ASSIGNMENT 2

STATISTICAL TREATMENT FOR RETAIL DATASETS

HANI NAFISAH AMALIYA

KELAS PYTN-10

Berikut ini merupakan Assignment 2 mengenai Statistics. Macam statistics yang akan dibahas pada Assignment 2 ini, antara lain :

1. Measure of Central Tendency : Mean
2. Measure of Cntral Tendency : Median
3. Measure of Central Tendency : Modus
4. Measure of Spread : Range
5. Measure of Spread : Variance
6. Measure of Spread : Standard Deviation
7. Probability Distribution
8. Convidence Intervals
9. Hypothesis Testing

```
In [1]: # Import Library
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import string
        import seaborn as sns
        import datetime
        import statsmodels.api as sm

        import warnings
        warnings.filterwarnings("ignore")
        %matplotlib inline
```

In [2]:

```
# Membaca File
df_sales = pd.read_csv('./dataset/nyc-rolling-sales.csv', skipinitialspac
e=True)
df_sales
```

Out[2]:

	Unnamed: 0	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	I
0	4	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	392	6	
1	5	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	26	
2	6	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	39	
3	7	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	402	21	
4	8	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	404	55	
...	...	...	...	...	...	...	...	
84543	8409	5	WOODROW	02 TWO FAMILY DWELLINGS	1	7349	34	
84544	8410	5	WOODROW	02 TWO FAMILY DWELLINGS	1	7349	78	
84545	8411	5	WOODROW	02 TWO FAMILY DWELLINGS	1	7351	60	
84546	8412	5	WOODROW	22 STORE BUILDINGS	4	7100	28	
84547	8413	5	WOODROW	35 INDOOR PUBLIC AND CULTURAL FACILITIES	4	7105	679	

84548 rows × 22 columns

In [3]:
```
df_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84548 entries, 0 to 84547
Data columns (total 22 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Unnamed: 0                      84548 non-null  int64  
 1   BOROUGH                         84548 non-null  int64  
 2   NEIGHBORHOOD                    84548 non-null  object 
 3   BUILDING CLASS CATEGORY         84548 non-null  object 
 4   TAX CLASS AT PRESENT            83810 non-null  object 
 5   BLOCK                           84548 non-null  int64  
 6   LOT                             84548 non-null  int64  
 7   EASE-MENT                       0 non-null      float64
 8   BUILDING CLASS AT PRESENT       83810 non-null  object 
 9   ADDRESS                         84548 non-null  object 
 10  APARTMENT NUMBER                19052 non-null  object 
 11  ZIP CODE                        84548 non-null  int64  
 12  RESIDENTIAL UNITS               84548 non-null  int64  
 13  COMMERCIAL UNITS                84548 non-null  int64  
 14  TOTAL UNITS                     84548 non-null  int64  
 15  LAND SQUARE FEET                84548 non-null  object 
 16  GROSS SQUARE FEET               84548 non-null  object 
 17  YEAR BUILT                      84548 non-null  int64  
 18  TAX CLASS AT TIME OF SALE       84548 non-null  int64  
 19  BUILDING CLASS AT TIME OF SALE  84548 non-null  object 
 20  SALE PRICE                      84548 non-null  object 
 21  SALE DATE                       84548 non-null  object 
dtypes: float64(1), int64(10), object(11)
memory usage: 14.2+ MB
```

In [4]:
```
df_sales.shape
```

Out[4]:
```
(84548, 22)
```

In [5]:

```
df_sales.isnull().sum().sort_values(ascending=False)
```

Out[5]:

```
EASE-MENT                         84548
APARTMENT NUMBER                  65496
TAX CLASS AT PRESENT                738
BUILDING CLASS AT PRESENT           738
SALE PRICE                            0
BOROUGH                               0
NEIGHBORHOOD                          0
BUILDING CLASS CATEGORY               0
BLOCK                                 0
LOT                                   0
ADDRESS                               0
SALE DATE                             0
ZIP CODE                              0
RESIDENTIAL UNITS                     0
COMMERCIAL UNITS                      0
TOTAL UNITS                           0
LAND SQUARE FEET                      0
GROSS SQUARE FEET                     0
YEAR BUILT                            0
TAX CLASS AT TIME OF SALE             0
BUILDING CLASS AT TIME OF SALE        0
Unnamed: 0                            0
dtype: int64
```

In [6]:

```
# Menghapus kolom "Unnamed : 0" karena tidak diperlukan dalam proses analisis
df_sales.drop('Unnamed: 0', axis=1, inplace=True)
```

In [7]:

```
# Menghapus kolom "EASE-MENT" karena berisi NaN
df_sales.drop('EASE-MENT', axis=1, inplace=True)
```

