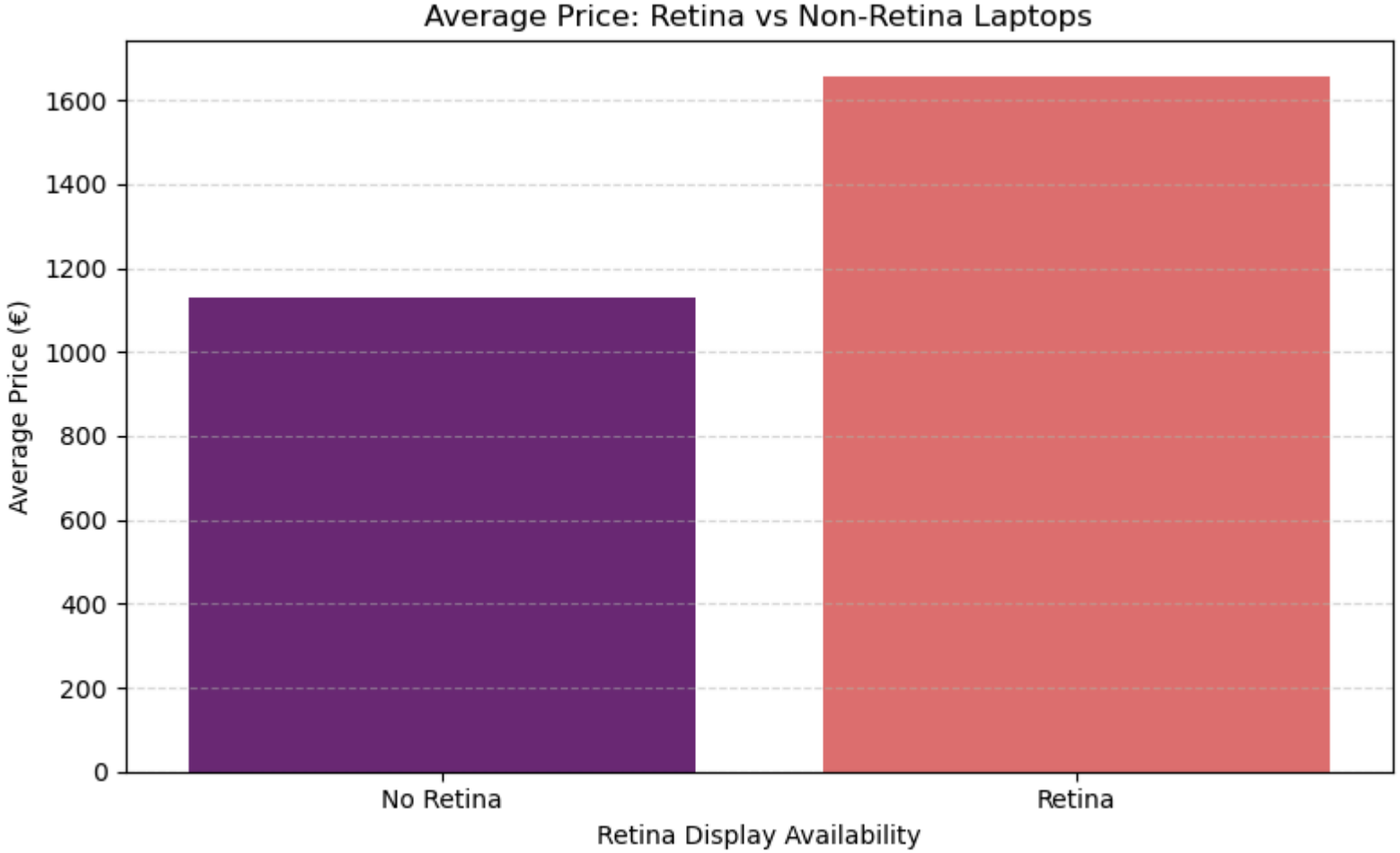
In [43]:
```
plt.figure(figsize=(8, 5))
ips_avg_price = df.groupby('IPSpanel')['Price_euros'].mean()
sns.barplot(x=ips_avg_price.index, y=ips_avg_price.values, palette='coolwarm')
plt.xticks([0, 1], ['No IPS', 'IPS'])
plt.title('Average Price: IPS Panel vs Non-IPS Laptops')
plt.ylabel('Average Price (€)')
plt.xlabel('IPS Panel Availability')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_6688/1794520785.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=ips_avg_price.index, y=ips_avg_price.values, palette='coolwarm')
```



In [83]:
```
plt.figure(figsize=(8, 5))
retina_avg_price = df.groupby('RetinaDisplay')['Price_euros'].mean()
sns.barplot(x=retina_avg_price.index, y=retina_avg_price.values, palette='magma')
plt.xticks([0, 1], ['No Retina', 'Retina'])
plt.title('Average Price: Retina vs Non-Retina Laptops')
plt.ylabel('Average Price (€)')
plt.xlabel('Retina Display Availability')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_1258/3488182014.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=retina_avg_price.index, y=retina_avg_price.values, palette='magma')
```



