Find a Graphics Processing Unit Analysis

by Joshua Nguyen

Abstract Graphics Processing Units or GPUs are currently being sold at an all time high due to a plethora of factors such as chip shortages and coin mining. Low supply and high demand drives GPU's prices high. Fortunately, Microcenter is an American computer retail store that sells most GPU at a manufacture suggested retail price.

Project Overview

1. This project explores different statistical metrics to analyze and assay *GPU* prices sold at MicroCenter. Additionally, this project will display the results that is easily read and interpreted by the non-technical population who may not easily understand statistical vernacular.
2. This project aims to provide analytical and visual **real-time** data for any potential GPU buyers at MicroCenter.
3. This project utilizes **Python** for data extraction, transformation and data analytics and visualization and the **Streamlit** framework for web application.

Objectives

The overall goal of this project is to build an interactive dashboard that showcases real time metrics along with visual analytical representation about current GPU's sold in all available Microcenter stores in the USA.

Questions of Interest

1. What are the top five most expensive GPU's sold at the Houston, TX Microcenter store?
2. What are the top five least expensive GPU's sold at the Houston, TX Microcenter store?
3. What are the mean and median price for all GPU products sold at the Houston, TX Microcenter store?
4. At the Houston, TX Microcenter store, how many different GPU products are being sold and what is the inventory count?
5. At the Houston, TX Microcenter store, what is the mean review score and total review count for all of their GPU products?
6. When filtering for specific product brands and product prices, how are the inventory count distributed?
7. When filtering for specific product brands and product prices, what is the distribution of the prices?
8. When filtering for specific product brands and product prices, what is the distribution of the review count?

Extraction, Transformation and Preparation

Import Libraries

Extracting The Data

This project uses **BeautifulSoup** for webscraping and data extraction. First, we will create a function to get the user's headers.

```python
# Import Libraries For Data Backend
from modulefinder import STORE_NAME
import re
from bs4 import BeautifulSoup
from numpy import choose
from prometheus_client import Metric
import requests
from urllib.request import urlopen
import pandas as pd
from pprint import pprint

# Import Libraries For Streamlit App
import plotly.express as px
from soupsieve import select
import streamlit as st
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```python
def get_headers():
    r = requests.get('http://httpbin.org/headers')

    your_head = r.json()
    user_agent = your_head['headers']['User-Agent']

    header_dictionary = {
    'User-Agent' : user_agent
    }
    return header_dictionary

your_header = get_headers()
```

Next, we will create the functions to paginate through each products page.

```python
# Function to Find Soup JSON For Page 1
def find_soup(integer):
    (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36'}
    headers  = your_header
    URL = f'https://www.microcenter.com/search/search_results.aspx?
        Ntk=all&sortby=match&rpp=96&N=4294966937&myStore=true&storeid={integer}&page=1'
    response = requests.get(URL, headers=headers)
    soup = BeautifulSoup(response.content, 'html.parser')
    return soup
```

```python
# Function to Find Soup JSON For Page 2
def find_soup2(integer):
    (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36'}
    headers  = your_header
    URL = f'https://www.microcenter.com/search/search_results.aspx?
        Ntk=all&sortby=match&rpp=96&N=4294966937&myStore=true&storeid={integer}&page=2'
    response = requests.get(URL, headers=headers)
    soup = BeautifulSoup(response.content, 'html.parser')
    return soup
```

Lets get the product names, prices and brands.

```python
# Function to Find Product Information For One Store
def find_info(soup, integer):
    categories = soup.find_all('a', {'id':f'hypProductH2_{integer}'})
    for d in categories:
        data_name = (d.get('data-name'))
        data_brand = (d.get('data-brand'))
        data_price = (d.get('data-price'))
        href_link = 'https://www.microcenter.com'+ d.get('href')
        return data_name, data_brand, data_price, href_link
```

Here, we seek to find the rating counts and number of reviews.

```python
# Function to Find Ratings For Store Products
def find_ratings(soup, integer):
    categories = soup.find_all('li', {'id':f'pwrapper_{integer}', 'class' :
        'product_wrapper'})
    ratelist = []
    for soup_class in categories:
        reviews = soup_class.find_all('div', {'class' : 'ratingstars'})
        for soup_class in reviews:
            div = soup_class.find_all('div')
            for i in div:
                for j in i:
                    ratelist.append(j.text)

    if ratelist[0] == '0 Reviews':
        stars = 0
        numbers = 0
        return stars, numbers
    else:
        stars = int(str(ratelist[0])[0:1])
        numbers = int(re.findall(r'\d+', ratelist[1])[0])
        return stars, numbers
```

It is important to get the exact amount of products on each paginated page of the products.

```python
# Function to Find Inventory Count For Store Products
def find_inventory(soup, integer):
    categories = soup.find_all('li', {'id':f'pwrapper_{integer}', 'class' :
        'product_wrapper'})
    for soup_class in categories:
        inventory = soup_class.find_all('div', {'class' : 'stock'})
        for cut in inventory:
            return(str(cut.text))
```

```python
# Function to Find Maximum Number Of Items Displayed For One Store
def find_item_num(soup):
    emplist = []
    categories = soup.find_all('div', {'id':'bottomPagination', 'class' : 'pagination'})
    for soup_class in categories:
        num = soup_class.find_all('p', {'class' : 'status'})
        for i in num:
            emplist.append(i.text)

    numbers = (re.findall(r'\d+', emplist[0]))
    return (int(numbers[2]))
```

