GPU MEMORY CLOCK
EXPLATORY DATA ANALYSIS (EDA)

GUARDIAN TRI ANGGORO

DATA SCIENTIST PORTOFOLIO

TABLE OF CONTENT

DATASET INFORMATION

The dataset contains Graphics Card Memory NVIDIA and AMD. A Graphics Card is nothing more than another processor that is specially design and made to handle graphics. Adding one of these to your computer will take the load of processing graphics away from your CPU, allowing your CPU to handle other tasks.

source:

https://www.kaggle.com/datasets/alanjo/graphics-card-full-specs

OBJECTIVES

This project objectives:

1. Which brand dominates the most?
2. Top five product with the best GPU Memory Performance
3. Showing the correlation between variables

MISSING VALUES & DUPLICATED DATA

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2889 entries, 0 to 2888
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   manufacturer   2889 non-null   object
 1   productName    2889 non-null   object
 2   memSize        2477 non-null   float64
 3   memBusWidth    2477 non-null   float64
 4   gpuClock       2889 non-null   int64
 5   memClock       2477 non-null   float64
 6   unifiedShader  2065 non-null   float64
 7   tmu            2889 non-null   int64
 8   rop            2889 non-null   int64
 9   bus            2889 non-null   object
 10  memType        2889 non-null   object
dtypes: float64(4), int64(3), object(4)
memory usage: 248.4+ KB
```

```
[ ]  df.duplicated().sum()

     9
```

Observation :

1. There are 2889 rowss and 11 columns in dataset
2. There are four data that have a missing value. (memSize, memBusWidth, MemClock, and unifiedShader).
3. There are nine duplicated data

MISSING VALUES & DUPLICATED DATA

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2889 entries, 0 to 2888
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   manufacturer   2889 non-null   object
 1   productName    2889 non-null   object
 2   memSize        2477 non-null   float64
 3   memBusWidth    2477 non-null   float64
 4   gpuClock       2889 non-null   int64
 5   memClock       2477 non-null   float64
 6   unifiedShader  2065 non-null   float64
 7   tmu            2889 non-null   int64
 8   rop            2889 non-null   int64
 9   bus            2889 non-null   object
 10  memType        2889 non-null   object
dtypes: float64(4), int64(3), object(4)
memory usage: 248.4+ KB
```

```
[ ]  df.duplicated().sum()

     9
```

Observation :

1. There are 2889 rowss and 11 columns in dataset
2. There are four data that have a missing value. (memSize, memBusWidth, MemClock, and unifiedShader).
3. There are nine duplicated data

HANDLING MISSING VALUES

```
df['memSize'].fillna(df['memSize'].median)
```

```
0        8.0
1        4.0
2        4.0
3        4.0
4        8.0
       ...
2884   0.016
2885   0.016
2886   0.032
2887     4.0
2888     8.0
Name: memSize, Length: 2889, dtype: object
```

```
[ ] df['memBusWidth'].fillna(df['memBusWidth'].median)
```

```
0       128.0
1        64.0
2        64.0
3        64.0
4       128.0
        ...
2884    128.0
2885    128.0
2886    128.0
2887    128.0
2888    128.0
Name: memBusWidth, Length: 2889, dtype: object
```

```
[ ] df['memClock'].fillna(df['memClock'].median)
```

```
0       2250.0
1       1500.0
2       1500.0
3       1500.0
4       1500.0
         ...
2884     166.0
2885     166.0
2886     166.0
2887    2133.0
2888    2133.0
Name: memClock, Length: 2889, dtype: object
```

```
[ ] df['unifiedShader'].fillna(df['unifiedShader'].median)
```

```
0                                                    3840.0
1                                                     768.0
2                                                    1024.0
3                                                    1024.0
4                                                    2048.0
                                  ...
2884    <bound method NDFrame._add_numeric_operations....
2885    <bound method NDFrame._add_numeric_operations....
2886    <bound method NDFrame._add_numeric_operations....
2887                                                  640.0
2888                                                  768.0
Name: unifiedShader, Length: 2889, dtype: object
```

```
[ ] df = df.dropna()
```

```
df.isna().sum()
```

```
manufacturer    0
productName     0
memSize         0
memBusWidth     0
gpuClock        0
memClock        0
unifiedShader   0
tmu             0
rop             0
bus             0
memType         0
dtype: int64
```

