Time Series and Forecasting Laboratory (201DSL416**)**
Experiment No. 1: Implementation of Time Series Data - Cleaning, Loading and Handling Times series data.

Instructions:
Begin with Part 1, where you'll read and comprehend the time series data using examples. Once you've completed this step, proceed to Part 2 and carry out the implementation accordingly.

Part 1) Understanding and Exploring Time Series Data

1.1. What is Time series data?

Time series data is data that is recorded over consistent intervals of time. In time series analysis, analysts record data points at consistent intervals over a set period of time rather than just recording the data points intermittently or randomly. However, this type of analysis is not merely the act of collecting data over time. While time-series data is information gathered over time, various types of information describe how and when that information was gathered. For example:

1. **Time series data:** It is a collection of observations on the values that a variable takes at various points in time.
2. **Cross-sectional data:** Data from one or more variables that were collected simultaneously.
3. **Pooled data:** It is a combination of cross-sectional and time-series data.

1.2. Examples of Time series data

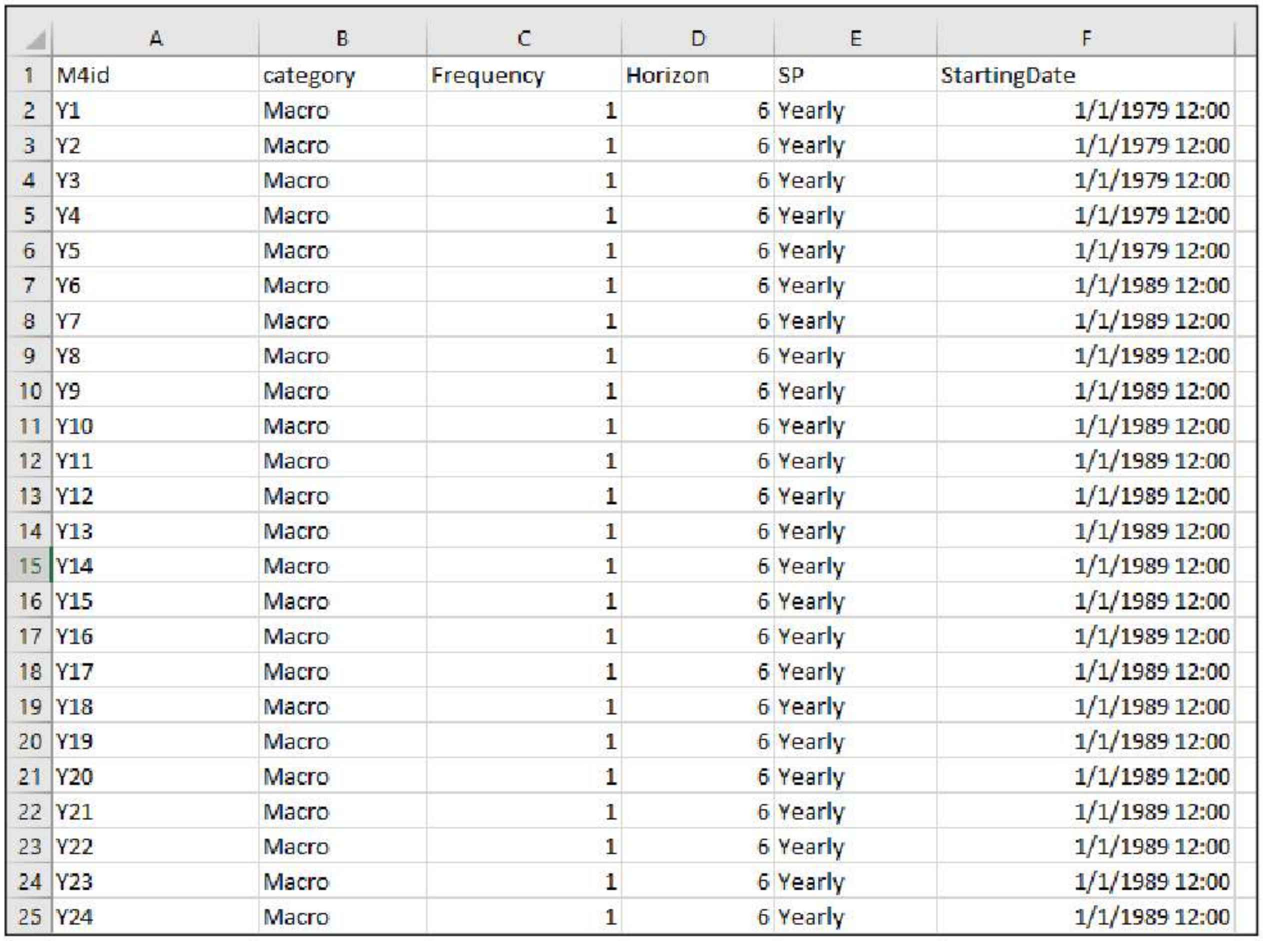
- Electrical activity in the brain
- Rainfall measurements
- Stock prices
- Number of sunspots
- Annual retail sales
- Monthly subscribers
- Heartbeats per minute

Source for examples - https://www.influxdata.com/what-is-time-series-data/#examples

1.3. Ten best Time series datasets.

Use below link to explore different time series datasets.

- https://hackernoon.com/10-best-datasets-for-time-series-analysis

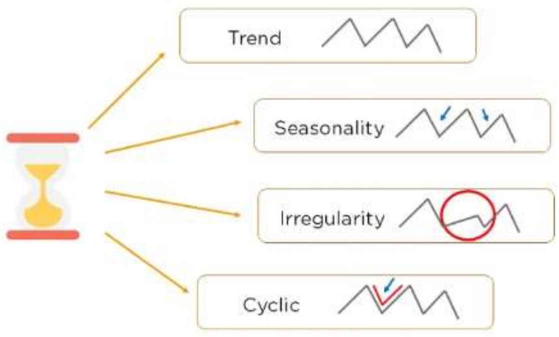


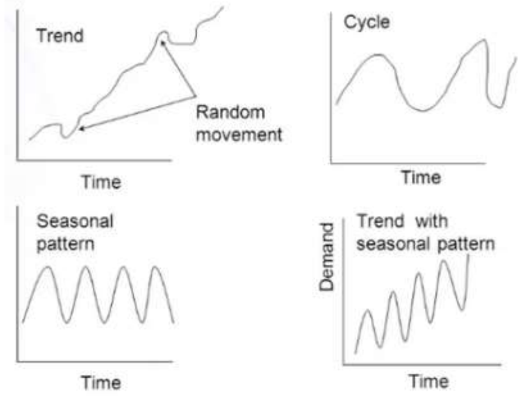
	A	B	C	D	E	F
1	M4id	category	Frequency	Horizon	SP	StartingDate
2	Y1	Macro	1	6	Yearly	1/1/1979 12:00
3	Y2	Macro	1	6	Yearly	1/1/1979 12:00
4	Y3	Macro	1	6	Yearly	1/1/1979 12:00
5	Y4	Macro	1	6	Yearly	1/1/1979 12:00
6	Y5	Macro	1	6	Yearly	1/1/1979 12:00
7	Y6	Macro	1	6	Yearly	1/1/1989 12:00
8	Y7	Macro	1	6	Yearly	1/1/1989 12:00
9	Y8	Macro	1	6	Yearly	1/1/1989 12:00
10	Y9	Macro	1	6	Yearly	1/1/1989 12:00
11	Y10	Macro	1	6	Yearly	1/1/1989 12:00
12	Y11	Macro	1	6	Yearly	1/1/1989 12:00
13	Y12	Macro	1	6	Yearly	1/1/1989 12:00
14	Y13	Macro	1	6	Yearly	1/1/1989 12:00
15	Y14	Macro	1	6	Yearly	1/1/1989 12:00
16	Y15	Macro	1	6	Yearly	1/1/1989 12:00
17	Y16	Macro	1	6	Yearly	1/1/1989 12:00
18	Y17	Macro	1	6	Yearly	1/1/1989 12:00
19	Y18	Macro	1	6	Yearly	1/1/1989 12:00
20	Y19	Macro	1	6	Yearly	1/1/1989 12:00
21	Y20	Macro	1	6	Yearly	1/1/1989 12:00
22	Y21	Macro	1	6	Yearly	1/1/1989 12:00
23	Y22	Macro	1	6	Yearly	1/1/1989 12:00
24	Y23	Macro	1	6	Yearly	1/1/1989 12:00
25	Y24	Macro	1	6	Yearly	1/1/1989 12:00

