Laptop Specs by Rating and Price - Exploratory and Inferential Data Analysis

Dataset: https://www.kaggle.com/datasets/kuchhbhi/latest-laptop-price-list?select=Cleaned_Laptop_data.csv

Chris Heimbuch: https://github.com/chrisheimbuch

laptopIntro.jpg

Image link

Overview

This dataset is scraped from flipkart.com, and was gathered using an automated chrome web extension tool called Instant Data Scrapped.

The columns that this dataset includes are:

- **brand** - The name of the brand of laptop.
- **model** - The model name/number associated with the laptop.
- **processor_brand** - The brand of processor in the laptop such as AMD or Intel.
- **processor_name** - The model name/number associated with a processor.
- **processor_gnrtn** - The generational version of a processor.
- **ram_gb** - How much RAM is in a particular laptop model.
- **ram_type** - What type of RAM is in a laptop, for example DDR3 or DDR4.
- **ssd** - Indicating if a Solid State Drive storage is in a laptop, and how much in Gigabytes.
- **hdd** - Indicating if a Hard Disk Drive storage is in a laptop, and how much in Gigabytes.
- **os** - The operating system on the laptop.
- **os_bit** - The operating system bit that's on a laptop (such as 32 or 64 bit Windows).
- **graphic_card_gb** - How much video random access memory a graphic card has.
- **weight** - Weight of laptop.
- **display_size** - How large a screen is for a respective laptop.
- **warranty** - How many years of active warranty a particular laptop has.
- **touchscreen** - Indicator if a laptop has a touch screen.
- **msoffice** - Indicator if a laptop comes with Microsoft Office.
- **latest_price** - The latest price of the laptop in INR.
- **old_price** - Old price of laptop if it was in the original dataset.
- **discount** - Indicator if there is a difference between the old price and latest price.
- **star_rating** - The overall rating of the laptop, from 1 star to 5 stars.
- **ratings** - The count of ratings overall for a laptop.
- **reviews** - Count of reviews for a laptop.

For my analysis, Section 1 comprised of getting familiar with my data using some common techniques, such as inspecting the shape, getting the head of my dataframe by invoking the .head() method, checking for null values, using the describe technique and the .info() technique to understand data types. Next, I cleaned my data and added some new columns added some additional columns for USD pricing instead of INR pricing. I created a new dataframe and focused on columns I was interested in working with. Section 2 comprised of Descriptive Questions and answers via beautified graphical representations. Section 3 comprised of inferential analysis and hypothesis testing. Finally, Section 4 is an analysis and conclusion of findings. I designed a powerpoint presentation catered to technology companies such as Dell, HP, Acer, Asus and many more on laptop findings. This was a fun project and hope you enjoy!

In [5]:
```
#Import libraries for analysis

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random
from matplotlib.lines import Line2D
from scipy import stats
from scipy.stats import mannwhitneyu
from scipy.stats import kruskal
from scipy.stats import ttest_ind
from scipy.stats import f_oneway
from itertools import combinations
import warnings
warnings.filterwarnings('ignore')
```

In [6]:
```
#Load in dataset and inspect shape.
df = pd.read_csv("Cleaned_Laptop_data.csv")
df.shape
```

Out[6]:
```
(896, 23)
```

In [5]:
```
#Inspect all columns
df.columns
```

Out[5]:
```
Index(['brand', 'model', 'processor_brand', 'processor_name',
       'processor_gnrtn', 'ram_gb', 'ram_type', 'ssd', 'hdd', 'os', 'os_bit',
       'graphic_card_gb', 'weight', 'display_size', 'warranty', 'Touchscreen',
       'msoffice', 'latest_price', 'old_price', 'discount', 'star_rating',
       'ratings', 'reviews'],
      dtype='object')
```

In [6]:
```
#Inspect head
df.head()
```

Out[6]:

	brand	model	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd
0	Lenovo	A6-9225	AMD	A6-9225 Processor	10th	4 GB GB	DDR4	0 GB	1024 GB

	brand	model	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	
1	Lenovo	Ideapad	AMD	APU Dual	10th	4 GB GB	DDR4	0 GB	512 GB	V
2	Avita	PURA	AMD	APU Dual	10th	4 GB GB	DDR4	128 GB	0 GB	V
3	Avita	PURA	AMD	APU Dual	10th	4 GB GB	DDR4	128 GB	0 GB	V
4	Avita	PURA	AMD	APU Dual	10th	4 GB GB	DDR4	256 GB	0 GB	V

5 rows × 23 columns

In [7]:
```
#Inspect data types and potential null values.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 896 entries, 0 to 895
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   brand            896 non-null    object 
 1   model            896 non-null    object 
 2   processor_brand  896 non-null    object 
 3   processor_name   896 non-null    object 
 4   processor_gnrtn  896 non-null    object 
 5   ram_gb           896 non-null    object 
 6   ram_type         896 non-null    object 
 7   ssd              896 non-null    object 
 8   hdd              896 non-null    object 
 9   os               896 non-null    object 
 10  os_bit           896 non-null    object 
 11  graphic_card_gb  896 non-null    int64  
 12  weight           896 non-null    object 
 13  display_size     896 non-null    object 
 14  warranty         896 non-null    int64  
 15  Touchscreen      896 non-null    object 
 16  msoffice         896 non-null    object 
 17  latest_price     896 non-null    int64  
 18  old_price        896 non-null    int64  
 19  discount         896 non-null    int64  
 20  star_rating      896 non-null    float64
 21  ratings          896 non-null    int64  
 22  reviews          896 non-null    int64  
dtypes: float64(1), int64(7), object(15)
memory usage: 161.1+ KB
```

In [8]:
```
#Check for null values.
df.isna().sum()
```

Out[8]:
```
brand              0
model              0
processor_brand    0
processor_name     0
processor_gnrtn    0
ram_gb             0
ram_type           0
ssd                0
hdd                0
```

```
os                0
os_bit            0
graphic_card_gb   0
weight            0
display_size      0
warranty          0
Touchscreen       0
msoffice          0
latest_price      0
old_price         0
discount          0
star_rating       0
ratings           0
reviews           0
dtype: int64
```

In [9]:
```
#Check out columns that have any meaningful numeric data.
df.describe()
```

Out[9]:

	graphic_card_gb	warranty	latest_price	old_price	discount	star_rating	ratings
count	896.000000	896.000000	896.000000	896.000000	896.000000	896.000000	896.000000
mean	1.198661	0.691964	76309.860491	88134.154018	18.527902	2.980469	367.391741
std	2.057454	0.606282	46613.354368	55719.645554	10.508486	1.965254	1106.309355
min	0.000000	0.000000	13990.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	45490.000000	54940.500000	11.000000	0.000000	0.000000
50%	0.000000	1.000000	63494.000000	78052.500000	19.000000	4.100000	19.000000
75%	2.000000	1.000000	89090.000000	111019.500000	26.000000	4.400000	179.500000
max	8.000000	3.000000	441990.000000	377798.000000	57.000000	5.000000	15279.000000

In [10]:
```
#Creating copy of DF to work with.
df_copy = df.copy()
```

In [11]:
```
#Clean the ram_gb column

gb_replacement = {
    '4 GB GB': "4 GB",
    "8 GB GB": "8 GB",
    "16 GB GB": "16 GB",
    "32 GB GB": "32 GB"
}

df_copy['ram_gb'] = df_copy['ram_gb'].replace(gb_replacement)
```

In [12]:
```
#Sanity Check
df_copy['ram_gb'].unique()
```

Out[12]:
```
array(['4 GB', '8 GB', '32 GB', '16 GB'], dtype=object)
```

In [13]:
```
#Inspect DF to see changes and other areas to clean/work on.
df_copy.head(30)
```

Out[13]:

	brand	model	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hd
0	Lenovo	A6-9225	AMD	A6-9225 Processor	10th	4 GB	DDR4	0 GB	102 G
1	Lenovo	Ideapad	AMD	APU Dual	10th	4 GB	DDR4	0 GB	51 G
2	Avita	PURA	AMD	APU Dual	10th	4 GB	DDR4	128 GB	0 G
3	Avita	PURA	AMD	APU Dual	10th	4 GB	DDR4	128 GB	0 G
4	Avita	PURA	AMD	APU Dual	10th	4 GB	DDR4	256 GB	0 G
5	Avita	PURA	AMD	APU Dual	10th	8 GB	DDR4	256 GB	0 G
6	HP	APU	AMD	APU Dual	10th	4 GB	DDR4	0 GB	102 G
7	Lenovo	APU	AMD	APU Dual	10th	4 GB	DDR4	0 GB	102 G
8	HP	Athlon	AMD	Athlon Dual	10th	32 GB	DDR4	32 GB	0 G
9	acer	Aspire	AMD	Athlon Dual	10th	4 GB	DDR4	256 GB	0 G
10	ASUS	ExpertBook	Intel	Core i3	10th	4 GB	DDR4	0 GB	102 G
11	Lenovo	Missing	Intel	Core i3	10th	4 GB	DDR4	0 GB	102 G
12	Lenovo	v15	Intel	Core i3	10th	4 GB	DDR4	0 GB	102 G
13	ASUS	ExpertBook	Intel	Core i5	10th	8 GB	DDR4	512 GB	0 G
14	ASUS	VivoBook	Intel	Celeron Dual	Missing	4 GB	DDR4	0 GB	51 G
15	ASUS	EeeBook	Intel	Celeron Dual	Missing	4 GB	DDR4	0 GB	51 G
16	ASUS	EeeBook	Intel	Celeron Dual	Missing	4 GB	DDR4	0 GB	51 G
17	ASUS	ExpertBook	Intel	Core i5	10th	8 GB	DDR4	0 GB	102 G
18	Lenovo	Missing	Intel	Core i5	10th	4 GB	DDR4	0 GB	102 G
19	acer	Aspire	AMD	Ryzen	10th	4 GB	DDR4	0 GB	51 G
20	acer	Nitro	AMD	Ryzen 5	10th	4 GB	DDR4	0 GB	51 G

	brand	model	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hd
21	acer	Nitro	AMD	Ryzen 5	10th	4 GB	DDR4	0 GB	51 G
22	acer	Nitro	AMD	Ryzen 5	10th	4 GB	DDR4	0 GB	51 G
23	Avita	Cosmos	Intel	Celeron Dual	Missing	4 GB	DDR4	0 GB	51 G
24	ASUS	EeeBook	Intel	Celeron Dual	Missing	4 GB	DDR4	0 GB	51 G
25	ASUS	ExpertBook	Intel	Core i3	11th	4 GB	DDR4	256 GB	0 G
26	HP	x360	Intel	Core i3	11th	8 GB	DDR4	256 GB	0 G
27	HP	x360	Intel	Core i5	11th	8 GB	DDR4	256 GB	0 G
28	Lenovo	IdeaPad	Intel	Celeron Dual	Missing	4 GB	DDR4	256 GB	0 G
29	HP	Celeron	Intel	Celeron Dual	Missing	8 GB	DDR4	256 GB	0 G

30 rows × 23 columns

In [14]:
```
#Add USD column for latest price and old price for USD
# https://www.xe.com/currencyconverter/ - source for converting as of 07/29/24

inrToUsd = 0.011939

df_copy['latest_price_usd'] = df_copy['latest_price'].map(lambda x: x * inrToUsd).round
df_copy['old_price_usd'] = df_copy['old_price'].map(lambda x: x * inrToUsd).round(2)
```

In [15]:
```
#Rename some columns for interpretability.

rename_columns = {
    "ssd": "solid_state_drive",
    "hdd": "hard_disk_drive",
    "os": "operating_system",
    "processor_gnrtn": "processor_generation",
    "Touchscreen": "touch_screen",
    "os_bit": "operating_system_bit",
    "reviews": "review_count"
}

df_copy.rename(columns=rename_columns, inplace=True)
```

In [16]:
```
#Sanity check
df_copy.columns
```

Out[16]:
```
Index(['brand', 'model', 'processor_brand', 'processor_name',
       'processor_generation', 'ram_gb', 'ram_type', 'solid_state_drive',
       'hard_disk_drive', 'operating_system', 'operating_system_bit',
       'graphic_card_gb', 'weight', 'display_size', 'warranty', 'touch_screen',
```

```
         'msoffice', 'latest_price', 'old_price', 'discount', 'star_rating',
         'ratings', 'review_count', 'latest_price_usd', 'old_price_usd'],
        dtype='object')
```

In [17]:
```
#Clean up data in brands column so its all title case and not upper case or lowercase.
df_copy['brand'] = df_copy['brand'].map(lambda x: x.title())
df_copy['brand'].unique()
```

Out[17]:
```
array(['Lenovo', 'Avita', 'Hp', 'Acer', 'Asus', 'Dell', 'Redmibook',
       'Realme', 'Infinix', 'Msi', 'Microsoft', 'Smartron', 'Lg', 'Nokia',
       'Apple', 'Vaio', 'Mi', 'Alienware', 'Iball', 'Samsung'],
      dtype=object)
```

In [18]:
```
#Update missing to say unknown model.
df_copy['model'] = df_copy['model'].replace("Missing", 'Unknown Model')
```

In [19]:
```
#Create dataframe with items i will only be working with.
columns = ['brand', 'model', 'processor_brand','processor_name','processor_generation',

clean_df = df_copy[columns]
clean_df.head()
```