Finally, we combine functions above to get our target information.

```python
# Function For A List of All Product Information For One Store
def create_list(store_id, store_integer):
    soup = find_soup(store_id)
    newlist = []
    if store_integer <= 96:
        for integer in range(0,store_integer):
            inventory_count = find_inventory(soup,integer).strip()
            inv_count = (re.findall(r'\d+', inventory_count))
            listprime = {
                'product_name' : find_info(soup,integer)[0],
                'brand_name' : find_info(soup,integer)[1],
                'product_price' : find_info(soup,integer)[2],
                'product_link' : find_info(soup,integer)[3],
                'product_star_count' : find_ratings(soup,integer)[0],
                'product_review_count' : find_ratings(soup,integer)[1],
                'product_inventory_count' : inv_count[0],
                'product_inventory_name' : inventory_count
            }
            newlist.append(listprime)

    else:
        newsoup = find_soup2(store_id)
        second_page_integer = int(store_integer - 96)

        # Getting products between 1 and max items of 96
        for integer in range(0,96):
            inventory_count = find_inventory(soup,integer).strip()
            inv_count = (re.findall(r'\d+', inventory_count))
            listprime = {
                'product_name' : find_info(soup,integer)[0],
                'brand_name' : find_info(soup,integer)[1],
                'product_price' : find_info(soup,integer)[2],
                'product_link' : find_info(soup,integer)[3],
                'product_star_count' : find_ratings(soup,integer)[0],
                'product_review_count' : find_ratings(soup,integer)[1],
                'product_inventory_count' : inv_count[0],
                'product_inventory_name' : inventory_count
            }
            newlist.append(listprime)

        # Getting products between 97 - store_integer
        newurl = bottom_pagination(find_soup(store_integer))

        for integer in range(0,second_page_integer):
            inventory_count = find_inventory(newsoup,integer).strip()
            inv_count = (re.findall(r'\d+', inventory_count))
            listprime = {
                'product_name' : find_info(newsoup,integer)[0],
                'brand_name' : find_info(newsoup,integer)[1],
                'product_price' : find_info(newsoup,integer)[2],
                'product_link' : find_info(newsoup,integer)[3],
                'product_star_count' : find_ratings(newsoup,integer)[0],
                'product_review_count' : find_ratings(newsoup,integer)[1],
                'product_inventory_count' : inv_count[0],
                'product_inventory_name' : inventory_count
            }
            newlist.append(listprime)
    return newlist
```

```
# Function For Pandas Dataframe
def get_df(store_list):
    store_data = pd.DataFrame(store_list, columns=['product_name', 'brand_name',
        'product_price', 'product_link', 'product_star_count','product_review_count',
            'product_inventory_count'])
    return store_data
```

```
# Final Call Function
@st.cache(allow_output_mutation=True, show_spinner=False)
def get_data(store_name):
        store_name_id = find_soup(store_ids[store_name])
        item_num = find_item_num(store_name_id)

        store_df = get_df(create_list(store_ids[store_name],item_num))
        store_df['product_inventory_count'] =
            store_df['product_inventory_count'].astype('int64')
        store_df['product_price'] = store_df['product_price'].astype('float')
        return store_df
```

Data Analysis and Visualizations

Pie Chart

```
def pie_chart(df):
    pie_data = df['product_inventory_count'].to_list()
    pie_labels = df.index
    colors = sns.color_palette("hls",8)

    pie_label_size = pie_labels.size
    explode_list = []
    for i in range(0,pie_label_size):
        explode_list.append(0.02)

    max_value = max(pie_data)
    max_value_index = pie_data.index(max_value)
    explode_list[max_value_index] = 0.03

    pie_explode = explode_list

    plt.pie(pie_data, labels=pie_labels, colors=colors, autopct = '%0.0f%%',
        explode=explode_list, shadow= False, startangle=90,
        textprops={'color': 'Black', 'fontsize':25},
        wedgeprops={'linewidth':6},
        center=(0.1,0.1), rotatelabels=True)
    plt.rcParams["figure.figsize"] = [50,50]

    gif_runner = st.image('processing.gif')

    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.pyplot()

    gif_runner.empty()
```

Bar Chart

```python
def bar_chart(df):
    top_five_price = df.sort_values(by='product_price', ascending=False).nlargest(5,
        'product_price')

    colors = sns.color_palette("hls",8)

    fig = plt.subplots(figsize=[15,7])
    ax = sns.barplot(x=top_five_price.index, y = 'product_price', data = top_five_price,
        palette= colors)

    for bar, label in zip(ax.patches, top_five_price['product_price']):
        x = bar.get_x()
        width = bar.get_width()
        height = bar.get_height()
        ax.text(x+width/2., height + 1, label, ha="center")

    gif_runner = st.image('processing.gif')

    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.pyplot()

    gif_runner.empty()
```

