وزارة التعليم العالي

جامعة حمص

الكلية التطبيقية

قسم تقنيات حاسوب

السنة الثالثة

قواعد معطيات٣

القسم العملي

المحاضرة الثانية

اعداد المدرسين:

م. زينب مراد **م. بتول الليوس**

Preprocessing Using Python

Attributes:

Attributes has four types:

- Nominal.
- Ordinal.
- Interval-Scaled.
- Ratio-Scaled.

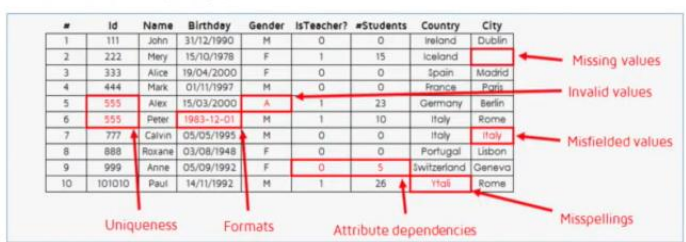
We can also classify attributes to two types (**discrete** and **continuous**):

Basic statistical Descriptions of Data:

Basic statistical descriptions can be used to give us overall picture of our data, and identify properties of the data and highlight which data values should be treated as noise or outliers.

I. **Central Tendency** They include measures like (***mean***, ***median***, ***mode*** and ***midrange***).
II. **Dispersion** of the data: In this kind we have some measures like (***variance*** and ***standard deviation***).
III. Graphs.

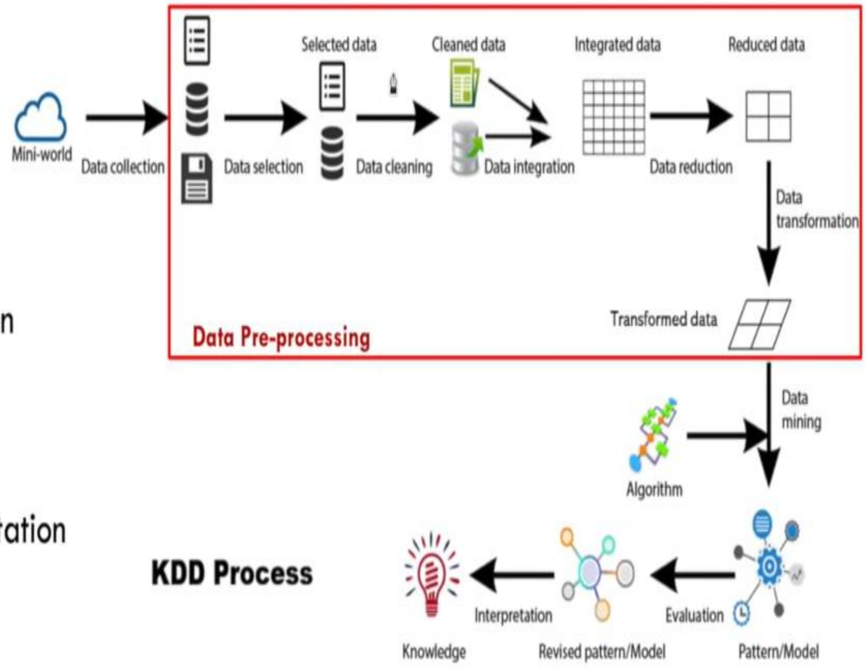
- Data in the Real World is Dirty: Lots of potentially incorrect data, e.g., instrument faulty, human or computer error, transmission error.
  - Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data.
    - e.g., Occupation=" " (missing data)
  - Noisy: containing noise, errors, or outliers.
    - e.g., Salary="−10" (an error)
  - Inconsistent: containing contradictories in codes or names, e.g.,
    - Age="42", Birthday="03/07/2010".
  - Intentional (e.g., disguised missing data)
    - Jan. 1 as everyone's birthday?