Out[19]:

	brand	model	processor_brand	processor_name	processor_generation	ram_gb	ram_type	solid_sta
0	Lenovo	A6-9225	AMD	A6-9225 Processor	10th	4 GB	DDR4	
1	Lenovo	Ideapad	AMD	APU Dual	10th	4 GB	DDR4	
2	Avita	PURA	AMD	APU Dual	10th	4 GB	DDR4	
3	Avita	PURA	AMD	APU Dual	10th	4 GB	DDR4	
4	Avita	PURA	AMD	APU Dual	10th	4 GB	DDR4	

In [20]:
```
clean_df['graphic_card_gb'].unique()
```

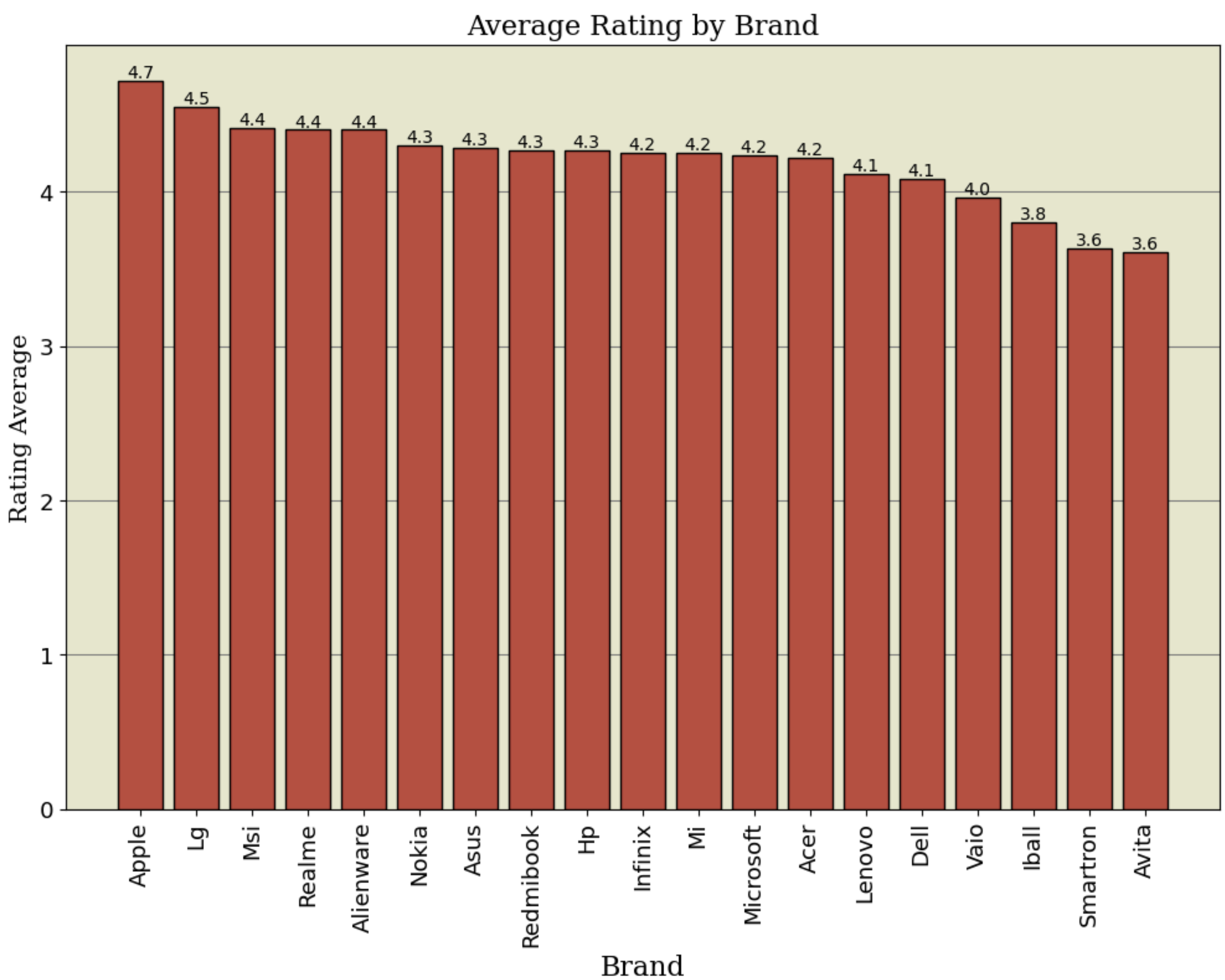
Out[20]:
```
array([0, 4, 2, 6, 8], dtype=int64)
```

Section 2: Descriptive Questions

1. Which brands are rated the highest?

In [21]:
```
#Get average rating per brand, sort data for plotting.
MASK_RATING = clean_df['star_rating'] > 0.0
average_ratings = clean_df[MASK_RATING].groupby('brand')['star_rating'].mean().reset_in
average_ratings = average_ratings.sort_values(by='star_rating', ascending=False)

font1 = {'family':'serif','color':'black','size':16}
font2 = {'family':'serif','color':'black','size':14}


# Plot the bar chart
fig, ax = plt.subplots(figsize=(12,8))
ax.bar(x=average_ratings['brand'], height=average_ratings['star_rating'], color='#B8504
```

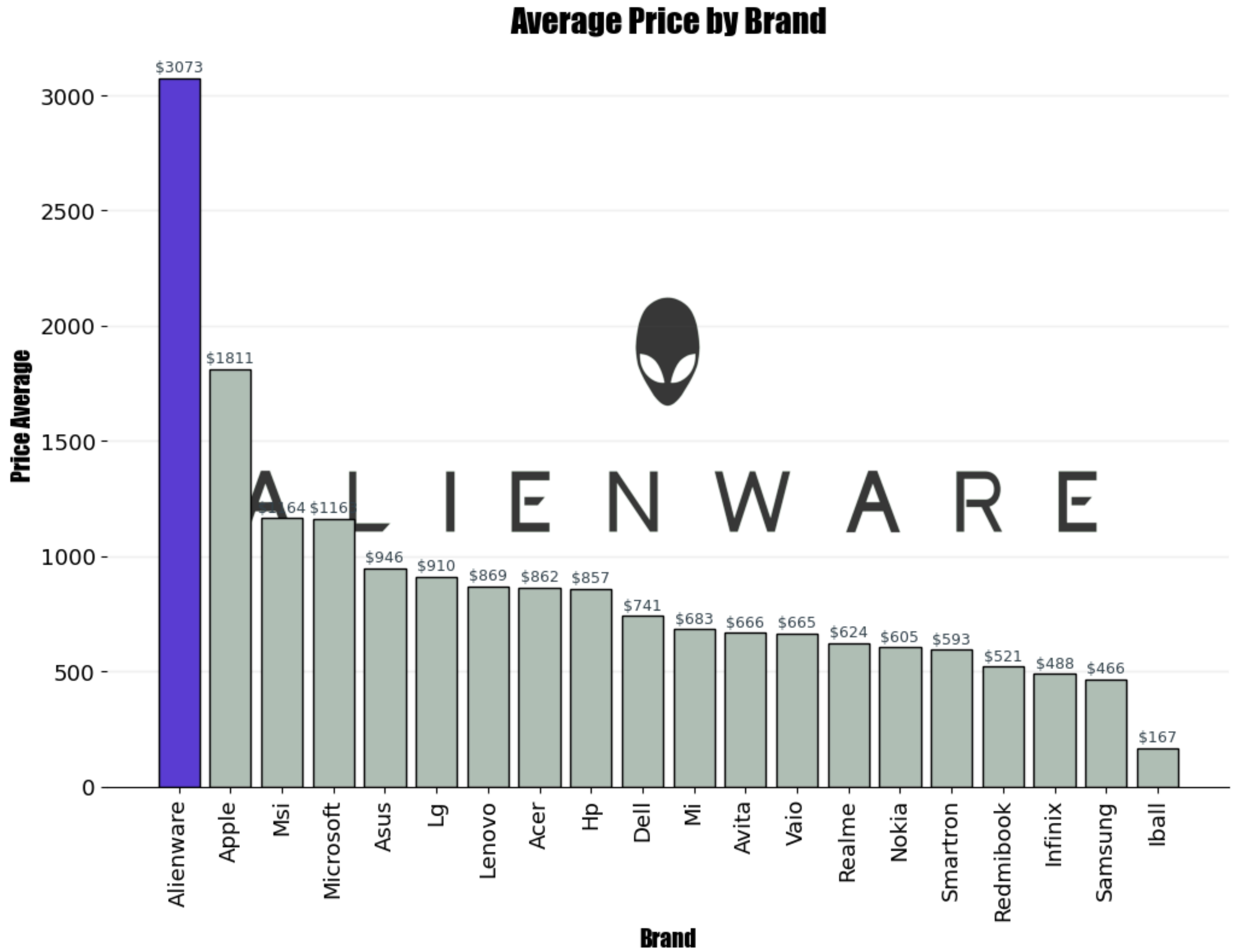
```
#Set title, axis names, customize fonts.
ax.set_xlabel("Brand", fontdict=font1)
ax.set_ylabel("Rating Average", fontdict=font2)
ax.set_title("Average Rating by Brand", fontdict=font1)

#Customize graph
ax.grid(axis = 'y', color='grey')
ax.set_facecolor('#E7E8D1')
plt.xticks(rotation=90, ha='center', fontsize=13)
plt.yticks(fontsize=13)

#Annotate each bar
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2,
        height + .11,
        f'{height:.1f}',
        ha='center',
        va='top',
        color='black',
        fontsize=10)


plt.show()
```



2. Which brand is the most expensive?

In [23]:

```
#Sort data for plotting.
average_price = clean_df.groupby('brand')['latest_price_usd'].mean().reset_index()
average_price = average_price.sort_values(by='latest_price_usd', ascending=False)

#Color mapping and font dictionaries for styling
color_mapping = ['#B2BEB5' for bar in range(20)]
color_mapping[0] = ('#5D3FD3')
font3 = {'family':'fantasy','color':'black','size':20}
font4= {'family':'fantasy','color':'black','size':14}

# Plot the bar chart.
fig, ax = plt.subplots(figsize=(12,8))
ax.bar(x=average_price['brand'], height=average_price['latest_price_usd'], color=color_

#Set title, axis names, customize fonts.
ax.set_xlabel("Brand", fontdict=font4)
ax.set_ylabel("Price Average", fontdict=font4)
ax.set_title("Average Price by Brand", fontdict=font3)
plt.xticks(rotation=90, ha='center', fontsize=13)
plt.yticks(fontsize=13)

# Get the current axis limits
x_min, x_max = ax.get_xlim()
y_min, y_max = ax.get_ylim()

#Customize the graph, set image background
ax.grid(axis = 'y', color='grey', alpha=0.1)
background = plt.imread(r'C:\Users\Chris\Documents\Flatiron\Course Materials\Phase_2\Ph
ax.imshow(background, extent=[x_min, x_max, y_min, y_max], aspect='auto', alpha=0.9)
ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)
ax.spines["top"].set_visible(False)

#Bar annotations
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2,
        height + 80,
        f'${height:.0f}',
        ha='center',
        va='top',
        color='#36454F',
        fontsize=9)

plt.show()
```