Grouped Circular Bar Chart

```python
def get_label_rotation(angle, offset):
    # Rotation must be specified in degrees :(
    rotation = np.rad2deg(angle + offset)
    if angle <= np.pi:
        alignment = "right"
        rotation = rotation + 180
    else:
        alignment = "left"
    return rotation, alignment

def add_labels(angles, values, labels, offset, ax):

    # This is the space between the end of the bar and the label
    padding = 4

    # Iterate over angles, values, and labels, to add all of them.
    for angle, value, label, in zip(angles, values, labels):
        angle = angle

        # Obtain text rotation and alignment
        rotation, alignment = get_label_rotation(angle, offset)

        # And finally add the text
        ax.text(
            x=angle,
            y=value + padding,
            s=label,
            ha=alignment,
            va="center",
            rotation=rotation,
            rotation_mode="anchor"
        )

def grouped_bar_chart(test_df):
    test_df_sorted = (test_df.groupby(['brand_name']).apply(
        lambda x: x.sort_values(["product_review_count"],
        ascending = False)).reset_index(drop=True))

    VALUES = test_df_sorted["product_review_count"].values
    LABELS = test_df_sorted["product_name"].values
    GROUP = test_df_sorted["brand_name"].values

    PAD = 3
    ANGLES_N = len(VALUES) + PAD * len(np.unique(GROUP))

    ANGLES = np.linspace(0, 2 * np.pi, num=ANGLES_N, endpoint=False)
    WIDTH = (2 * np.pi) / len(ANGLES)


    OFFSET = np.pi / 2

    # Specify offset
    #ax.set_theta_offset(OFFSET)
    offset = 0
    IDXS = []

    GROUPS_SIZE = []
    unique, counts = np.unique(GROUP, return_counts=True)
    result = np.column_stack((unique, counts))

    for i in range(0, len(result)):
        GROUPS_SIZE.append(result[i][1])
    for size in GROUPS_SIZE:
        IDXS += list(range(offset + PAD, offset + size + PAD))
        offset += size + PAD

    fig, ax = plt.subplots(figsize=(20, 10), subplot_kw={"projection": "polar"})

    ax.set_theta_offset(OFFSET)
    ax.set_ylim(-100, 100)
    ax.set_frame_on(False)
    ax.xaxis.grid(False)
    ax.yaxis.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])

    GROUPS_SIZE = []
    unique, counts = np.unique(GROUP, return_counts=True)
    result = np.column_stack((unique, counts))

    for i in range(0, len(result)):
        GROUPS_SIZE.append(result[i][1])
    COLORS = [f"C{i}" for i, size in enumerate(GROUPS_SIZE) for _ in range(size)]

    # Add bars to represent ...
    ax.bar(
        ANGLES[IDXS], VALUES, width=WIDTH, color=COLORS,
        edgecolor="white", linewidth=2
    )

    add_labels(ANGLES[IDXS], VALUES, LABELS, OFFSET, ax)

    offset = 0
    test_list = unique.tolist()
    for group, size in zip(test_list, GROUPS_SIZE):
        # Add line below bars
        x1 = np.linspace(ANGLES[offset + PAD], ANGLES[offset + size + PAD - 1], num=50)
        ax.plot(x1, [-5] * 50, color="#333333")

        # Add text to indicate group
        ax.text(
            np.mean(x1), -20, group, color="#333333", fontsize=14,
            fontweight="bold", ha="center", va="center"
        )

        # Add reference lines at 20, 40, 60, and 80
        x2 = np.linspace(ANGLES[offset], ANGLES[offset + PAD - 1], num=50)
        ax.plot(x2, [20] * 50, color="#bebebe", lw=0.8)
        ax.plot(x2, [40] * 50, color="#bebebe", lw=0.8)
        ax.plot(x2, [60] * 50, color="#bebebe", lw=0.8)
        ax.plot(x2, [80] * 50, color="#bebebe", lw=0.8)

        offset += size + PAD

    gif_runner = st.image('processing.gif')

    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.pyplot()

    gif_runner.empty()
```

Lollipops

```
def most_expensive(test_df):
    prices = test_df.sort_values(by=['product_price'],
        ascending=False).head(10).sort_values('product_price')
    my_range = range(0,len(prices))

    fig = plt.figure(figsize=(14,10))

    plt.hlines(y=prices['product_name'],
        xmin=0,
        xmax=prices['product_price'],
        color='black')
    plt.plot(prices['product_price'], my_range, "o", color = 'black')
    plt.xlabel('Price (USD)', fontsize=20)
    plt.ylabel('Product Name',fontsize=20)
    plt.yticks(fontsize=15)
    plt.xticks(fontsize=15)
    plt.xlim(0,max(prices['product_price'])+100)
    plt.grid()
    plt.title("Most Expensive Products", fontsize=20, x=0.5,y=1.02)

    gif_runner = st.image('processing.gif')
    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.pyplot()
    gif_runner.empty()

def least_expensive(test_df):
    prices = test_df.sort_values(by=['product_price'],
        ascending=True).head(10).sort_values('product_price')
    my_range = range(0,len(prices))

    fig = plt.figure(figsize=(14,10))

    plt.hlines(y=prices['product_name'],
        xmin=0,
        xmax=prices['product_price'],
        color='black')
    plt.plot(prices['product_price'], my_range, "o", color = 'black')
    plt.xlabel('Price (USD)', fontsize=20)
    plt.ylabel('Product Name',fontsize=20)
    plt.yticks(fontsize=15)
    plt.xticks(fontsize=15)
    plt.xlim(0,max(prices['product_price'])+100)
    plt.grid()
    plt.title("Least Expensive Products", fontsize=20, x=0.5,y=1.02)

    gif_runner = st.image('processing.gif')
    st.set_option('deprecation.showPyplotGlobalUse', False)
    st.pyplot()
    gif_runner.empty()
```