Sample time series dataset : M4 Competition Dataset

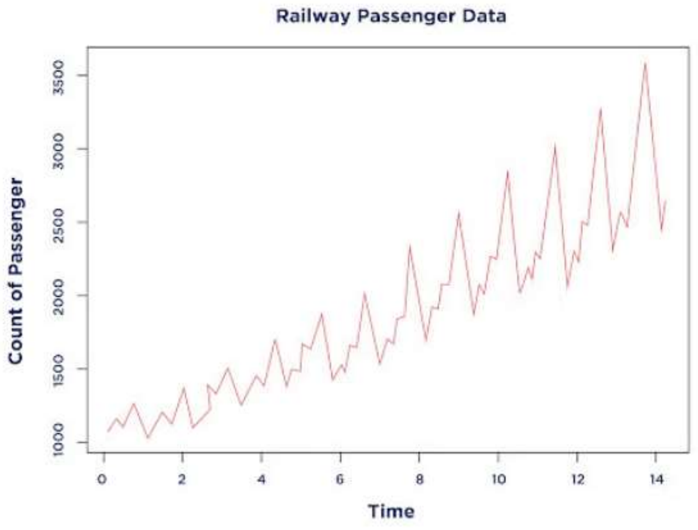
This dataset is a collection of over 100,000 time series of yearly, quarterly, monthly and other (weekly, daily and hourly) data, divided into training and test datasets and is used in the M4 Forecasting Competition, which is an annual competition organized by the International Institute of Forecasters (IIF). The competition aims to evaluate and compare different forecasting methods and models on a diverse set of time series.

1.4. Components of Time series -





Reference-https://medium.com/@ritusantra/what-is-time-series-and-components-of-time-series-c80b69ad5cb9



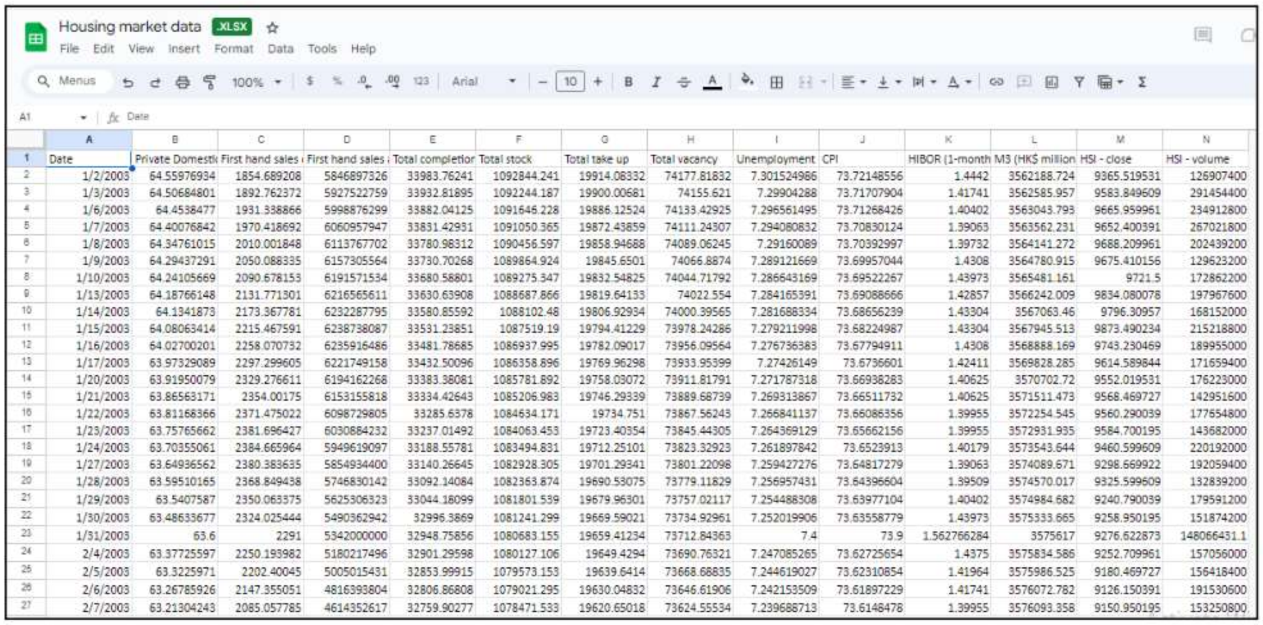
Reference-https://www.simplilearn.com/tutorials/statistics-tutorial/what-is-time-series-analysis#what_is_time_series_analysis

The following observations can be derived from the above image-
Trend: Over time, an increasing or decreasing pattern has been observed. The total number of passengers has risen over time.
Seasonality: Cyclic patterns are the ones that repeat after a certain interval of time. In the case of the railway passenger, you can see a cyclic pattern with a high and low point that is visible throughout the interval.

Part 2) Implementation of time series data loading, handling and cleaning

2.1. Dataset used for the experiment - HK macroeconomics data

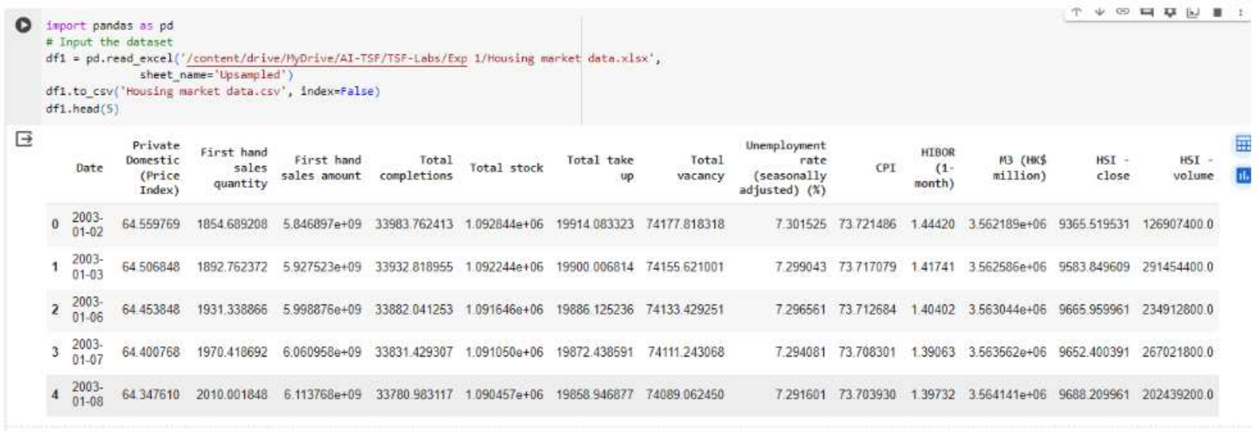
Download the dataset -
https://www.kaggle.com/datasets/stanley11291985/hk-macroeconomics-data