In [8]:

```
# Menghapus kolom "'ADDRESS','APARTMENT NUMBER', 'ZIP CODE'" karena tidak diperlukan dalam proses analisis
df_sales.drop(labels=['ADDRESS','APARTMENT NUMBER', 'ZIP CODE'], axis=1, inplace=True)
```

In [9]:

```
#'SALE PRICE' harus bertipe numerik, missing value akan diset sebagai NaN
df_sales['SALE PRICE'] = pd.to_numeric(df_sales['SALE PRICE'], errors='coerce')

# 'LAND SQUARE FEET' dan 'GROSS SQUARE FEET' harus bertipe numerik
df_sales['LAND SQUARE FEET'] = pd.to_numeric(df_sales['LAND SQUARE FEET'], errors='coerce')
df_sales['GROSS SQUARE FEET'] = pd.to_numeric(df_sales['GROSS SQUARE FEET'], errors='coerce')

# 'SALE DATE' harus bertipe datetime
df_sales['SALE DATE'] = pd.to_datetime(df_sales['SALE DATE'], errors='coerce')

# Kolom di bawah ini harus categorical
categorical = ['NEIGHBORHOOD', 'BUILDING CLASS CATEGORY', 'TAX CLASS AT PRESENT', 'BUILDING CLASS AT PRESENT',
         'BUILDING CLASS AT TIME OF SALE', 'TAX CLASS AT TIME OF SALE']
for col in categorical:
    df_sales[col] = df_sales[col].astype('category')
```

In [10]:

```
df_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84548 entries, 0 to 84547
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   BOROUGH                         84548 non-null  int64
 1   NEIGHBORHOOD                    84548 non-null  category
 2   BUILDING CLASS CATEGORY         84548 non-null  category
 3   TAX CLASS AT PRESENT            83810 non-null  category
 4   BLOCK                           84548 non-null  int64
 5   LOT                             84548 non-null  int64
 6   BUILDING CLASS AT PRESENT       83810 non-null  category
 7   RESIDENTIAL UNITS               84548 non-null  int64
 8   COMMERCIAL UNITS                84548 non-null  int64
 9   TOTAL UNITS                     84548 non-null  int64
 10  LAND SQUARE FEET                58296 non-null  float64
 11  GROSS SQUARE FEET               56936 non-null  float64
 12  YEAR BUILT                      84548 non-null  int64
 13  TAX CLASS AT TIME OF SALE       84548 non-null  category
 14  BUILDING CLASS AT TIME OF SALE  84548 non-null  category
 15  SALE PRICE                      69987 non-null  float64
 16  SALE DATE                       84548 non-null  datetime64[ns]
dtypes: category(6), datetime64[ns](1), float64(3), int64(7)
memory usage: 7.8 MB
```

In [11]:
```
df_sales
```

Out[11]:

	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	BUILDING CLASS AT PRESENT
0	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	392	6	C2
1	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	26	C7
2	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	39	C7
3	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	402	21	C4
4	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	404	55	C2
...	...	...	...	...	...	...	...
84543	5	WOODROW	02 TWO FAMILY DWELLINGS	1	7349	34	B9
84544	5	WOODROW	02 TWO FAMILY DWELLINGS	1	7349	78	B9
84545	5	WOODROW	02 TWO FAMILY DWELLINGS	1	7351	60	B2
84546	5	WOODROW	22 STORE BUILDINGS	4	7100	28	K6
84547	5	WOODROW	35 INDOOR PUBLIC AND CULTURAL FACILITIES	4	7105	679	P9

84548 rows × 17 columns

In [12]:
```
sum(df_sales.duplicated())
```

Out[12]:
```
959
```

In [13]:
```
df_sales = df_sales.drop_duplicates(df_sales.columns, keep='last')
```

In [14]:
```
sum(df_sales.duplicated())
```

Out[14]:
```
0
```

In [15]:

```
missing_value = df_sales.isnull().sum()/len(df_sales)*100
print(pd.DataFrame([missing_value[missing_value>0], pd.Series(df_sales.is
null().sum()[df_sales.isnull().sum()>1000])],
                   index=['percent missing', 'num of missing']))
```

```
                 TAX CLASS AT PRESENT  BUILDING CLASS AT PRESENT  \
percent missing              0.882891                   0.882891   
num of missing                    NaN                        NaN   

                 LAND SQUARE FEET  GROSS SQUARE FEET    SALE PRICE  
percent missing          31.04954          32.638266     16.837144  
num of missing        25954.00000       27282.000000  14074.000000  
```