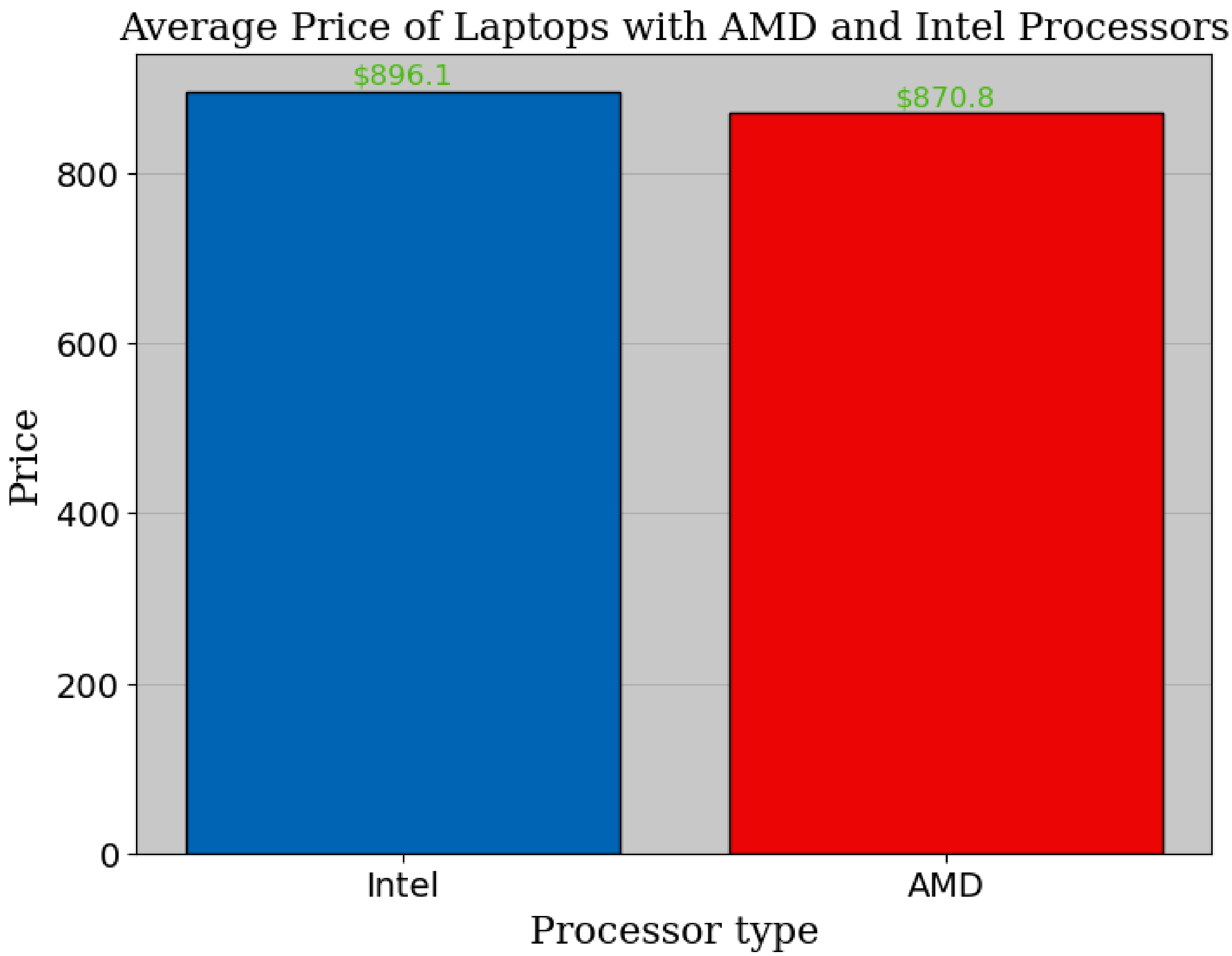


3. Are AMD or Intel processor laptops more expensive? Extra: How do their ratings compare?

In [24]:

```
#Sort data for graphing
AMD_INTEL_MASK = (clean_df['processor_brand'] == "Intel") | (clean_df['processor_brand'
amd_intel_df = clean_df[AMD_INTEL_MASK]
mean_amd_intel_df = amd_intel_df.groupby("processor_brand")['latest_price_usd'].mean().

#Color map for both bars and color dictionary for fonts of axis.
color_mapping = ['#ef0707' for bar in range(2)]
color_mapping[1] = ('#0068B5')
font1 = {'family':'serif','color':'black','size':16}
font2 = {'family':'serif','color':'black','size':14}

#Set up graph, label the axis + title
fig, ax = plt.subplots(figsize=(8, 6))
ax.bar(data=mean_amd_intel_df, x=mean_amd_intel_df['processor_brand'], height=mean_amd_
ax.set_xlabel("Processor type", fontdict=font1)
ax.set_ylabel("Price", fontdict=font1)
ax.set_title("Average Price of Laptops with AMD and Intel Processors", fontdict=font1)

#Further customize the graph by adjusting font size of ticks.
ax.grid(axis='y')
ax.invert_xaxis()
plt.xticks(ha='center', fontsize=14)
plt.yticks(fontsize=14)
ax.set_facecolor('#CCCCCC')
```

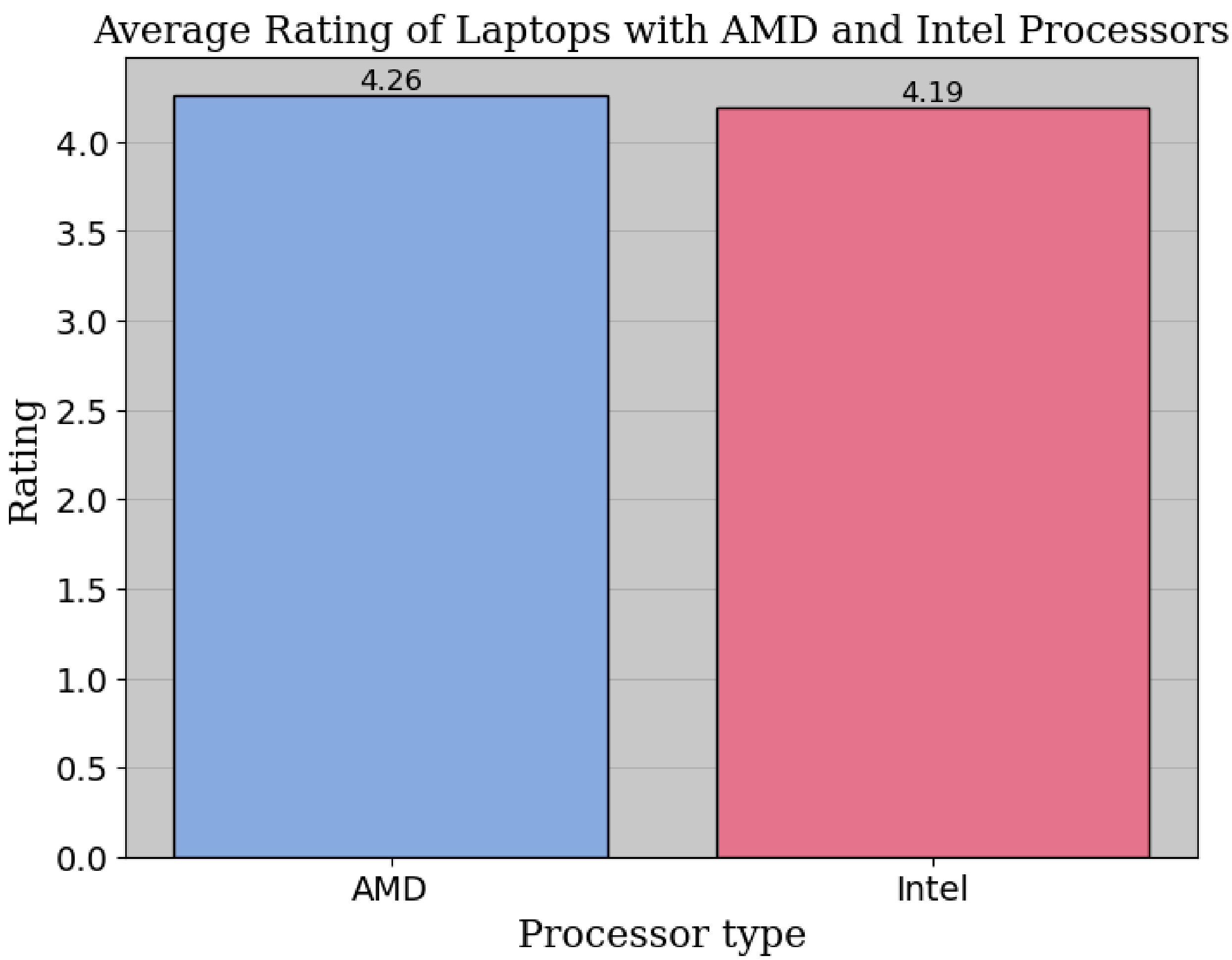
```
#Annotate the bars to add the average amount in USD.
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2,
        height + 34,
        f'${height:.1f}',
        ha='center',
        va='top',
        color='#4CBB17',
        fontsize=12)
```



In [25]:

```
#Sort data for graphing
amd_intel_no_zero = amd_intel_df[amd_intel_df['star_rating'] > 0.0]
mean_amd_intel_rating_df = amd_intel_no_zero.groupby("processor_brand")['star_rating'].

#Color map for both bars and color dictionary for fonts of axis.
color_mapping = ['#89ABE3' for bar in range(2)]
color_mapping[1] = ('#EA738D')
font1 = {'family':'serif','color':'black','size':16}
font2 = {'family':'serif','color':'black','size':14}

#Set up graph, label the axis + title, pass in font dictionaries to customize the label
fig, ax = plt.subplots(figsize=(8, 6))
ax.bar(data=mean_amd_intel_rating_df, x=mean_amd_intel_rating_df['processor_brand'], he
ax.set_xlabel("Processor type", fontdict=font1)
ax.set_ylabel("Rating", fontdict=font1)
ax.set_title("Average Rating of Laptops with AMD and Intel Processors", fontdict=font1)

#Further customize the graph by adjusting font size of ticks, set face color.
```

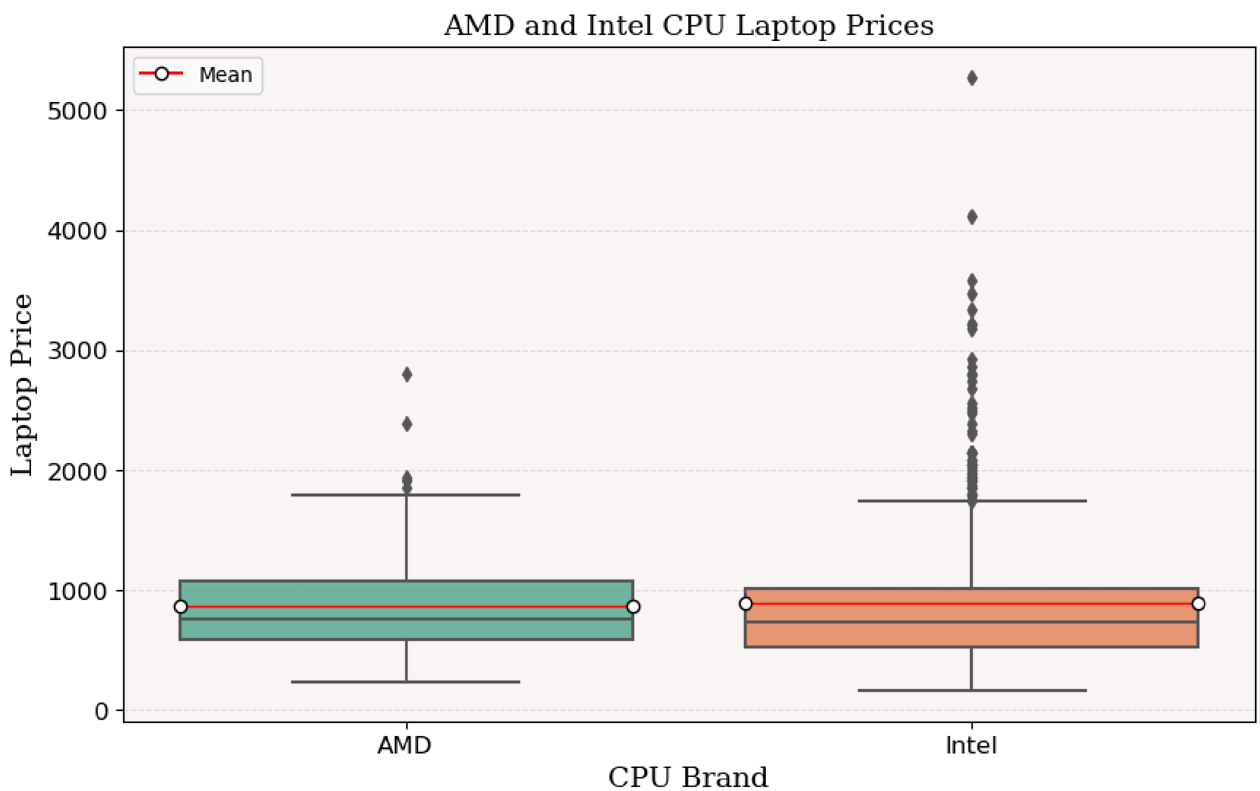
```
ax.grid(axis='y')
plt.xticks(ha='center', fontsize=14)
plt.yticks(fontsize=14)
ax.set_facecolor('#CCCCCC')

#Annotate the bars to add the average amount in USD.
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2,
        height + 0.15,
        f'{height:.2f}',
        ha='center',
        va='top',
        color='black',
        fontsize=12)
```



In [26]:

```
#set up Box plot visualization and plot it
plt.figure(figsize=(10, 6))
sns.boxplot(x='processor_brand',
            y='latest_price_usd',
            data=amd_intel_df,
            palette="Set2",
            zorder=2,
            showmeans=True,
            meanline=True,
            meanprops={"marker":"o", "linestyle":"-", "color":"#ff0000",  "markerfaceco
            )

#Set axis names and title, customize fonts
```