There is no missing value

Observation :

fill in the values that have missing values with the median

HANDLING DUPLICATED VALUE

```
[ ] df.duplicated().sum()

    9
```

```
df[df.duplicated(keep=False)].sort_values('manufacturer').head(10)
```

	manufacturer	productName	memSize	memBusWidth	gpuClock	memClock	unifiedShader	tmu	rop	bus	memType
1671	AMD	Radeon HD 6550M	1.024	128.0	600	900.0	400.0	20	8	MXM-II	DDR3
1670	AMD	Radeon HD 6550M	1.024	128.0	600	900.0	400.0	20	8	MXM-II	DDR3
652	AMD	FirePro W4170M	2.000	128.0	825	1000.0	384.0	24	8	PCIe 3.0 x8	GDDR5
824	AMD	FirePro W4170M	2.000	128.0	825	1000.0	384.0	24	8	PCIe 3.0 x8	GDDR5
157	NVIDIA	GeForce GTX 1650 Ti Mobile	4.000	128.0	1350	1500.0	1024.0	64	32	PCIe 3.0 x16	GDDR6
1244	NVIDIA	GeForce GTX 650	1.024	128.0	1058	1250.0	384.0	32	16	PCIe 3.0 x16	GDDR5
1227	NVIDIA	GeForce GT 630M	1.024	64.0	660	900.0	96.0	16	4	MXM-A (3.0)	DDR3
1226	NVIDIA	GeForce GT 630M	1.024	64.0	660	900.0	96.0	16	4	MXM-A (3.0)	DDR3
1003	NVIDIA	GeForce GTX 650	1.024	128.0	1058	1250.0	384.0	32	16	PCIe 3.0 x16	GDDR5
847	NVIDIA	GeForce GT 710	2.000	64.0	954	900.0	192.0	16	8	PCIe 2.0 x8	DDR3

```
[ ] df = df.drop_duplicates()
```

```
[ ] df.duplicated().sum()

    0
```

There is no duplicated data

Observation :

1. Check data location that have a duplicate value.
2. Drop data that have duplicated value

STATISTICAL SUMMARY

	memSize	memBusWidth	gpuClock	memClock	unifiedShader	tmu	rop
count	1738.000000	1738.000000	1738.000000	1738.000000	1738.000000	1738.000000	1738.000000
mean	4.366116	326.094361	862.724396	1103.786536	1182.071346	72.115075	27.642117
std	8.252672	763.899227	326.889643	409.809392	1771.561211	85.216487	28.855399
min	0.128000	32.000000	300.000000	266.000000	8.000000	4.000000	0.000000
25%	1.024000	128.000000	625.000000	800.000000	192.000000	20.000000	8.000000
50%	2.000000	128.000000	796.000000	1000.000000	496.000000	40.000000	16.000000
75%	4.000000	256.000000	1005.750000	1375.000000	1536.000000	96.000000	32.000000
max	128.000000	8192.000000	2331.000000	2257.000000	17408.000000	880.000000	256.000000

Observation :

1. Overall the minimum and maximum values make sense for each column.
2. All variable has skewed distribution (mean>median)