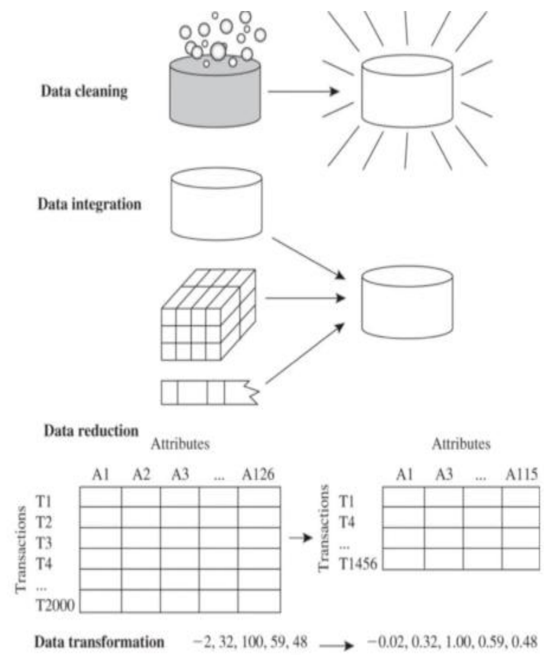
#	Id	Name	Birthday	Gender	IsTeacher?	#Students	Country	City
1	111	John	31/12/1990	M	0	0	Ireland	Dublin
2	222	Mery	15/10/1978	F	1	15	Iceland	
3	333	Alice	19/04/2000	F	0	0	Spain	Madrid
4	444	Mark	01/11/1997	M	0	0	France	Paris
5	555	Alex	15/03/2000	A	1	23	Germany	Berlin
6	555	Peter	1983-12-01	M	1	10	Italy	Rome
7	777	Calvin	05/05/1995	M	0	0	Italy	Italy
8	888	Roxane	03/08/1948	F	0	0	Portugal	Lisbon
9	999	Anne	05/09/1992	F	0	5	Switzerland	Geneva
10	101010	Paul	14/11/1992	M	1	26	Ytali	Rome

Data Pre-processing

1. Data collection
2. Data selection
3. Data cleaning
4. Data integration
5. Data transformation
6. Data mining
7. Pattern evaluation
8. Knowledge presentation



- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.
- Data integration
  - Integration of multiple databases or files.
- Data reduction
  - Dimensionality reduction.
  - Numerosity reduction.
- Data transformation
  - Normalization.
  - Concept hierarchy generation.
  - Discretization



Using Pandas to get statistical data description:

we'll use Pandas library to get some statistical descriptions about car sales dataset as an example.

The dataset has the following attributes:

#	Attribute	Explanation
1	Make	Manufactured Company.
2	Color	Car Color.
3	Odometer (KM)	The total distance that car walked.
4	Doors	The number of doors.
5	Price	Car price in $.

- ***Import required libraries:***

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

- ***Read car sales dataset:***

```
#read car sales dataset using read_csv function and store it into DataFrame
car_sdd = pd.read_csv('car_sdd.csv')
```

the result data frame would be like the following:

Out[3]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

- ***Describing Dataset:***

Pandas offers a lot of functions to get statistical information about any dataset.

✓ describe() function: give some statistical information about numeric attributes only such as (mean, std, min, max and more, ….).

```
car_sdd.describe()
```

Out[4]:

	Odometer (KM)	Doors
count	10.000000	10.000000
mean	78601.400000	4.000000
std	61983.471735	0.471405
min	11179.000000	3.000000
25%	35836.250000	4.000000
50%	57369.000000	4.000000
75%	96384.500000	4.000000
max	213095.000000	5.000000

✓ info() function: gives summary about dataset attributes (total objects, type, and many more).

```
car_sdd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Make           10 non-null     object
 1   Colour         10 non-null     object
 2   Odometer (KM)  10 non-null     int64
 3   Doors          10 non-null     int64
 4   Price          10 non-null     object
dtypes: int64(2), object(3)
memory usage: 528.0+ bytes
```

✓ To view central tendency measurements using built in Pandas functions as follows:

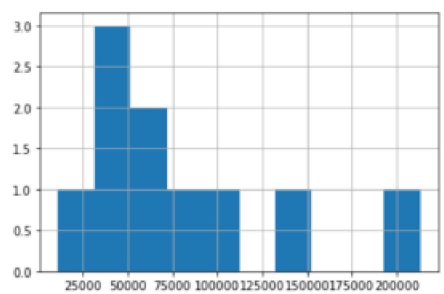
```
print('Odometer attribute mean:',car_sdd['Odometer (KM)'].mean())
print('Odometer attribute median:',car_sdd['Odometer (KM)'].median())
```

✓ To calculate standard deviation using std() function provided by Pandas library.

```
car_sdd['Odometer (KM)'].std()
```

```
#view Odometer distribution using hist() function:
car_sdd['Odometer (KM)'].hist()
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x2628e11e3a0>
```



this histogram shows:

- about average of odometer values fall into in the left part.
- two values could be considered outliers because they're not in the range of all values.