In [16]:

```
df_sales['SALE PRICE'].describe()
```

Out[16]:

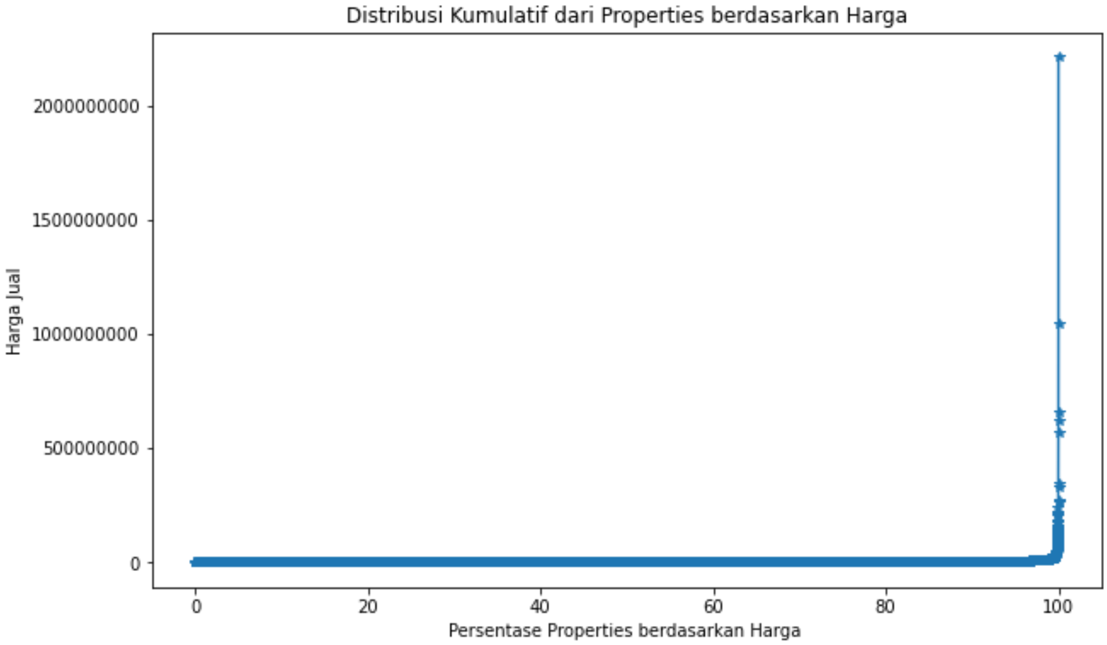
```
count    6.951500e+04
mean     1.282005e+06
std      1.143784e+07
min      0.000000e+00
25%      2.300000e+05
50%      5.345810e+05
75%      9.500000e+05
max      2.210000e+09
Name: SALE PRICE, dtype: float64
```

In [17]:

```
# Menghilangkan semua value yang mengandung null
df_sales = df_sales[df_sales['SALE PRICE'].notnull()]
df_sales = df_sales[df_sales['LAND SQUARE FEET'].notnull()]
df_sales = df_sales[df_sales['GROSS SQUARE FEET'].notnull()]
```

In [18]:

```
#get data property proportion
x= df_sales[['SALE PRICE']].sort_values(by='SALE PRICE').reset_index()
x['PROPERTY PROPORTION']= 1
x['PROPERTY PROPORTION']= x['PROPERTY PROPORTION'].cumsum()
x['PROPERTY PROPORTION'] = 100 * x['PROPERTY PROPORTION']/len(x['PROPERTY PROPORTION'])

#set size for the plot
plt.figure(figsize=(10,6))

#plot the data
plt.plot(x['PROPERTY PROPORTION'], x['SALE PRICE'], linestyle=None, marker='*')
plt.title("Distribusi Kumulatif dari Properties berdasarkan Harga")
plt.xlabel("Persentase Properties berdasarkan Harga ")
plt.ylabel("Harga Jual")
plt.ticklabel_format(style='plain',axis='y')
plt.show()
```