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Date	Private Domesti	First hand sales	First hand sales	Total completior	Total stock	Total take up	Total vacancy	Unemployment	CPI	HIBOR (1-month	M3 (HK$ million	HSI - close	HSI - volume
2	1/2/2003	64.55976934	1854.689208	5846897326	33983.76241	1092844.241	19914.08332	74177.81832	7.301524986	73.72148556	1.4442	3562188.724	9365.519531	126907400
3	1/3/2003	64.50684801	1892.762372	5927522759	33932.81895	1092244.187	19900.00681	74155.621	7.29904288	73.71707904	1.41741	3562585.957	9583.849609	291454400
4	1/6/2003	64.4538477	1931.338866	5998876299	33882.04125	1091646.228	19886.12524	74133.42925	7.296561495	73.71268426	1.40402	3563043.793	9665.959961	234912800
5	1/7/2003	64.40076842	1970.418692	6060957947	33831.42931	1091050.365	19872.43859	74111.24307	7.294080832	73.70830124	1.39063	3563562.231	9652.400391	267021800
6	1/8/2003	64.34761015	2010.001848	6113767702	33780.98312	1090456.597	19858.94688	74089.06245	7.29160089	73.70392997	1.39732	3564141.272	9688.209961	202439200
7	1/9/2003	64.29437291	2050.088335	6157305564	33730.70268	1089864.924	19845.6501	74066.8874	7.289121669	73.69957044	1.4308	3564780.915	9675.410156	129623200
8	1/10/2003	64.24105669	2090.678153	6191571534	33680.58801	1089275.347	19832.54825	74044.71792	7.286643169	73.69522267	1.43973	3565481.161	9721.5	172862200
9	1/13/2003	64.18766148	2131.771301	6216565611	33630.63908	1088687.866	19819.64133	74022.554	7.284165391	73.69088666	1.42857	3566242.009	9834.080078	197967600
10	1/14/2003	64.1341873	2173.367781	6232287795	33580.85592	1088102.48	19806.92934	74000.39565	7.281688334	73.68656239	1.43304	3567063.46	9796.30957	168152000
11	1/15/2003	64.08063414	2215.467591	6238738087	33531.23851	1087519.19	19794.41229	73978.24286	7.279211998	73.68224987	1.43304	3567945.513	9873.490234	215218800
12	1/16/2003	64.02700201	2258.070732	6235916486	33481.78685	1086937.995	19782.09017	73956.09564	7.276736383	73.67794911	1.4308	3568888.169	9743.230469	189955000
13	1/17/2003	63.97329089	2297.299605	6221749158	33432.50096	1086358.896	19769.96298	73933.95399	7.27426149	73.6736601	1.42411	3569828.285	9614.589844	171659400
14	1/20/2003	63.91950079	2329.276611	6194162268	33383.38081	1085781.892	19758.03072	73911.81791	7.271787318	73.66938283	1.40625	3570702.72	9552.019531	176223000
15	1/21/2003	63.86563171	2354.00175	6153155818	33334.42643	1085206.983	19746.29339	73889.68739	7.269313867	73.66511732	1.40625	3571511.473	9568.469727	142951600
16	1/22/2003	63.81168366	2371.475022	6098729805	33285.6378	1084634.171	19734.751	73867.56243	7.266841137	73.66086356	1.39955	3572254.545	9560.290039	177654800
17	1/23/2003	63.75765662	2381.696427	6030884232	33237.01492	1084063.453	19723.40354	73845.44305	7.264369129	73.65662156	1.39955	3572931.935	9584.700195	143682000
18	1/24/2003	63.70355061	2384.665964	5949619097	33188.55781	1083494.831	19712.25101	73823.32923	7.261897842	73.6523913	1.40179	3573543.644	9460.599609	220192000
19	1/27/2003	63.64936562	2380.383635	5854934400	33140.26645	1082928.305	19701.29341	73801.22098	7.259427276	73.64817279	1.39063	3574089.671	9298.669922	192059400
20	1/28/2003	63.59510165	2368.849438	5746830142	33092.14084	1082363.874	19690.53075	73779.11829	7.256957431	73.64396604	1.39509	3574570.017	9325.599609	132839200
21	1/29/2003	63.5407587	2350.063375	5625306323	33044.18099	1081801.539	19679.96301	73757.02117	7.254488308	73.63977104	1.40402	3574984.682	9240.790039	179591200
22	1/30/2003	63.48633677	2324.025444	5490362942	32996.3869	1081241.299	19669.59021	73734.92961	7.252019906	73.63558779	1.43973	3575333.665	9258.950195	151874200
23	1/31/2003	63.6	2291	5342000000	32948.75856	1080683.155	19659.41234	73712.84363	7.4	73.9	1.562766284	3575617	9276.622873	148066431.1
24	2/4/2003	63.37725597	2250.193982	5180217496	32901.29598	1080127.106	19649.4294	73690.76321	7.247085265	73.62725654	1.4375	3575834.586	9252.709961	157056000
25	2/5/2003	63.3225971	2202.40045	5005015431	32853.99915	1079573.153	19639.6414	73668.68835	7.244619027	73.62310854	1.41964	3575986.525	9180.469727	156418400
26	2/6/2003	63.26785926	2147.355051	4816393804	32806.86808	1079021.295	19630.04832	73646.61906	7.242153509	73.61897229	1.41741	3576072.782	9126.150391	191530600
27	2/7/2003	63.21304243	2085.057785	4614352617	32759.90277	1078471.533	19620.65018	73624.55534	7.239688713	73.6148478	1.39955	3576093.358	9150.950195	153250800

2.2 Exploratory Data Analysis (inspection, data profiling, visualizations)

- To begin with, import the necessary python libraries (for this example use - pandas, NumPy, matplotlib etc) and load the data set.
- Look at the data and check field types. Now, for simplicity you can download the .csv file for the dataset.

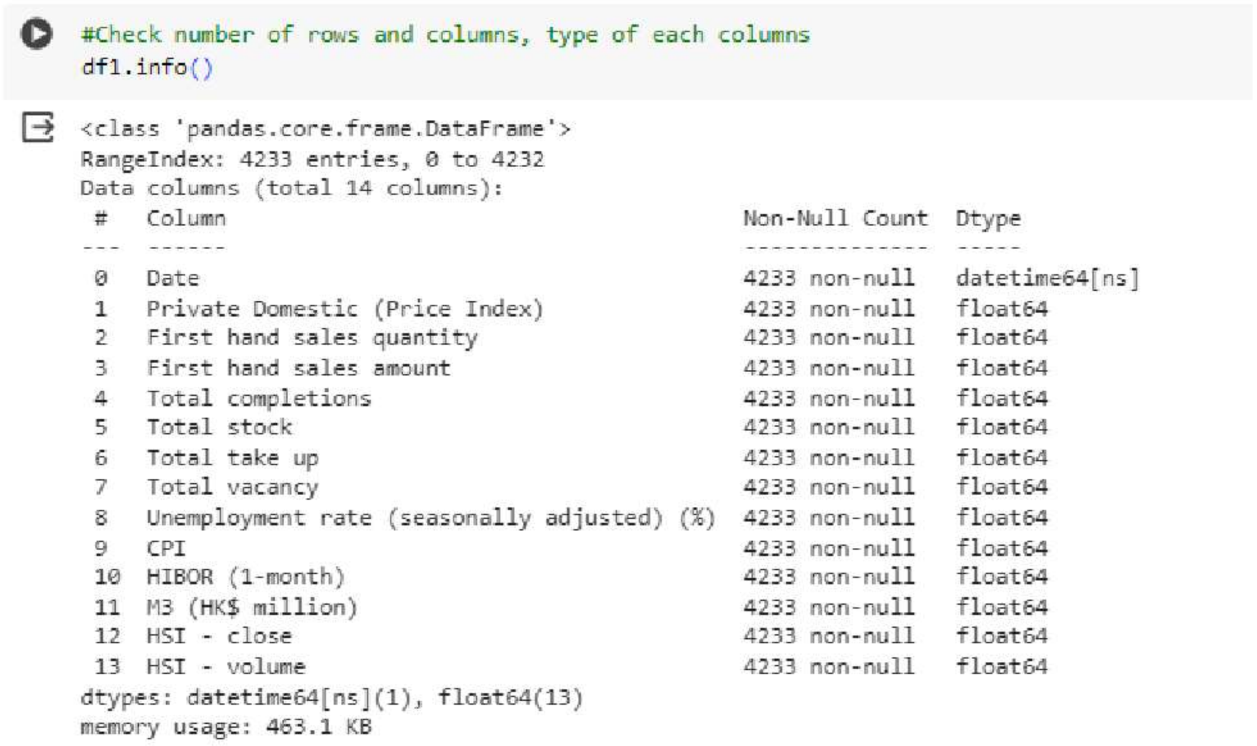


```python
import pandas as pd
# Input the dataset
df1 = pd.read_excel('/content/drive/MyDrive/AI-TSF/TSF-Labs/Exp 1/Housing market data.xlsx',
                    sheet_name='Upsampled')
df1.to_csv('Housing market data.csv', index=False)
df1.head(5)
```

	Date	Private Domestic (Price Index)	First hand sales quantity	First hand sales amount	Total completions	Total stock	Total take up	Total vacancy	Unemployment rate (seasonally adjusted) (%)	CPI	HIBOR (1-month)	M3 (HK$ million)	HSI - close	HSI - volume
0	2003-01-02	64.559769	1854.689208	5.846897e+09	33983.762413	1.092844e+06	19914.083323	74177.818318	7.301525	73.721486	1.44420	3.562189e+06	9365.519531	126907400.0
1	2003-01-03	64.506848	1892.762372	5.927523e+09	33932.818955	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
2	2003-01-06	64.453848	1931.338866	5.998876e+09	33882.041253	1.091646e+06	19886.125236	74133.429251	7.296561	73.712684	1.40402	3.563044e+06	9665.959961	234912800.0
3	2003-01-07	64.400768	1970.418692	6.060958e+09	33831.429307	1.091050e+06	19872.438591	74111.243068	7.294081	73.708301	1.39063	3.563562e+06	9652.400391	267021800.0
4	2003-01-08	64.347610	2010.001848	6.113768e+09	33780.983117	1.090457e+06	19858.946877	74089.062450	7.291601	73.703930	1.39732	3.564141e+06	9688.209961	202439200.0

```
from google.colab import files
files.download('Housing market data.csv')
```