In [45]:
```
plt.figure(figsize=(10, 6))
sns.violinplot(x='RetinaDisplay', y='Price_euros', data=df, palette='magma', inner="quartile")
plt.xticks([0, 1], ['No Retina', 'Retina'])
plt.title('Price Distribution: Retina vs Non-Retina Laptops')
```

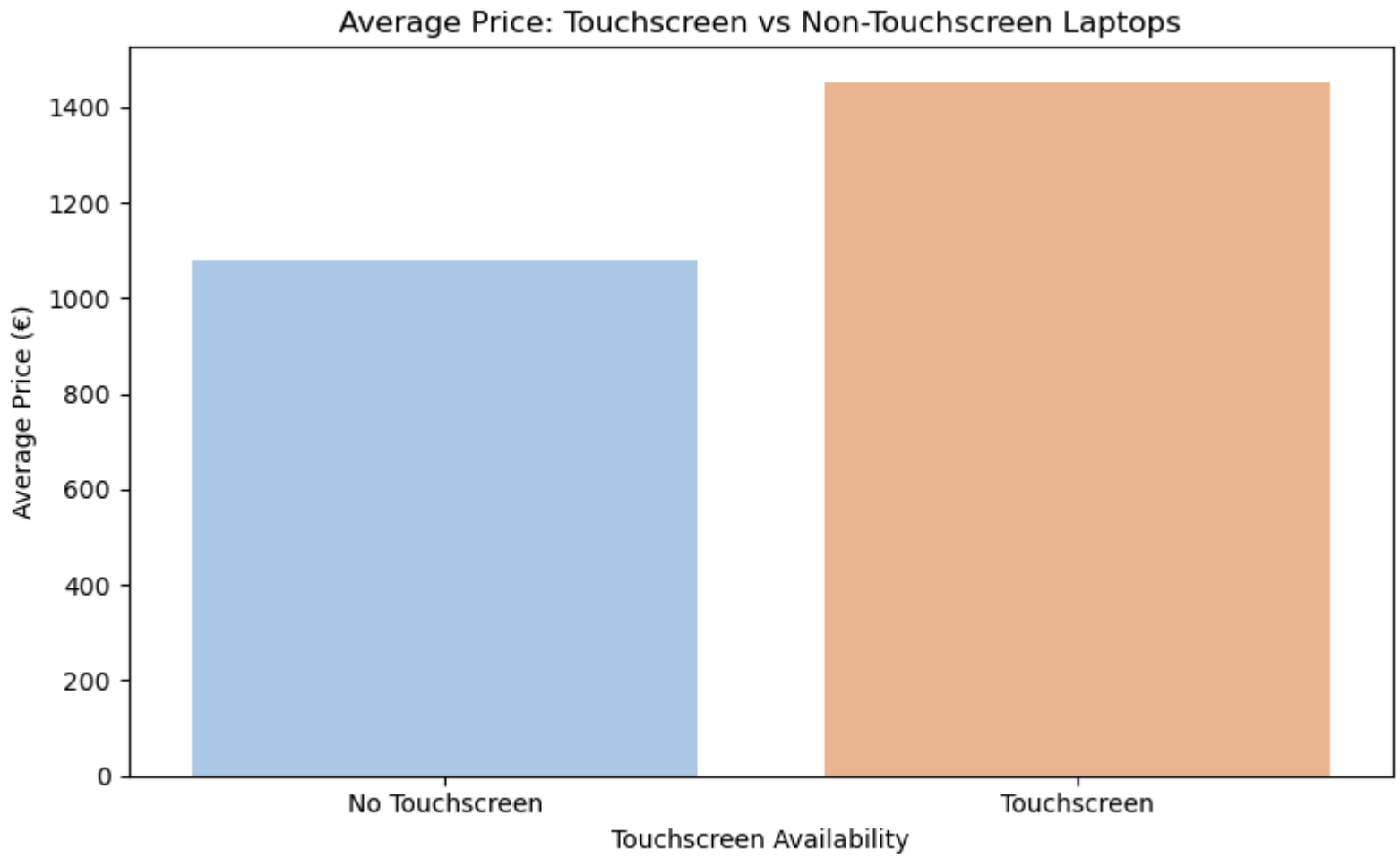
```
plt.xlabel('Retina Display Availability')
plt.ylabel('Price (€)')
plt.tight_layout()
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_6688/1621181651.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(x='RetinaDisplay', y='Price_euros', data=df, palette='magma', inner="quartile")
```