STATISTICAL SUMMARY

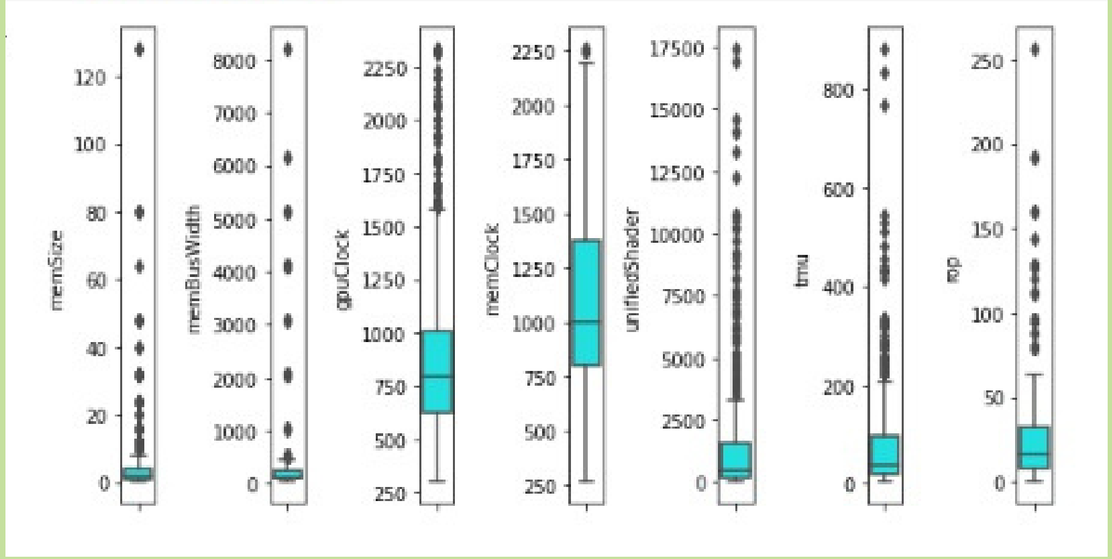
	manufacturer	productName	bus	memType
count	1738	1738	1738	1738
unique	4	1618	21	15
top	NVIDIA	GeForce GT 555M	PCIe 2.0 x16	GDDR5
freq	906	5	547	721

Observation :

1. For Manufacturer column there are 4 unique data from 1738 data with the top data is NVIDIA
2. For ProductName column there are 1618 unique data from 1738 data with the top data is GeForce GT 55M
3. For ProductName column there are 21 unique data from 1738 data with the top data is PCcle 2.0 x16
4. For MemType column there are 15 unique data from 1738 data with the top data is GDDR5

DETECT OUTLIERS VIA BOXPLOT

```
plt.figure(figsize=(8,4))

features = numericals
for i in range(0, len(features)):
    plt.subplot(1, len(features), i+1)
    sns.boxplot(y=df[features[i]], color='cyan')
    plt.tight_layout()
```



Observation :

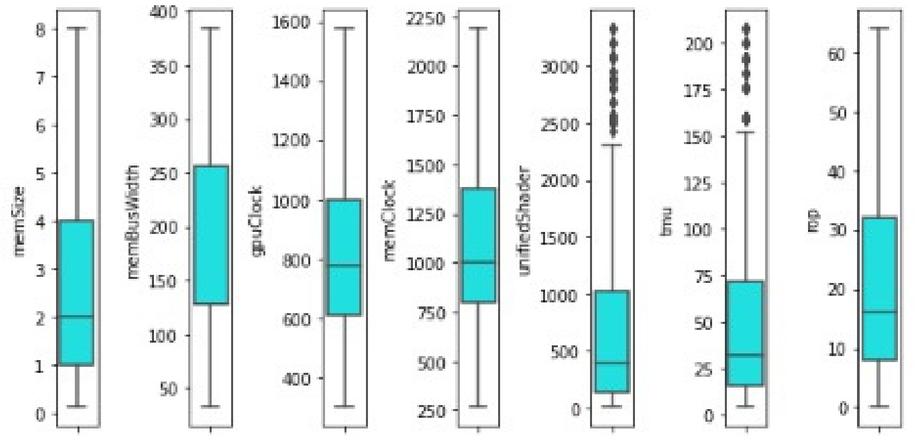
1. There are outliers for each variable
2. Handling Outliers with IQR