Web Application

First, we need a dictionary to store all the available Microcenter stores and their respective ID's in the USA.

```
# Dictionary Of All Micro Center Stores in The USA
store_ids = {
'CA-Tustin' : 101,
'CO-Denver' : 181,
'GA-Duluth' : 65,
'GA-Marietta' : 41,
'IL-Chicago' : 151,
'IL-Wesmont' : 25,
'KS-Overland Park' : 191,
'MA-Cambridge' : 121,
'MD-Rockville' : 85,
'MI-Madison Heights' : 55,
'MN-St. Louis Park' : 45,
'MO-Brentwood' : 95,
'NJ-North Jersey' : 75,
'NY-Westbury' : 171,
'NY-Brooklyn' : 115,
'NY-Flushing' : 145,
'NY-Yonkers' : 105,
'OH-Columbus' : 141,
'OH-Mayfield Heights' : 51,
'OH-Sharonville' : 71,
'PA-St. Davids' : 61,
'TX-Houston' : 155,
'TX-Dallas' : 131,
'VA-Fairfax' : 81,
}
```

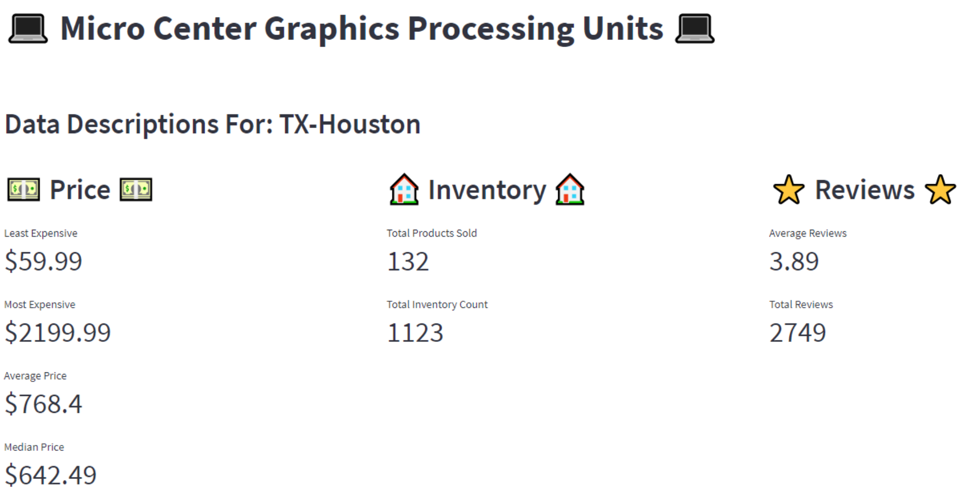
Total Dataset For Entire Store

```python
if start_execution or st.session_state.load_state:
    st.session_state.load_state = True
    with col2:
        main_gif.empty()
        gif_runner = st.image('processing.gif')
        st.session_state['data_frame'] = get_data(dropdown)
        data_frame = st.session_state['data_frame']
        data_frame['product_price'] = data_frame['product_price'].astype(float)
        data_frame['product_inventory_count'] = data_frame['product_inventory_count']
            .astype(int)
        gif_runner.empty()

    # ----- Top KPI ------
    max_high = data_frame['product_price'].max()
    min_low = data_frame['product_price'].min()
    average_price = round(data_frame['product_price'].mean(),2)
    median_price = data_frame['product_price'].median()
    total_inventory = data_frame['product_inventory_count'].sum()
    average_review = round(data_frame['product_star_count'].mean(),2)
    total_review_count = data_frame['product_review_count'].sum()

    # ----- SIDE BAR ----- #
    st.sidebar.subheader('Brand')
    selected_brands = st.sidebar.multiselect(label = "",
        options = data_frame['brand_name'].unique())
    st.sidebar.subheader('Price')
    min_value = st.sidebar.slider(label="Minimum Price",
        min_value= round(min_low),
        max_value=round(max_high))
    if min_value:
        max_value = st.sidebar.slider(label="Maximum Price",
            min_value= round(min_value),
            max_value=round(max_high))
    filter_inventory = st.sidebar.radio(label='Filter Inventory',
        options= ('Include All Items', 'Exclude Sold Out Items'))
    filter_reviews = st.sidebar.radio(label='Filter Reviews',
        options= ('Include All Items', 'Exclude Items With No Reviews'))
    filter_data = st.sidebar.button('Filter Products')

    selected_store = [key for key, value in store_ids.items()
        if value == store_ids[dropdown]][0]
    st.header(f"Data Descriptions For: {selected_store}")
    left_column, middle_column, right_column = st.columns(3)
    with left_column:
        st.header(":dollar: Price :dollar:")
        st.metric(label="Least Expensive", value = f"${min_low}")
        st.metric(label="Most Expensive", value = f"${max_high}")
        st.metric(label="Average Price", value = f"${average_price}")
        st.metric(label="Median Price", value = f"${median_price}")
    with middle_column:
        st.header(":house: Inventory :house:")
        st.metric(label="Total Products Sold", value = data_frame['product_name'].count())
        st.metric(label="Total Inventory Count", value = total_inventory)
    with right_column:
        st.header(":star: Reviews :star:")
        st.metric(label="Average Reviews", value = average_review)
        st.metric(label="Total Reviews", value = total_review_count)

    st.markdown("---")
    st.header("All Products For Your Store :point_down:")
    st.dataframe(data_frame)

    most_expensive(data_frame)

    least_expensive(data_frame)
```