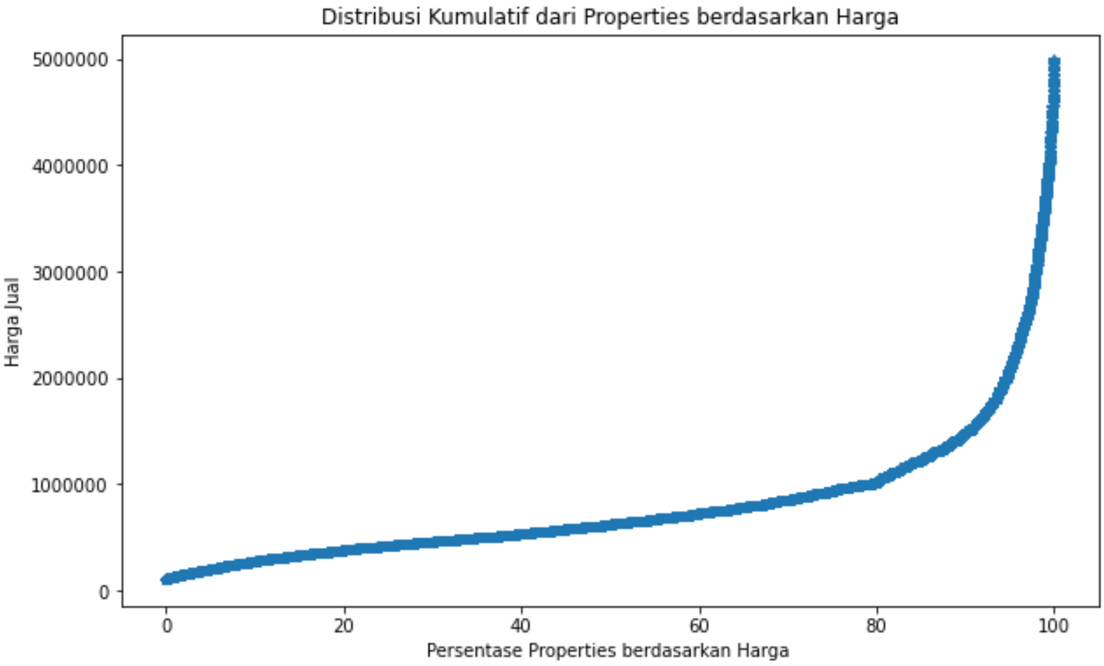
Untuk menghilangkan outliers kita bisa mengambil data antara 100.000 USD sampai 5.000.000 USD

In [19]:

```
df_sales = df_sales[(df_sales['SALE PRICE']>100000) & (df_sales['SALE PRICE']<5000000)]
```

In [20]:

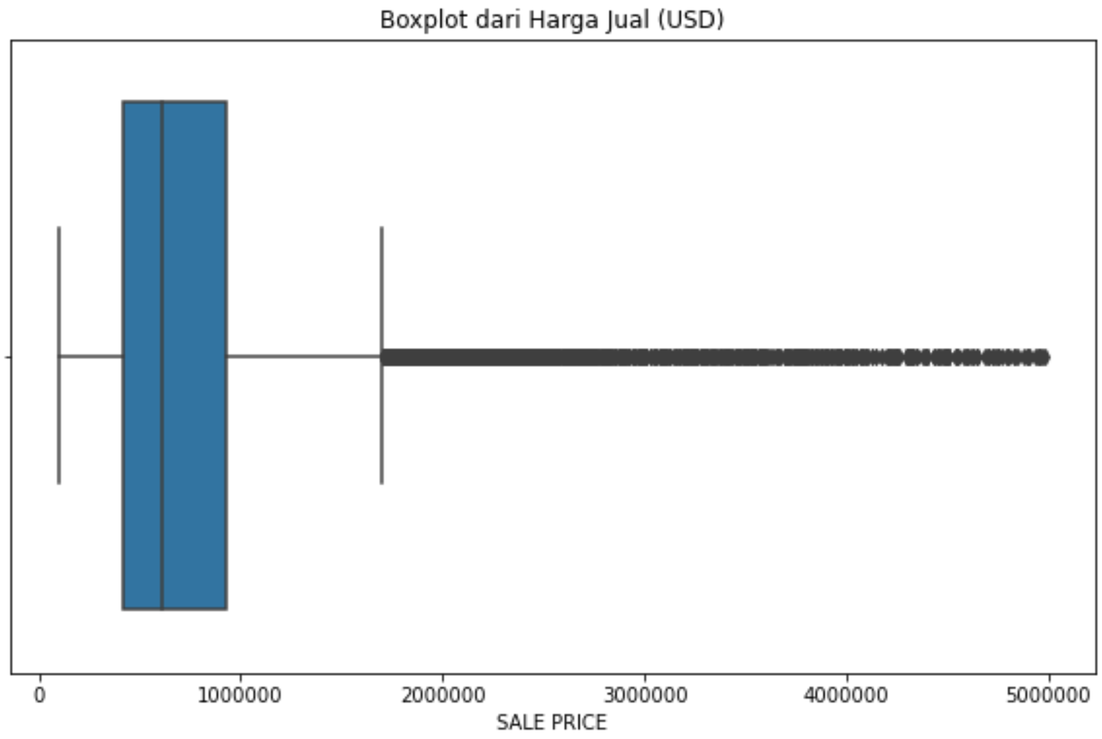
```
# do the same as above to get data property proportion and plot the data,
then see the result
x = df_sales[['SALE PRICE']].sort_values(by='SALE PRICE').reset_index()
x['PROPERTY PROPORTION']= 1
x['PROPERTY PROPORTION']= x['PROPERTY PROPORTION'].cumsum()
x['PROPERTY PROPORTION'] = 100 * x['PROPERTY PROPORTION']/len(x['PROPERTY
PROPORTION'])

plt.figure(figsize=(10,6))
plt.plot(x['PROPERTY PROPORTION'], x['SALE PRICE'], linestyle=None, marke
r='*')
plt.title("Distribusi Kumulatif dari Properties berdasarkan Harga")
plt.xlabel("Persentase Properties berdasarkan Harga ")
plt.ylabel("Harga Jual")
plt.ticklabel_format(style='plain',axis='y')
plt.show()
```