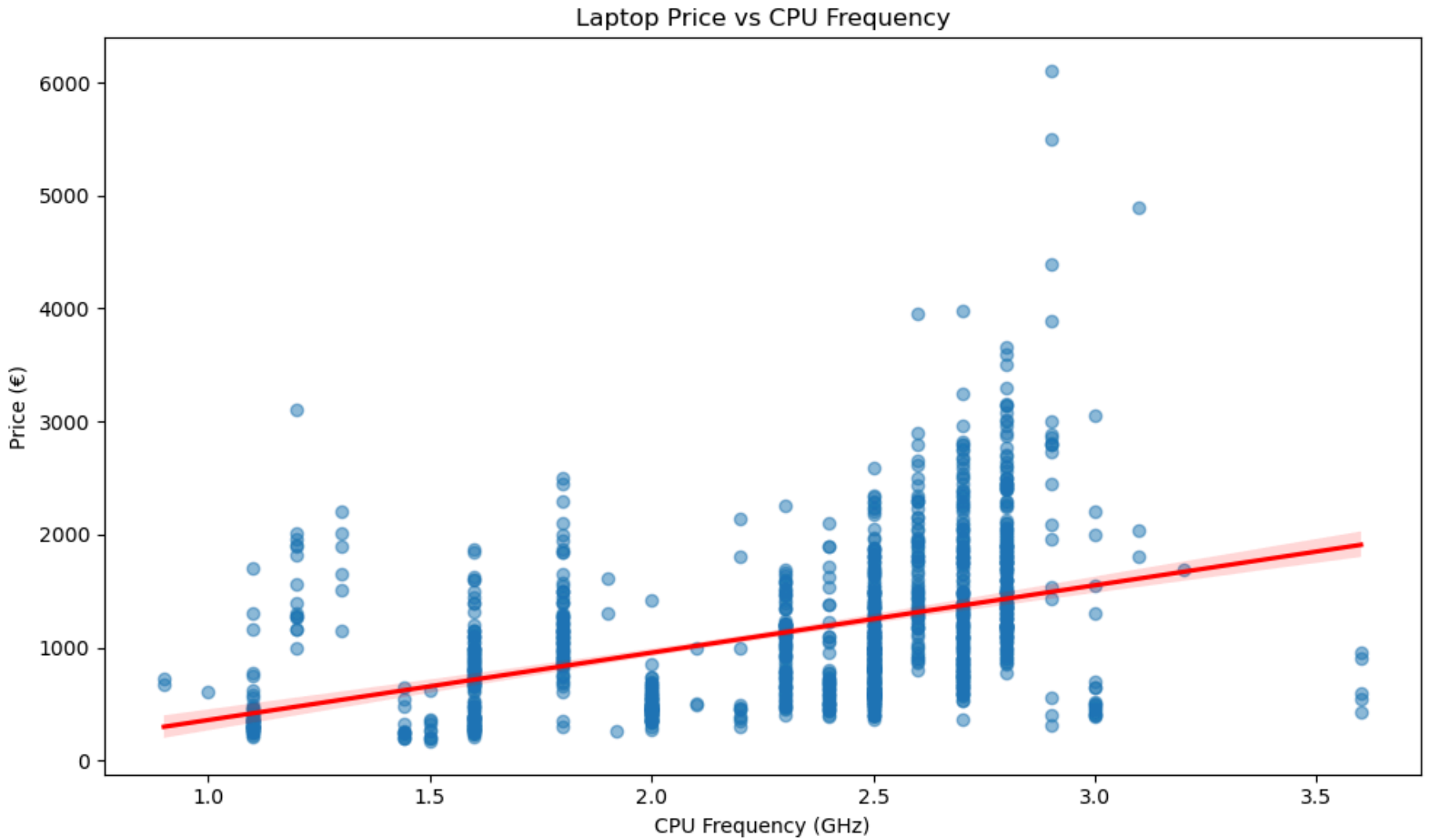
In [87]:
```
#Touchscreen vs Price
plt.figure(figsize=(8, 5))
touchscreen_counts = df.groupby('Touchscreen')['Price_euros'].mean()
sns.barplot(x=touchscreen_counts.index, y=touchscreen_counts.values, palette='pastel')
plt.xticks([0, 1], ['No Touchscreen', 'Touchscreen'])
plt.title('Average Price: Touchscreen vs Non-Touchscreen Laptops')
plt.ylabel('Average Price (€)')
```

```
plt.xlabel('Touchscreen Availability')
plt.tight_layout()
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_1258/3123336635.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=touchscreen_counts.index, y=touchscreen_counts.values, palette='pastel')
```