HANDLING OUTLIERS WITH IQR

```
# Detect Outliers Using IQR
Q1 = df[numericals].quantile(0.25)

Q3 = df[numericals].quantile(0.75)

iqr = Q3 - Q1

# Finding upper and lower limit
upper_limit = Q3 + 1.5 * iqr
lower_limit = Q1 - 1.5 * iqr

# Finding Outliers
df[df[numericals] > upper_limit]
df[df[numericals] < lower_limit]

# Trimming
new_df = df[df[numericals] < upper_limit]
new_df.shape
```

```
(1738, 11)
```

```
[ ]  plt.figure(figsize=(8,4))

     features = numericals
     for i in range(0, len(features)):
         plt.subplot(1, len(features), i+1)
         sns.boxplot(y=new_df[features[i]], color='cyan')
         plt.tight_layout()
```



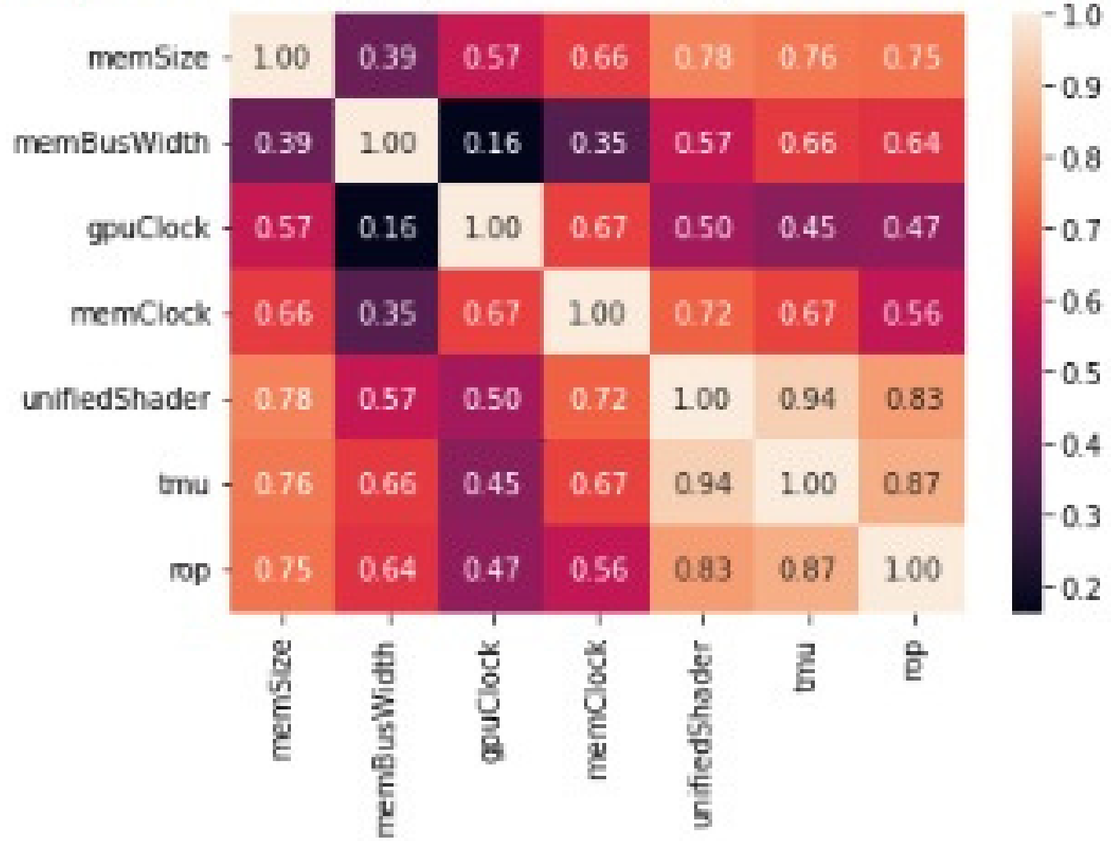
After Handling outliers with IQR.

MemSize, MemBusWidth, gpuClock, memClock, and rop are handful from outliers.