Filtered Dataset For User Selected Brands And Price Range

```
if filter_data or st.session_state.load_state2:
        st.session_state.load_state2 = True
        if filter_inventory == 'Include All Items':
            data_frame_prime = data_frame[data_frame['brand_name'].isin(selected_brands)]
            data_frame_prime = data_frame_prime[data_frame_prime['product_price']
                > min_value]
            data_frame_prime = data_frame_prime[data_frame_prime['product_price']
                < max_value]
            if filter_reviews == 'Include All Items':
                data_frame_prime = data_frame_prime
            else:
                data_frame_prime =
                    data_frame_prime[data_frame_prime['product_review_count'] > 0]
        else:
            data_frame_prime = data_frame[data_frame['product_inventory_count']>0]
            data_frame_prime = data_frame_prime[data_frame['brand_name']
                .isin(selected_brands)]
            data_frame_prime = data_frame_prime[data_frame_prime['product_price']
                > min_value]
            data_frame_prime = data_frame_prime[data_frame_prime['product_price']
                < max_value]
            if filter_reviews == 'Include All Items':
                data_frame_prime = data_frame_prime
            else:
                data_frame_prime =
                    data_frame_prime[data_frame_prime['product_review_count'] > 0]

        # ----- Filtered KPI ------ #
        max_high_prime = data_frame_prime['product_price'].max()
        min_low_prime = data_frame_prime['product_price'].min()
        average_price_prime = round(data_frame_prime['product_price'].mean(),2)
        median_price_prime = data_frame_prime['product_price'].median()
        total_inventory_prime = data_frame_prime['product_inventory_count'].sum()
        average_review_prime = round(data_frame_prime['product_star_count'].mean(),2)
        total_review_count_prime = data_frame_prime['product_review_count'].sum()

        st.markdown("---")
        st.header(f"Data Descriptions For:
            {selected_brands} With Prices Between {min_value} And {max_value} USD")
        left_column, middle_column, right_column = st.columns(3)
        with left_column:
            st.header(":dollar: Price :dollar:")
            st.metric(label="Least Expensive", value = f"${min_low_prime}")
            st.metric(label="Most Expensive", value = f"${max_high_prime}")
            st.metric(label="Average Price", value = f"${average_price_prime}")
            st.metric(label="Median Price", value = f"${median_price_prime}")
        with middle_column:
            st.header(":house: Inventory :house:")
            st.metric(label="Total Products Sold", value =
                data_frame_prime['product_name'].count())
            st.metric(label="Total Inventory Count", value = total_inventory_prime)
        with right_column:
            st.header(":star: Reviews :star:")
            st.metric(label="Average Reviews", value = average_review_prime)
            st.metric(label="Total Reviews", value = total_review_count_prime)

        # ----- Pie Chart ------ #
        # Distribution of Product Inventory
        grouped_df = data_frame_prime.groupby(by=['brand_name']).sum()

        graph_col1, graph_col2 = st.columns(2)

        with graph_col1:
            st.header(':bank: Distribution of Inventory :bank:')
            pie_chart(grouped_df)

        # ----- Bar Plot ------ #
        # Top Most Expensive Brand
        with graph_col2:
            st.header(':moneybag: Distribution of Price :moneybag:')
            bar_chart(grouped_df)

        # ----- Grouped Bar Chart ------ #
        # Total Review Count
        st.header(':star2: Distribution of Review Count :star2:')
        grouped_bar_chart(data_frame_prime)


        st.header("Your Products :point_down:")
        st.dataframe(data_frame_prime)
```