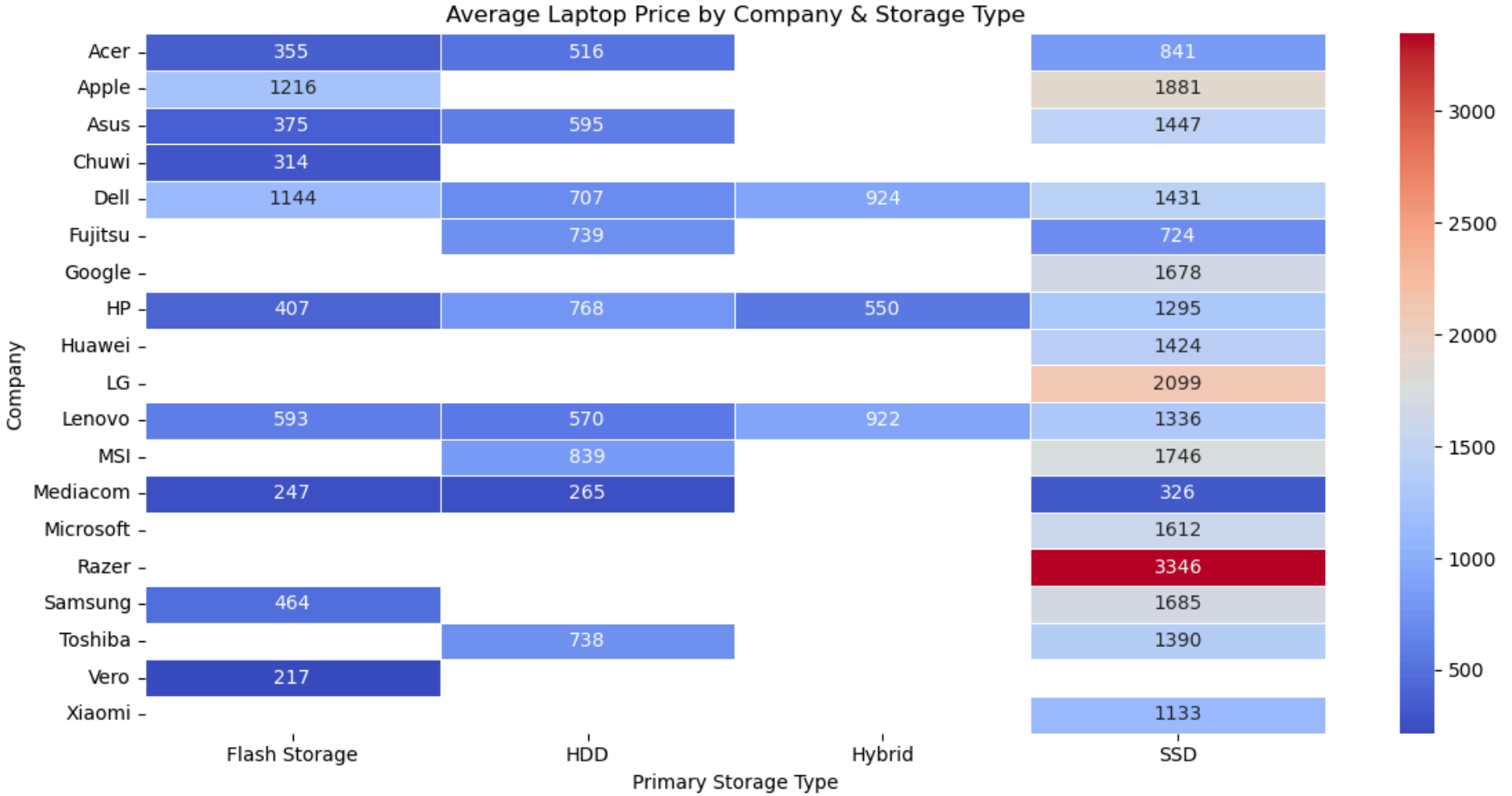


In [47]:
```
#Price vs CPU Frequency (Regression Line)
plt.figure(figsize=(10, 6))
sns.regplot(x='CPU_freq', y='Price_euros', data=df, scatter_kws={'alpha':0.5}, line_kws={'color':'red'})
plt.title('Laptop Price vs CPU Frequency')
plt.xlabel('CPU Frequency (GHz)')
plt.ylabel('Price (€)')
plt.tight_layout()
plt.show()
```