Data distribusi sudah tidak ada outliers

In [21]:

```
# plot curve using boxplot to see another view of the data
plt.figure(figsize=(10,6))
sns.boxplot(x='SALE PRICE', data = df_sales)
plt.ticklabel_format(style='plain', axis='x')
plt.title("Boxplot dari Harga Jual (USD)")
plt.show()
```



Ploting menggunakan kurva boxplot dapat kita lihat sudah tidak ada outliers, data sudah lebih baik

1. Measure of Central Tendency : Mean

Mean atau Average adalah central tendency dari data, angka diantara seluruh data tersebar, angka tunggal yang dapat memperkirakan nilai seluruh kumpulan data. Rata-rata dihitung dengan jumlah semua nilai, dibagi dengan jumlah nilai.

In [22]:

```
mean = df_sales['SALE PRICE'].mean()

print(mean)
```

```
795972.4573388677
```

2. Measure of Central Tendency : Median

Median adalah nilai yang membagi data dalam 2 bagian yang sama. Median merupakan nilai tengah atau titik tengah dalam data dan juga disebut **persentil ke-50.**

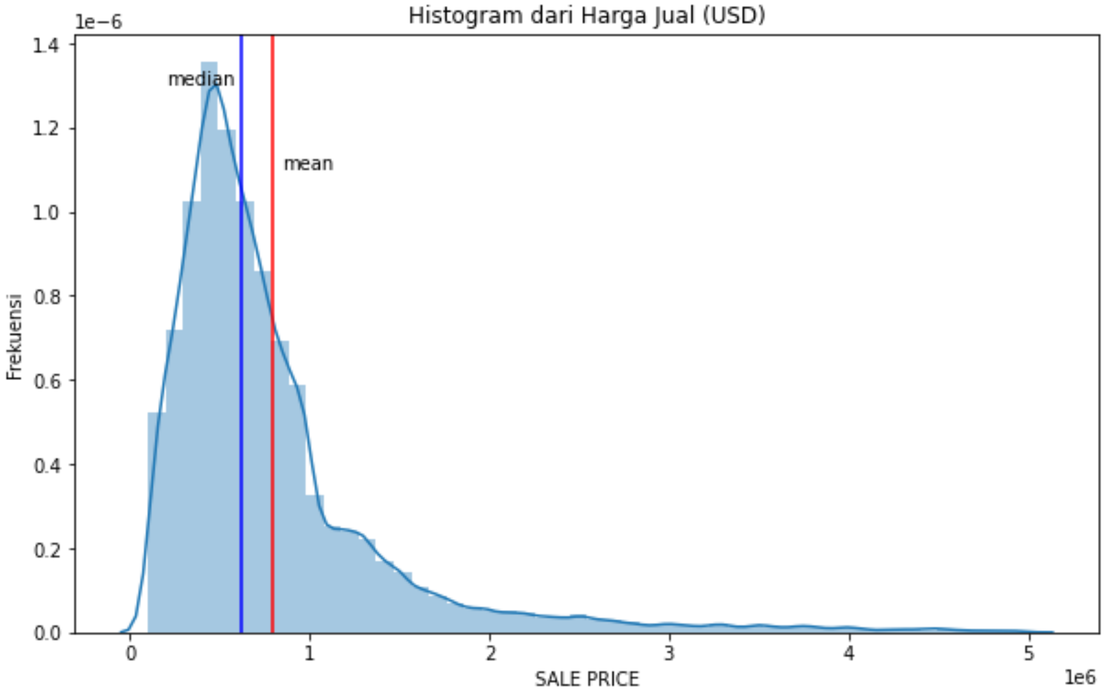
In [23]:

```
median = df_sales['SALE PRICE'].median()

print(median)
```

```
615000.0
```