HEATMAP CORRELATION

```
correlation = new_df.corr()
sns.heatmap(correlation, annot=True, fmt='.2f')
```

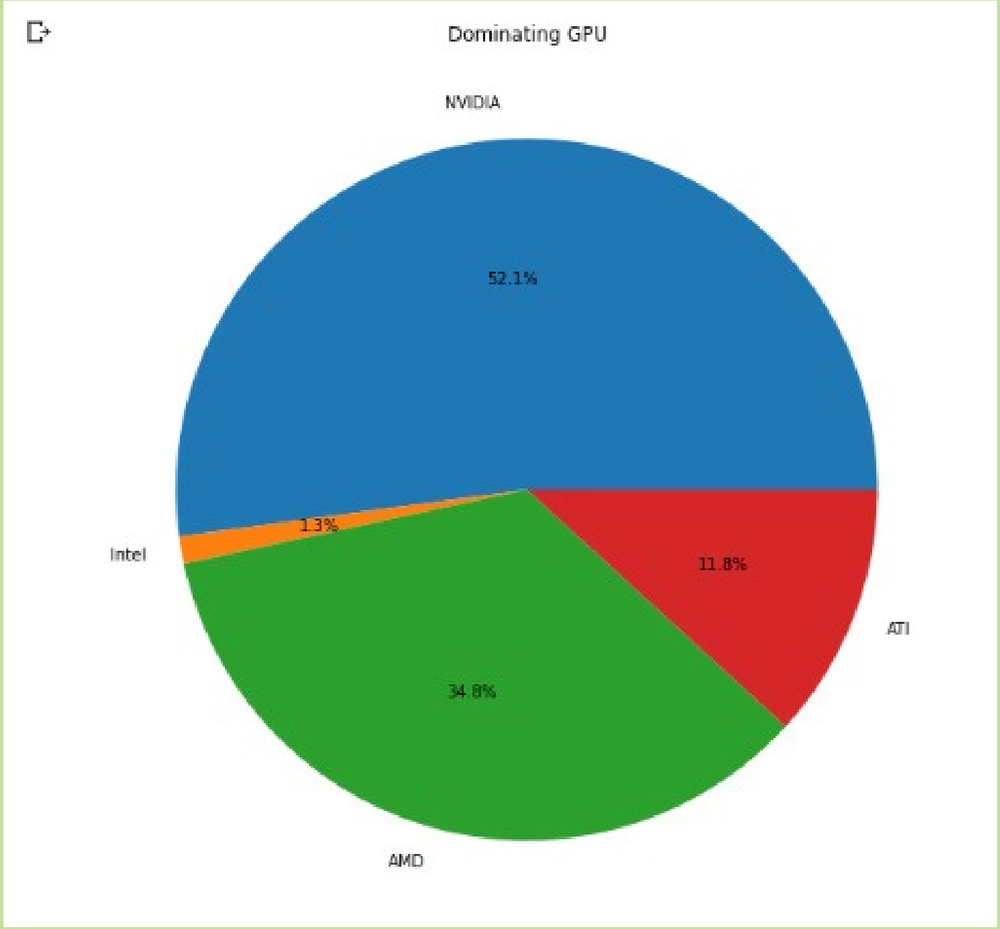
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f20f53510>
```



Observation :

1. MemSize have strong correlation between tmu, unifiedshader, and rop
2. memClock have strong correlation between unifiedshader.
3. unifiedshader have strong correlation beetween tmu, rop, memSize, and memClock,

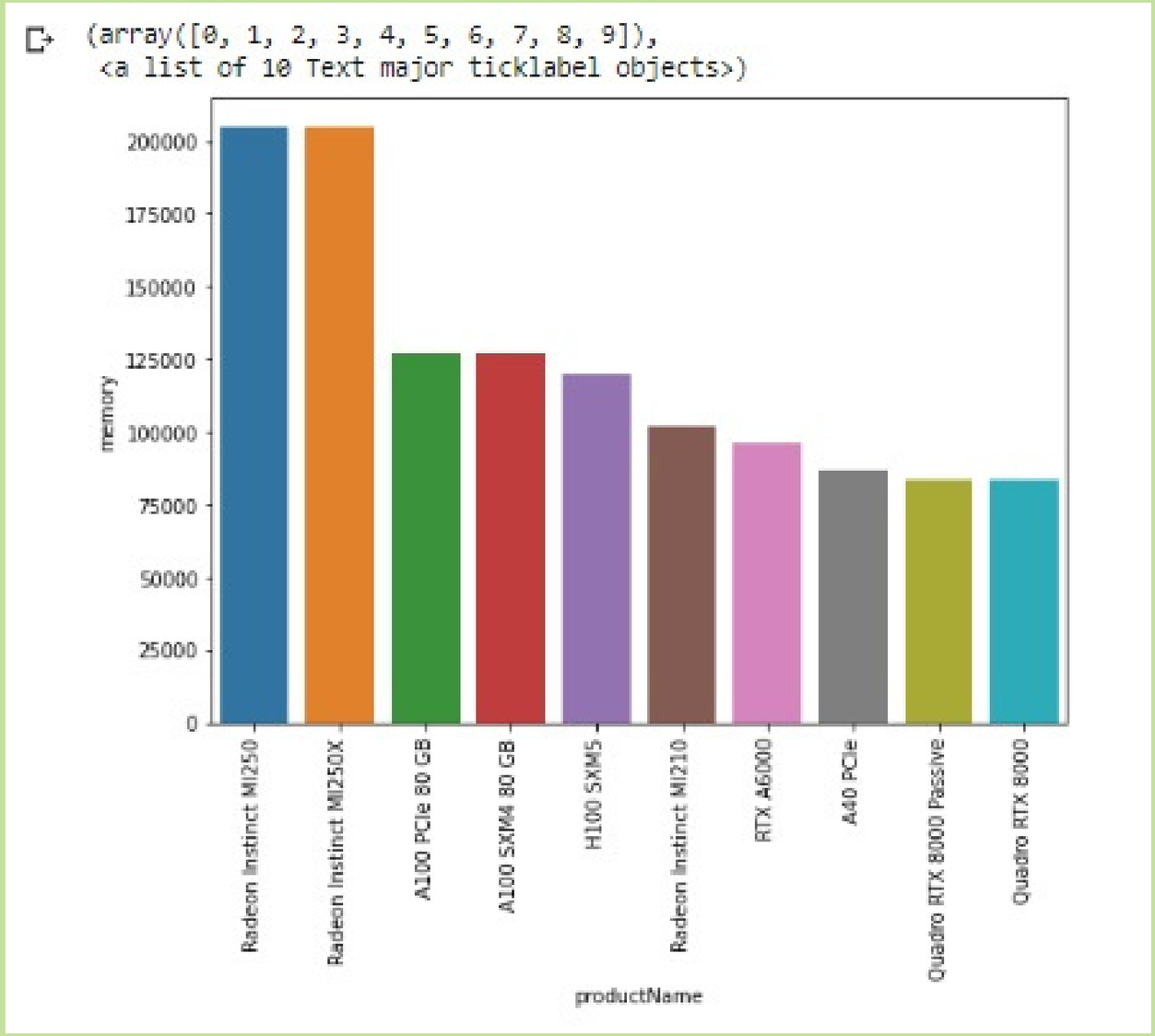
DATA VISUALIZASTION (RESULT)



Observation :

Nvidia is a dominating GPU in the market with 52.1% and AMD is the second dominating GPU in the market with 34.8%

DATA VISUALIZASTION (RESULT)



The top 10 Product with the Best GPU MemoryClock Performance are :

1. Radeon Instinct MI250
2. Radeon Instinct MI250X
3. A100 PCIe 80 GB
4. A100 SXM4 80 GB
5. H100 SXM5
6. Radeon Instinct MI210
7. RTX A6000
8. A40 PCIe
9. Quadro RTX 8000 Passive
10. Quadro RTX 8000

THANKYOU