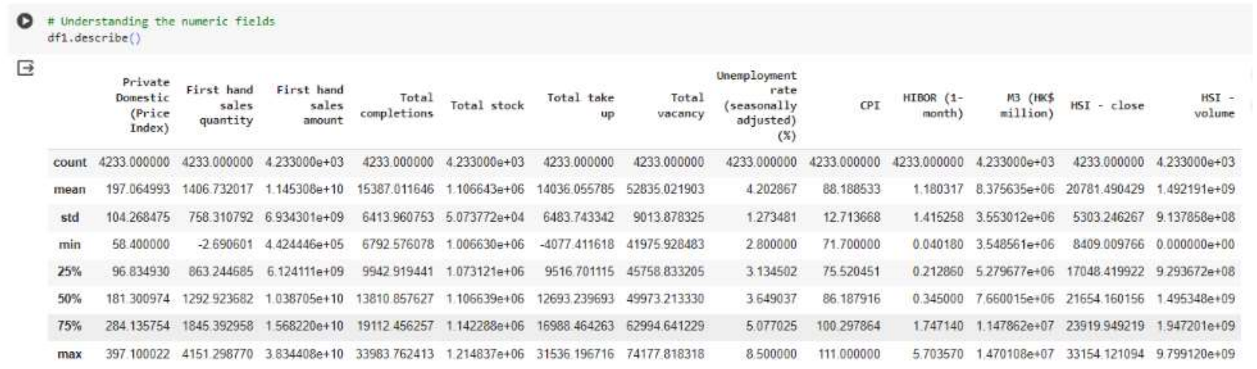
- Check number of rows and columns, type of each columns df1.info()



```
#Check number of rows and columns, type of each columns
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4233 entries, 0 to 4232
Data columns (total 14 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   Date                                     4233 non-null   datetime64[ns]
 1   Private Domestic (Price Index)           4233 non-null   float64
 2   First hand sales quantity                4233 non-null   float64
 3   First hand sales amount                  4233 non-null   float64
 4   Total completions                        4233 non-null   float64
 5   Total stock                              4233 non-null   float64
 6   Total take up                            4233 non-null   float64
 7   Total vacancy                            4233 non-null   float64
 8   Unemployment rate (seasonally adjusted) (%)  4233 non-null   float64
 9   CPI                                      4233 non-null   float64
 10  HIBOR (1-month)                          4233 non-null   float64
 11  M3 (HK$ million)                         4233 non-null   float64
 12  HSI - close                              4233 non-null   float64
 13  HSI - volume                             4233 non-null   float64
dtypes: datetime64[ns](1), float64(13)
memory usage: 463.1 KB
```

- To take a closer look at the data, used headfunction of the pandas library which returns the first five observations of the data. Similarly, tail returns the last five observations of the data set.



```
# Understanding the numeric fields
df1.describe()
```

	Private Domestic (Price Index)	First hand sales quantity	First hand sales amount	Total completions	Total stock	Total take up	Total vacancy	Unemployment rate (seasonally adjusted) (%)	CPI	HIBOR (1-month)	M3 (HK$ million)	HSI - close	HSI - volume
count	4233.000000	4233.000000	4.233000e+03	4233.000000	4.233000e+03	4233.000000	4233.000000	4233.000000	4233.000000	4233.000000	4.233000e+03	4233.000000	4.233000e+03
mean	197.064993	1406.732017	1.145308e+10	15387.011646	1.106643e+06	14036.055785	52835.021903	4.202867	88.188533	1.180317	8.375635e+06	20781.490429	1.492191e+09
std	104.268475	758.310792	6.934301e+09	6413.960753	5.073772e+04	6483.743342	9013.878325	1.273481	12.713668	1.415258	3.553012e+06	5303.246267	9.137858e+08
min	58.400000	-2.690601	4.424446e+05	6792.576078	1.006630e+06	-4077.411618	41975.928483	2.800000	71.700000	0.040180	3.548561e+06	8409.009766	0.000000e+00
25%	96.834930	863.244685	6.124111e+09	9942.919441	1.073121e+06	9516.701115	45758.833205	3.134502	75.520451	0.212860	5.279677e+06	17048.419922	9.293672e+08
50%	181.300974	1292.923682	1.038705e+10	13810.857627	1.106639e+06	12693.239693	49973.213330	3.649037	86.187916	0.345000	7.660015e+06	21654.160156	1.495348e+09
75%	284.135754	1845.392958	1.568220e+10	19112.456257	1.142288e+06	16988.464263	62994.641229	5.077025	100.297864	1.747140	1.147862e+07	23919.949219	1.947201e+09
max	397.100022	4151.298770	3.834408e+10	33983.762413	1.214837e+06	31536.196716	74177.818318	8.500000	111.000000	5.703570	1.470108e+07	33154.121094	9.799120e+09

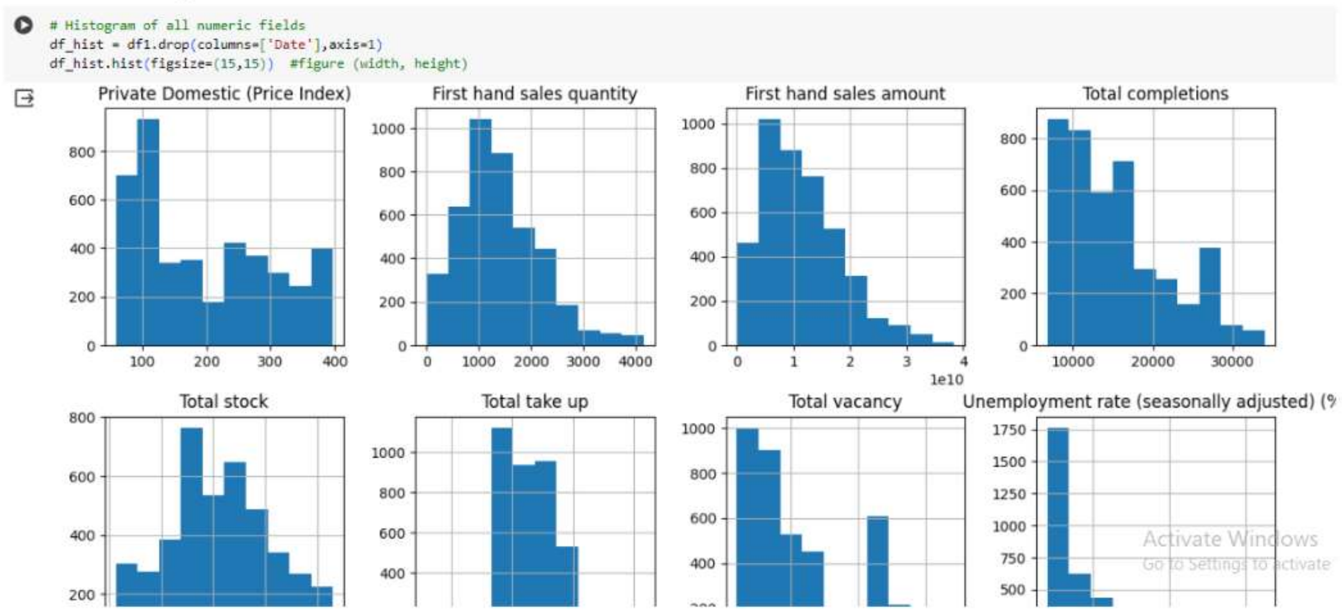
Now,observe this result and try to come up with some notable information about the given dataset. i.e.

Observation 1) *Here as you can notice the mean value is more than the median value of most columns which is represented by 50% (50th percentile) in the index column.*

Observation 2) *There is notably a big difference between 75th percentile and max values of certain fields like "First hand sales quantity","First hand sales amount"," Total completions" etc.*

Thus observations 1 and 2 suggest that there are extreme values-Outliers in our data set. We get the same conclusion once we look at the histograms of all the numeric fields.