In [24]:

```
#let's take a look distribution of the data, the data is skewness or norm
al distribution
plt.figure(figsize=(10,6))
sns.distplot(df_sales['SALE PRICE'])
plt.title('Histogram dari Harga Jual (USD)')
plt.ylabel("Frekuensi")
plt.axvline(df_sales[(df_sales['SALE PRICE']>100000) & (df_sales['SALE PR
ICE'] < 5000000)]['SALE PRICE'].mean(), c='red')
plt.axvline(df_sales[(df_sales['SALE PRICE']>100000) & (df_sales['SALE PR
ICE'] < 5000000)]['SALE PRICE'].median(), c='blue')
plt.text(200000,0.0000013, "median")
plt.text(850000,0.0000011, "mean")
plt.show()
```



Sebaran data diatas menunjukkan adanya positive skewness karena nilai Mean lebih besar daripada Median

3. Measure of Central Tendency : Modus

Modus adalah nilai atau kategori yang paling sering muncul dalam data.

In [25]:

```
total_bangunan = df_sales['BUILDING CLASS CATEGORY'].value_counts()

total_bangunan
```

Out[25]:

```
01 ONE FAMILY DWELLINGS                         12354
02 TWO FAMILY DWELLINGS                          9526
10 COOPS - ELEVATOR APARTMENTS                   2649
13 CONDOS - ELEVATOR APARTMENTS                  2634
03 THREE FAMILY DWELLINGS                        2243
07 RENTALS - WALKUP APARTMENTS                   1352
15 CONDOS - 2-10 UNIT RESIDENTIAL                 775
04 TAX CLASS 1 CONDOS                             534
09 COOPS - WALKUP APARTMENTS                      493
12 CONDOS - WALKUP APARTMENTS                     365
22 STORE BUILDINGS                                355
14 RENTALS - 4-10 UNIT                            284
05 TAX CLASS 1 VACANT LAND                        198
29 COMMERCIAL GARAGES                             196
21 OFFICE BUILDINGS                               128
30 WAREHOUSES                                     122
27 FACTORIES                                       69
31 COMMERCIAL VACANT LAND                          63
44 CONDO PARKING                                   63
37 RELIGIOUS FACILITIES                            48
41 TAX CLASS 4 - OTHER                             33
43 CONDO OFFICE BUILDINGS                          30
17 CONDO COOPS                                     29
06 TAX CLASS 1 - OTHER                             24
08 RENTALS - ELEVATOR APARTMENTS                   21
16 CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT         14
33 EDUCATIONAL FACILITIES                          13
32 HOSPITAL AND HEALTH FACILITIES                  13
35 INDOOR PUBLIC AND CULTURAL FACILITIES           13
46 CONDO STORE BUILDINGS                           12
26 OTHER HOTELS                                     9
11A CONDO-RENTALS                                   8
38 ASYLUMS AND HOMES                                7
23 LOFT BUILDINGS                                   6
48 CONDO TERRACES/GARDENS/CABANAS                   4
11 SPECIAL CONDO BILLING LOTS                       1
28 COMMERCIAL CONDOS                                1
36 OUTDOOR RECREATIONAL FACILITIES                  1
39 TRANSPORTATION FACILITIES                        1
42 CONDO CULTURAL/MEDICAL/EDUCATIONAL/ETC           1
18 TAX CLASS 3 - UNTILITY PROPERTIES                0
25 LUXURY HOTELS                                    0
34 THEATRES                                         0
40 SELECTED GOVERNMENTAL FACILITIES                 0
45 CONDO HOTELS                                     0
47 CONDO NON-BUSINESS STORAGE                       0
49 CONDO WAREHOUSES/FACTORY/INDUS                   0
Name: BUILDING CLASS CATEGORY, dtype: int64
```

BUILDING CLASS CATEGORY yang paling banyak muncul pada dataset ini adalah **ONE FAMILY DWELLINGS** dengan jumlah total 12327

4. Measure of Spread : Range

Range atau Rentang adalah salah satu teknik statistik deskriptif yang paling sederhana. Range adalah perbedaan antara nilai terendah dan tertinggi.

```
In [26]: minimal = total_bangunan.min()

         print(minimal)

         0
```

```
In [27]: maximal = total_bangunan.max()

         print(maximal)

         12354
```

```
In [28]: # Menghitung Range
         jarak = maximal - minimal

         print(jarak)

         12354
```

5. Measure of Spread : Variance

Variance atau Varians adalah kuadrat jarak rata-rata antara setiap kuantitas dan mean. Variance adalah kuadrat dari **Standar Deviasi**

```
In [29]: var_ = df_sales['SALE PRICE'].var()
         var_

Out[29]: 426769738804.34357
```

6. Measure of Spread : Standard Deviation

Standard Deviation atau Simpangan Baku adalah pengukuran jarak rata-rata antara setiap besaran dan mean.