Final Website



💻 Micro Center Graphics Processing Units 💻

Data Descriptions For: TX-Houston

💵 Price 💵

Least Expensive
$59.99

Most Expensive
$2199.99

Average Price
$768.4

Median Price
$642.49

🏠 Inventory 🏠

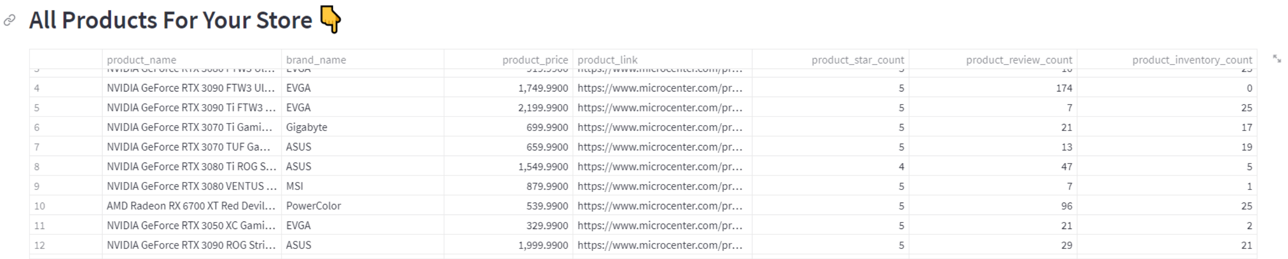
Total Products Sold
132

Total Inventory Count
1123

⭐ Reviews ⭐

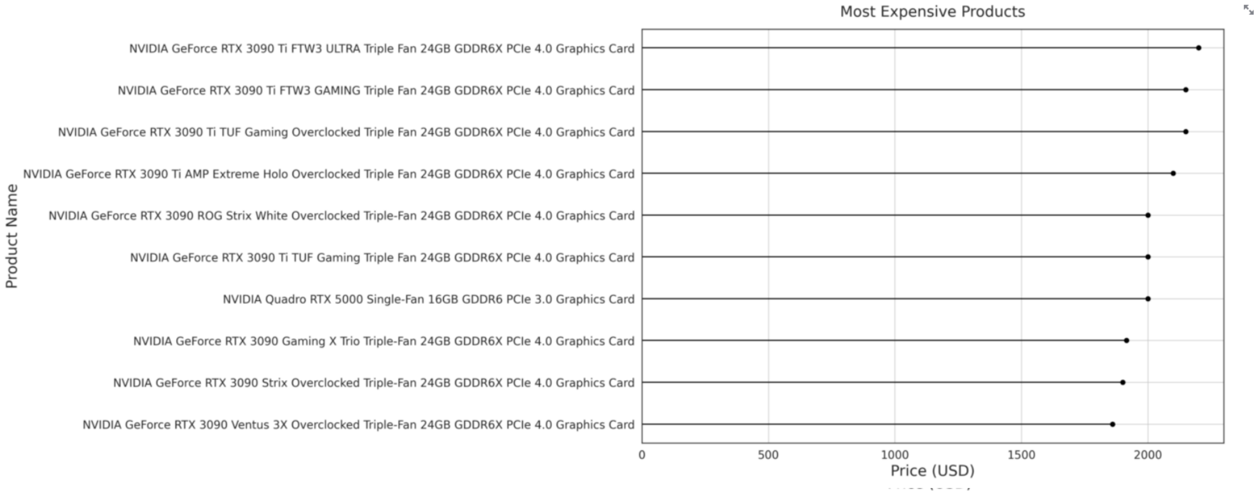
Average Reviews
3.89

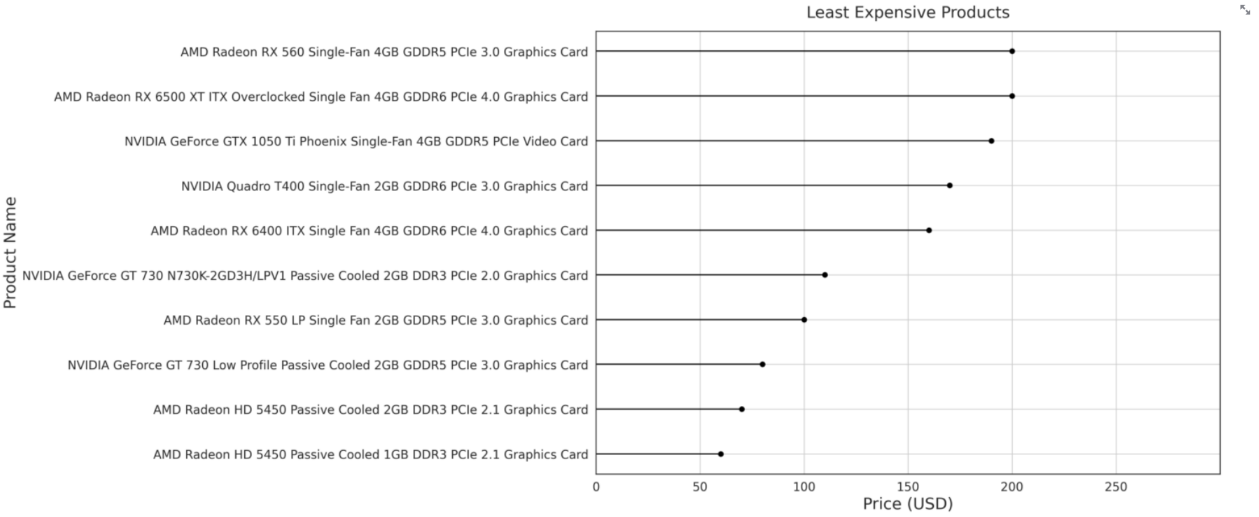
Total Reviews
2749



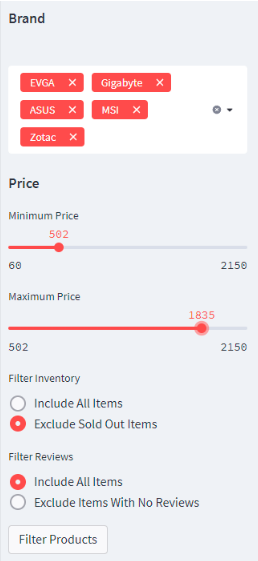
All Products For Your Store 👇

	product_name	brand_name	product_price	product_link	product_star_count	product_review_count	product_inventory_count
3	NVIDIA GeForce RTX 3080 FTW3 Ul…	EVGA	[illegible]	https://www.microcenter.com/pr…	5	[illegible]	[illegible]
4	NVIDIA GeForce RTX 3090 FTW3 Ul…	EVGA	1,749.9900	https://www.microcenter.com/pr…	5	174	0
5	NVIDIA GeForce RTX 3090 Ti FTW3 …	EVGA	2,199.9900	https://www.microcenter.com/pr…	5	7	25
6	NVIDIA GeForce RTX 3070 Ti Gami…	Gigabyte	699.9900	https://www.microcenter.com/pr…	5	21	17
7	NVIDIA GeForce RTX 3070 TUF Ga…	ASUS	659.9900	https://www.microcenter.com/pr…	5	13	19
8	NVIDIA GeForce RTX 3080 Ti ROG S…	ASUS	1,549.9900	https://www.microcenter.com/pr…	4	47	5
9	NVIDIA GeForce RTX 3080 VENTUS …	MSI	879.9900	https://www.microcenter.com/pr…	5	7	1
10	AMD Radeon RX 6700 XT Red Devil…	PowerColor	539.9900	https://www.microcenter.com/pr…	5	96	25
11	NVIDIA GeForce RTX 3050 XC Gami…	EVGA	329.9900	https://www.microcenter.com/pr…	5	21	2
12	NVIDIA GeForce RTX 3090 ROG Stri…	ASUS	1,999.9900	https://www.microcenter.com/pr…	5	29	21