- Histogram of all numeric fields



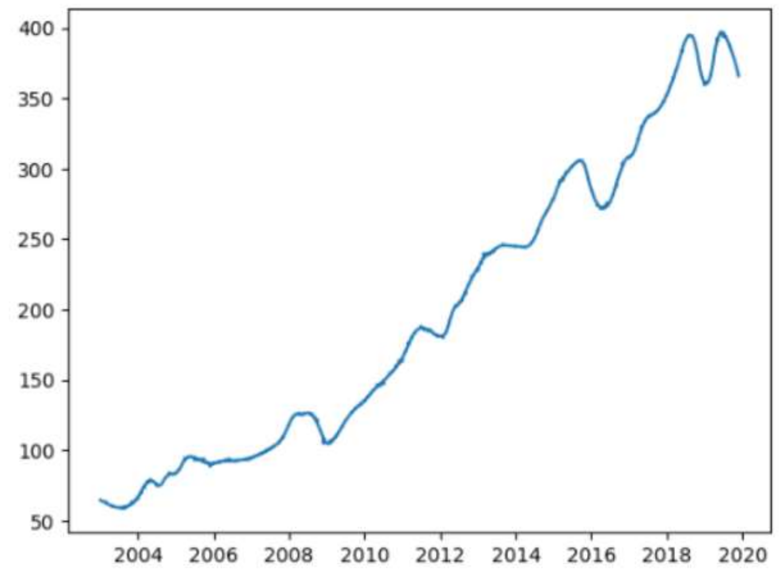
```
# Histogram of all numeric fields
df_hist = df1.drop(columns=['Date'],axis=1)
df_hist.hist(figsize=(15,15))  #figure (width, height)
```

- Visualize the target variable

```
[30] import matplotlib.pyplot as plt

     # Visualise the target variable
     plt.plot(df1['Date'], df1['Private Domestic (Price Index)'])
```

```
[<matplotlib.lines.Line2D at 0x7fe9d53be9e0>]
```

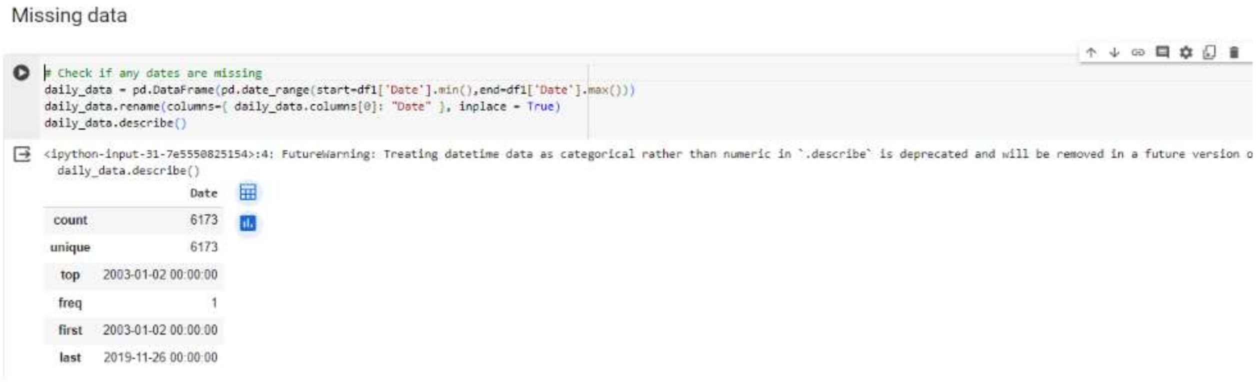


Target variable/Dependent variable ('Private Domestic (Price Index)')has a rising trend

2.3. Data Cleaning (missing data, outliers detection and treatment)

Data cleaning is the process of identifying and correcting inaccurate records from a dataset along with recognizing unreliable or irrelevant parts of the data. We will be focusing on handling missing data and outliers in this experiment.

- Missing Data



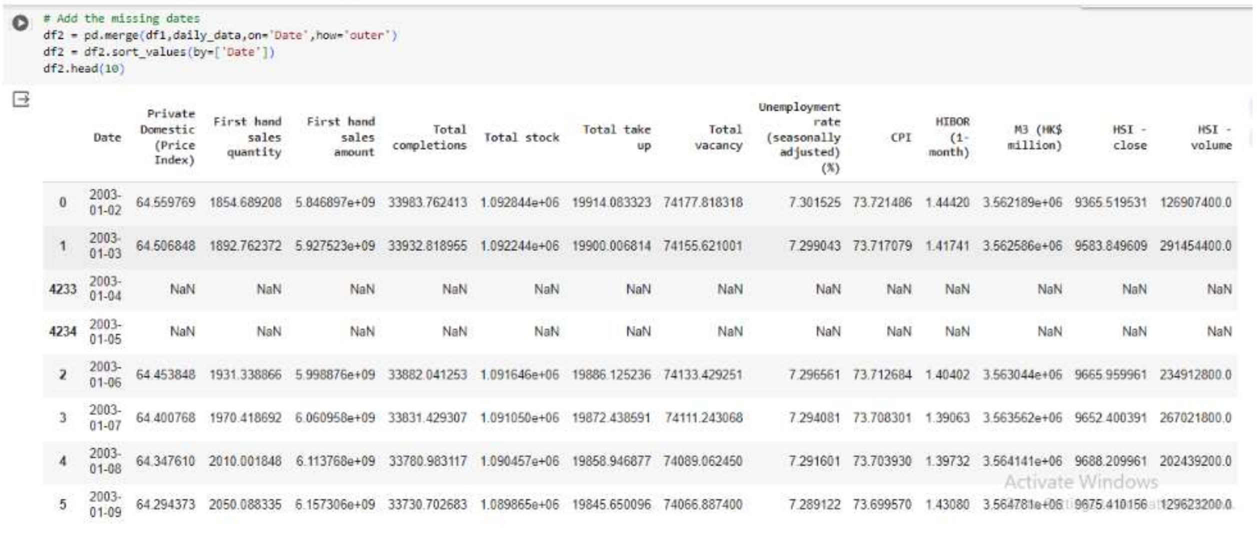
Missing data

```
# Check if any dates are missing
daily_data = pd.DataFrame(pd.date_range(start=df1['Date'].min(),end=df1['Date'].max()))
daily_data.rename(columns={ daily_data.columns[0]: "Date" }, inplace = True)
daily_data.describe()
```

```
<ipython-input-31-7e5550825154>:4: FutureWarning: Treating datetime data as categorical rather than numeric in `.describe` is deprecated and will be removed in a future version o
  daily_data.describe()
```

	Date
count	6173
unique	6173
top	2003-01-02 00:00:00
freq	1
first	2003-01-02 00:00:00
last	2019-11-26 00:00:00

- Adding the missing dates



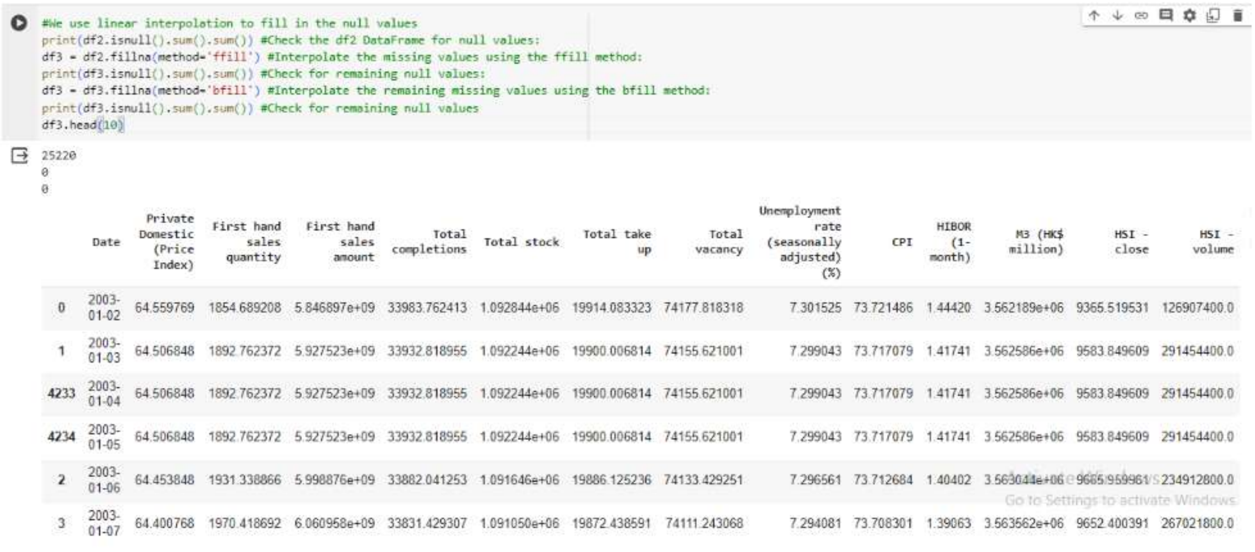
```
# Add the missing dates
df2 = pd.merge(df1,daily_data,on='Date',how='outer')
df2 = df2.sort_values(by=['Date'])
df2.head(10)
```