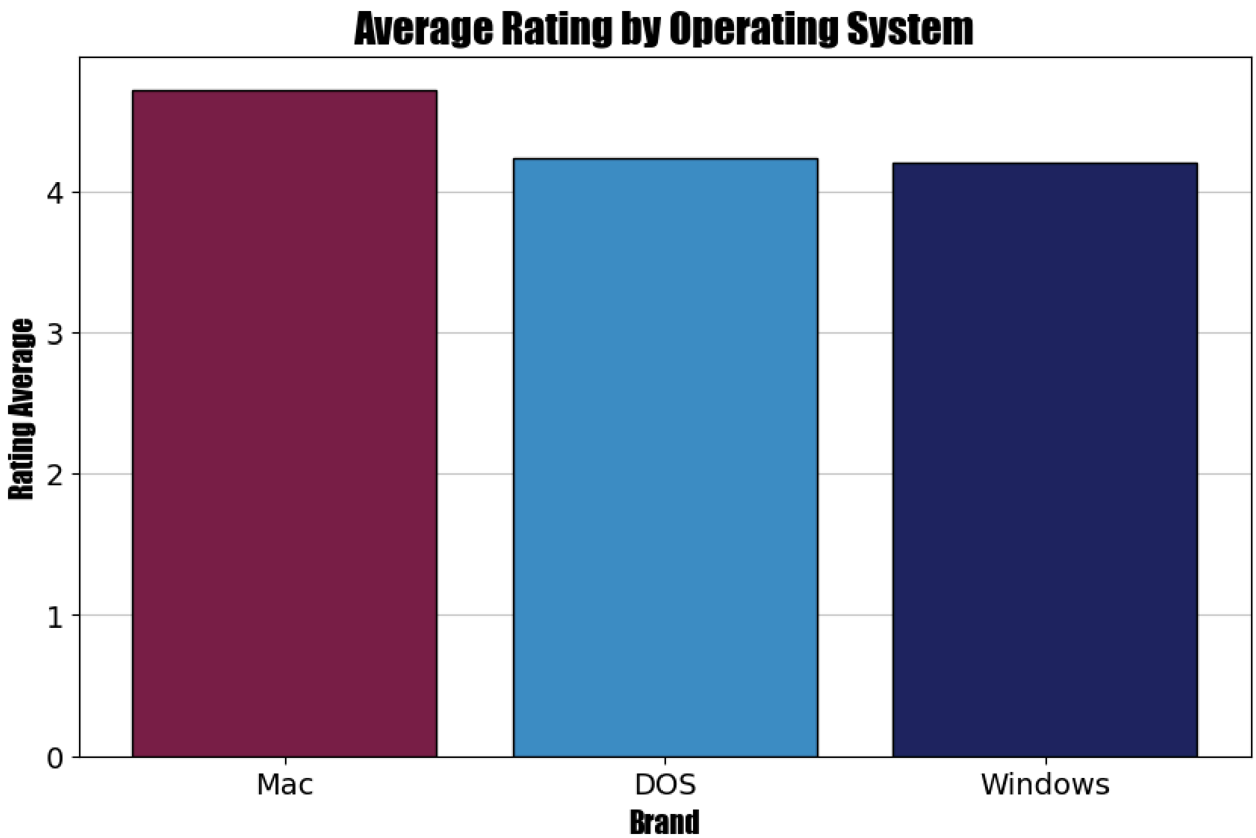
```
plt.title('AMD and Intel CPU Laptop Prices', fontdict=font2)
plt.xlabel('CPU Brand', fontdict=font2)
plt.ylabel('Laptop Price',fontdict=font2)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

#Set background of grid to custom color and add y-axis gridlines.
ax = plt.gca()
ax.set_facecolor('#FCF6F5')
plt.grid(True, axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.2, zorder

# Add custom legend for the mean line and add it to the plot
mean_line = Line2D([0], [0], color='#ff0000', marker='o', markerfacecolor='white', mark
plt.legend(handles=[mean_line], loc='upper left')

plt.show()
```



4. Which operating systems are rated the highest?

In [27]:
```
#Sort data for plotting
MASK_RATING = clean_df['star_rating'] > 0.0
os_ratings = clean_df[MASK_RATING].groupby("operating_system")['star_rating'].mean().re
os_ratings = os_ratings.sort_values(by='star_rating', ascending=False)

#Color map
color_mapping = ['#1E2761' for bar in range(3)]
color_mapping[0] = ('#7A2048')
color_mapping[1] = ('#408EC6')

# Plot the bar chart
fig, ax = plt.subplots(figsize=(10,6))
ax.bar(x=os_ratings['operating_system'], height=os_ratings['star_rating'],color=color_m

#Set axis and title
```

```
ax.set_xlabel("Brand", fontdict=font4)
ax.set_ylabel("Rating Average", fontdict=font4)
ax.set_title("Average Rating by Operating System", fontdict=font3)
plt.xticks(ha='center', fontsize=14)
plt.yticks(fontsize=14)
ax.grid(axis = 'y', color='grey', alpha=0.4)

plt.show()
```

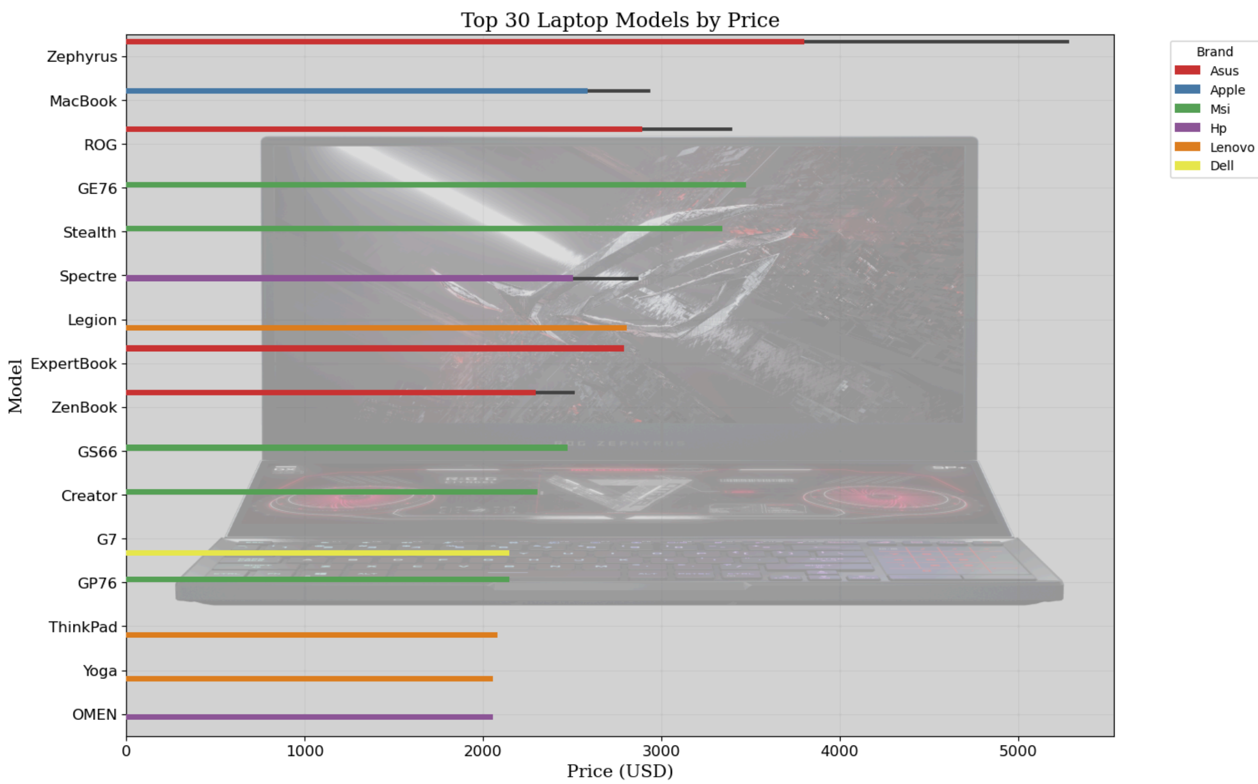


5. Of the top thirty models, which models are the most expensive?

In [28]:

```
#Sort Data for plotting.
MASK_MODEL = clean_df['model'] != "Unknown Model"
columns_models = ['brand','model','latest_price_usd']
model_brand_price = clean_df[MASK_MODEL][columns_models].sort_values(by="latest_price_u

#Set up the plot and plot the bar plot.
fig, ax = plt.subplots(figsize=(14, 10))
ax.set_facecolor('#D3D3D3')
ax = sns.barplot(x='latest_price_usd', y='model', hue='brand', data=model_brand_price,

#Set axis names, title, axis tick sizes.
plt.title('Top 30 Laptop Models by Price', fontdict=font1)
plt.xlabel('Price (USD)', fontdict=font2)
plt.ylabel('Model', fontdict=font2)
plt.xticks(ha='center', fontsize=12)
plt.yticks(fontsize=12)

# Get the current axis limits.
x_min, x_max = ax.get_xlim()
y_min, y_max = ax.get_ylim()
```

```
#Customize the graph, set image background
ax.grid(axis = 'both', color='grey', alpha=0.1)
background = plt.imread(r'C:\Users\Chris\Documents\Flatiron\Course Materials\Phase_2\Ph
ax.imshow(background, extent=[x_min, x_max, y_min, y_max], aspect='auto', alpha=0.3)

#Graph Customizations
plt.legend(title='Brand', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.show()
```

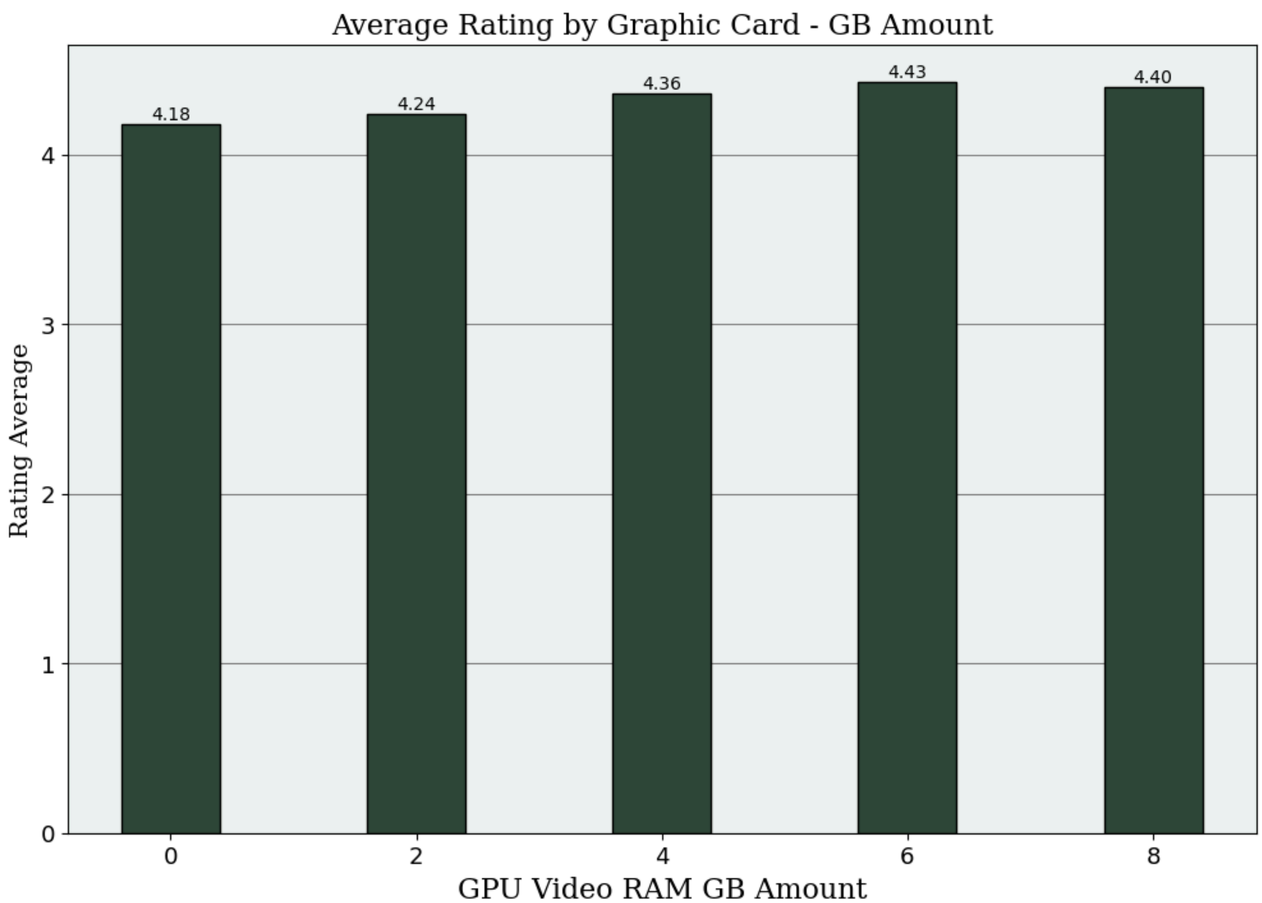


6. What are the average ratings for laptops by their GPU's GB Amount?

In [29]:

```
#Get average rating per GPU - sort data for plotting.
MASK_RATING = clean_df['star_rating'] > 0.0
average_gpu = clean_df[MASK_RATING].groupby('graphic_card_gb')['star_rating'].mean().re
average_gpu = average_gpu.sort_values(by='star_rating', ascending=False)

font1 = {'family':'serif','color':'black','size':16}
font2 = {'family':'serif','color':'black','size':14}


# Plot the bar chart
fig, ax = plt.subplots(figsize=(12,8))
ax.bar(x=average_gpu['graphic_card_gb'], height=average_gpu['star_rating'], color='#314

#Set title, axis names, customize fonts.
ax.set_xlabel("GPU Video RAM GB Amount", fontdict=font1)
ax.set_ylabel("Rating Average", fontdict=font2)
ax.set_title("Average Rating by Graphic Card - GB Amount", fontdict=font1)

#Customize graph
ax.grid(axis = 'y', color='grey')
```

```
ax.set_facecolor('#EDF4F2')
plt.xticks(ha='center', fontsize=13)
plt.yticks(fontsize=13)

#Annotate each bar
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2,
        height + .11,
        f'{height:.2f}',
        ha='center',
        va='top',
        color='black',
        fontsize=10)


plt.show()
```