- Standar Deviasi yang **rendah** menunjukkan bahwa titik data cenderung mendekati rata-rata kumpulan data.
- Standar Deviasi yang **tinggi** menunjukkan bahwa titik data tersebar di nilai yang lebih luas.

In [30]:
```
std_ = var_ ** 0.5
std_
```

Out[30]:
```
653276.1581477956
```

atau

In [31]:
```
standar = df_sales['SALE PRICE'].std()

print(standar)
```

```
653276.1581477956
```

7. Probability Distribution

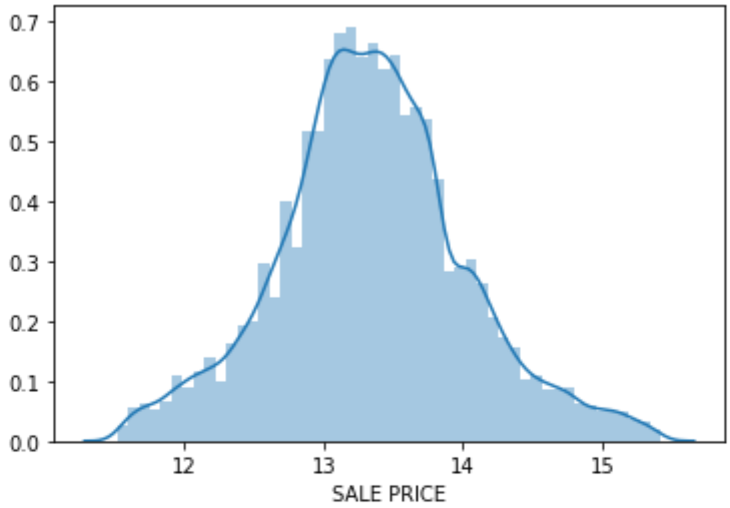
Probability Distribution adalah fungsi di bawah teori dan statistik probabilitas yang memberi seberapa besar kemungkinan hasil yang berbeda dalam sebuah eksperimen. Menggambarkan peristiwa dalam hal probabilitasnya dari semua kemungkinan hasil.

In [32]:
```
# Menghitung Probability Distribution dari Office Buildings dari Building Class Category
# p_of = jumlah office building / jumlah seluruh building dari Class Category
p_of = 128/34692
p_of
```

Out[32]:
```
0.003689611437795457
```

In [33]:
```
# Distribusi Normal
df_sales['SALE PRICE']=np.log(df_sales['SALE PRICE'])
print(df_sales['SALE PRICE'].skew())
sns.distplot(df_sales['SALE PRICE']);
```

```
0.14759631798190956
```



Data distribusi sudah tidak terdapat skewness yang berarti sebaran data sudah normal

8. Convidence Intervals

Confidence Interval (CI) adalah jenis estimasi yang dihitung dari data statistik yang diamati. CI digunakan unuk mengukur seberapa akurat Mean sebuah sample mewakili (mencakup) nilai Mean Populasi sesungguhnya. Jadi, Confidence Interval adalah rentang antara dua nilai dimana nilai suatu Sample Mean tepat berada di tengah-tengahnya.

```
In [34]: # Menghitung CI dari Office Buildings dari Building Class Category
         Office = df_sales[df_sales['BUILDING CLASS CATEGORY'] =='OFFICE BUILDINGS
         ']
```

```
In [35]: n = 34692
```

```
In [36]: p_of = 128/n
         p_of

Out[36]: 0.003689611437795457
```

```
In [37]: se_of = np.sqrt(p_of * (1-p_of) / n)
         se_of

Out[37]: 0.0003255164769176591
```

```
In [38]: z_score = 1.96

         lcb = p_of - z_score * se_of #lower limit dari CI
         ucb = p_of + z_score * se_of #upper limit dari CI

         lcb,ucb

Out[38]: (0.0030515991430368453, 0.004327623732554069)
```

atau

```
In [39]: sm.stats.proportion_confint(n * p_of, n)

Out[39]: (0.0030516108666624815, 0.004327612008928433)
```

Jadi, Convidence Interval adalah 0.0030516108666624815 dan 0.004327612008928433

9. Hypothesis Testing

Hipotesis adalah anggapan dasar atau jawaban sementara terhadap masalah yang masih bersifat praduga karena masih harus dibuktikan kebenarannya. Hipotesis harus dapat diuji, baik dengan eksperimen atau observasi.