Data Cleaning:

using Pandas library we're going to handle missing value problem:

- ***Import required libraries:***

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

- ***Load Dataset with Missing Values:***

```
missing_c_df = pd.read_csv('car_pre.csv')
missing_c_df
```

We can notice that python use 'NaN' when we don't have values.

	Make	Colour	Odometer (KM)	Doors	Price
0	Honda	White	35431.0	4.0	15323.0
1	BMW	Blue	192714.0	5.0	19943.0
2	Honda	White	84714.0	4.0	28343.0
3	Toyota	White	154365.0	4.0	13434.0
4	Nissan	Blue	181577.0	3.0	14043.0
...	...	...	...	...	...
995	Toyota	Black	35820.0	4.0	32042.0
996	NaN	White	155144.0	3.0	5716.0
997	Nissan	Blue	66604.0	4.0	31570.0
998	Honda	White	215883.0	4.0	4001.0
999	Toyota	Blue	248360.0	4.0	12732.0

1000 rows × 5 columns

- ***Apply Data Cleaning Methods using Pandas:***

1. Fill in the missing value automatically using (fillna(value, inplace=True or False)).

   'inplace' has False value by default. if it is true then inplace will modify any other views on this object.

   Note: if you want to change an attribute values we must mention that Pandas requires to reassign these values to that attribute.

To fill Odometer (KM) and Price attributes with the mean value:

```
#use fillna() function with just passing the value to replace NaN:
mean_odo = missing_c_df['Odometer (KM)'].mean()
missing_c_df['Odometer (KM)'].fillna(mean_odo)
missing_c_df.head(10)
```

	Make	Colour	Odometer (KM)	Doors	Price
0	Honda	White	35431.0	4.0	15323.0
1	BMW	Blue	192714.0	5.0	19943.0
2	Honda	White	84714.0	4.0	28343.0
3	Toyota	White	154365.0	4.0	13434.0
4	Nissan	Blue	181577.0	3.0	14043.0
5	Honda	Red	42652.0	4.0	23883.0
6	Toyota	Blue	163453.0	4.0	8473.0
7	Honda	White	NaN	4.0	20306.0
8	NaN	White	130538.0	4.0	9374.0
9	Honda	Blue	51029.0	4.0	26683.0

Now let's execute the previous code and set 'inplace' parameter to true, then notice the changes:

```
#use fillna() function with passing the value to replace NaN and inplace=True:
missing_c_df['Odometer (KM)'].fillna(mean_odo, inplace=True)
missing_c_df["Price"].fillna(missing_c_df['Price'].mean(), inplace=True)
#view first 10 lines:
missing_c_df.head(10)
```

Out[5]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Honda	White	35431.000000	4.0	15323.0
1	BMW	Blue	192714.000000	5.0	19943.0
2	Honda	White	84714.000000	4.0	28343.0
3	Toyota	White	154365.000000	4.0	13434.0
4	Nissan	Blue	181577.000000	3.0	14043.0
5	Honda	Red	42652.000000	4.0	23883.0
6	Toyota	Blue	163453.000000	4.0	8473.0
7	Honda	White	131253.237895	4.0	20306.0
8	NaN	White	130538.000000	4.0	9374.0
9	Honda	Blue	51029.000000	4.0	26683.0

2. fill missing values in Make attribute using most frequent value:

let's find the most frequent value using value_counts() function:

```
#get value counts for Make attribute:
missing_c_df['Make'].value_counts()
```

Out[6]:

```
Toyota    379
Honda     292
Nissan    183
BMW        97
Name: Make, dtype: int64
```

```
#filling Make attribute with most frequent value:
missing_c_df['Make'].fillna(missing_c_df['Make'].value_counts().index[0],inplace=True)
missing_c_df.head(10)
```