Section 3: Inferential Questions Analysis

In [31]:

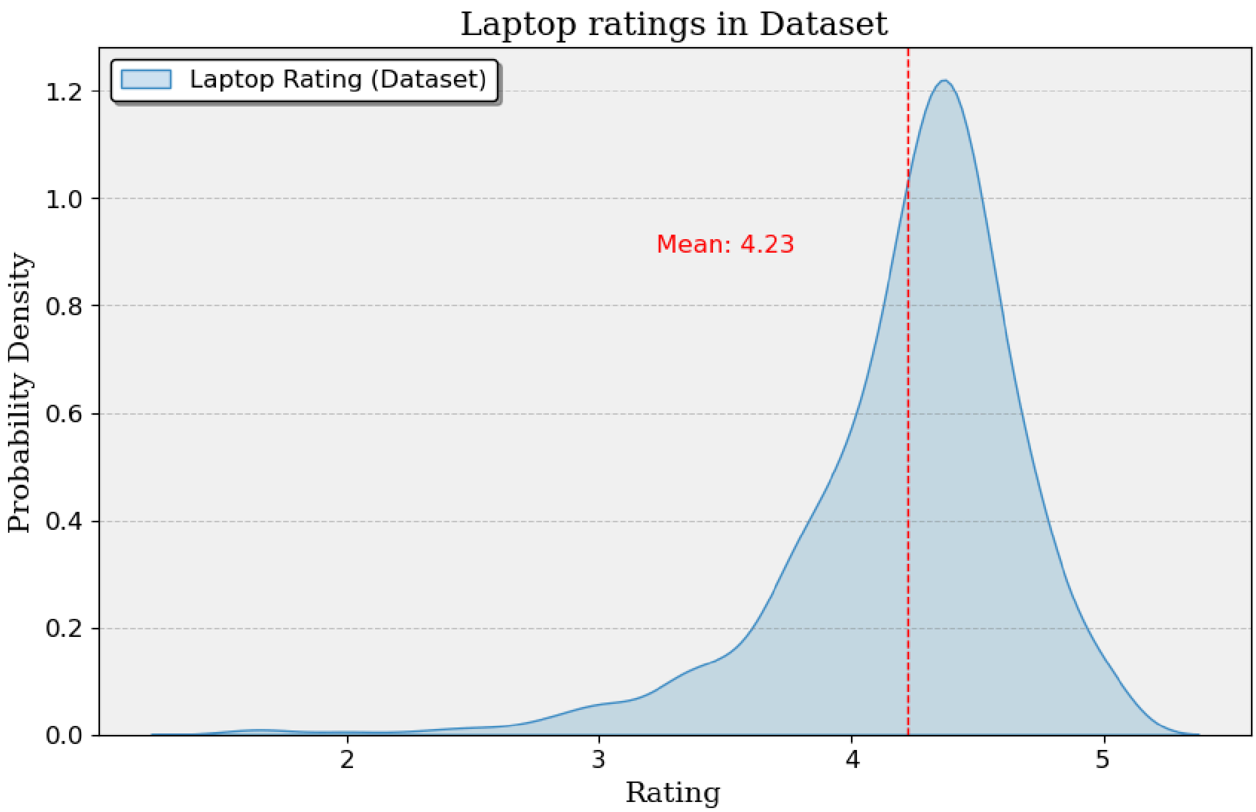
```
#Define function for testing our hypothesis.

def test_outcome(pvalue, alpha=0.05):
    if pvalue < alpha:
        return "Reject the null hypothesis."
    else:
        return "Fail to reject the null hypothesis."
```

```
#Creating sample for analysis.
df_sample_a = clean_df.sample(n=200, random_state=1)

#Slicing the sample to include only ratings that have merit (0.0 were null.)
good_rating_sample = df_sample_a[df_sample_a['star_rating'] > 0.0]
good_rating_df = clean_df[clean_df['star_rating'] > 0.0]
```

3.1 Let's check visually and run a Shapiro-Wilkes test to see if our data distribution is normal.

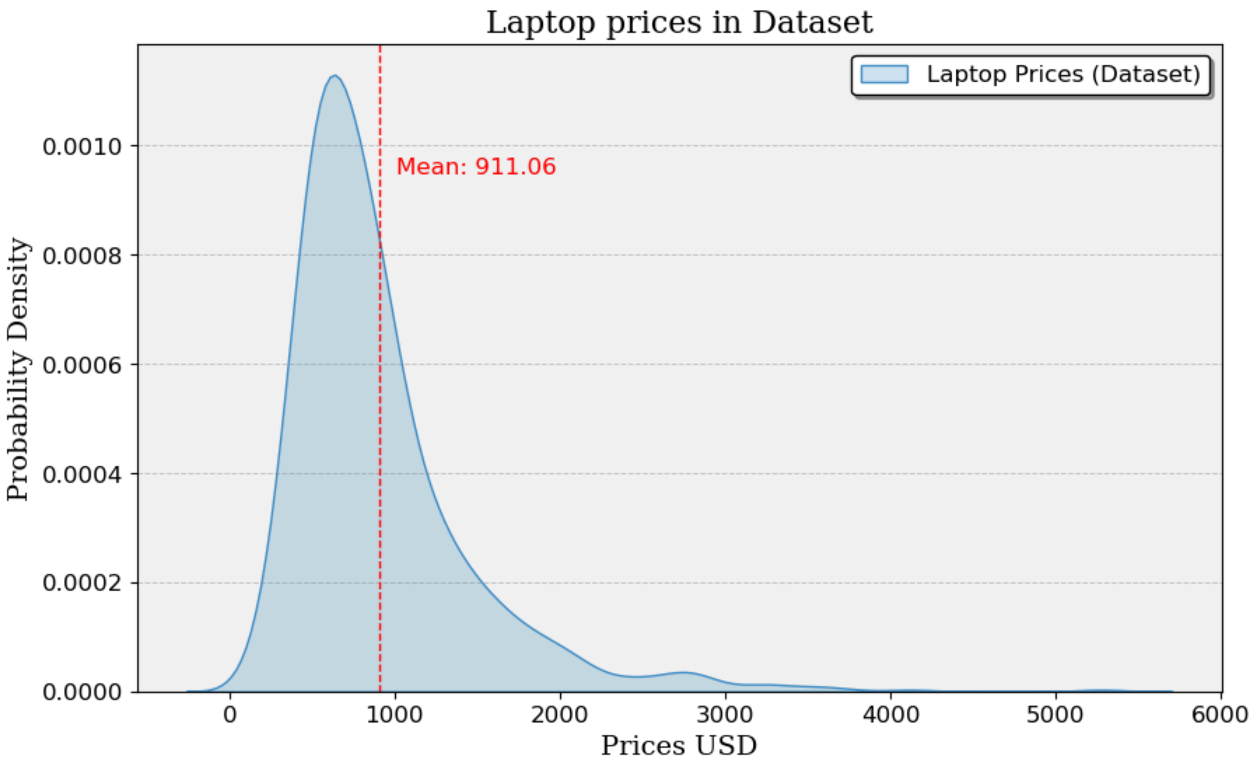
In [32]:

```
#Inspected the Dataset to see how the rating data is distributed for ratings (data does

#Set up the plot
fig, ax = plt.subplots(figsize=(10, 6))

#KDE plot
sns.kdeplot(data=good_rating_df, x="star_rating", color='#408EC6', label="Laptop Rating

#Labeling axes, customizing font sizes and styles, adjust tick sizes, and setting face
plt.title("Laptop ratings in Dataset", fontdict=font1)
plt.xlabel('Rating', fontdict=font2)
plt.ylabel('Probability Density', fontdict=font2)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.4)
ax.set_facecolor('#f0f0f0')

# Add a vertical line at the mean
mean_points = good_rating_df['star_rating'].mean()
plt.axvline(mean_points, color='red', linestyle='--', linewidth=1)
plt.text(mean_points - 1, 0.9, f'Mean: {mean_points:.2f}', color='red', fontsize=12)

# Add Legend
plt.legend(loc='upper left', fontsize=12, frameon=True, fancybox=True, shadow=True, fac

plt.show()
```



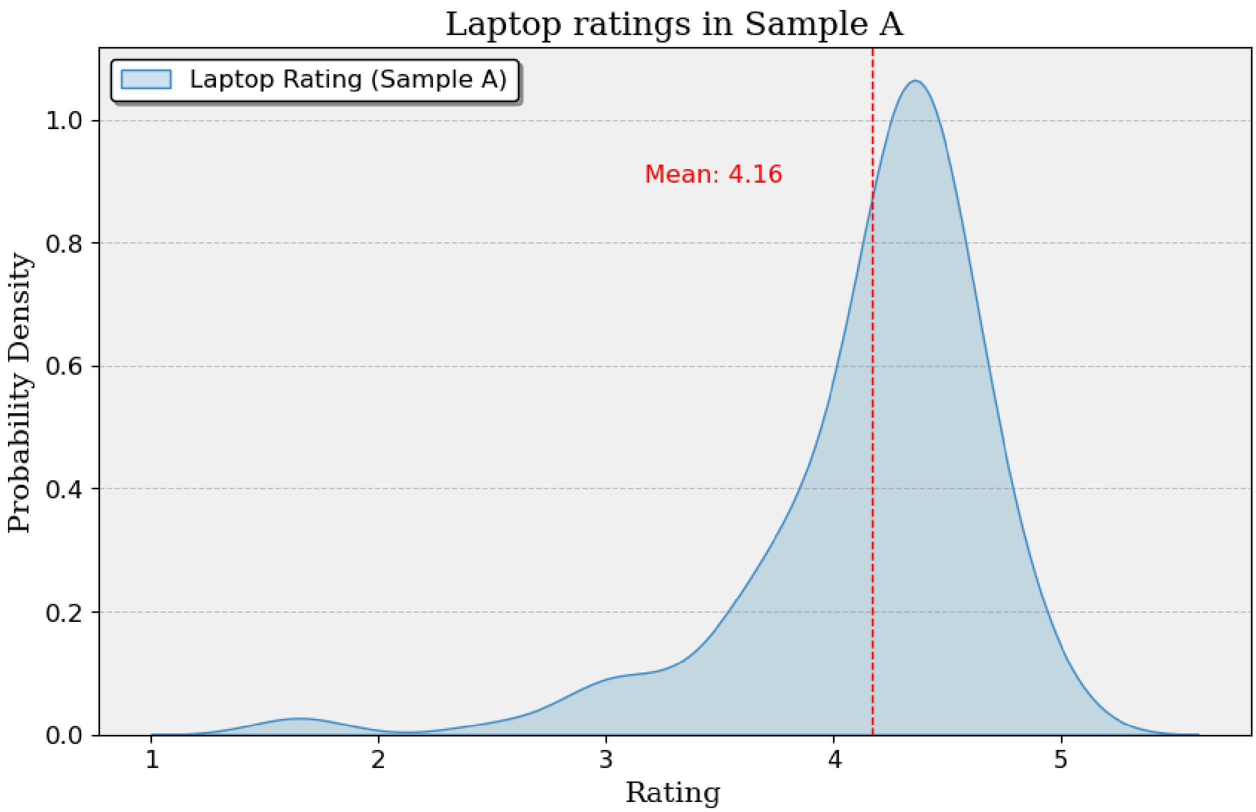
In [33]:

```
#Inspected the Dataset to see how the laptop price data is distributed for price.

#Set up the plot
fig, ax = plt.subplots(figsize=(10, 6))

#KDE plot
sns.kdeplot(data=clean_df, x="latest_price_usd", color='#408EC6', label="Laptop Prices

#Labeling axes, customizing font sizes and styles, adjust tick sizes, and setting face
plt.title("Laptop prices in Dataset", fontdict=font1)
plt.xlabel('Prices USD', fontdict=font2)
plt.ylabel('Probability Density', fontdict=font2)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.4)
ax.set_facecolor('#f0f0f0')

# Add a vertical line at the mean
mean_points = clean_df['latest_price_usd'].mean()
plt.axvline(mean_points, color='red', linestyle='--', linewidth=1)
plt.text(mean_points + 100, plt.gca().get_ylim()[1] * 0.8, f'Mean: {mean_points:.2f}',

# Add legend
plt.legend(loc='upper right', fontsize=12, frameon=True, fancybox=True, shadow=True, fa

plt.show()
```