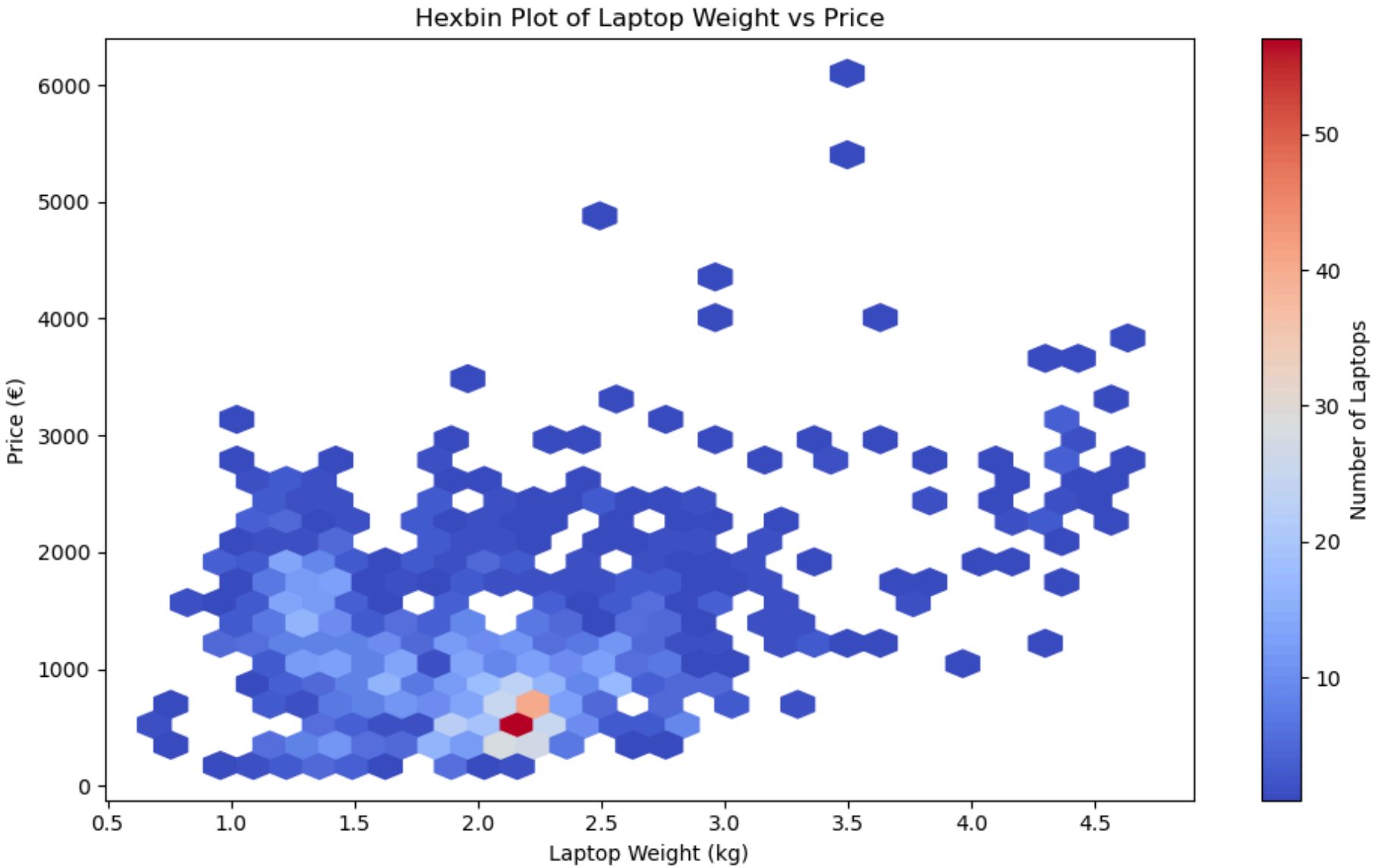
In [49]:
```
#Heatmap of Categorical Features vs Price

plt.figure(figsize=(12, 6))
pivot_table = df.pivot_table(values='Price_euros', index='Company', columns='PrimaryStorageType', aggfunc='mean')
sns.heatmap(pivot_table, cmap='coolwarm', annot=True, fmt='.0f', linewidths=0.5)
plt.title('Average Laptop Price by Company & Storage Type')
plt.xlabel('Primary Storage Type')
```

```
plt.ylabel('Company')
plt.tight_layout()
plt.show()
```