Out[7]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Honda	White	35431.000000	4.0	15323.0
1	BMW	Blue	192714.000000	5.0	19943.0
2	Honda	White	84714.000000	4.0	28343.0
3	Toyota	White	154365.000000	4.0	13434.0
4	Nissan	Blue	181577.000000	3.0	14043.0
5	Honda	Red	42652.000000	4.0	23883.0
6	Toyota	Blue	163453.000000	4.0	8473.0
7	Honda	White	131253.237895	4.0	20306.0
8	Toyota	White	130538.000000	4.0	9374.0
9	Honda	Blue	51029.000000	4.0	26683.0

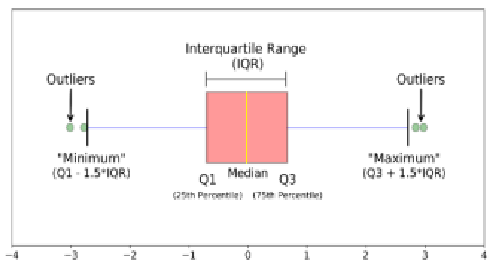
3. drop rows with missing values:

We want to remove (ignore) each row that has a 'NaN' for Doors attribute, . To do that we'll use dropna() function,

```
#drop records with missing Doors attribute:
#axis to specify reads, how determines if there is at least one NaN values.
#subset specifies the attributes we want to examine against.
missing_c_df.dropna(axis=0, how=”any”, inplace=True, subset=[“Doors”])
```

Boxplot Graph:

Boxplot displays five-number summary of a distribution minimum, Q1, Median, Q3, maximum.

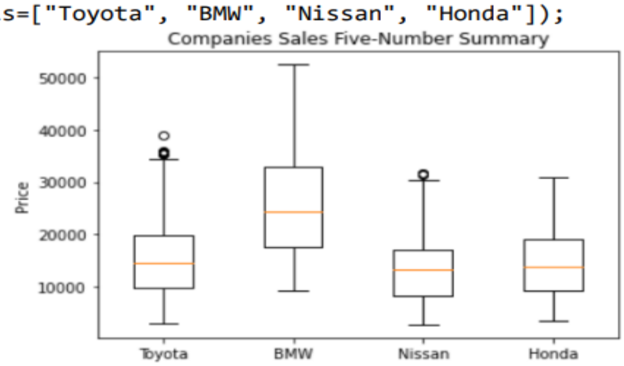


After filling in missing values for price attribute, we'll extract each manufactured company sales alone:

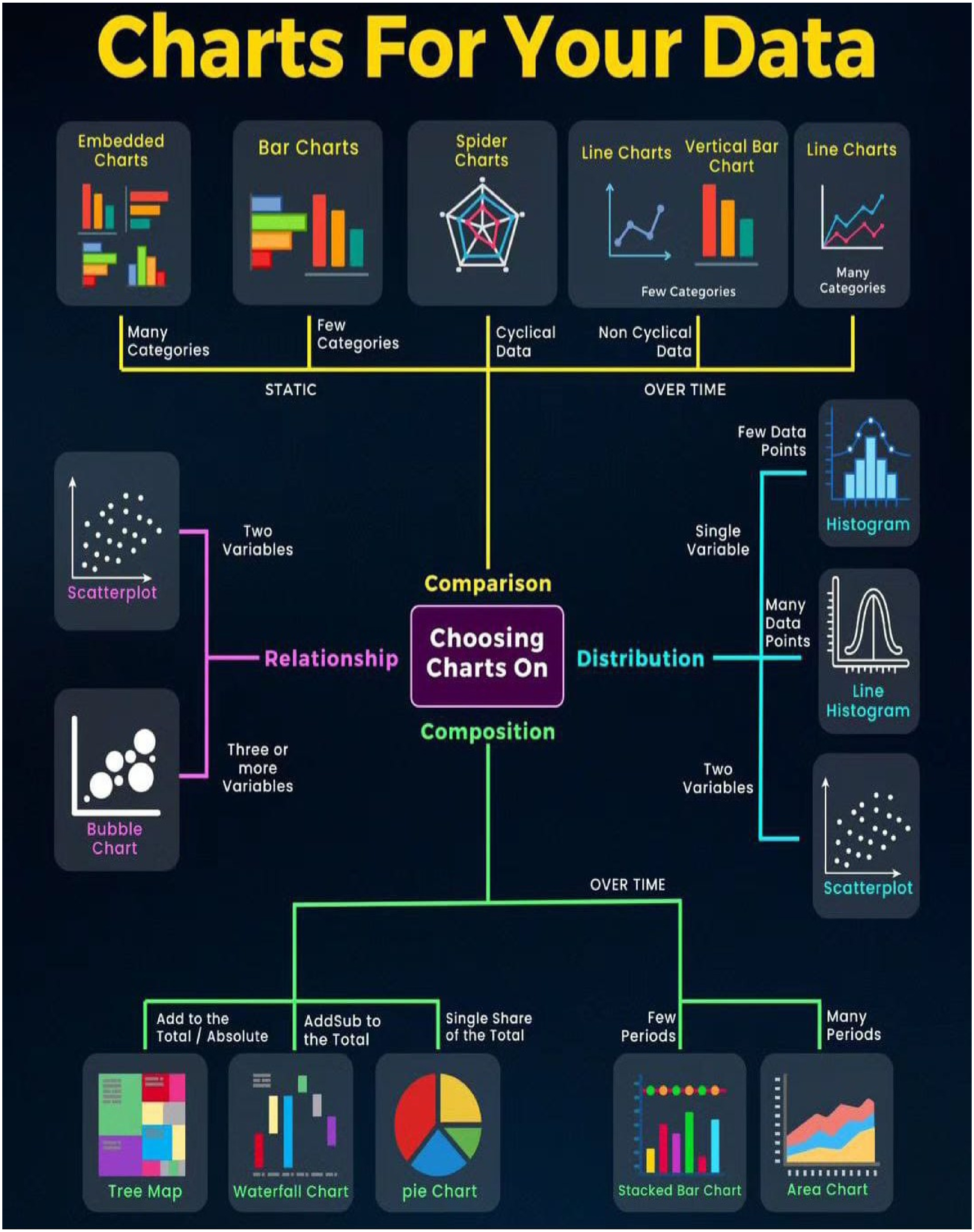
```
#visulaize all cars sales using boxplot graph:
#get all companies in new dataframes:
toyota = missing_c_df[missing_c_df["Make"]=="Toyota"] #get only objects that has Toyota as make value
bmw = missing_c_df[missing_c_df["Make"]=="BMW"]

nissan = missing_c_df[missing_c_df["Make"]=="Nissan"]
honda = missing_c_df[missing_c_df["Make"]=="Honda"]
```

Then visualize boxplot for each company sales in the same graph:

```
prices = [toyota["Price"], bmw["Price"], nissan["Price"], honda["Price"]] #extract price feature values to visualize
fig, ax = plt.subplots() #create multiple plots on the same figure
plt.title("Companies Sales Five-Number Summary")
plt.ylabel("Price")
plt.xticks(rotation=0)
ax.boxplot(prices, labels=["Toyota", "BMW", "Nissan", "Honda"]);
```



For Toyota, we see that median price of items sold is 15000, Q1 is 10000, Q3 is 20000 and we have some outlying observations were plotted individually, as their values are more than Q3 + 1.5*IQR.



Charts For Your Data
Embedded Charts
Bar Charts
Spider Charts
Line Charts
Vertical Bar Chart
Few Categories
Line Charts
Many Categories
Many Categories
Few Categories
Cyclical Data
Non Cyclical Data
STATIC
OVER TIME
Few Data Points
Histogram
Single Variable
Scatterplot
Two Variables
Comparison
Many Data Points
Relationship
Choosing Charts On
Distribution
Line Histogram
Composition
Three or more Variables
Two Variables
Bubble Chart
Scatterplot
OVER TIME
Add to the Total / Absolute
AddSub to the Total
Single Share of the Total
Few Periods
Many Periods
Tree Map
Waterfall Chart
pie Chart
Stacked Bar Chart
Area Chart