	Date	Private Domestic (Price Index)	First hand sales quantity	First hand sales amount	Total completions	Total stock	Total take up	Total vacancy	Unemployment rate (seasonally adjusted) (%)	CPI	HIBOR (1-month)	M3 (HK$ million)	HSI - close	HSI - volume
0	2003-01-02	64.559769	1854.689208	5.846897e+09	33983.762413	1.092844e+06	19914.083323	74177.818318	7.301525	73.721486	1.44420	3.562189e+06	9365.519531	126907400.0
1	2003-01-03	64.506848	1892.762372	5.927523e+09	33932.818955	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
4233	2003-01-04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4234	2003-01-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2003-01-06	64.453848	1931.338866	5.998876e+09	33882.041253	1.091646e+06	19886.125236	74133.429251	7.296561	73.712684	1.40402	3.563044e+06	9665.959961	234912800.0
3	2003-01-07	64.400768	1970.418692	6.060958e+09	33831.429307	1.091050e+06	19872.438591	74111.243068	7.294081	73.708301	1.39063	3.563562e+06	9652.400391	267021800.0
4	2003-01-08	64.347610	2010.001848	6.113768e+09	33780.983117	1.090457e+06	19858.946877	74089.062450	7.291601	73.703930	1.39732	3.564141e+06	9688.209961	202439200.0
5	2003-01-09	64.294373	2050.088335	6.157306e+09	33730.702683	1.089865e+06	19845.650096	74066.887400	7.289122	73.699570	1.43080	3.564781e+06	9675.410156	129623200.0

->Our raw data starts from "2003–01–02" and ends at "2019–11–26". There are 6173 days between "2003–01–02" and "2019–11–26" but the original data only had 4233 records. So a few dates are missing.

->We create a new dataset with all the 6173 dates and join the original dataset with this new dataset. This leads to null values for all the records not available in the original dataset.

- **Interpolation** is a technique used to estimate values within a range of known values. In the context of time series data, interpolation can be used to estimate missing values based on the values of neighboring points in the time series.
- Interpolation is mostly used while working with time-series data because, in time-series data, we like to fill missing values with the previous one or two values.



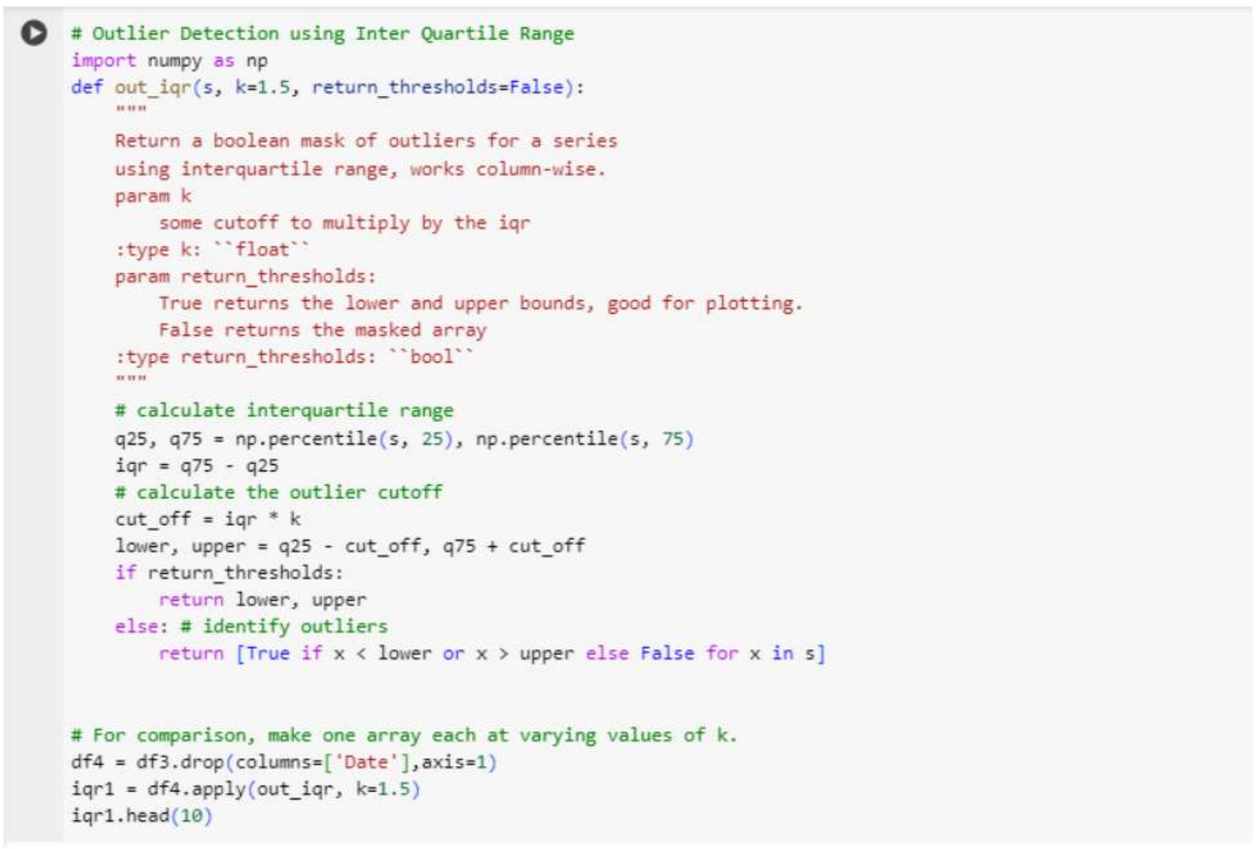
```python
#We use linear interpolation to fill in the null values
print(df2.isnull().sum().sum()) #Check the df2 DataFrame for null values:
df3 = df2.fillna(method='ffill') #Interpolate the missing values using the ffill method:
print(df3.isnull().sum().sum()) #Check for remaining null values:
df3 = df3.fillna(method='bfill') #Interpolate the remaining missing values using the bfill method:
print(df3.isnull().sum().sum()) #Check for remaining null values
df3.head(10)
```

```
25220
0
0
```

	Date	Private Domestic (Price Index)	First hand sales quantity	First hand sales amount	Total completions	Total stock	Total take up	Total vacancy	Unemployment rate (seasonally adjusted) (%)	CPI	HIBOR (1-month)	M3 (HK$ million)	HSI - close	HSI - volume
0	2003-01-02	64.559769	1854.689208	5.846897e+09	33983.762413	1.092844e+06	19914.083323	74177.818318	7.301525	73.721486	1.44420	3.562189e+06	9365.519531	126907400.0
1	2003-01-03	64.506848	1892.762372	5.927523e+09	33932.818955	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
4233	2003-01-04	64.506848	1892.762372	5.927523e+09	33932.818955	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
4234	2003-01-05	64.506848	1892.762372	5.927523e+09	33932.818955	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
2	2003-01-06	64.453848	1931.338866	5.998876e+09	33882.041253	1.091646e+06	19886.125236	74133.429251	7.296561	73.712684	1.40402	3.563044e+06	9665.959961	234912800.0
3	2003-01-07	64.400768	1970.418692	6.060958e+09	33831.429307	1.091050e+06	19872.438591	74111.243068	7.294081	73.708301	1.39063	3.563562e+06	9652.400391	267021800.0

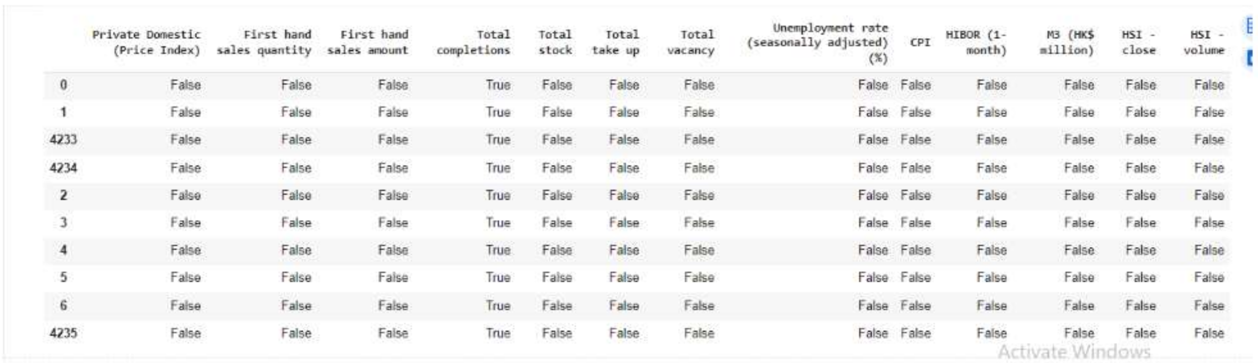
- **Outlier detection**- In statistics, an outlier is an observation point that is distant from other observations.
- You can find more information on outlier detection by clicking here
  Various techniques to detect outlier in the dataset -
  1. **Z-Score method**: This method calculates the distance between each data point and the mean of the data, expressed as a number of standard deviations.
  2. **Interquartile Range (IQR) method:** This method looks at the range of the middle 50% of the data, and considers data points that fall outside of this range to be outliers.