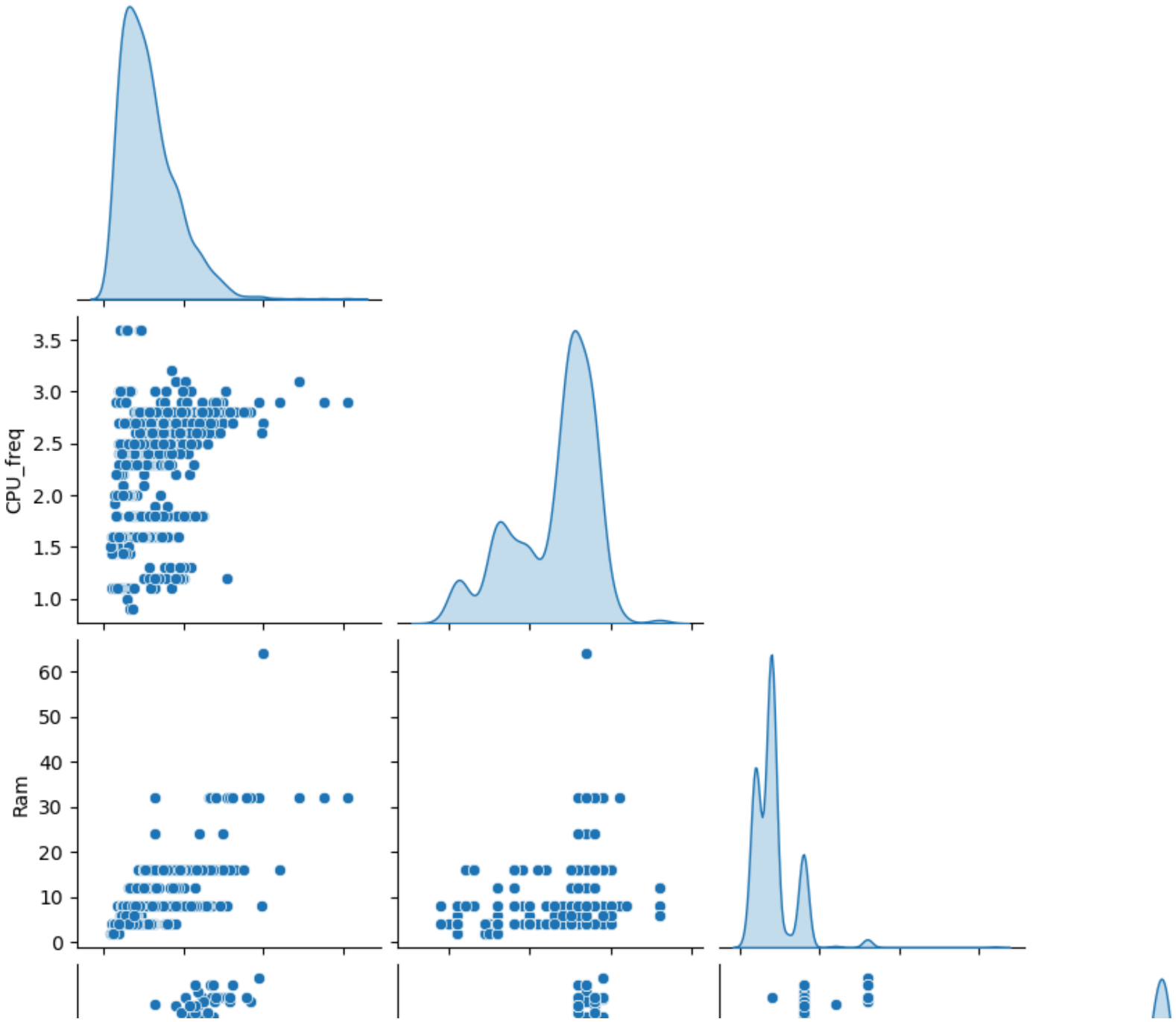


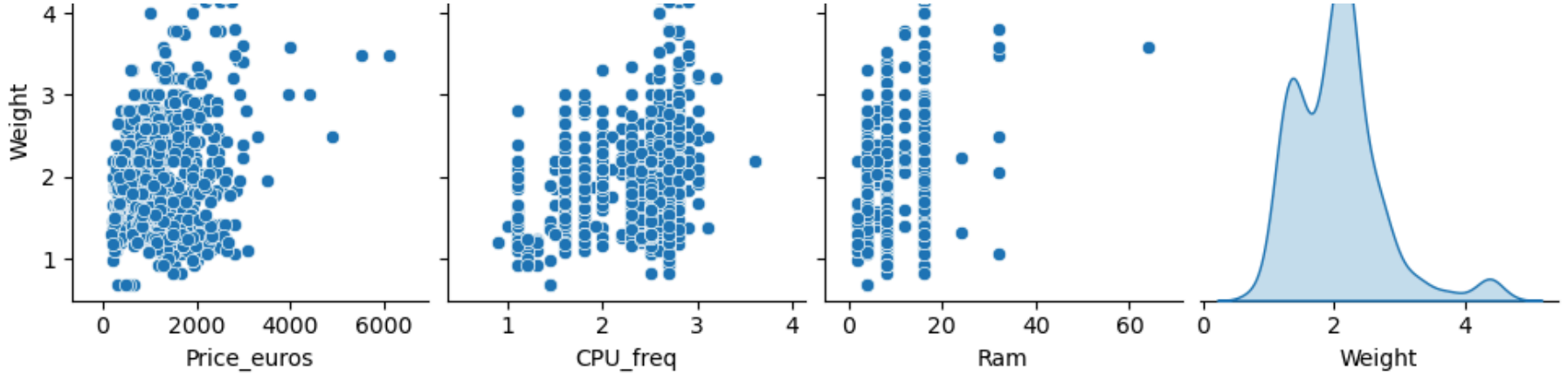
In [51]:
```
plt.figure(figsize=(10, 6))
plt.hexbin(df['Weight'], df['Price_euros'], gridsize=30, cmap='coolwarm', mincnt=1)
plt.colorbar(label='Number of Laptops')
plt.xlabel('Laptop Weight (kg)')
plt.ylabel('Price (€)')
plt.title('Hexbin Plot of Laptop Weight vs Price')
```

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



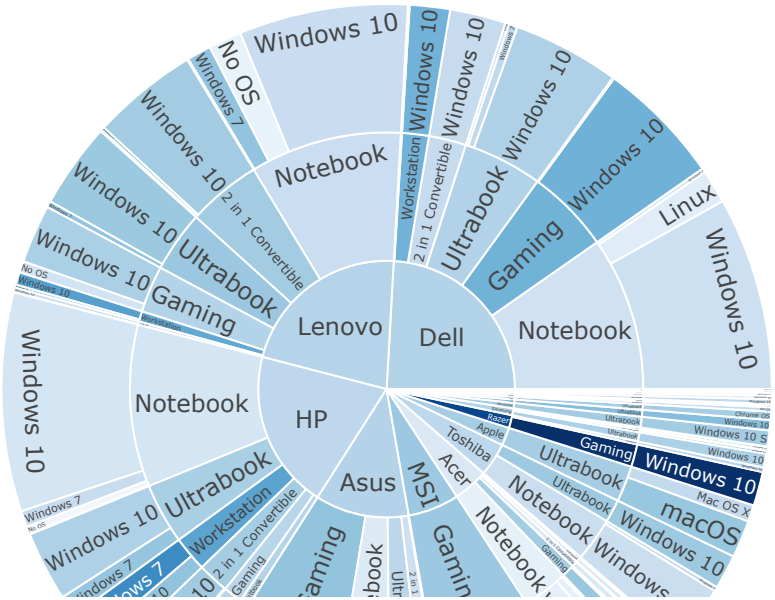
In [53]:
```
sns.pairplot(df[['Price_euros', 'CPU_freq', 'Ram', 'Weight']], diag_kind='kde', markers='o', corner=True)
plt.show()
```