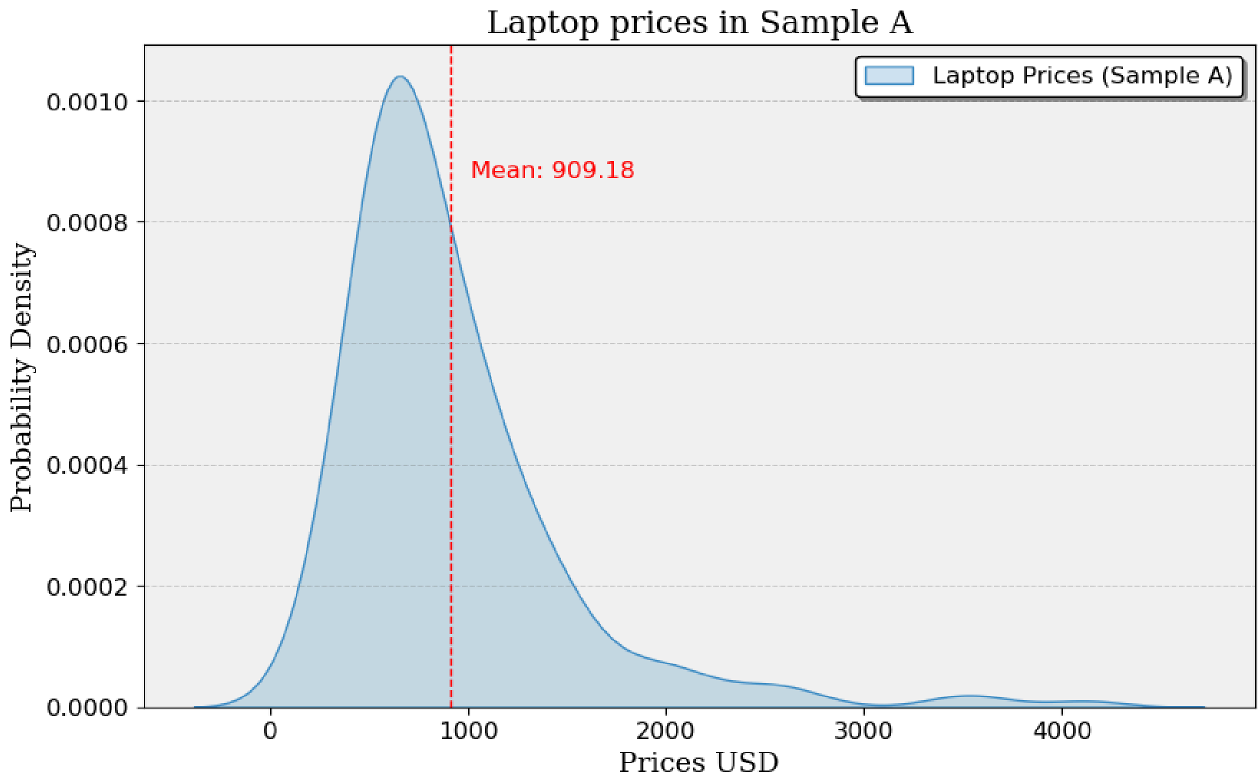
In [34]:

```
#Inspecting the sample to see how the rating data is distributed for ratings, does not

#Set up the plot
fig, ax = plt.subplots(figsize=(10, 6))

#KDE plot
sns.kdeplot(data=good_rating_sample, x="star_rating", color='#408EC6', label="Laptop Ra

#Labeling axes, customizing font sizes and styles, adjust tick sizes, and setting face 
plt.title("Laptop ratings in Sample A", fontdict=font1)
plt.xlabel('Rating', fontdict=font2)
plt.ylabel('Probability Density', fontdict=font2)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.4)
ax.set_facecolor('#f0f0f0')

# Add a vertical line at the mean
mean_points = good_rating_sample['star_rating'].mean()
plt.axvline(mean_points, color='red', linestyle='--', linewidth=1)
plt.text(mean_points - 1, 0.9, f'Mean: {mean_points:.2f}', color='red', fontsize=12)

# Add legend
plt.legend(loc='upper left', fontsize=12, frameon=True, fancybox=True, shadow=True, fac

plt.show()
```



In [35]:

```
#Inspected the sample to see how the laptop price data is distributed for price. - Matc

#Set up the plot
fig, ax = plt.subplots(figsize=(10, 6))

#KDE plot
sns.kdeplot(data=df_sample_a, x="latest_price_usd", color='#408EC6', label="Laptop Pric

#Labeling axes, customizing font sizes and styles, adjust tick sizes, and setting face 
plt.title("Laptop prices in Sample A", fontdict=font1)
plt.xlabel('Prices USD', fontdict=font2)
plt.ylabel('Probability Density', fontdict=font2)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.4)
ax.set_facecolor('#f0f0f0')

# Add a vertical line at the mean
mean_points = df_sample_a['latest_price_usd'].mean()
plt.axvline(mean_points, color='red', linestyle='--', linewidth=1)
plt.text(mean_points + 100, plt.gca().get_ylim()[1] * 0.8, f'Mean: {mean_points:.2f}',

# Add legend
plt.legend(loc='upper right', fontsize=12, frameon=True, fancybox=True, shadow=True, fa

plt.show()
```



In [36]:
```
#Performing normality test for ratings.

stat, p_val = stats.shapiro(good_rating_sample['star_rating'])
p_val

# Interpret the results
alpha = 0.05
if p_val > alpha:
 print('Sample looks Gaussian (fail to reject H0)')
else:
 print('Sample does not look Gaussian (reject H0)')
```

```
Sample does not look Gaussian (reject H0)
```

In [37]:
```
#Performing normality test for price.

stat, p_val = stats.shapiro(df_sample_a['latest_price_usd'])
p_val

# Interpret the results
alpha = 0.05
if p_val > alpha:
 print('Sample looks Gaussian (fail to reject H0)')
else:
 print('Sample does not look Gaussian (reject H0)')
```

```
Sample does not look Gaussian (reject H0)
```

Interpretation

After examining the distribution of latest prices and ratings, we have a Non-Parametric distribution for our dataset (Non-Gaussian or non-normal), and therefore we will need to perform

Nonparametric statistical significance tests such as Mann-Whitney U Test (Non Parametric version of Students T-Test) and Kruskal-Wallis H Test (Non Parametric version of ANOVA Test)

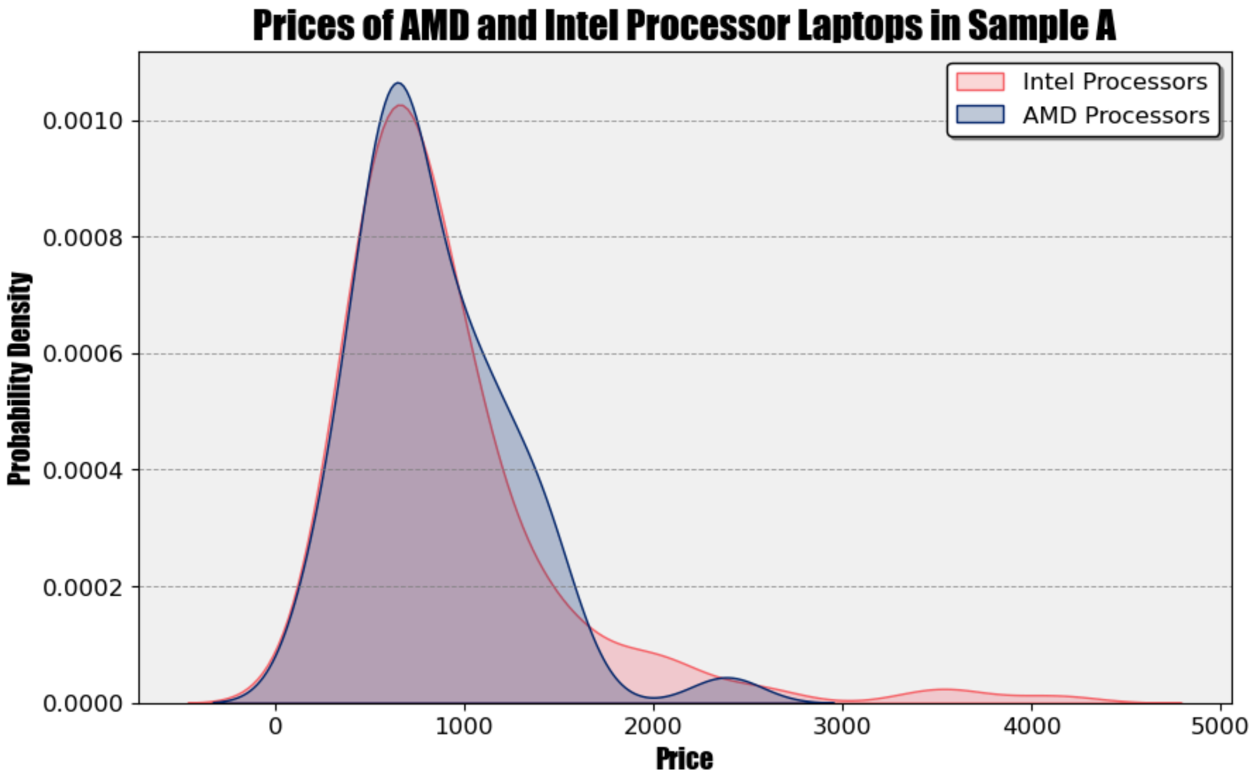
1. Is there a significant difference in the prices between laptops with different processor brands (e.g., Intel vs. AMD)? (Mann-Whitney U Test)

H_0: *There is no difference in price between laptops with different processor brands such as AMD vs. Intel.*

H_1: *There is a difference in price between laptops with different processor brands such as AMD vs. Intel.*

In [38]:

```
sample_processor_intel = df_sample_a[df_sample_a['processor_brand'] == "Intel"]['latest_
sample_processor_amd = df_sample_a[df_sample_a['processor_brand'] == "AMD"]['latest_pri

#Set up the plot
plt.figure(figsize=(10, 6))

#KDE plot
sns.kdeplot(data=sample_processor_intel, x=sample_processor_intel, color='#F96167', lab
sns.kdeplot(data=sample_processor_amd, x=sample_processor_amd, color='#00246B', label="

#Labeling axes, customizing font sizes and styles, adjust tick sizes, and setting face 
plt.title("Prices of AMD and Intel Processor Laptops in Sample A", fontdict=font3)
plt.xlabel('Price', fontdict=font4)
plt.ylabel('Probability Density', fontdict=font4)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)


ax = plt.gca()
ax.set_facecolor('#f0f0f0')

# Add legend
plt.grid(True, which='both', axis='y', linestyle='--', linewidth=0.7, color='gray', alp
plt.legend(loc='upper right', fontsize=12, frameon=True, fancybox=True, shadow=True, fa

plt.show()
```



In [39]:
```
#Interpret Results
stat, p_val = mannwhitneyu(sample_processor_intel, sample_processor_amd)
print(p_val)

alpha = 0.05

test_outcome(p_val, alpha=alpha)
```

```
0.9964533022037422
```

Out[39]: 'Fail to reject the null hypothesis.'

Interpretation

Based on our test outcome, we have found that there is no significant difference between the means of price in each sample of laptops that have Intel processors and laptops that have AMD processors.

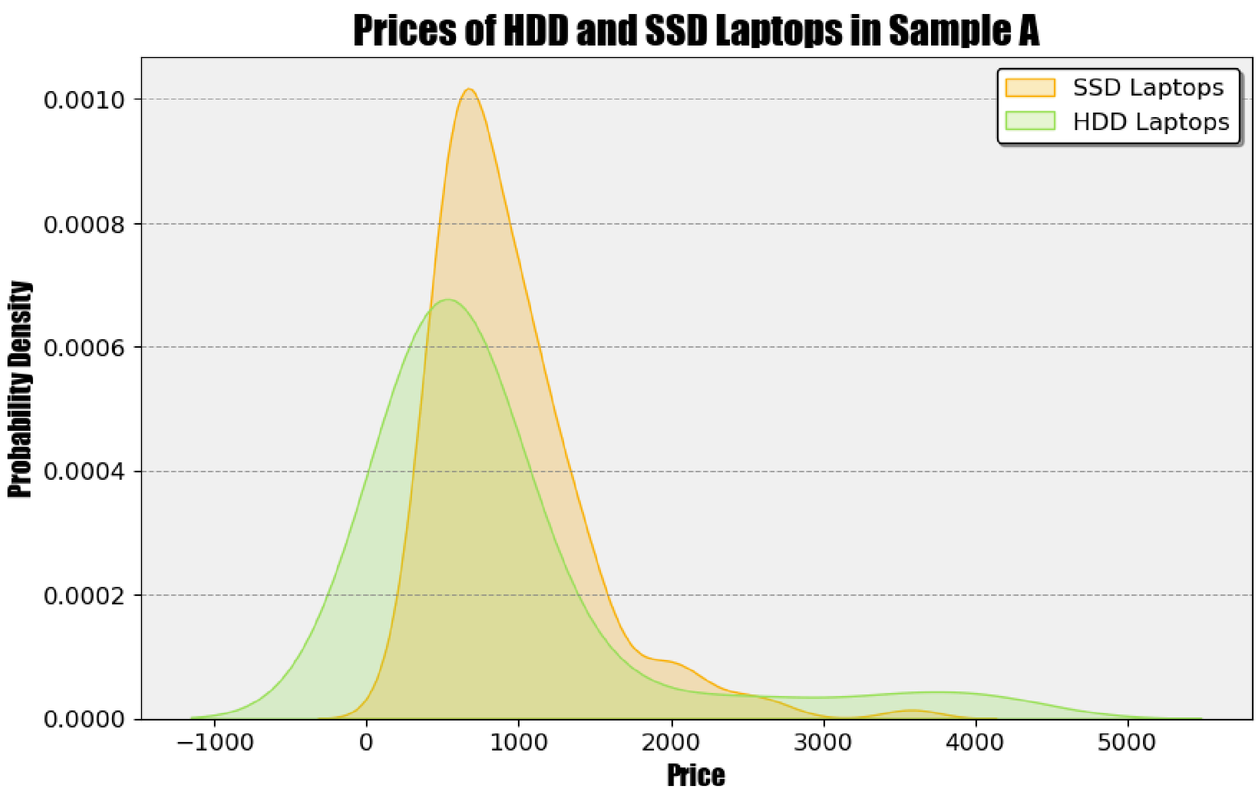
2. Do laptops with solid-state drives (SSD's) have significantly different prices compared to those with hard disk drives (HDD's)? (Mann-Whitney U Test)