Hypothesis Testing dalam statistik adalah cara menguji hasil survey atau eksperimen untuk melihat apakah memiliki hasil yang bermakna. Pada dasarnya menguji apakah hasil valid dengan mencari tahu kemungkinan bahwa hasil terjadi secara kebetulan.

In [40]:

```
df_sales['BOROUGH'] = df_sales['BOROUGH'].astype(str)
df_sales['BOROUGH'] = df_sales['BOROUGH'].str.replace("1", "Manhattan")
df_sales['BOROUGH'] = df_sales['BOROUGH'].str.replace("2", "Bronx")
df_sales['BOROUGH'] = df_sales['BOROUGH'].str.replace("3", "Brooklyn")
df_sales['BOROUGH'] = df_sales['BOROUGH'].str.replace("4", "Queens")
df_sales['BOROUGH'] = df_sales['BOROUGH'].str.replace("5", "Staten Islan
d")
```

In [41]:

```
df_sales
```

Out[41]:

	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	BUILDING CLASS AT PRESENT
3	Manhattan	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	402	21	C4
6	Manhattan	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	406	32	C4
172	Manhattan	ALPHABET CITY	14 RENTALS - 4-10 UNIT	2A	391	19	S3
174	Manhattan	ALPHABET CITY	14 RENTALS - 4-10 UNIT	2A	394	5	S5
195	Manhattan	ALPHABET CITY	22 STORE BUILDINGS	4	390	34	K4
...	...	...	...	...	...	...	...
84540	Staten Island	WOODROW	02 TWO FAMILY DWELLINGS	1	7316	93	B2
84541	Staten Island	WOODROW	02 TWO FAMILY DWELLINGS	1	7317	126	B2
84543	Staten Island	WOODROW	02 TWO FAMILY DWELLINGS	1	7349	34	B9
84544	Staten Island	WOODROW	02 TWO FAMILY DWELLINGS	1	7349	78	B9
84545	Staten Island	WOODROW	02 TWO FAMILY DWELLINGS	1	7351	60	B2

34692 rows × 17 columns

In [42]:
```
df_borough = df_sales.groupby('BOROUGH', axis=0).sum()

df_borough
```

Out[42]:

	BLOCK	LOT	RESIDENTIAL UNITS	COMMERCIAL UNITS	TOTAL UNITS	LAND SQUARE FEET	GRO SQUA FE
BOROUGH							
Bronx	19585559	1370003	8550	398	8947	10882624.0	970122
Brooklyn	61616042	5466898	22069	1182	23341	22044154.0	2185940
Manhattan	506732	17482	2384	242	2625	861416.0	403367
Queens	84318984	568145	20304	3141	23437	39427791.0	2568726
Staten Island	16335608	397679	6442	464	6900	24941536.0	1062452

Dengan melihat data diatas, apakah harga rata rata per unit pada borough Staten Island lebih besar secara signifikan daripada Bronx?

- h0 = Tidak ada perbedaan secara signifikan pada harga rata rata perunit antara borough Staten Island dan Bronx
- h1 = Terdapat perbedaan secara signifikan pada harga rata rata per unit antara borugh Staten Island dan Bronx

In [43]:
```
Queens = df_sales[df_sales['BOROUGH']=='Queens']
Staten_Island = df_sales[df_sales['BOROUGH']=='Staten Island']
```

In [44]:
```
total_unit_Queens = df_borough.iloc[-2, 4]
mu_Queens = Queens['SALE PRICE'].mean()
std_Queens = Queens['SALE PRICE'].std()
total_unit_Queens, mu_Queens, std_Queens
```

Out[44]:
```
(23437, 13.346150061358832, 0.5459676075936886)
```

In [45]:
```
total_unit_SI = df_borough.iloc[-1, 4]
mu_SI = Staten_Island['SALE PRICE'].mean()
std_SI = Staten_Island['SALE PRICE'].std()
total_unit_SI, mu_SI, std_SI
```

Out[45]:
```
(6900, 13.091598037978054, 0.4305626958529941)
```

In [46]:
```
from statsmodels.stats.weightstats import ztest
ztest, pval= ztest(Staten_Island['SALE PRICE'],Queens['SALE PRICE'])
print("pval: ",float(pval))
if pval<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

```
pval:  1.4730188768955306e-179
reject null hypothesis
```

Dengan hasil ini dapat ditarik kesimpulan bahwa terdapat perbedaan yang cukup signifikan pada harga rata-rata per unit antara Borough Staten Island dengan Queens