Here, we are discussing the IQR method for the outlier detection.



```python
# Outlier Detection using Inter Quartile Range
import numpy as np
def out_iqr(s, k=1.5, return_thresholds=False):
    """
    Return a boolean mask of outliers for a series
    using interquartile range, works column-wise.
    param k
        some cutoff to multiply by the iqr
    :type k: ``float``
    param return_thresholds:
        True returns the lower and upper bounds, good for plotting.
        False returns the masked array
    :type return_thresholds: ``bool``
    """
    # calculate interquartile range
    q25, q75 = np.percentile(s, 25), np.percentile(s, 75)
    iqr = q75 - q25
    # calculate the outlier cutoff
    cut_off = iqr * k
    lower, upper = q25 - cut_off, q75 + cut_off
    if return_thresholds:
        return lower, upper
    else: # identify outliers
        return [True if x < lower or x > upper else False for x in s]


# For comparison, make one array each at varying values of k.
df4 = df3.drop(columns=['Date'],axis=1)
iqr1 = df4.apply(out_iqr, k=1.5)
iqr1.head(10)
```



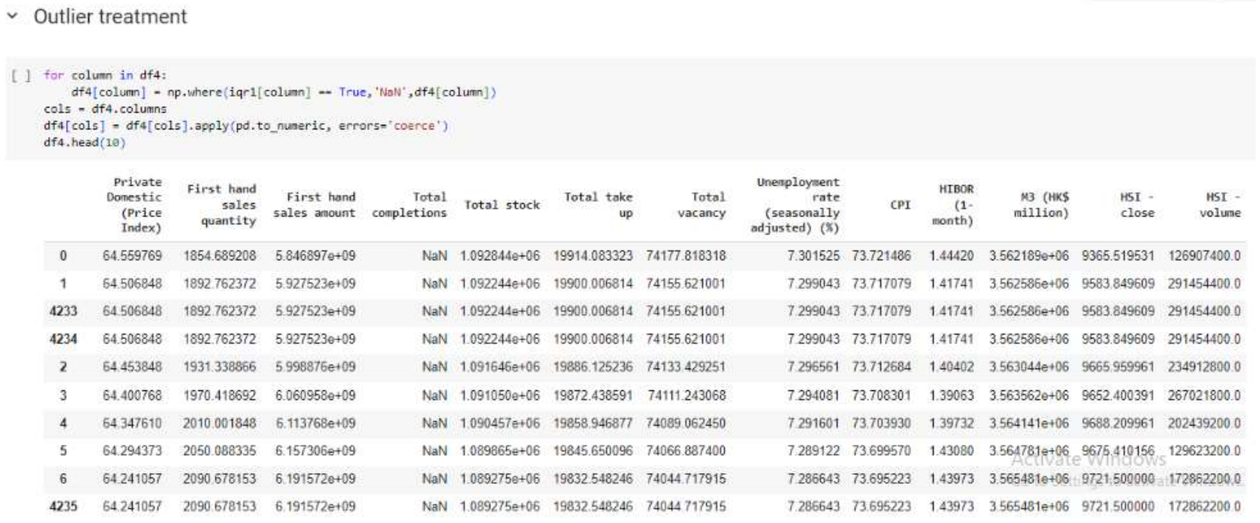
	Private Domestic (Price Index)	First hand sales quantity	First hand sales amount	Total completions	Total stock	Total take up	Total vacancy	Unemployment rate (seasonally adjusted) (%)	CPI	HIBOR (1-month)	M3 (HK$ million)	HSI - close	HSI - volume
0	False	False	False	True	False	False	False	False	False	False	False	False	False
1	False	False	False	True	False	False	False	False	False	False	False	False	False
4233	False	False	False	True	False	False	False	False	False	False	False	False	False
4234	False	False	False	True	False	False	False	False	False	False	False	False	False
2	False	False	False	True	False	False	False	False	False	False	False	False	False
3	False	False	False	True	False	False	False	False	False	False	False	False	False
4	False	False	False	True	False	False	False	False	False	False	False	False	False
5	False	False	False	True	False	False	False	False	False	False	False	False	False
6	False	False	False	True	False	False	False	False	False	False	False	False	False
4235	False	False	False	True	False	False	False	False	False	False	False	False	False

- The Interquartile range (IQR) is calculated as the difference between the 75th and the 25th percentiles of the data. The IQR can be used to identify outliers by defining limits on the sample values that are a factor k of the IQR below the 25th percentile or above the 75th percentile. The common value for the factor k is the value 1.5 (which we have used here). A factor k of 3 or more can be used to identify values that are extreme outliers or "far outs".
- For the initial values, fields like 'Total Completion' have a lot of outliers.
- If we know that the distribution of values in the sample is Gaussian or Gaussian-like, we can use the standard deviation of the sample as a cut-off for identifying outliers. Three standard deviations from the mean is a common cut-off in practice for

identifying outliers in a Gaussian or Gaussian-like distribution. For smaller samples of data, perhaps a value of 2 standard deviations (95%) can be used, and for larger samples, perhaps a value of 4 standard deviations (99.9%) can be used.

- **Outlier treatment**

All the identified outliers are replaced by nulls first.

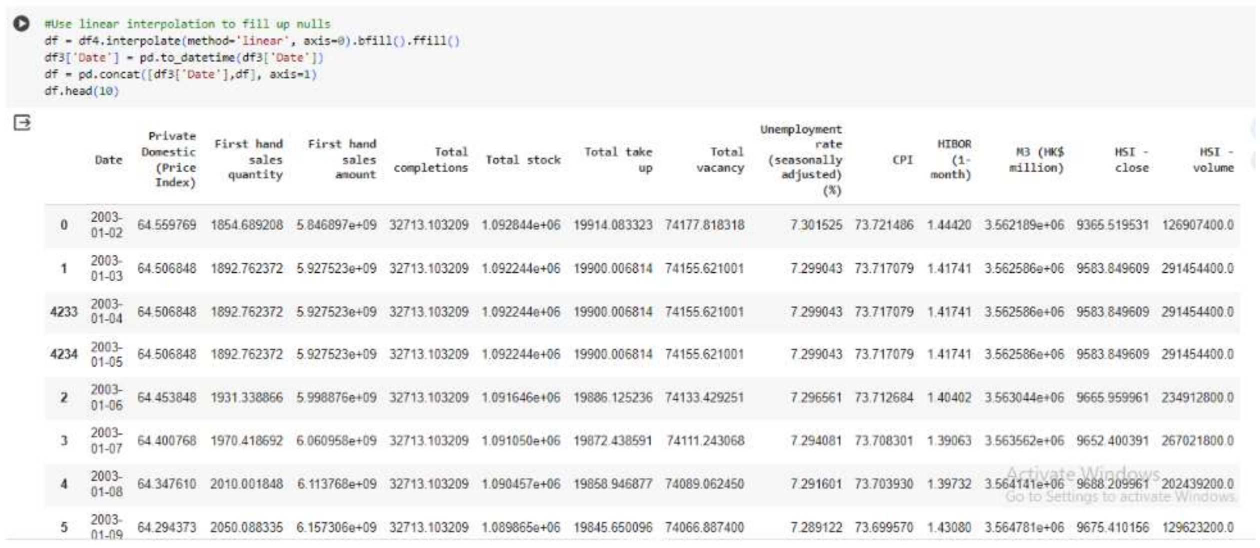


Outlier treatment

```python
for column in df4:
    df4[column] = np.where(iqr1[column] == True,'NaN',df4[column])
cols = df4.columns
df4[cols] = df4[cols].apply(pd.to_numeric, errors='coerce')
df4.head(10)
```