H_0: *There is no difference in price between laptops with SSD's compared to Laptops with HDD's.*

H_1: *There is a difference in price between laptops with SSD's compared to Laptops with HDD's.*

In [40]:
```
sample_ssd = df_sample_a[(df_sample_a['solid_state_drive'] != "0 GB") & (df_sample_a['h
sample_hdd = df_sample_a[(df_sample_a['solid_state_drive'] == "0 GB") & (df_sample_a['h

#Set up the plot
plt.figure(figsize=(10, 6))

#KDE plot
```

```
sns.kdeplot(data=sample_ssd, x=sample_ssd, color='#fbb30b', label="SSD Laptops", fill=T
sns.kdeplot(data=sample_hdd, x=sample_hdd, color='#9ce15b', label="HDD Laptops", fill=T

#Labeling axes, customizing font sizes and styles, adjust tick sizes, and setting face 
plt.title("Prices of HDD and SSD Laptops in Sample A", fontdict=font3)
plt.xlabel('Price', fontdict=font4)
plt.ylabel('Probability Density', fontdict=font4)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)


ax = plt.gca()
ax.set_facecolor('#f0f0f0')


# Add legend
plt.grid(True, which='both', axis='y', linestyle='--', linewidth=0.7, color='gray', alp
plt.legend(loc='upper right', fontsize=12, frameon=True, fancybox=True, shadow=True, fa

plt.show()

#The KDE plot is assuming some negative values, despite there being no negative values.
```



In [41]:
```
stat, p_val = mannwhitneyu(sample_ssd, sample_hdd)
print(p_val)

alpha = 0.05

test_outcome(p_val, alpha=alpha)
```

```
0.0016371099590585666
```

Out[41]:
```
'Reject the null hypothesis.'
```

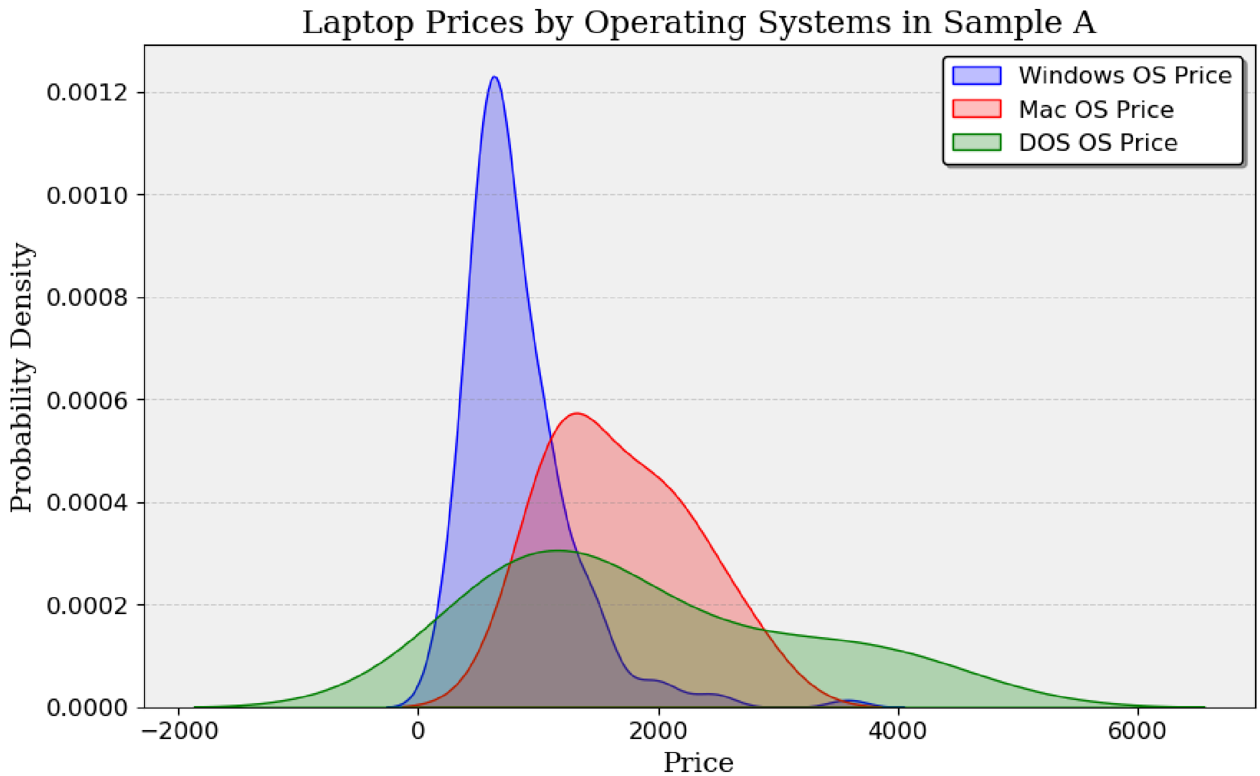
Interpretation

Based on our test outcome, we have found that there is a statistically significant difference between the means of both samples of laptops that have SSD's and laptops that have hard disk drives and as a result, further research is warranted.

3. Is there a significant difference in the prices of laptops based on their operating system? (Kruskal-Wallis H Test)

- $H_0 : \mu_{Windows} = \mu_{Mac} = \mu_{DOS}$
- $H_a : H_0$ is not true

In [42]:
```
#Data sorting for plotting.
windows_sample = df_sample_a[df_sample_a['operating_system'] == "Windows"]['latest_pric
mac_sample = df_sample_a[df_sample_a['operating_system'] == "Mac"]['latest_price_usd']
dos_sample = df_sample_a[df_sample_a['operating_system'] == "DOS"]['latest_price_usd']

#Set up the plot.
plt.figure(figsize=(10, 6))

#KDE plot
sns.kdeplot(data=windows_sample, x=windows_sample, color='blue', label="Windows OS Pric
sns.kdeplot(data=mac_sample, x=mac_sample, color='red', label="Mac OS Price", fill=True
sns.kdeplot(data=dos_sample, x=dos_sample, color='green', label="DOS OS Price", fill=Tr

#Labeling axes, customizing font sizes and styles, adjust tick sizes, and setting face 
plt.title("Laptop Prices by Operating Systems in Sample A", fontdict=font1)
plt.xlabel('Price', fontdict=font2)
plt.ylabel('Probability Density', fontdict=font2)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

#Grid customization
ax = plt.gca()
ax.set_facecolor('#f0f0f0')
plt.grid(True, which='both', axis='y', linestyle='--', linewidth=0.7, color='gray', alp

# Add Legend
plt.legend(loc='upper right', fontsize=12, frameon=True, fancybox=True, shadow=True, fa


plt.show()
```



In [43]:
```
stat, p_value = kruskal(windows_sample, mac_sample, dos_sample)
p_value

test_outcome(p_value, alpha=alpha)
```

Out[43]:
```
'Reject the null hypothesis.'
```

Interpretation

Based on our test outcome, we have found that our Null hypothesis was not true and there is a statistical significance between the means of each group, therefore warranting further research.

3.2 Inferential Analysis via Central Limit Theorem (CLT)

4. Is there a significant difference in the ratings between different brands, such as Asus and HP? (Independent Student T-Test)

H_0: *There is no difference between the ratings in different brand such as Asus and HP.*

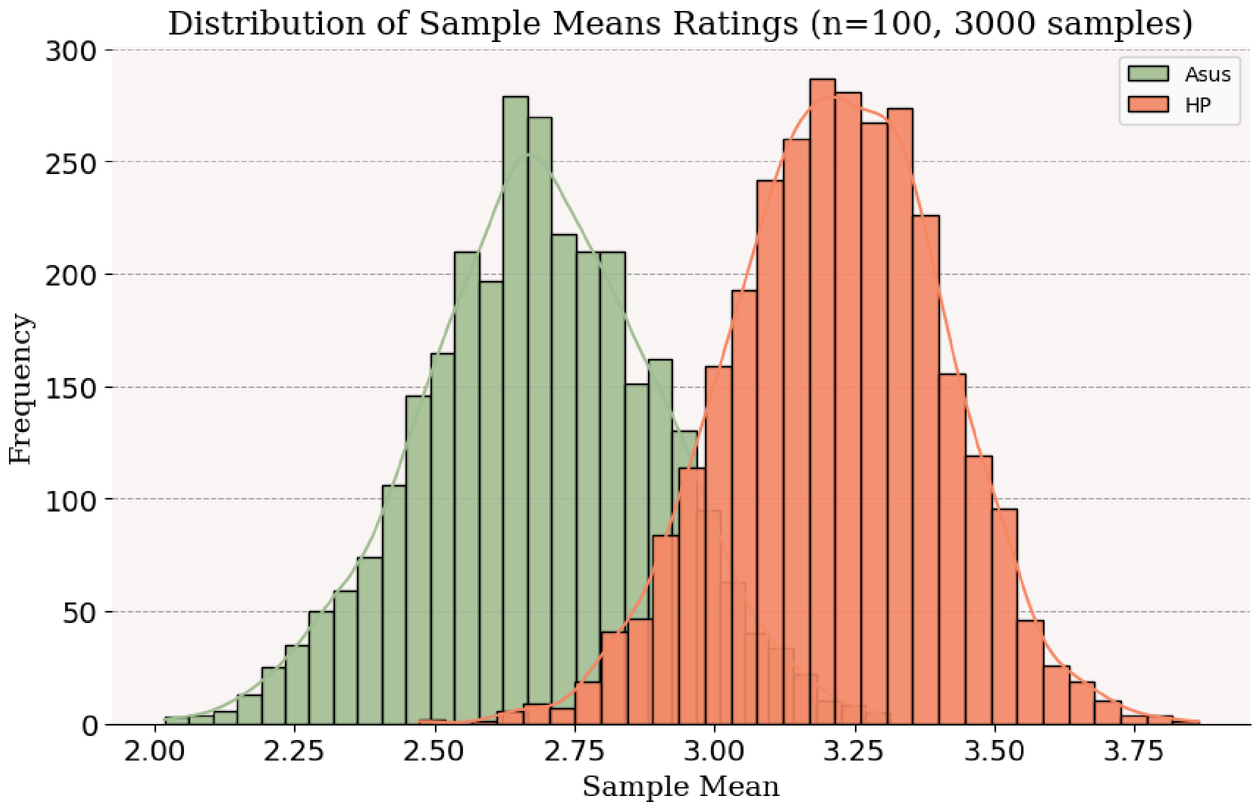
H_1: *There is a difference between the ratings in different brand such as Asus and HP.*

In [44]:
```
#Create a function to easily create a sample of means to apply CLT.

def generate_sample_means(data, sample_size, num_samples):
```

```
    sample_means = []
    for _ in range(num_samples):
        sample = np.random.choice(data, size=sample_size, replace=True)
        sample_means.append(np.mean(sample))
    return sample_means
```

In [49]:

```
#Slice main DF by each computer brand.
asus_df = clean_df[clean_df['brand'] == "Asus"]
hp_df = clean_df[clean_df['brand'] == "Hp"]

#Only include star ratings in stored variables.
asus_data = asus_df['star_rating'].values
hp_data = hp_df['star_rating'].values

# Parameters for sampling
sample_size = 100
num_samples = 3000

# Generate sample means
asus_sample_means = generate_sample_means(asus_data, sample_size, num_samples)
hp_sample_means = generate_sample_means(hp_data, sample_size, num_samples)

# Plot the distribution of sample means
fig, ax = plt.subplots(figsize=(10, 6))
sns.histplot(asus_sample_means, kde=True, color='#A1BE95', bins=30, label="Asus", alpha
sns.histplot(hp_sample_means, kde=True, color='#F98866', bins=30, label="HP", alpha=0.9

#Set title and axis, additional plot customization
plt.title(f'Distribution of Sample Means Ratings (n={sample_size}, {num_samples} sample
plt.xlabel('Sample Mean', fontdict=font2)
plt.ylabel('Frequency', fontdict=font2)
plt.grid(True, axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.7, zorder
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.legend()

#Remove lines of grid perimeter to make more appealing.
ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)
ax.spines["top"].set_visible(False)

#Set background of grid to custom color.
ax = plt.gca()
ax.set_facecolor('#FCF6F5')

plt.show()
```



In [50]:
```
# Shapiro-Wilk test for normality for asus sample means.
_, p_value = stats.shapiro(asus_sample_means)
print(p_value)

if p_value > 0.05:
    print("The sample means are approximately normally distributed (fail to reject the
else:
    print("The sample means are not normally distributed (reject the null hypothesis of

_, p_value = stats.shapiro(hp_sample_means)
print(p_value)

if p_value > 0.05:
    print("The sample means are approximately normally distributed (fail to reject the
else:
    print("The sample means are not normally distributed (reject the null hypothesis of
```

```
0.18792724609375
The sample means are approximately normally distributed (fail to reject the null hypothe
sis of normality).
0.08553383499383926
The sample means are approximately normally distributed (fail to reject the null hypothe
sis of normality).
```