CPU_freq
Ram
3.5
3.0
2.5
2.0
1.5
1.0
60
50
40
30
20
10
0



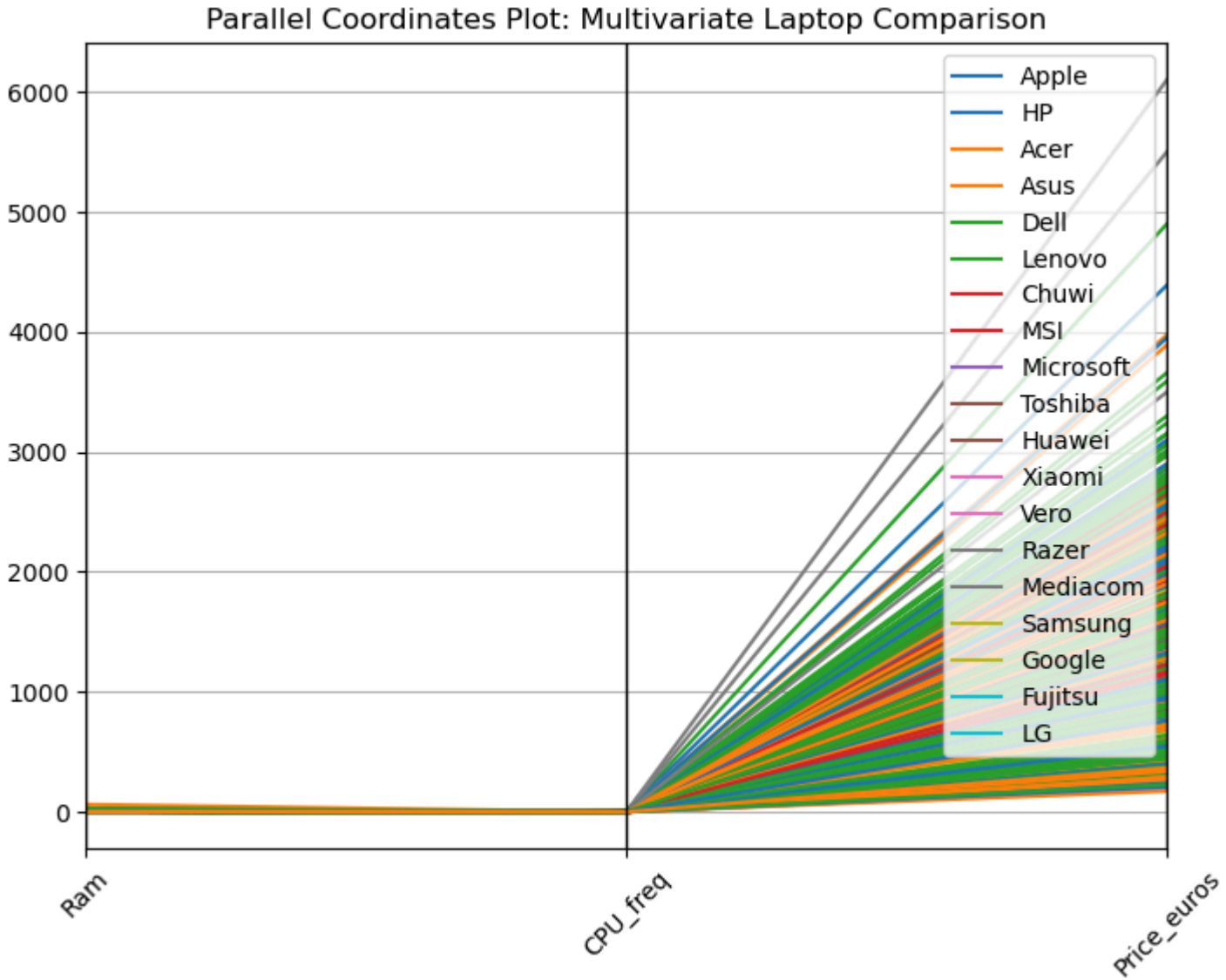
In [55]:
```
import plotly.express as px

fig = px.sunburst(df, path=['Company', 'TypeName', 'OS'], values='Price_euros', color='Price_euros', color_continuo
fig.update_layout(title='Sunburst Chart: Brand, Type & OS Market Share')
fig.show()
```