Filtered Products



Brand

EVGA ✕ Gigabyte ✕ ASUS ✕ MSI ✕ Zotac ✕

Price

Minimum Price: 502 (60 – 2150)

Maximum Price: 1835 (502 – 2150)

Filter Inventory

○ Include All Items
● Exclude Sold Out Items

Filter Reviews

● Include All Items
○ Exclude Items With No Reviews

Filter Products

Data Descriptions For: ['EVGA', 'Gigabyte', 'ASUS', 'MSI', 'Zotac'] With Prices Between 502 And 1835 USD

💵 Price 💵

Least Expensive
$529.99

Most Expensive
$1699.99

Average Price
$1009.16

Median Price
$919.99

🏠 Inventory 🏠

Total Products Sold
36

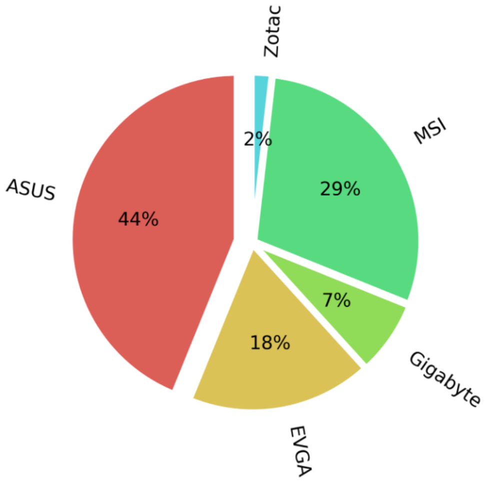
Total Inventory Count
447

⭐ Reviews ⭐

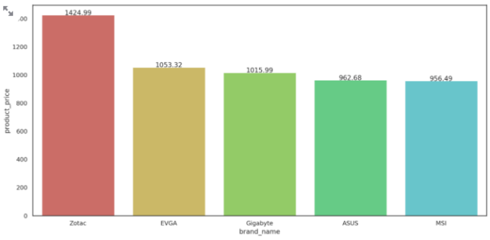
Average Reviews
3.92

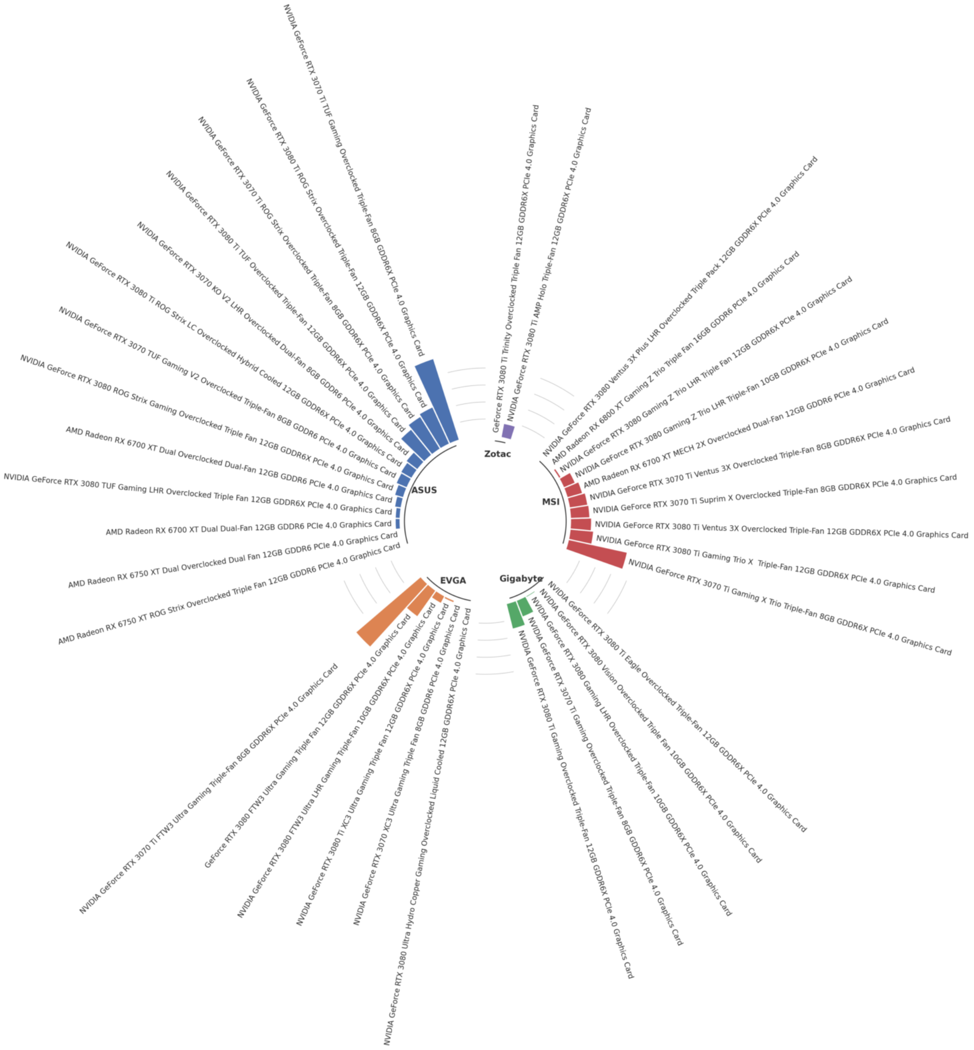
Total Reviews
782

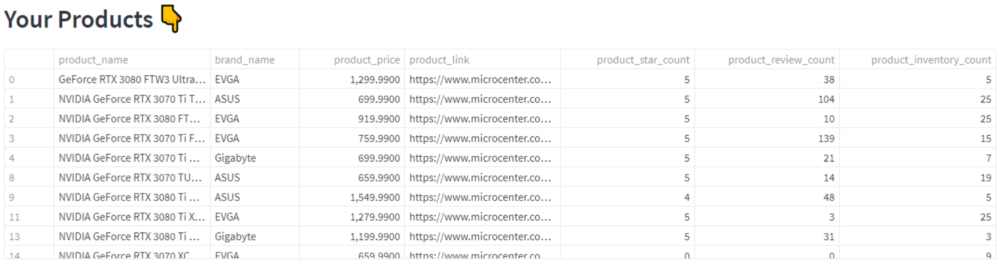
🏢 Distribution of Inventory 🏢



💰 Distribution of Average Price 💰







Your Products 👇

	product_name	brand_name	product_price	product_link	product_star_count	product_review_count	product_inventory_count
0	GeForce RTX 3080 FTW3 Ultra…	EVGA	1,299.9900	https://www.microcenter.co…	5	38	5
1	NVIDIA GeForce RTX 3070 Ti T…	ASUS	699.9900	https://www.microcenter.co…	5	104	25
2	NVIDIA GeForce RTX 3080 FT…	EVGA	919.9900	https://www.microcenter.co…	5	10	25
3	NVIDIA GeForce RTX 3070 Ti F…	EVGA	759.9900	https://www.microcenter.co…	5	139	15
4	NVIDIA GeForce RTX 3070 Ti …	Gigabyte	699.9900	https://www.microcenter.co…	5	21	7
8	NVIDIA GeForce RTX 3070 TU…	ASUS	659.9900	https://www.microcenter.co…	5	14	19
9	NVIDIA GeForce RTX 3080 Ti …	ASUS	1,549.9900	https://www.microcenter.co…	4	48	5
11	NVIDIA GeForce RTX 3080 Ti X…	EVGA	1,279.9900	https://www.microcenter.co…	5	3	25
13	NVIDIA GeForce RTX 3080 Ti …	Gigabyte	1,199.9900	https://www.microcenter.co…	5	31	3
14	NVIDIA GeForce RTX 3070 XC	EVGA	659.9900	https://www.microcenter.co	0	0	9

Conclusion