	Private Domestic (Price Index)	First hand sales quantity	First hand sales amount	Total completions	Total stock	Total take up	Total vacancy	Unemployment rate (seasonally adjusted) (%)	CPI	HIBOR (1-month)	M3 (HK$ million)	HSI - close	HSI - volume
0	64.559769	1854.689208	5.846897e+09	NaN	1.092844e+06	19914.083323	74177.818318	7.301525	73.721486	1.44420	3.562189e+06	9365.519531	126907400.0
1	64.506848	1892.762372	5.927523e+09	NaN	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
4233	64.506848	1892.762372	5.927523e+09	NaN	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
4234	64.506848	1892.762372	5.927523e+09	NaN	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
2	64.453848	1931.338866	5.998876e+09	NaN	1.091646e+06	19886.125236	74133.429251	7.296561	73.712684	1.40402	3.563044e+06	9665.959961	234912800.0
3	64.400768	1970.418692	6.060958e+09	NaN	1.091050e+06	19872.438591	74111.243068	7.294081	73.708301	1.39063	3.563562e+06	9652.400391	267021800.0
4	64.347610	2010.001848	6.113768e+09	NaN	1.090457e+06	19858.946877	74089.062450	7.291601	73.703930	1.39732	3.564141e+06	9688.209961	202439200.0
5	64.294373	2050.088335	6.157306e+09	NaN	1.089865e+06	19845.650096	74066.887400	7.289122	73.699570	1.43080	3.564781e+06	9675.410156	129623200.0
6	64.241057	2090.678153	6.191572e+09	NaN	1.089275e+06	19832.548246	74044.717915	7.286643	73.695223	1.43973	3.565481e+06	9721.500000	172862200.0
4235	64.241057	2090.678153	6.191572e+09	NaN	1.089275e+06	19832.548246	74044.717915	7.286643	73.695223	1.43973	3.565481e+06	9721.500000	172862200.0

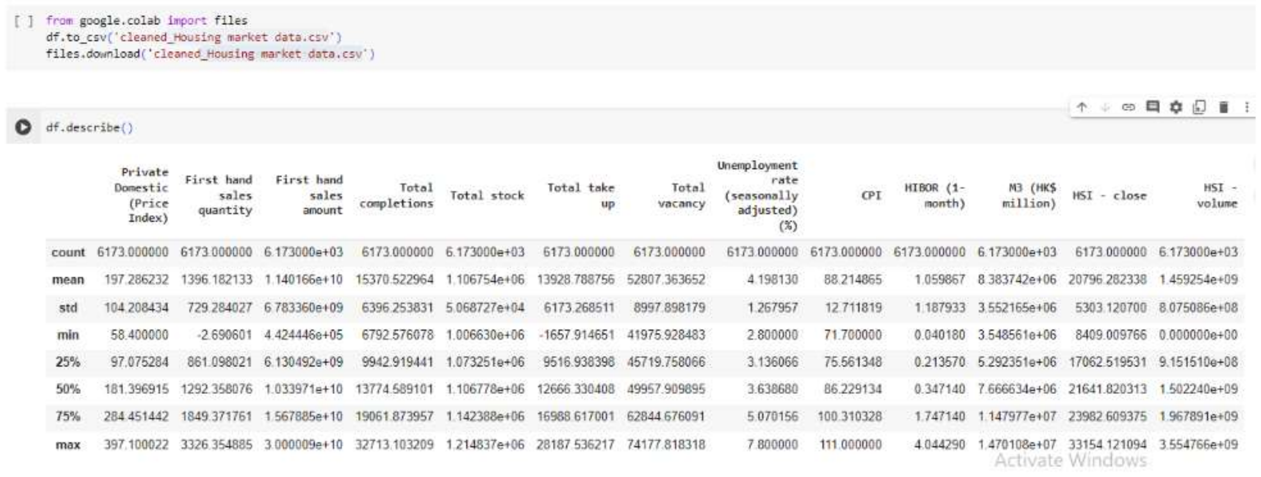
- Then the nulls are filled by linear interpolation



```python
#Use linear interpolation to fill up nulls
df = df4.interpolate(method='linear', axis=0).bfill().ffill()
df3['Date'] = pd.to_datetime(df3['Date'])
df = pd.concat([df3['Date'],df], axis=1)
df.head(10)
```

	Date	Private Domestic (Price Index)	First hand sales quantity	First hand sales amount	Total completions	Total stock	Total take up	Total vacancy	Unemployment rate (seasonally adjusted) (%)	CPI	HIBOR (1-month)	M3 (HK$ million)	HSI - close	HSI - volume
0	2003-01-02	64.559769	1854.689208	5.846897e+09	32713.103209	1.092844e+06	19914.083323	74177.818318	7.301525	73.721486	1.44420	3.562189e+06	9365.519531	126907400.0
1	2003-01-03	64.506848	1892.762372	5.927523e+09	32713.103209	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
4233	2003-01-04	64.506848	1892.762372	5.927523e+09	32713.103209	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
4234	2003-01-05	64.506848	1892.762372	5.927523e+09	32713.103209	1.092244e+06	19900.006814	74155.621001	7.299043	73.717079	1.41741	3.562586e+06	9583.849609	291454400.0
2	2003-01-06	64.453848	1931.338866	5.998876e+09	32713.103209	1.091646e+06	19886.125236	74133.429251	7.296561	73.712684	1.40402	3.563044e+06	9665.959961	234912800.0
3	2003-01-07	64.400768	1970.418692	6.060958e+09	32713.103209	1.091050e+06	19872.438591	74111.243068	7.294081	73.708301	1.39063	3.563562e+06	9652.400391	267021800.0
4	2003-01-08	64.347610	2010.001848	6.113768e+09	32713.103209	1.090457e+06	19858.946877	74089.062450	7.291601	73.703930	1.39732	3.564141e+06	9688.209961	202439200.0
5	2003-01-09	64.294373	2050.088335	6.157306e+09	32713.103209	1.089865e+06	19845.650096	74066.887400	7.289122	73.699570	1.43080	3.564781e+06	9675.410156	129623200.0

- Now, you can download the cleaned .csv file for the dataset and observe the difference between the original **Housing market data.csv** and **cleaned_Housing market data.csv**



```
[ ] from google.colab import files
    df.to_csv('cleaned_Housing market data.csv')
    files.download('cleaned_Housing market data.csv')
```

```
df.describe()
```

	Private Domestic (Price Index)	First hand sales quantity	First hand sales amount	Total completions	Total stock	Total take up	Total vacancy	Unemployment rate (seasonally adjusted) (%)	CPI	HIBOR (1-month)	M3 (HK$ million)	HSI - close	HSI - volume
count	6173.000000	6173.000000	6.173000e+03	6173.000000	6.173000e+03	6173.000000	6173.000000	6173.000000	6173.000000	6173.000000	6.173000e+03	6173.000000	6.173000e+03
mean	197.286232	1396.182133	1.140166e+10	15370.522964	1.106754e+06	13928.788756	52807.363652	4.198130	88.214865	1.059867	8.383742e+06	20796.282338	1.459254e+09
std	104.208434	729.284027	6.783360e+09	6396.253831	5.068727e+04	6173.268511	8997.898179	1.267957	12.711819	1.187933	3.552165e+06	5303.120700	8.075086e+08
min	58.400000	-2.690601	4.424446e+05	6792.576078	1.006630e+06	-1657.914651	41975.928483	2.800000	71.700000	0.040180	3.548561e+06	8409.009766	0.000000e+00
25%	97.075284	861.098021	6.130492e+09	9942.919441	1.073251e+06	9516.938398	45719.758066	3.136066	75.561348	0.213570	5.292351e+06	17062.519531	9.151510e+08
50%	181.396915	1292.358076	1.033971e+10	13774.589101	1.106778e+06	12666.330408	49957.909895	3.638680	86.229134	0.347140	7.666634e+06	21641.820313	1.502240e+09
75%	284.451442	1849.371761	1.567885e+10	19061.873957	1.142388e+06	16988.617001	62844.676091	5.070156	100.310328	1.747140	1.147977e+07	23982.609375	1.967891e+09
max	397.100022	3326.354885	3.000009e+10	32713.103209	1.214837e+06	28187.536217	74177.818318	7.800000	111.000000	4.044290	1.470108e+07	33154.121094	3.554766e+09

Additional reading for cleaning process -
https://www.linkedin.com/advice/0/what-common-data-cleaning-techniques-time-series-ojnff#:~:text=MBA%2CMS%2CPhD-,Common%20data%20cleaning%20techniques%20for%20time%20series%20data%20in%20machine,engineering%20to%20summarize%2C%20lag%2C%20or

References-

1. https://medium.com/analytics-vidhya/cleaning-and-understanding-multivariate-time-series-data-6554eefbda9c
2. Additional learning -https://www.linkedin.com/advice/0/what-common-data-cleaning-techniques-time-series-ojnff#:~:text=MBA%2CMS%2CPhD-,Common%20data%20cleaning%20techniques%20for%20time%20series%20data%20in%20machine,engineering%20to%20summarize%2C%20lag%2C%20or