In [51]:
```
# Perform a two-sample t-test on the sample means
stat, p_value = ttest_ind(asus_sample_means, hp_sample_means, equal_var=False)
print(p_value)

if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in the ratings
else:
    print("Fail to reject the null hypothesis: There is no significant difference in the
```

```
0.0
Reject the null hypothesis: There is a significant difference in the ratings of laptops
```

```
based on their brand between HP and Asus.
```

Interpretation

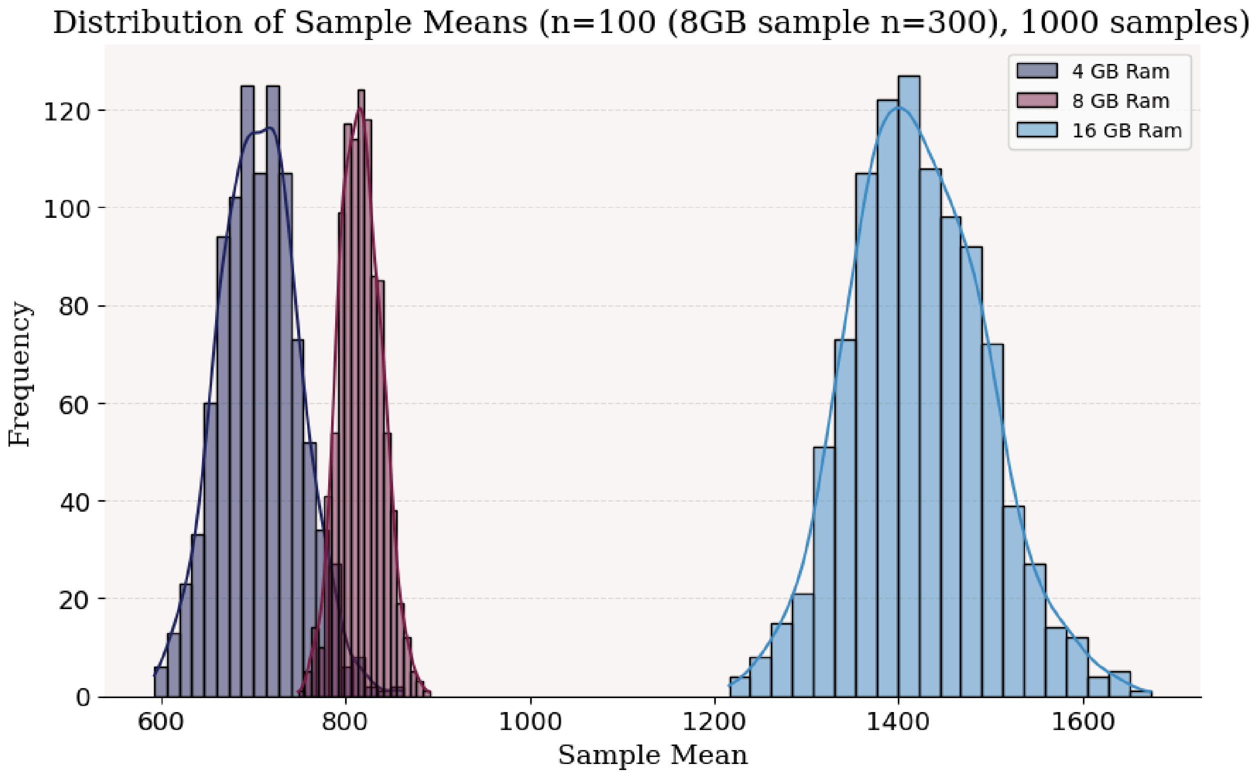
After performing our test, we have rejected the null hypothesis. There was found to be a significant difference in the means between the ratings of laptops based on their brand between HP and Asus and therefore warrants additional research.

5. Do laptops with higher RAM tend to have higher prices compared to those that do not? (ANOVA Test)

- $H_0 : \mu_{4GB} = \mu_{8GB} = \mu_{16GB}$
- $H_a : H_0$ is not true

In [64]:

```
#Slice main DF by each RAM amount type and only include the USD value.
gb4_data = clean_df[clean_df['ram_gb'] == "4 GB"]['latest_price_usd'].values
gb8_data = clean_df[clean_df['ram_gb'] == "8 GB"]['latest_price_usd'].values
gb16_data = clean_df[clean_df['ram_gb'] == "16 GB"]['latest_price_usd'].values

# Parameters for sampling
sample_size = 100
num_samples = 1000

# Generate sample means
gb4_sample_means = generate_sample_means(gb4_data, sample_size, num_samples)
gb8_sample_means = generate_sample_means(gb8_data, sample_size + 200, num_samples)
gb16_sample_means = generate_sample_means(gb16_data, sample_size, num_samples)

# Plot the distribution of sample means
fig, ax = plt.subplots(figsize=(10, 6))
sns.histplot(gb4_sample_means, kde=True, color='#1E2761', bins=20, zorder=2, label="4 G
sns.histplot(gb8_sample_means, kde=True, color='#7A2048', alpha=0.5, bins=20, zorder=2,
sns.histplot(gb16_sample_means, kde=True, color='#408EC6', bins=20, zorder=2, label="16

#Customize the plot and label axis.
plt.title(f'Distribution of Sample Means (n={sample_size} (8GB sample n=300), {num_samp
plt.xlabel('Sample Mean', fontdict=font2)
plt.ylabel('Frequency', fontdict=font2)
plt.grid(True, axis='y', linestyle='--', linewidth=0.7, color='gray', alpha=0.2, zorder
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)

#Remove lines of grid perimeter to make more appealing.
ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)
ax.spines["top"].set_visible(False)

#Set background of grid to custom color.
ax = plt.gca()
ax.set_facecolor('#FCF6F5')
plt.legend()

plt.show()
```



In [65]:

```
# Shapiro-Wilk test for normality

#Test for 4 GB samples
_, p_value_one = stats.shapiro(gb4_sample_means)
print(p_value_one)

if p_value_one > 0.05:
    print("The sample means are approximately normally distributed (fail to reject the
else:
    print("The sample means are not normally distributed (reject the null hypothesis of

#Test for 8GB samples
_, p_value_two = stats.shapiro(gb8_sample_means)
print(p_value_two)

if p_value_two > 0.05:
    print("The sample means are approximately normally distributed (fail to reject the
else:
    print("The sample means are not normally distributed (reject the null hypothesis of

#Test for 16GB samples
    _, p_value_three = stats.shapiro(gb16_sample_means)
print(p_value_three)

if p_value_three > 0.05:
    print("The sample means are approximately normally distributed (fail to reject the
else:
    print("The sample means are not normally distributed (reject the null hypothesis of
```

```
0.20212385058403015
The sample means are approximately normally distributed (fail to reject the null hypothe
sis of normality).

0.10074173659086227
The sample means are approximately normally distributed (fail to reject the null hypothe
```

```
sis of normality).

0.12228406965732574
The sample means are approximately normally distributed (fail to reject the null hypothe
sis of normality).
```

In [66]:
```python
# ANOVA Test to determine if samples are significantly different.
stat, p_value = f_oneway(gb4_sample_means, gb8_sample_means, gb16_sample_means)
print(p_value)

#Interpret results
test_outcome(p_value, alpha)
```

```
0.0
```

Out[66]:
```
'Reject the null hypothesis.'
```

Interpretation

Based on our test outcome, we have found that our Null hypothesis was not true and there is a statistical significance between the means of each group, therefore warranting further research.

Section 4: Analysis and Conclusions

Analysis

Laptops are a convenient way to either get work done or game compared to desktop computers due to their portability. This dataset revealed some interesting insights for major technology companies such as Apple, LG, Alienware, Asus, Microsoft, MSI, Lenovo, Acer, HP, and Dell based on their prices and ratings received. The average prices of each brand revealed that Alienware is the most expensive brand on average, which makes sense considering Alienware manufactures high-end video gaming PCs, followed by Apple products which are fairly expensive compared to brands such as MSI, Asus, and Lenovo laptops. The average ratings per brand revealed that, on average, Apple laptops received the highest ratings of all brands, followed by LG, MSI, Nokia, Realme, and Alienware. This is interesting considering that nearly all of the top 10 brands with the highest ratings of their laptops were priced under 900, compared to the top 10 laptops by price and their respective ratings being slightly lower.

Many factors can impact price and ratings, such as having more RAM, a better processor, a better GPU, and a larger display, which all can drive up the price. Additionally, one's needs will differ, whether for gaming, programming, video editing, or simply surfing the web. According to the data, AMD processor laptops received better ratings and are also slightly cheaper on average when compared to Intel. Intel had many more outliers compared to AMD, with Intel CPU laptops being as expensive as roughly 5,500, and the most expensive AMD CPU laptop reaching nearly 3,000! That is a massive price for both, but an over 2,000 difference in price. Based on the ratings of operating systems, the Mac operating system is rated the highest, at roughly a 4.8 rating, whereas the popular competitor Microsoft Windows is rated at roughly 4.2. Additionally, Asus Zephyrus laptop was priced to be one of the most expensive, with one model roughly 3,800 dollars and another Zephyrus model exceeding 5,000 dollars. These must be high end gaming laptops with high end GPU's. The Macbook was also one of the top models of laptop that was most expensive from the dataset.

In the Inferential Analysis portion, I asked specific questions on my data. My data was not normally distributed after plotting a KDE plot and testing for normality via a Shapiro Wilkes test. The first question I asked was, "is a significant difference in the prices between laptops with different processor brands, such as Intel and AMD?". My null hypothesis was that there is no difference in price between laptops with different processor brands such as AMD vs. Intel and my alternative hypothesis was there is a difference in price between laptops with different processor brands such as AMD vs. Intel. I used a Mann-Whitney U Test which is basically a Students Independent T-Test for nonparametric data. After performing my test, we failed to reject the null hypothesis. This means that there was no significant difference between the means of price in each sample of laptops that have intel processors and laptops that have AMD processors.

My second question was, "Do laptops with solid-state drives (SSD's) have significantly different prices compared to those with hard disk drives (HDD's)?". My null hypothesis was that there is no difference in price between laptops with SSDs compared to laptops with HDDs, and my alternative hypothesis was that there is a difference in price between laptops with SSDs compared to laptops with HDDs. I conducted a Mann-Whitney U test for this, and after running my test, we rejected the null hypothesis using an alpha value of 0.05. Based on our test outcome, it was found that there was a statistically significant difference between the means of both samples of laptops with SSDs and laptops with HDDs, and as a result, further research is warranted.

For my third inferential question, I asked, "Is there a significant difference in the prices of laptops based on their operating system?". For this test, I conducted a Kruskal-Wallis H Test, which is essentially an ANOVA test for non-parametric data. My null hypothesis was that the mean price of Windows operating system laptops is equal to the mean price of Mac operating system laptops, which is equal to the mean price of DOS operating system laptops. My alternative hypothesis is that the null hypothesis is not true. After running the test, we rejected the null hypothesis. This means that we found that our null hypothesis was not true and there was a statistically significant difference between the means of each group, which therefore warrants further research.

For my fourth question, I applied the Central Limit Theorem (CLT) to normalize my data and asked two additional questions. First, I asked, "Is there a significant difference in the ratings between different brands, such as Asus and HP?". My null hypothesis was that there is no difference between the ratings of different brands such as Asus and HP, and my alternative hypothesis was that there is a difference between the ratings of different brands such as Asus and HP. I conducted an Independent Student's T-Test for this question, and after running the test, we rejected the null hypothesis. There was found to be a significant difference in the means between the ratings of laptops based on their brand, specifically between HP and Asus, and therefore warrants additional research.

Finally, the last inferential question I asked was, "Do laptops with higher RAM tend to have higher prices compared to those that do not?". This involved three groups: 4GB RAM, 8GB RAM, and 16GB RAM. My null hypothesis was that the mean prices for 4GB RAM laptops are equal to the mean prices for 8GB RAM laptops, which are also equal to the mean prices for 16GB RAM laptops. My alternative hypothesis is that the null hypothesis is not true. For this, I conducted an ANOVA test. After running my test, we rejected the null hypothesis. We found that our null hypothesis was not true and there is a statistical significance between the means of each group, therefore warranting further research.

Recommendation

Companies like Apple are doing a really good job, and I encourage Apple to continue doing what they are doing and investing in market research so that they can continue to understand the market and consumer needs. More well-known companies such as Dell, Acer, and HP laptops have an average rating above 4.0, which is great, but there is room for improvement, being on the lower end of 4's when you compare them to Apple with an average rating of 4.7. Alienware is rated highly at 4.4, despite being on average $3,000 per laptop. Dell, Acer, and HP should consider making more laptops with AMD processors as they are cheaper than Intel processors on average and rated higher, which may indicate that consumers are happier with AMD and spending less money. These companies should also consider adding a GPU in their laptops, regardless of VRAM amount, as ratings on laptops with a GPU are higher than those without. I would not recommend Apple to use Windows operating systems as they are rated lower compared to the Mac operating system. However, companies like Dell, Acer, and HP can continue to use Windows, or potentially partner with Apple to explore their operating system and work together to build a new OS. Laptops with SSDs are more expensive on average compared to laptops with HDDs, but SSDs are much faster, which can improve customer satisfaction. I recommend making more laptops with SSDs.