First, the five most expensive GPU's sold at the Houston Texas MicroCenter Store are:

1. NVIDIA GeForce RTX 3090 Ti FTW3 Ulta Triple Fan 24 GB
2. NVIDIA GeForce RTX 3090 Ti FTW3 Gaming Triple Fan 24 GB
3. NVIDIA GeForce RTX Ti TuF Gaming Overclocked Triple Fan 24 GB
4. NVIDIA GeForce RTX 3090 Ti AMP Extreme Holo Overclocked Triple Fan 24 GB
5. NVIDIA FeForce RTX 3090 ROG Strix White Overclocked Triple-Fan 24 GB

The five least expensive GPU's sold at this store are:

1. AMD Radeon RX 560 Single-Fan 4 GB

2. AMD Radeon RX 6500 XT ITX Overclocked Single Fan 4 GB
3. NVIDIA Geforce GTX 1050 Ti Phoenix Single-Fan 4 GB
4. NVIDIA Quadro T400 Single-Fan 2 GB
5. AMD Radeon RX 6400 ITX Single Fan 4 GB

Next, the mean price for all GPU's sold at the Houston Texas store is $768.40, and the median price for all GPU's sold is $642.49. There are a total of 132 different unique GPU's sold here and a total GPU inventory count of 1123 GPU's. Furthermore, the average review score here is 3.89 and the total review count for all GPU's sold here is 2749 reviews.

When filtering for the following GPU brands,

1. EVGA
2. Gigabyte
3. ASUS
4. MSI
5. Zotac

We see that the ASUS brand makes up the majority of the GPU Brands in our filtered products at the Houston Texas MicroCenter store.

In addition, we see that the price distribution for the filtered products is that the Zotac Brand is on average most expensive, and the MSI brand is the cheapest brand on average.

Lastly, the review distribution for our filtered GPU brands is that the ASUS brand has the most reviews and the Zotac brand possesses the least reviews.

Implications

Overall, if you are looking for a GPU in the Houston Texas MicroCenter, then you should buy the ASUS brand which on average has the most review counts and inventory here at this store. It is also in the intermediate price range.