Sunburst Chart: Brand, Type & OS Market Share

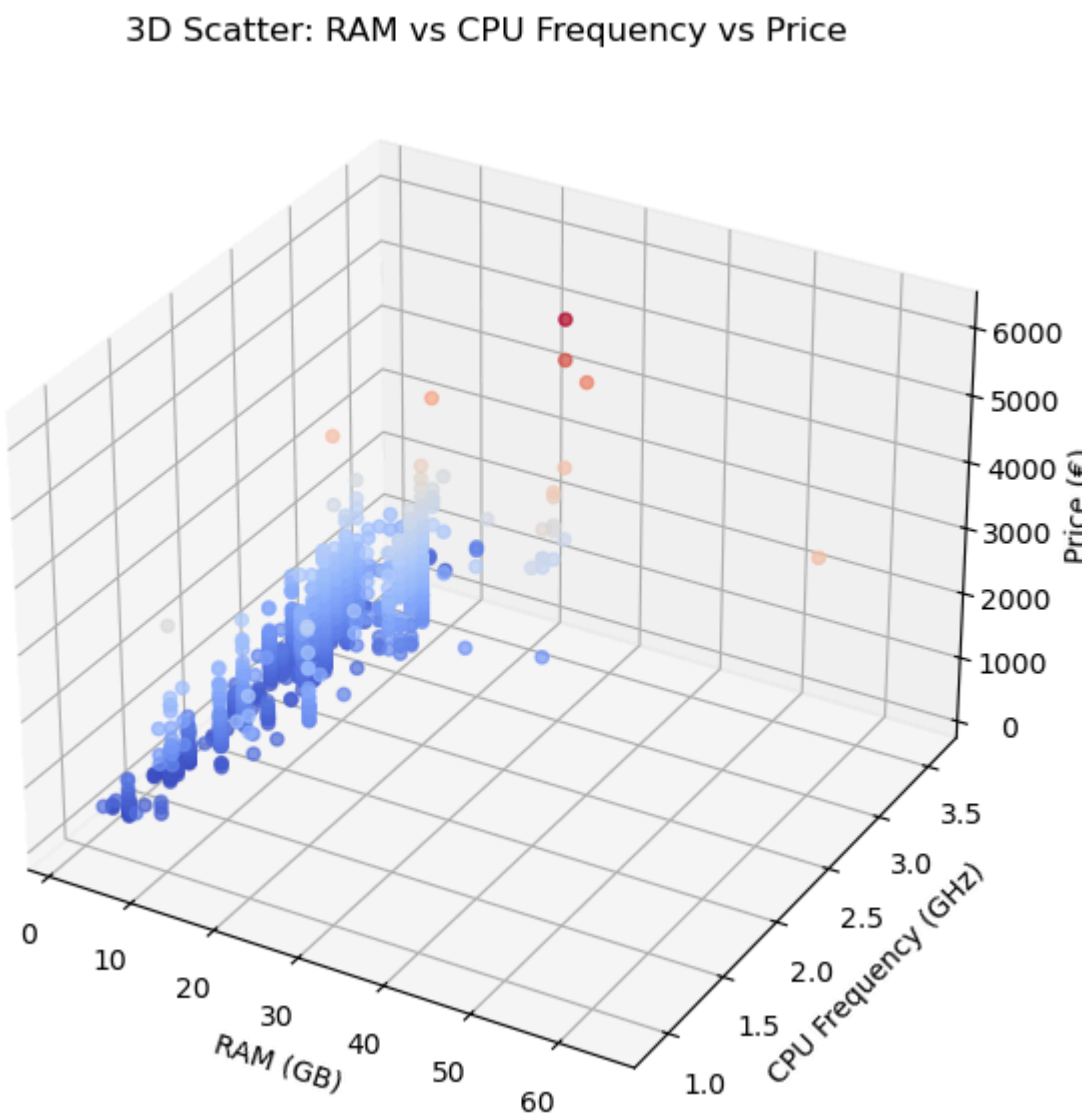


In [57]:
```
from pandas.plotting import parallel_coordinates

plt.figure(figsize=(8, 6))
parallel_coordinates(df[['Company', 'Ram', 'CPU_freq', 'Price_euros']], 'Company', colormap=plt.cm.get_cmap('tab10'
plt.xticks(rotation=45)
plt.title('Parallel Coordinates Plot: Multivariate Laptop Comparison')
plt.show()
```

```
/var/folders/pm/cnlmdnjj5g1ct4r7rrx83vnr0000gn/T/ipykernel_6688/3621600638.py:4: MatplotlibDeprecationWarning:

The get_cmap function was deprecated in Matplotlib 3.7 and will be removed two minor releases later. Use ``matplotli
b.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)`` instead.
```



In [59]:
```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df['Ram'], df['CPU_freq'], df['Price_euros'], c=df['Price_euros'], cmap='coolwarm', alpha=0.7)

ax.set_xlabel('RAM (GB)')
ax.set_ylabel('CPU Frequency (GHz)')
ax.set_zlabel('Price (€)')
ax.set_title('3D Scatter: RAM vs CPU Frequency vs Price')

plt.show()
```


3D Scatter: RAM vs CPU Frequency vs Price
RAM (GB)
CPU Frequency (GHz)
Price (€)
0
10
20
30
40
50
60
1.0
1.5
2.0
2.5
3.0
3.5
0
1000
2000
3000
4000
5000
6000

In [ ]: