LAPORAN

MATA KULIAH ANALISIS DATA EKSPLORATIF (A)

"EKSPLORASI PERTANYAAN DASAR PADA DATA, PENGGABUNGAN DATA (MERGING), TRANSFORMASI DATA, DAN DETEKSI OUTLIER"

DISUSUN OLEH:

Reza Putri Angga (22083010006)

DOSEN PENGAMPU:

Tresna Maulana Fahrudin S.ST., M.T. (NIP. 199305012022031007)

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN" JAWA TIMUR

2023

STUDI KASUS DAN PEMBAHASAN

1. EKSPLORASI PERTANYAAN DASAR (STUDI KASUS : *BREAST CANCER DATASET*)

Sebelum melakukan eksplorasi ke dalam *breast cancer dataset* atau dataset kanker payudara, harus di lakukan langkah-langkah awal untuk *meload* (membaca) dataset sebagai berikut.

- **Melakukan Load Dataset *Breast Cancer Dataset***

• Melakukan Load Dataset Breast Cancer Dataset

In [1]:
```
import pandas as pd

BreastCancer = pd.read_csv("breast-cancer.csv", sep = ";")
BreastCancer
```

Out[1]:

	age	menopause	tumor-size	inv-nodes	node-caps	deg-malig	breast	breast-quad	irradiat	Class
0	'40-49'	'premeno'	'15-19'	'0-2'	'yes'	'3'	'right'	'left_up'	'no'	'recurrence-events'
1	'50-59'	'ge40'	'15-19'	'0-2'	'no'	'1'	'right'	'central'	'no'	'no-recurrence-events'
2	'50-59'	'ge40'	'35-39'	'0-2'	'no'	'2'	'left'	'left_low'	'no'	'recurrence-events'
3	'40-49'	'premeno'	'35-39'	'0-2'	'yes'	'3'	'right'	'left_low'	'yes'	'no-recurrence-events'
4	'40-49'	'premeno'	'30-34'	'3-5'	'yes'	'2'	'left'	'right_up'	'no'	'recurrence-events'
...	...	...	...	...	...	...	...	...	...	...
272	'50-59'	'ge40'	'30-34'	'6-8'	'yes'	'2'	'left'	'left_low'	'no'	'no-recurrence-events'
273	'50-59'	'premeno'	'25-29'	'3-5'	'yes'	'2'	'left'	'left_low'	'yes'	'no-recurrence-events'
274	'30-39'	'premeno'	'30-34'	'6-8'	'yes'	'2'	'right'	'right_up'	'no'	'no-recurrence-events'
275	'50-59'	'premeno'	'15-19'	'0-2'	'no'	'2'	'right'	'left_low'	'no'	'no-recurrence-events'
276	'50-59'	'ge40'	'40-44'	'0-2'	'no'	'3'	'left'	'right_up'	'no'	'no-recurrence-events'

277 rows × 10 columns

Pada kode script di atas, di lakukan *load* dan pembacaan *breast-cancer* dataset menggunakan *library* pandas dengan jenis file csv. Kemudian, *breast-cancer* dataset tersebut di simpan ke dalam variabel BreastCancer dengan 277 baris dan 10 kolom. Di mana nama variabel ini akan di pergunakan untuk proses lebih lanjut dalam melakukan eskplorasi data.

- **Memeriksa Dan Menampilkan Nama-Nama Kolom Pada *Breast Cancer Dataset***

• Memeriksa Dan Menampilkan Nama-Nama Kolom Pada Breast Cancer Dataset

In [2]:
```
BreastCancerKolom = BreastCancer.columns

print("Di Tampilkan Nama-Nama Kolom Yang Ada Pada Breast Cancer Dataset : ")
for kolom in BreastCancerKolom:
    print(kolom)
```

```
Di Tampilkan Nama-Nama Kolom Yang Ada Pada Breast Cancer Dataset :
age
menopause
tumor-size
inv-nodes
node-caps
deg-malig
breast
breast-quad
irradiat
Class
```

Pada kode script di atas, di lakukan proses pemeriksaan dan pembacaan nama-nama kolom yang terdapat pada *breast-cancer* dataset untuk di lakukan penjelasan lebih lanjut mengenai informasi yang terdapat dalam kolom atau atribut dalam dataset tersebut yang akan di jelaskan lebih lanjut seperti di bawah. Terdapat nama kolom atau atribut *age*, *menopause*, *tumor-size*, *inv-nodes*, *node-caps*, *deg-malig*, *breast*, *breast-quad*, *irradiat*, dan *class*.

Selanjutnya, bisa di lanjutkan dengan menjawab pertanyaan-pertanyaan dan melakukan eksplorasi lebih lanjut pada *breast cancer dataset* sebagai berikut.

A. Deskripsikan Definisi Masing-Masing Atribut Pada Dataset Kanker Payudara Berdasarkan Pengertian Di Dalam Domain Keilmuan Medis Dan Refrensi Yang Dapat Di Pertanggungjawabkan.

Terdapat beberapa atribut pada dataset kanker payudara, dengan penjelasan lebih lanjut sebagai berikut.

- *Age* : Merupakan atribut untuk merepresentasikan informasi mengenai umur pasien ketika di diagnosis terkena kanker payudara, dalam kolom ini age (umur) di tuliskan dengan rentang nilai, seperti 40-49.
- *Menopause* : Merupakan atribut untuk merepresentasikan informasi mengenai status *menopause* pasien, dalam kolom ini *menopause* (status menopause) di tuliskan dengan premeno (sebelum menopause atau memasuki tahap menopause) dan ge40 (pasien diatas 40 tahun dan setelah menopause).
- *Tumor-Size* : Merupakan atribut untuk merepresentasikan informasi mengenai ukuran tumor payudara pasien, dalam kolom ini *tumor-size* (ukuran tumor) di tuliskan dengan rentang nilai, seperti 15-19.
- *Inv-Nodes* : Merupakan atribut untuk merepresentasikan informasi mengenai kemunculan dan keberadaan kanker payudara yang mengacu pada jumlah kelenjar getah bening yang terlibat atau terkena sel-sel kanker, dalam kolom ini *inv-nodes* (kelenjar getah bening yang terlibat) di tuliskan dengan rentang nilai, seperti 0-2.
- *Node-Caps* : Merupakan atribut untuk merepresentasikan informasi mengenai kapsul limfonodal (kapsul kelenjar getah bening) yang mengacu pada apakah kelenjar getah bening telah menembus kapsul dan sekitarnya atau belum, dalam kolom *node-caps* (kapsul kelenjar getah bening) di tuliskan dengan *yes* (kelenjar getah bening telah menembus kapsul) dan *no* (kelenjar getah bening belum menembus kapsul).
- *Deg-Malig* : Merupakan atribut untuk merepresentasikan informasi mengenai tingkat keganasan tumor payudara, dalam kolom *deg-malig* (tingkat keganasan kanker) di tuliskan dengan tingkat keganasan 1 (rendah), 2 (sedang), dan 3 (tinggi).
- *Breast* : Merupakan atribut untuk merepresentasikan informasi mengenai sisi sebelah mana yang terkena kanker payudara, dalam kolom *breast* (sisi yang terkena kanker) di tuliskan dengan *right* (kanan) dan *left* (kiri).
- *Breast-Quad* : Merupakan atribut untuk merepresentasikan informasi mengenai lokasi atau kuadran mana dari bagian payudara yang terkena kanker, dalam kolom *breast-quad* (kuadran yang terkena kanker) di tuliskan dengan *left-up* (di kuadran atas kiri), *central* (di bagian pusat atau tengah), *left_low* (di kuadran bawah), dan *right_up* (di kuadran atas kanan).
- *Irradiat* : Merupakan atribut untuk merepresentasikan informasi mengenai apakah pasien menjalani terapi radiasi atau tidak, dalam kolom *irradiat* (terapi radiasi) di tuliskan dengan *yes* (melakukan terapi radiasi) dan *no* (tidak melakukan terapi radiasi).
- *Class* : Merupakan atribut untuk merepresentasikan informasi mengenai status rekurensi (kemunculan kembali atau kembalinya kanker payudara yang telah di obati), dalam kolom *class* (status rekurensi) di tuliskan dengan *recurrence-events* (terjadinya rekurensi/muncul kembali) dan *no-recurrence-events* (tidak terjadinya rekurensi/tidak muncul kembali).

B. Lakukan Eksplorasi Pada Dataset Kanker Payudara Menggunakan Python Dan Jawablah Pertanyaan Sebagai Berikut.

1) Berapa Banyak Pasien Yang Berumur 50-59 Tahun Dengan Derajat Keganasan Kanker Payudara Sebesar 2?

```
1) Berapa Banyak Pasien Yang Berumur 50-59 Tahun Dengan Derajat Keganasan Kanker Payudara Sebesar 2?

In [3]: KategoriPasien1 = BreastCancer[(BreastCancer["age"] == "'50-59'") & (BreastCancer["deg-malig"] == "'2'")]
        JumlahPasien1 = KategoriPasien1.shape[0]

        print("Di Peroleh Jumlah Pasien Yang Berumur 50-59 Tahun Dengan Derajat Keganasan Kanker Payudara 2 Sebesar :", \
              JumlahPasien1, "Pasien")

        Di Peroleh Jumlah Pasien Yang Berumur 50-59 Tahun Dengan Derajat Keganasan Kanker Payudara 2 Sebesar : 39 Pasien
```

Pada kode script di atas, di lakukan pencarian jumlah pasien yang sesuai dengan kriteria tersebut. Terdapat variabel KategoriPasien1 yang di pergunakan untuk menuliskan kriteria kondisi pasien dengan *age* (umur) di rentang 50-59 dan *deg-malig* (derajat keganasan kanker) sebesar 2 menggunakan operator & (dan) yang di pergunakan sebagai operator bahwa pasien harus memenuhi kedua kriteria tersebut dari *dataframe* BreastCancer.

Di mana, jika kondisi pasien memenuhi kedua kriteria tersebut, akan tersimpan ke dalam variabel JumlahPasien1 yang akan menghitung banyak pasien yang memenuhi kriteria tersebut menggunakan shape. Maka, akan di tampilkan jumlah pasien yang berumur 50-59 dengan derajat keganasan kanker 2 sebesar 39 pasien.

2) Berapa Banyak Pasien Dengan Ukuran Tumor 0-4 (mm) Dengan Kondisi Kelenjar Getah Bening Belum Atau Tidak Menembus Kapsul Dan Sekitarnya (Node-Caps)?

```
2) Berapa Banyak Pasien Dengan Ukuran Tumor 0-4 (mm) Dengan Kondisi Kelenjar Getah Bening Belum Atau Tidak Menembus Kapsul Dan Sekitarnya
(Node-Caps)?

In [4]: KategoriPasien2 = BreastCancer[(BreastCancer["tumor-size"] == "'0-4'") & (BreastCancer["node-caps"] == "'no'")]
        JumlahPasien2 = len(KategoriPasien2)

        print("Di Peroleh Jumlah Pasien Dengan Ukuran Tumor 0-4 (mm) Dan Kelenjar Getah Bening Belum Menembus Sebesar :", \
              JumlahPasien2, "Pasien")

        Di Peroleh Jumlah Pasien Dengan Ukuran Tumor 0-4 (mm) Dan Kelenjar Getah Bening Belum Menembus Sebesar : 8 Pasien
```

Pada kode script di atas, di lakukan pencarian jumlah pasien yang sesuai dengan kriteria tersebut. Terdapat variabel KategoriPasien2 yang di pergunakan untuk menuliskan kriteria kondisi pasien dengan *tumor-size* (ukuran tumor) di rentang 0-4 (mm) dan *node-caps* (kelenjar getah bening) *no* (belum menembus sekitarnya) menggunakan operator & (dan) yang di pergunakan sebagai operator bahwa pasien harus memenuhi kedua kriteria tersebut dari *dataframe* BreastCancer.

Di mana jika kondisi pasien memenuhi kedua kriteria tersebut, akan tersimpan ke dalam variabel JumlahPasien2 yang akan menghitung banyak pasien yang memenuhi kriteria tersebut menggunakan len. Maka, akan di tampilkan jumlah pasien dengan ukuran tumor 0-4 (mm) dengan kelenjar getah bening belum menembus kapsul sebesar 8 pasien.

3) Berapa Banyak Pasien Dengan Tumor Yang Melakukan Terapi Radiasi Dengan Derajat Keganasan Kanker Sebesar 2 Dan 3?

```
3) Berapa Banyak Pasien Dengan Tumor Yang Melakukan Terapi Radiasi Dengan Derajat Keganasan Kanker Sebesar 2 Dan 3?

In [5]: KategoriPasien3 = BreastCancer[(BreastCancer["irradiat"] == "'yes'") & ((BreastCancer["deg-malig"] == "'2'") | \
                                                                                 (BreastCancer["deg-malig"] == "'3'"))]
        JumlahPasien3 = len(KategoriPasien3)

        print("Di Peroleh Jumlah Pasien Yang Melakukan Terapi Radiasi Dengan Derajat Keganasan Kanker Payudara 2 Dan 3 Sebesar :", \
              JumlahPasien3, "Pasien")

        Di Peroleh Jumlah Pasien Yang Melakukan Terapi Radiasi Dengan Derajat Keganasan Kanker Payudara 2 Dan 3 Sebesar : 58 Pasien
```

Pada kode script di atas, di lakukan pencarian jumlah pasien yang memenuhi kriteria tersebut. Terdapat variabel KategoriPasien3 yang di pergunakan untuk menuliskan kriteria kondisi pasien dengan *irradiat* (terapi radiasi) *yes* (melakukan terapi radiasi) dan *deg-malig* (derajat keganasan kanker) sebesar 2 dan 3 menggunakan operator | (*or*) yang di gunakan untuk menggabungkan *deg-malig* sebesar 2 dan 3. Kemudian, menggunakan operator & (dan) yang di pergunakan untuk menggabungkan kriteria-kriteria yang harus di penuhi tersebut dari *dataframe* BreastCancer.

Di mana jika kondisi pasien memenuhi 3 kriteria tersebut, akan tersimpan ke dalam variabel JumlahPasien3 yang akan menghitung banyak pasien yang memenuhi kriteria menggunakan len. Maka, akan di tampilkan jumlah pasien yang melakukan terapi radiasi dengan derajat keganasan kanker sebesar 2 dan 3 sebanyak 58 pasien.

3.1) Masing-Masing Jumlah Pasien Yang Melakukan Terapi Radiasi Dengan Derajat Keganasan Kanker 2 Dan 3

```
3.1) Masing-Masing Jumlah Pasien Yang Melakukan Terapi Radiasi Dengan Derajat Keganasan Kanker 2 Dan 3

In [6]: KategoriPasien4 = BreastCancer[(BreastCancer["irradiat"] == "'yes'") & (BreastCancer["deg-malig"] == "'2'")]
        KategoriPasien5 = BreastCancer[(BreastCancer["irradiat"] == "'yes'") & (BreastCancer["deg-malig"] == "'3'")]

        JumlahPasien4 = len(KategoriPasien4)
        JumlahPasien5 = len(KategoriPasien5)

        print("Di Peroleh Jumlah Pasien Yang Melakukan Terapi Radiasi Dengan Derajat Keganasan Kanker Payudara 2 Sebesar :", \
              JumlahPasien4, "Pasien")
        print("Di Peroleh Jumlah Pasien Yang Melakukan Terapi Radiasi Dengan Derajat Keganasan Kanker Payudara 3 Sebesar :", \
              JumlahPasien5, "Pasien")

        Di Peroleh Jumlah Pasien Yang Melakukan Terapi Radiasi Dengan Derajat Keganasan Kanker Payudara 2 Sebesar : 31 Pasien
        Di Peroleh Jumlah Pasien Yang Melakukan Terapi Radiasi Dengan Derajat Keganasan Kanker Payudara 3 Sebesar : 27 Pasien
```

Pada kode script di atas, di lakukan pencarian masing-masing jumlah pasien yang melakukan terapi radiasi dengan derajat keganasan kanker sebesar 2 dan 3. Dengan menggunakan operator yang sama yakni & (dan) dan menggunakan perhitungan menggunakan len.

Maka, akan di tampilkan bahwa masing-masing jumlah pasien yang melakukan terapi radiasi dengan derajat keganasan kanker payudara 2 sebesar 31 pasien dan jumlah pasien yang melakukan terapi radiasi dengan derajat keganasan kanker payudara 3 sebesar 27 pasien.

4) Berapa Banyak Pasien Yang Memiliki Tumor Yang Terletak Di Sebelah Kanan Dan Kiri Serta Tepatnya Pada Kuadran Pusat?

```
4) Berapa Banyak Pasien Yang Memiliki Tumor Yang Terletak Di Sebelah Kanan Dan Kiri Serta Tepatnya Pada Kuadran Pusat?

In [8]: KategoriPasien6 = BreastCancer[((BreastCancer["breast"] == "'right'") | (BreastCancer["breast"] == "'left'")) & \
                                       (BreastCancer["breast-quad"] == "'central'")]
        JumlahPasien6 = len(KategoriPasien6)

        print("Di Peroleh Jumlah Pasien Yang Memiliki Tumor Di Kanan Dan Kiri Yang Tepatnya Di Kuadran Pusat Sebesar :", \
              JumlahPasien6, "Pasien")

        Di Peroleh Jumlah Pasien Yang Memiliki Tumor Di Kanan Dan Kiri Yang Tepatnya Di Kuadran Pusat Sebesar : 21 Pasien
```

Pada kode script di atas, di lakukan pencarian jumlah pasien yang memenuhi kriteria tersebut. Terdapat variabel KategoriPasien6 yang di pergunakan untuk menuliskan kriteria kondisi pasien dengan *breast* (letak tumor) di *right* (kanan) dan *left* (kiri) menggunakan operator | (*or*) yang di

gunakan untuk menggabungkan breast di kanan dan kiri dan *breast-quad* (kuadran kanker) di *central* (pusat). Kemudian, menggunakan operator & (dan) untuk menggabungkan kriteria-kriteria yang harus di penuhi tersebut dari *dataframe* BreastCancer.

Di mana jika kondisi pasien memenuhi 3 kriteria tersebut, akan tersimpan ke dalam variabel JumlahPasien6 yang akan menghitung banyak pasien yang memenuhi kriteria menggunakan len. Maka, akan di tampilkan jumlah pasien yang memiliki tumor di bagian kanan dan kiri serta tepat di kuadran pusat sebesar 21 pasien.

4.1) Masing-Masing Jumlah Pasien Yang Memiliki Tumor Terletak Di Kanan, Kiri, Serta Di Kuadran Pusat

```
4.1) Masing-Masing Jumlah Pasien Yang Memiliki Tumor Terletak Di Kanan, Kiri, Serta Di Kuadran Pusat

In [9]: KategoriPasien7 = BreastCancer[(BreastCancer["breast"] == "'right'") & (BreastCancer["breast-quad"] == "'central'")]
        KategoriPasien8 = BreastCancer[(BreastCancer["breast"] == "'left'") & (BreastCancer["breast-quad"] == "'central'")]

        JumlahPasien7 = len(KategoriPasien7)
        JumlahPasien8 = len(KategoriPasien8)

        print("Di Peroleh Jumlah Pasien Yang Memiliki Tumor Di Kanan Serta Tepat Di Kuadran Pusat Sebesar :", \
              JumlahPasien7, "Pasien")
        print("Di Peroleh Jumlah Pasien Yang Memiliki Tumor Di Kiri Serta Tepat Di Kuadran Pusat Sebesar :", \
              JumlahPasien8, "Pasien")

        Di Peroleh Jumlah Pasien Yang Memiliki Tumor Di Kanan Serta Tepat Di Kuadran Pusat Sebesar : 10 Pasien
        Di Peroleh Jumlah Pasien Yang Memiliki Tumor Di Kiri Serta Tepat Di Kuadran Pusat Sebesar : 11 Pasien
```

Pada kode script di atas, di lakukan pencarian masing-masing jumlah pasien yang memiliki tumor di sebelah kanan tepat di kuadran pusat dan di sebelah kiri tepat di kuadran pusat. Dengan menggunakan operator yang sama, yakni & (dan) dan menggunakan perhitungan menggunakan len.

Maka, akan di tampilkan bahwa jumlah pasien yang memiliki tumor di sebelah kanan dan tepat di kuadran pusat sebesar 10 pasien dan jumlah pasien yang memiliki tumor di kiri dan tepat di kuadran pusat sebesar 11.

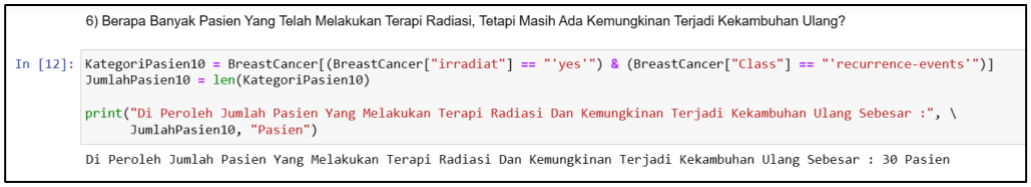
5) Berapa Banyak Pasien Yang Sedang Premenopause Dengan Kelenjar Getah Bening Yang Mengandung Kanker Payudara Pada Range 6-8?

```
5) Berapa Banyak Pasien Yang Sedang Premenopause Dengan Kelenjar Getah Bening Yang Mengandung Kanker Payudara Pada Range 6-8?

In [11]: KategoriPasien9 = BreastCancer[(BreastCancer["menopause"] == "'premeno'") & (BreastCancer["inv-nodes"] == "'6-8'")]
         JumlahPasien9 = len(KategoriPasien9)

         print("Di Peroleh Jumlah Pasien Yang Premenopause Dengan Kelejar Getah Bening Di Range 6-8 Sebesar :", \
               JumlahPasien9, "Pasien")

         Di Peroleh Jumlah Pasien Yang Premenopause Dengan Kelejar Getah Bening Di Range 6-8 Sebesar : 7 Pasien
```

Pada kode script di atas, di lakukan pencarian jumlah pasien yang sesuai dengan kriteria tersebut. Terdapat variabel KategoriPasien9 yang di pergunakan untuk menuliskan kriteria kondisi pasien dengan *menopause premeno* dan *inv-nodes* (kelenjar getah bening) di range 6-8 menggunakan operator & (dan) yang di pergunakan sebagai operator bahwa pasien harus memenuhi kedua kriteria tersebut dari *dataframe* BreastCancer.

Di mana jika kondisi pasien memenuhi kedua kriteria tersebut, akan tersimpan ke dalam variabel JumlahPasien9 yang akan menghitung banyak pasien yang memenuhi kriteria tersebut menggunakan len. Maka, akan di tampilkan jumlah pasien yang *premenopause* dengan kelenjar getah bening yang mengandung kanker payudara di range 6-8 sebesar 7 pasien.

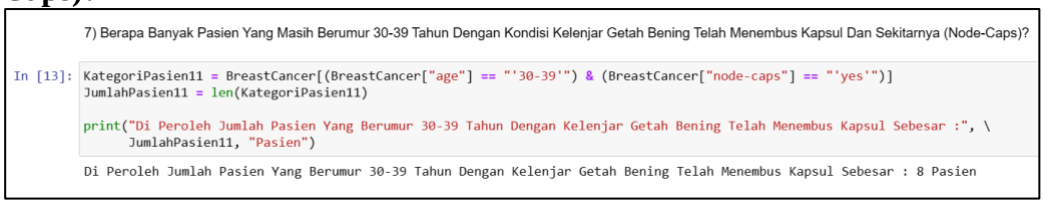
6) Berapa Banyak Pasien Yang Telah Melakukan Terapi Radiasi, Tetapi Masih Ada Kemungkinan Terjadi Kekambuhan Ulang?



Pada kode script di atas, di lakukan pencarian jumlah pasien yang sesuai dengan kriteria tersebut. Terdapat variabel KategoriPasien10 yang di pergunakan untuk menuliskan kriteria kondisi pasien dengan *irradiat* (terapi radiasi) *yes* (melakukan terapi radiasi) dan *class* (terjadinya rekurensi/kekambuhan ulang) *recurrence-events* (terjadi kekambuhan ulang) menggunakan operator & (dan) yang di pergunakan sebagai operator bahwa pasien harus memenuhi kedua kriteria tersebut dari *dataframe* BreastCancer.

Di mana jika kondisi pasien memenuhi kedua kriteria tersebut, akan tersimpan ke dalam variabel JumlahPasien10 yang akan menghitung banyak pasien yang memenuhi kriteria tersebut menggunakan len. Maka, akan di tampilkan jumlah pasien yang melakukan terapi radiasi dan kemungkinan terjadi kekambuhan ulang sebesar 30 pasien.

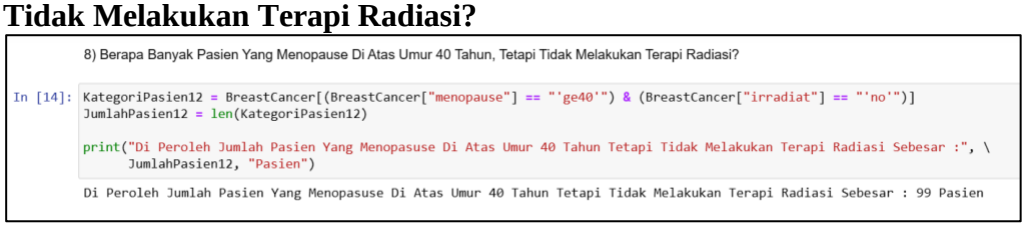
7) Berapa Banyak Pasien Yang Masih Berumur 30-39 Tahun Dengan Kondisi Kelenjar Getah Bening Telah Menembus Kapsul Dan Sekitarnya (Node-Caps)?



Pada kode script di atas, di lakukan pencarian jumlah pasien yang sesuai dengan kriteria tersebut. Terdapat variabel KategoriPasien11 yang di pergunakan untuk menuliskan kriteria kondisi pasien dengan *age* (umur) di rentang 30-39 dan *node-caps* (kelenjar getah bening) *yes* (telah menembus kapsul) menggunakan operator & (dan) yang di pergunakan sebagai operator bahwa pasien harus memenuhi kedua kriteria tersebut dari *dataframe* BreastCancer.

Di mana jika kondisi pasien memenuhi kedua kriteria tersebut, akan tersimpan ke dalam variabel JumlahPasien11 yang akan menghitung banyak pasien yang memenuhi kriteria tersebut menggunakan len. Maka, akan di tampilkan jumlah pasien yang berumur 30-39 tahun dengan kelenjar getah bening yang telah menembus kapsul sebesar 8 pasien.

8) Berapa Banyak Pasien Yang Menopause Di Atas Umur 40 Tahun, Tetapi Tidak Melakukan Terapi Radiasi?

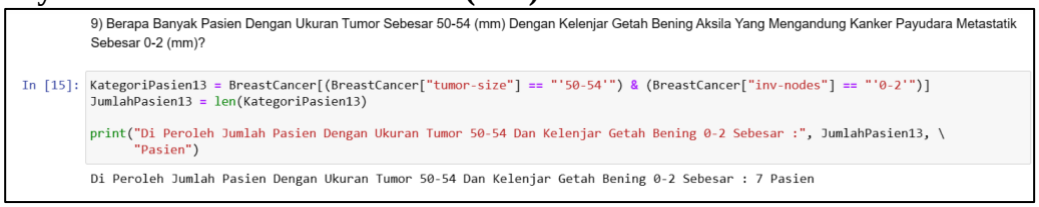


Pada kode script di atas, di lakukan pencarian jumlah pasien yang sesuai dengan kriteria tersebut. Terdapat variabel KategoriPasien12 yang di pergunakan untuk menuliskan kriteria kondisi pasien dengan *menopause ge40* dan *irradiat* (terapi radiasi) *no* (tidak melakukan terapi radiasi) menggunakan

operator & (dan) yang di pergunakan sebagai operator bahwa pasien harus memenuhi kedua kriteria tersebut dari *dataframe* BreastCancer.

Di mana jika kondisi pasien memenuhi kedua kriteria tersebut, maka akan tersimpan ke dalam variabel JumlahPasien12 yang akan menghitung banyak pasien yang memenuhi kriteria tersebut menggunakan len. Maka, akan di tampilkan jumlah pasien yang menopause di atas 40 tahun tetapi tidak melakukan terapi radiasi sebanyak 99 pasien.

9) Berapa Banyak Pasien Dengan Ukuran Tumor Sebesar 50-54 (mm) Dengan Kelenjar Getah Bening Aksila Yang Mengandung Kanker Payudara Metastatik Sebesar 0-2 (mm)?



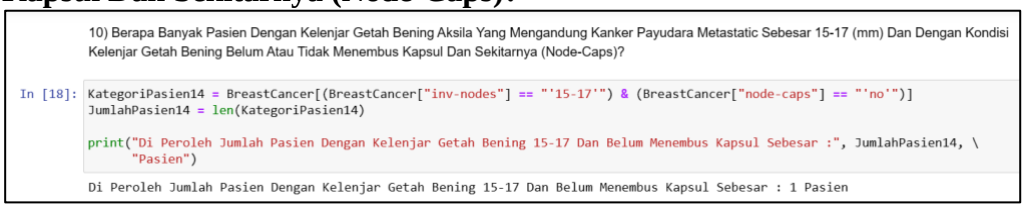
9) Berapa Banyak Pasien Dengan Ukuran Tumor Sebesar 50-54 (mm) Dengan Kelenjar Getah Bening Aksila Yang Mengandung Kanker Payudara Metastatik Sebesar 0-2 (mm)?

```
In [15]: KategoriPasien13 = BreastCancer[(BreastCancer["tumor-size"] == "'50-54'") & (BreastCancer["inv-nodes"] == "'0-2'")]
         JumlahPasien13 = len(KategoriPasien13)

         print("Di Peroleh Jumlah Pasien Dengan Ukuran Tumor 50-54 Dan Kelenjar Getah Bening 0-2 Sebesar :", JumlahPasien13, \
               "Pasien")

         Di Peroleh Jumlah Pasien Dengan Ukuran Tumor 50-54 Dan Kelenjar Getah Bening 0-2 Sebesar : 7 Pasien
```

Pada kode script di atas, di lakukan pencarian jumlah pasien yang sesuai dengan kriteria tersebut. Terdapat variabel KategoriPasien13 yang di pergunakan untuk menuliskan kriteria kondisi pasien dengan *tumor-size* (ukuran tumor) di range 50-54 dan *inv-nodes* (kelenjar getah bening) di range 0-2 menggunakan operator & (dan) yang di pergunakan sebagai operator bahwa pasien harus memenuhi kedua kedua kriteria tersebut dari *dataframe* BreastCancer.

Di mana jika kondisi pasien memenuhi kedua kriteria tersebut, maka akan tersimpan ke dalam variabel JumlahPasien13 yang akan menghitung banyak pasien yang memenuhi kedua kriteria tersebut menggunakan len. Maka, akan di tampilkan jumlah pasien dengan ukuran tumor 50-54 dan kelenjar getah bening aksila yang mengandung kanker payudara di range 0-2 sebesar 7 pasien.

10) Berapa Banyak Pasien Dengan Kelenjar Getah Bening Aksila Yang Mengandung Kanker Payudara Metastatic Sebesar 15-17 (mm) Dan Dengan Kondisi Kelenjar Getah Bening Belum Atau Tidak Menembus Kapsul Dan Sekitarnya (Node-Caps)?



10) Berapa Banyak Pasien Dengan Kelenjar Getah Bening Aksila Yang Mengandung Kanker Payudara Metastatic Sebesar 15-17 (mm) Dan Dengan Kondisi Kelenjar Getah Bening Belum Atau Tidak Menembus Kapsul Dan Sekitarnya (Node-Caps)?

```
In [18]: KategoriPasien14 = BreastCancer[(BreastCancer["inv-nodes"] == "'15-17'") & (BreastCancer["node-caps"] == "'no'")]
         JumlahPasien14 = len(KategoriPasien14)

         print("Di Peroleh Jumlah Pasien Dengan Kelenjar Getah Bening 15-17 Dan Belum Menembus Kapsul Sebesar :", JumlahPasien14, \
               "Pasien")

         Di Peroleh Jumlah Pasien Dengan Kelenjar Getah Bening 15-17 Dan Belum Menembus Kapsul Sebesar : 1 Pasien
```

Pada kode script di atas, di lakukan pencarian jumlah pasien yang sesuai dengan kriteria tersebut. Terdapat variabel KategoriPasien14 yang di pergunakan untuk menuliskan kriteria pasien dengan *inv-nodes* (kelenjar getah bening) di range 15-17 dan *node-caps* (kondisi kelenjar getah bening) *no* (belum menembus kapsul) menggunakan operator & (dan) yang di pergunakan sebagai operator bahwa pasien harus memenuhi kedua kriteria tersebut dari *dataframe BreastCancer*.

Di mana jika kondisi pasien memenuhi kedua kriteria tersebut, maka akan tersimpan ke dalam variabel JumlahPasien14 yang akan menghitung banyaknya pasien menggunakan len. Maka, akan di tampilkan jumlah pasien dengan kelenjar getah bening yang mengandung kanker payudara sebesar 15-

17 dengan kondisi kelenjar getah bening belum menembus kapsul sebesar 1 pasien.

2. PENGGABUNGAN DATA MENGGUNAKAN INNER JOIN, LEFT JOIN, RIGHT JOIN, DAN OUTER JOIN (STUDI KASUS : BREAST CANCER DATASET)

Sebelum melakukan penggabungan data dalam *breast cancer dataset* atau dataset kanker payudara, harus di lakukan langkah-langkah awal untuk *meload* (membaca) dataset sebagai berikut.

- **Load Dataset *Tumor-Size* Ganjil**

• Load Dataset Tumor-Size Ganjil

```
In [1]: import pandas as pd

        TumorSizeOdd = pd.read_csv("df1tumor_size_odd.csv", sep = ";")
        TumorSizeOdd
```

Out[1]:

	PatientID	tumor-size
0	1	'15-19'
1	3	'35-39'
2	5	'30-34'
3	7	'40-44'
4	9	'0-4'

Pada kode di atas, di lakukan *load* dan pembacaan *df1tumor_size_odd* dataset menggunakan *library* pandas dengan jenis file csv. Kemudian, *df1tumor_size_odd* dataset tersebut di simpan ke dalam variabel TumorSizeOdd dengan 120 baris dan 2 kolom yang berisi *tumor-size* (ukuran tumor) pasien dengan *PatienID* ganjil. Di mana nama variabel ini akan di pergunakan untuk proses lebih lanjut dalam melakukan penggabungan data berdasarkan *PatientID*.

- **Load Dataset *Tumor-Size* Genap**

• Load Dataset Tumor-Size Genap

```
In [2]: TumorSizeEven = pd.read_csv("df2tumor_size_even.csv", sep = ";")
        TumorSizeEven
```

Out[2]:

	PatientID	tumor-size
0	4	'35-39'
1	6	'25-29'
2	8	'10-14'
3	12	'15-19'
4	14	'25-29'

Pada kode di atas, di lakukan *load* dan pembacaan *df2tumor_size_even* dataset menggunakan *library* pandas dengan jenis file csv. Kemudian, *df2tumor_size_even* dataset tersebut di simpan ke dalam variabel TumorSizeEven dengan 118 baris dan 2 kolom yang berisi *tumor-size* (ukuran tumor) pasien dengan *PatienID* genap. Di mana nama variabel ini akan di pergunakan untuk proses lebih lanjut dalam melakukan penggabungan data berdasarkan *PatientID*.

- **Load Dataset *Breastquad* Ganjil**

- Load Dataset Breastquad Ganjil

```
In [3]: BreastquadOdd = pd.read_csv("df1breastquad_odd.csv", sep = ";")
        BreastquadOdd
```

Out[3]:

	PatientID	breast-quad
0	1	'left_up'
1	3	'left_low'
2	7	'left_up'
3	9	'right_low'
4	11	'left_low'

Pada kode di atas, di lakukan *load* dan pembacaan *df1breastquad_odd* dataset menggunakan *library* pandas dengan jenis file csv. Kemudian, *df1breastquad_odd* dataset tersebut di simpan ke dalam variabel BreastquadOdd dengan 119 baris dan 2 kolom yang berisi *breastquad* (letak kuadran tumor) pasien dengan *PatienID* ganjil. Di mana nama variabel ini akan di pergunakan untuk proses lebih lanjut dalam melakukan penggabungan data berdasarkan *PatientID*.

- **Load Dataset *Breastquad* Genap**

- Load Dataset Breastquad Genap

```
In [4]: BreastquadEven = pd.read_csv("df2breastquad_even.csv", sep = ";")
        BreastquadEven
```

Out[4]:

	PatientID	breast-quad
0	2	'central'
1	6	'left_up'
2	8	'left_up'
3	10	'left_up'
4	12	'left_up'

Pada kode di atas, di lakukan *load* dan pembacaan *df2breastquad_even* dataset menggunakan *library* pandas dengan jenis file csv. Kemudian, *df2breastquad_even* dataset tersebut di simpan ke dalam variabel BreastquadEven dengan 118 baris dan 2 kolom yang berisi *breastquad* (letak kuadran tumor) pasien dengan *PatienID* genap. Di mana nama variabel ini akan di pergunakan untuk proses lebih lanjut dalam melakukan penggabungan data berdasarkan *PatientID*.

Selanjutnya, bisa di lanjutkan dengan menjawab pertanyaan-pertanyaan dan melakukan penggabungan data lebih lanjut pada *breast cancer dataset* sebagai berikut.

A. Implementasikan Penggabungan Data (Merge) Menggunakan Inner Join, Left Join, Right Join, Dan Outer Join Menggunakan Python Untuk Menjawab Pertanyaan Sebagai Berikut.

1) Berapa Banyak Record Pasien Yang Memiliki Nilai Atribut Ukuran Tumor Dan Nilai Kuadran Kanker Payudara Secara Lengkap Keduanya?

1) Berapa Banyak Record Pasien Yang Memiliki Nilai Atribut Ukuran Tumor Dan Nilai Kuadran Kanker Payudara Secara Lengkap Keduanya?

```
In [5]: dfSize = pd.concat([TumorSizeOdd, TumorSizeEven], ignore_index = True)
        dfBreastquad = pd.concat([BreastquadOdd, BreastquadEven], ignore_index = True)

        dfJoin1 = dfSize.merge(dfBreastquad, how = "inner", on = "PatientID")
        dfJoin1
```

Out[5]:

	PatientID	tumor-size	breast-quad
0	1	'15-19'	'left_up'
1	3	'35-39'	'left_low'
2	7	'40-44'	'left_up'
3	9	'0-4'	'right_low'
4	11	'25-29'	'left_low'

```
In [6]: JumlahPasien15 = len(dfJoin1)

        print("Di Peroleh Jumlah Record Pasien Yang Memiliki Nilai Atribut Ukuran Tumor Dan Nilai Kuadran Kanker Payudara Sebesar :", \
              JumlahPasien15, "Pasien")

        Di Peroleh Jumlah Record Pasien Yang Memiliki Nilai Atribut Ukuran Tumor Dan Nilai Kuadran Kanker Payudara Sebesar : 203 Pasien
```

Pada kode script di atas, di lakukan proses penggabungan data dari masing-masing dataset yang telah di load sebelumnya menggunakan *inner join*. Terdapat variabel dfSize sebagai *dataframe* baru yang berisi gabungan dari *dataframe* TumorSizeOdd dan TumorSizeEven, yakni data ukuran tumor pasien berdasarkan *PatienID* ganjil dan genap, serta di lakukan pengaturan *index* baris menggunakan parameter *ignore_index*, sehingga menghasilkan gabungan *dataframe* yang berurutan. Kemudian, terdapat variabel dfBreastquad sebagai *dataframe* baru yang berisi gabungan dari *dataframe* BreasquatOdd dan BreatquadEven, yakni data kuadran pusat tumor pasien berdasarkan *PatienID* ganjil dan genap.

Selanjutnya, terdapat variabel dfJoin1 yang di pergunakan untuk melakukan penggabungan dua *dataframe*, yakni *dataframe* dfSize dan dfBreastquad menggunakan metode *merge*. Penggabungan ini di lakukan berdasarkan kolom *PatienID* dengan jenis penggabungan *inner join*, yakni hanya akan menampilkan hasil dengan nilai yang sesuai (cocok) dalam kedua *dataframe* pada kolom *PatienID*, yakni menghitung jumlah pasien yang memilik nilai ukuran tumor dan kuadran payudara berdasarkan kolom *PatienID*.

Di lakukan perhitungan jumlah pasien dari *dataframe* hasil penggabungan tersebut menggunakan len. Maka, akan di tampilkan jumlah record pasien yang memiliki nilai atribut ukuran tumor dan nilai kuadran payudara sebesar 203 pasien.

2) Berapa Banyak Record Pasien Yang Memiliki Nilai Atribut Ukuran Tumor, Tetapi Tidak Memiliki Nilai Kuadran Kanker Payudara ? Serta Berapa Banyak Record Nilai Atribut Kuadran Kanker Payudara Yang NaN Tersebut ?

2) Berapa Banyak Record Pasien Yang Memiliki Nilai Atribut Ukuran Tumor, Tetapi Tidak Memiliki Nilai Kuadran Kanker Payudara ? Serta Berapa Banyak Record Nilai Atribut Kuadran Kanker Payudara Yang NaN Tersebut ?

```
In [7]: dfSize = pd.concat([TumorSizeOdd, TumorSizeEven], ignore_index = True)
        dfBreastquad = pd.concat([BreastquadOdd, BreastquadEven], ignore_index = True)

        dfJoin2 = dfSize.merge(dfBreastquad, how = "left", on = "PatientID")
        dfJoin2
```

Out[7]:

	PatientID	tumor-size	breast-quad
0	1	'15-19'	'left_up'
1	3	'35-39'	'left_low'
2	5	'30-34'	NaN
3	7	'40-44'	'left_up'
4	9	'0-4'	'right_low'

```
In [13]: JumlahPasien16No = len(dfJoin2)
         JumlahPasien17Nan = dfJoin2["breast-quad"].isna().sum()

         print("Di Peroleh Jumlah Record Pasien Yang Memiliki Nilai Atribut Ukuran Tumor Tetapi Tidak Memiliki Nilai Kuadran Payudara \
         Sebesar :", JumlahPasien16No, "Pasien")
         print("Di Peroleh Jumlah Record Pasien Yang Memiliki Nilai Atribut Kuadran Kanker Payudara NaN Sebesar :", JumlahPasien17Nan, \
               "Pasien")

         Di Peroleh Jumlah Record Pasien Yang Memiliki Nilai Atribut Ukuran Tumor Tetapi Tidak Memiliki Nilai Kuadran Payudara Sebesar :
         238 Pasien
         Di Peroleh Jumlah Record Pasien Yang Memiliki Nilai Atribut Kuadran Kanker Payudara NaN Sebesar : 35 Pasien
```

Pada kode script di atas, di lakukan proses penggabungan data dari masing-masing dataset yang telah di *load* sebelumnya menggunakan *left join*. Terdapat variabel dfSize sebagai *dataframe* baru yang berisi gabungan dari *dataframe* TumorSizeOdd dan TumorSizeEven, yakni data ukuran tumor pasien berdasarkan *PatienID* ganjil dan genap, serta di lakukan pengaturan *index* baris menggunakan parameter *ignore_index*, sehingga menghasilkan gabungan

dataframe yang berurutan. Kemudian, terdapat variabel dfBreastquad sebagai *dataframe* baru yang berisi gabungan dari *dataframe* BreasquatOdd dan BreatquadEven, yakni data kuadran pusat tumor berdasarkan *PatienID* ganjil dan genap.

Selanjutnya, terdapat variabel dfJoin2 yang di pergunakan untuk melakukan penggabungan dua *dataframe*, yakni *dataframe* dfSize dan dfBreastquad menggunakan metode *merge*. Penggabungan ini di lakukan berdasarkan kolom *PatienID* dengan jenis penggabungan *left join*, yakni semua baris dari dfSize akan tetap masuk ke dalam hasil penggabungan dan hanya baris-baris dari dfBreastquad yang memiliki nilai cocok dengan *PatienID* dari dfSize yang akan di gabungkan, yakni menghitung pasien yang memiliki atribut ukuran tumor tetapi tidak memiliki atribut kuadran payudara, serta pasien yang memiliki atribut kuadran payudara dengan nilai nan.

Terdapat variabel JumlahPasien16No yang di pergunakan untuk melakukan perhitungan dari jumlah pasien yang tidak memiliki nilai kuadran payudara menggunakan len dan variabel JumlahPaien17Nan yang di pergunakan untuk melakukan perhitungan dari jumlah pasien yang memiliki kuadran kanker payudara dengan nilai *nan* menggunakan *isna* (pengecekan nilai nan/hilang). Maka, akan di tampilkan jumlah record pasien yang memiliki atribut ukuran tumor tetapi tidak memiliki kuadran payudara sebesar 238 pasien dan jumlah record pasien yang memiliki atribut kuadran payudara dengan nilai *nan* sebesar 35 pasien.

3) Berapa Banyak Record Pasien Yang Tidak Memiliki Nilai Atribut Ukuran Tumor, Tetapi Memiliki Nilai Kuadran Kanker Payudara? Serta Berapa Banyak Record Nilai Atribut Ukuran Tumor Yang NaN Tersebut?

3) Berapa Banyak Record Pasien Yang Tidak Memiliki Nilai Atribut Ukuran Tumor, Tetapi Memiliki Nilai Kuadran Kanker Payudara? Serta Berapa Banyak Record Nilai Atribut Ukuran Tumor Yang NaN Tersebut?

In [16]:
```
dfSize = pd.concat([TumorSizeOdd, TumorSizeEven], ignore_index = True)
dfBreastquad = pd.concat([BreastquadOdd, BreastquadEven], ignore_index = True)

dfJoin3 = dfSize.merge(dfBreastquad, how = "right", on = "PatientID")
dfJoin3
```

Out[16]:

	PatientID	tumor-size	breast-quad
0	1	'15-19'	'left_up'
1	3	'35-39'	'left_low'
2	7	'40-44'	'left_up'
3	9	'0-4'	'right_low'
4	11	'25-29'	'left_low'

In [18]:
```
JumlahPasien18No = len(dfJoin3)
JumlahPasien19Nan = dfJoin3["tumor-size"].isna().sum()

print("Di Peroleh Jumlah Record Pasien Yang Tidak Memiliki Nilai Atribut Ukuran Tumor Tetapi Memiliki Nilai Kuadran Kanker \
Payudara Sebesar : ", JumlahPasien18No, "Pasien")
print("Di Peroleh Jumlah Record Pasien Yang Memiliki Nilai Atribut Ukuran Tumor Nan Sebesar :", JumlahPasien19Nan, "Pasien")
```

```
Di Peroleh Jumlah Record Pasien Yang Tidak Memiliki Nilai Atribut Ukuran Tumor Tetapi Memiliki Nilai Kuadran Kanker Payudara
Sebesar :  237 Pasien
Di Peroleh Jumlah Record Pasien Yang Memiliki Nilai Atribut Ukuran Tumor Nan Sebesar : 34 Pasien
```

Pada kode script di atas, di lakukan proses penggabungan data dari masing-masing dataset yang telah di load sebelumnya menggunakan *right join*. Terdapat variabel dfSize sebagai *dataframe* baru yang berisi gabungan dari *dataframe* TumorSizeOdd dan TumorSizeEven, yakni data ukuran tumor pasien berdasarkan *PatienID* ganjil dan genap, serta di lakukan pengaturan *index* baris menggunakan parameter *ignore_index*, sehingga menghasilkan gabungan *dataframe* yang berurutan. Kemudian, terdapat variabel dfBreastquad sebagai *dataframe* baru yang berisi gabungan dari *dataframe* BreasquatOdd dan BreatquadEven, yakni data kuadran pusat tumor berdasarkan *PatienID* ganjil dan genap.

Selanjutnya, terdapat variabel dfJoin3 yang di pergunakan untuk melakukan penggabungan dua *dataframe*, yakni *dataframe* dfSize dan dfBreastquad menggunakan metode *merge*. Penggabungan data ini di lakukan berdasarkan kolom *PatienID* dengan jenis penggabungan *right join*, yakni hanya akan menampilkan semua baris dari dfBreastquad dan hanya baris-baris dari dfSize yang memiliki nilai yang cocok berdasarkan kolom *PatienID* dari dfBreastquad yang akan di gabungkan, yakni menghitung nilai pasien yang tidak memiliki atribut ukuran tumor tetapi memiliki atribut kuadran payudara, serta pasien yang memiliki atribut ukuran payudara dengan nilai *nan*.

Terdapat variabel JumlahPasien19No yang di pergunakan untuk melakukan perhitungan dari jumlah pasien yang tidak memiliki atribut ukuran tumor tetapi memiliki atribut ukuran payudara menggunakan len dan variabel JumlahPasien19Nan yang di pergunakan untuk melakukan perhitungan dari jumlah pasien yang memiliki nilai atribut ukuran tumor dengan nilai *nan* menggunakan *isna* (pengecekan nilai nan/hilang). Maka, akan di tampilkan jumlah record pasien yang tidak memiliki nilai atribut ukuran tumor tetapi memiliki nilai kuadran kanker payudara sebesar 237 pasien dan jumlah record pasien yang memiliki ukuran tumor dengan nilai nan sebesar 34 pasien.

4) Jika Menggunakan Outer Join, Berapa Banyak Record Pasien Yang NaN Pada Masing-Masing Atribut, Baik Pada Atribut Ukuran Tumor Maupun Kuadran Kanker Payudara?

4) Jika Menggunakan Outer Join, Berapa Banyak Record Pasien Yang NaN Pada Masing-Masing Atribut, Baik Pada Atribut Ukuran Tumor Maupun Kuadran Kanker Payudara?

```
In [22]: dfSize = pd.concat([TumorSizeOdd, TumorSizeEven], ignore_index = True)
         dfBreastquad = pd.concat([BreastquadOdd, BreastquadEven], ignore_index = True)

         dfJoin4 = dfSize.merge(dfBreastquad, how = "outer", on = "PatientID")
         dfJoin4
```

Out[22]:

	PatientID	tumor-size	breast-quad
0	1	'15-19'	'left_up'
1	3	'35-39'	'left_low'
2	5	'30-34'	NaN
3	7	'40-44'	'left_up'
4	9	'0-4'	'right_low'

```
In [25]: JumlahPasien19 = dfJoin4.isna().sum()

         print("Di Peroleh Jumlah Record Pasien Yang Memiliki Nan Pada Masing-Masing Atribut : ")
         print(JumlahPasien19)
```

```
Di Peroleh Jumlah Record Pasien Yang Memiliki Nan Pada Masing-Masing Atribut :
PatientID      0
tumor-size    34
breast-quad   35
dtype: int64
```

Pada kode script di atas, di lakukan proses penggabungan data dari masing-masing dataset yang telah di load sebelumnya menggunakan *outer join*. Terdapat variabel dfSize sebagai *dataframe* baru yang berisi gabungan dari *dataframe* TumorSizeOdd dan TumorSizeEven, yakni data ukuran tumor pasien berdasarkan *PatienID* ganjil dan genap, serta di lakukan pengaturan *index* baris menggunakan parameter *ignore_index*, sehingga menghasilkan gabungan *dataframe* yang berurutan. Kemudian, terdapat variabel dfBreastquad sebagai *dataframe* baru yang berisi gabungan dari *dataframe* BreasquatOdd dan BreatquadEven, yakni data kuadran pusat tumor dengan *PatienID* ganjil dan genap.

Selanjutnya, terdapat variabel dfJoin4 yang di pergunakan untuk melakukan penggabungan dua *dataframe*, yakni *dataframe* dfSize dan dfBreastquad menggunakan metode *merge*. Penggabungan ini di lakukan

berdasarkan kolom PatienID dengan jenis penggabungan *outer join*, yakni penggabungan semua baris yang ada di salah satu atau kedua *dataframe* yang ada, dan menghitung semua nilai *nan* yang ada di masing-masing atribut atau kolom.

Terdapat variabel JumlahPasien19 yang di pergunakan untuk melakukan perhitungan nilai *nan* pada masing-masing atribut atau kolom menggunakan *isna* (pengecekan nilai nan/hilang) dan sum (melakukan perhitungan pada total nilai *nan*). Maka, akan di tampilkan jumlah record pasien yang memiliki nilai *nan* pada masing-masing atribut, meliputi *PatienID* memiliki 0 nilai *nan*, *tumor-size* memiliki 34 nilai *nan*, dan *breast-quad* memiliki 35 nilai *nan*.

3. TRANSFORMASI DATA : DUPLIKASI DATA, *MISSING VALUES*, DAN IMPUTASI *MISSING VALUES*

Sebelum melakukan transformasi data dalam hepatitis dataset, harus di lakukan langkah-langkah awal untuk *meload* (membaca) dataset sebagai berikut.

- **Load Hepatitis Dataset Awal**

• Load Hepatitis Dataset Awal

In [1]:
```
import pandas as pd

Hepatitis = pd.read_excel("Hepatitis.xlsx")
Hepatitis
```

```
C:\Users\rreza\anaconda3\lib\site-packages\openpyxl\worksheet\_reader.py:312: UserWarning: Unknown extension is not supported a
nd will be removed
  warn(msg)
```

Out[1]:

	#	1	2	3	4	5	6	7	8	9	...	11	12	13	14	15	16	17	18	
0	NaN	Age	Sex	Steroid	Antivirals	Fatique	Malaise	Anorexia	Liver Big	Liver Firm	...	Speiders	Ascites	Varices	Bilirubin	Alk Phosphate	SGOT	Albumin	Protime	Histol
1	1.0	30	2	1	2	2	2	2	1	2	...	2	2	2	1	85	18	4	?	
2	2.0	50	1	1	2	1	2	2	1	2	...	2	2	2	0.9	135	42	3.5	?	
3	3.0	78	1	2	2	1	2	2	2	2	...	2	2	2	0.7	96	32	4	?	
4	4.0	31	1	?	1	2	2	2	2	2	...	2	2	2	0.7	46	52	4	80	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
151	151.0	46	1	2	2	1	1	1	2	2	...	1	1	1	7.6	?	242	3.3	50	
152	152.0	44	1	2	2	1	2	2	2	1	...	2	2	2	0.9	126	142	4.3	?	
153	153.0	61	1	1	2	1	1	2	1	1	...	1	2	2	0.8	75	20	4.1	?	
154	154.0	53	2	1	2	1	2	2	2	2	...	1	2	1	1.5	81	19	4.1	48	
155	155.0	43	1	2	2	1	2	2	2	2	...	1	1	2	1.2	100	19	3.1	42	

156 rows × 21 columns

Pada kode di atas, di lakukan *load* dan pembacaan hepatitis dataset menggunakan *library* pandas dengan jenis file xlsx (excel). Kemudian, hepatitis dataset tersebut di simpan ke dalam variabel Hepatitis dengan 156 baris dan 21 kolom.

- **Melakukan Perapian Nama Kolom Untuk Mempermudah Dalam Melakukan Transformasi Data**

• Melakukan Perapian Nama Kolom Untuk Mempermudah Dalam Melakukan Transformasi Data

In [2]:
```
HepatitisNew = pd.read_excel("Hepatitis.xlsx", skiprows = 1)
HepatitisNew
```

```
C:\Users\rreza\anaconda3\lib\site-packages\openpyxl\worksheet\_reader.py:312: UserWarning: Unknown extension is not supported a
nd will be removed
  warn(msg)
```

Out[2]:

	Unnamed: 0	Age	Sex	Steroid	Antivirals	Fatique	Malaise	Anorexia	Liver Big	Liver Firm	...	Speiders	Ascites	Varices	Bilirubin	Alk Phosphate	SGOT	Albumin	
0	1	30	2	1	2	2	2	2	1	2	...	2	2	2	1	85	18	4	
1	2	50	1	1	2	1	2	2	1	2	...	2	2	2	0.9	135	42	3.5	
2	3	78	1	2	2	1	2	2	2	2	...	2	2	2	0.7	96	32	4	
3	4	31	1	?	1	2	2	2	2	2	...	2	2	2	0.7	46	52	4	
4	5	34	1	2	2	2	2	2	2	2	...	2	2	2	1	?	200	4	

Pada kode di atas, di lakukan *load* dan pembacaan hepatitis dataset menggunakan library pandas dengan jenis file xlsx (excel). Dan di lakukan proses

skiprows yang di pergunkaan untuk mengabaikan satu baris pertama (header) dari file excel tersebut, dan menjadikan baris kedua sebagai *header*. Kemudian, hepatitis dataset tersebut di simpan ke dalam variabel HepatitisNew dengan 155 baris dan 21 kolom.

- **Melakukan Penghapusan Kolom Unnamed: 0 Karena Tidak Di Pergunakan**

• Melakukan Penghapusan Kolom Unnamed:0 Karena Tidak Di Pergunakan

```
In [3]: HepatitisNew.drop("Unnamed: 0", axis = 1, inplace = True)
        HepatitisNew
```

Out[3]:

	Age	Sex	Steroid	Antivirals	Fatique	Malaise	Anorexia	Liver Big	Liver Firm	Spleen Palpable	Speiders	Ascites	Varices	Bilirubin	Alk Phosphate	SGOT	Albumin
0	30	2	1	2	2	2	2	1	2	2	2	2	2	1	85	18	4
1	50	1	1	2	1	2	2	1	2	2	2	2	2	0.9	135	42	3.5
2	78	1	2	2	1	2	2	2	2	2	2	2	2	0.7	96	32	4
3	31	1	?	1	2	2	2	2	2	2	2	2	2	0.7	46	52	4
4	34	1	2	2	2	2	2	2	2	2	2	2	2	1	?	200	4

Pada kode di atas, di lakukan penghapusan kolom atau atribut unnamed: 0 yang terdapat pada hepatitis dataset yang di simpan dalam variabel HepatitisNew. Hal ini di karenakan, kolom tersebut di anggap tidak penting (tidak di perlukan untuk melakukan proses transformasi data).

- **Memeriksa Dan Menampilkan Nama-Nama Kolom Pada Hepatitis Dataset**

• Memeriksa Dan Menampilkan Nama-Nama Kolom Pada Hepatitis Dataset

```
In [5]: HepatitisKolom = HepatitisNew.columns

        print("Di Tampilkan Nama-Nama Kolom Yang Ada Pada Hepatitis Dataset : ")
        for kolom in HepatitisKolom:
            print(kolom)
```

```
Di Tampilkan Nama-Nama Kolom Yang Ada Pada Hepatitis Dataset :
Age
Sex
Steroid
Antivirals
Fatique
Malaise
Anorexia
Liver Big
Liver Firm
Spleen Palpable
Speiders
Ascites
Varices
Bilirubin
Alk Phosphate
SGOT
Albumin
Protime
Histology
CLASS
```

Pada kode di atas, di lakukan proses pemeriksaan dan pembacaan nama-nama kolom yang terdapat pada hepatitis dataset untuk di lakukan penjelasan lebih lanjut mengenai informasi yang terdapat dalam kolom atau atribut yang terdapat dalam dataset tersebut yang akan di jelaskan lebih lanjut seperti di bawah. Terdapat nama kolom *sex, streoid, antivirals, fatique, malaise, anorexia, liver big, liver firm, spleen, palpable, speiders, ascites, varices, bilirubin, alk phosphate, SGOT, albumin, protime, histology,* dan, *class*.

Selanjutnya, bisa di lanjutkan dengan menjawab pertanyaan-pertanyaan dan melakukan transformasi data lebih lanjut pada hepatitis dataset sebagai berikut.

A. Deskripsikan Definisi Masing-Masing Atribut Pada Dataset Hepatitis Berdasarkan Pengertian Di Dalam Domain Keilmuan Media Dan Refrensi Yang Dapat Di Pertanggungjawabkan.

Terdapat beberapa atribut pada dataset hepatitis, dengan penjelasan lebih lanjut sebagai berikut.

- *Age* : Merupakan atribut untuk merepresentasikan informasi mengenai umur pasien ketika di diagnosis terkena hepatitis, dalam kolom ini *age* (umur) di tuliskan dengan bilangan integer, seperti 30.
- *Sex* : Merupakan atribut untuk merepresentasikan informasi mengenai jenis kelamin pasien, dalam kolom ini *sex* (jenis kelamin) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean pria atau wanita.
- *Steroid* : Merupakan atribut untuk merepresentasikan informasi mengenai apakah pasien menerima obat *steroid* atau tidak menerima obat *steroid*, dalam kolom ini *streoid* (penerimaan obat) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean menerima atau tidaknya.
- *Antivirals* : Merupakan atribut untuk merepresentasikan informasi mengenai apakah pasien menerima obat *antivirals* atau tidak menerima obat *antivirals*, dalam kolom ini *antivirals* (penerimaan obat) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean menerima atau tidaknya.
- *Fatique* : Merupakan atribut untuk merepresentasikan informasi mengenai apakah pasien mengalami kelelahan atau kelemahan atau tidak mengalaminya, dalam kolom ini *fatique* (kelelahan/kelemahan) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean mengalami kelelahan/kelemahan atau tidaknya.
- *Malaise* : Merupakan atribut untuk merepresentasikan informasi mengenai apakah pasien mengalami ketidaknyamanan atau kebingungan atau pasien tidak mengalaminya, dalam kolom ini malaise (ketidaknyamanan/kebingungan) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean mengalami ketidaknyamanan/kebingungan atau tidaknya.
- *Anorexia* : Merupakan atribut untuk merepresentasikan informasi mengenai apakah pasien mengalami kehilangan nafsu makan atau tidak mengalami kehilangan nafsu makan, dalam kolom ini *anorexia* (nafsu makan) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean mengalami tidak nafsu makan atau tidaknya.
- *Liver Big* : Merupakan atribut untuk merepresentasikan infromasi mengenai apakah ukuran hati pasien mengalami pembesaran (*hepatomegali*) atau tidak mengalami pembesaran (*non-hepatomegali*), dalam kolom ini liver big (pembesaran ukuran hati) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean mengalami pembesaran hati atau tidaknya.
- *Liver Firm* : Merupakan atribut untuk merepresentasikan informasi mengenai apakah kondisi hati pasien mengalami pengerasan (terasa keras) atau tidak megalami pengerasan, dalam kolom ini *liver firm* (pengerasan hati) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean mengalami pengerasan hati atau tidaknya.
- *Spleen Palpable* : Merupakan atribut untuk merepresentasikan informasi mengenai apakah limpa pasien dapat di raba atau tidak dapat di raba selama pemeriksaan fisik pasien, dalam kolom ini *spleen palpable* (limpa) di

tuliskan dengan bilangan 1 dan 2 untuk pengkodean limpa dapat di raba atau tidaknya.

- *Speiders* : Merupakan atribut untuk merepresentasikan informasi mengenai apakah terdapat pembuluh darah yang tampak pada kulit pasien atau tidak terdapat pembuluh darah yang tampak, dalam kolom ini *speiders* (pembuluh darah tampak) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean pembuluh darah terlihat atau tidaknya.
- *Ascites* : Merupakan atribut untuk merepresentasikan informasi mengenai apakah pasien mengalami akumulasi cairan dalam perut atau tidak mengalami akumulasi cairan dalam perut, dalam kolom ini *ascites* (akumulasi cairan perut) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean mengalami akumulasi cairan dalam perut atau tidaknya.
- *Varices* : Merupakan atribut untuk merepresentasikan informasi megenai apakah pasien memili pembuluh darah yang melebar dan berbelok atau pasien tidak memiliki pembuluh darah yang melebar dan berbelok, dalam kolom ini *varices* (pembelokkan pembuluh darah) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean mengalami pembelokan atau pelebaran pembuluh darah atau tidaknya.
- *Bilirubin* : Merupakan informasi yang merepresentasikan informasi mengenai indikasi masalah hati yang di hasilkan dari pemecahan *hemoglobin* dalam sel darah merah yang di hilangkan oleh hati, dalam kolom ini bilirubin (masalah hati) di tuliskan dengan bilangan, seperti 1 dan 0,9 yang merupakan representasi konsentrasi *bilirubin* dalam darah pasien.
- *Alk Phosphate* : Merupakan informasi yang merepresentasikan informasi mengenai tingkat *alkali fosfatase* dalam darah pasien, dalam kolom ini *alk phosphate* (tingkat *alkali fosfate*) di tuliskan dengan bilangan, seperti 18 dan 42 yang merupakan representasi konsentrasi *alkali fosfate* dalam darah pasien.
- *SGOT* : Merupakan informasi yang merepresentasikan informasi menganai *Serum Glutamic Oxaloacetic Transaminase* dalam mengukur *SGOT* atau enzim yang dapat mengindikasikan masalah hati, dalam kolom ini *SGOT* di tuliskan dengan bilangan, seperti 18 dan 42 yang merupakan representasi konsentrasi *SGOT* dalam darah pasien.
- *Albumin* : Merupakan informasi yang merepresentasikan informasi mengenai tingkat *albumin* dalam darah pasien yang merupakan protein penting dalam tubuh, dalam kolom ini *albumin* (protein penting) di tuliskan dengan bilangan 4 dan 3,5.
- *Protime :* Merupakan informasi yang merepresentasikan mengenai waktu untuk pembekuan darah pasien, dalam kolom ini *protime* (waktu pembekuan darah) di tuliskan dengan bilangan 80 dan 75.
- *Histology* : Merupakan informasi yang merepresentasikan informasi mengenai pemeriksaan jaringan atau organ pasien, dalam kolom ini histology (pemeriksaan jaringan atau organ) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean baik atau tidaknya kondisi jaringan atau organ.
- *Class* : Merupakan informasi yang merepresentasikan informasi mengenai kelas atau status pasien dengan hepatitis, dalam kolom ini *class* (status pasien) di tuliskan dengan bilangan 1 dan 2 untuk pengkodean kelas atau status pasien dengan hepatitis.

B. Lakukan Transformasi Data Menggunakan Python Dengan Langkah-Langkah Sebagai Berikut.

1) Lakukan Pemeriksaan Apakah Terdapat Duplikasi Data (Row) Pada Dataset Hepatitis

```
1) Lakukan Pemeriksaan Apakah Terdapat Duplikasi Data (Row) Pada Dataset Hepatitis

In [6]: DataDuplikat = HepatitisNew[HepatitisNew.duplicated()]

        if DataDuplikat.empty:
            print("Tidak Di Peroleh Data Duplikat Dalam Hepatitis Dataset")
        else:
            print("Di Peroleh Data Duplikat Dalam Hepatitis Dataset")
            print(DataDuplikat)

        Tidak Di Peroleh Data Duplikat Dalam Hepatitis Dataset
```

Pada kode script di atas, di lakukan pemeriksaan mengenai keberadaan data duplikat yang ada dalam hepatitis dataset. Terdapat variabel DataDuplikat yang di pergunakan untuk menuliskan dan melakukan pengecekan baris-baris yang mengandung data duplikat dari *dataframe* HepatitisNew menggunakan fungsi *duplicated*. Selanjutnya di buat kondisi *if else* menggunakan DataDuplikat.empty yang di pergunakan untuk melakukan pengecekan jika DataDuplikat kosong, maka tidak di peroleh data duplikat dalam hepatitis dataset dan kondisi *if* terpenuhi.

Namun, jika DataDuplikat tidak kosong atau terdapat data duplikat maka kondisi *if* tidak terpenuhi dan akan memenuhi kondisi *else*, serta akan di tampilkan di peroleh data duplikat dalam hepatitis dataset beserta di tampilkan bagian mana yang terdapat data duplikatnya. Setelah melakukan rangkaian proses ini, maka akan di tampilkan bahwa tidak di peroleh data duplikat dalam hepatitis dataset.

1.1) Pemeriksaan Data Duplikat Pada Datframe Hepatitis Dataset

```
1.1) Pemeriksaan Data Duplikat Pada Dataframe Hepatitis Dataset

In [8]: DataDuplikat2 = HepatitisNew.duplicated()
        DataDuplikat2

Out[8]: 0      False
        1      False
        2      False
        3      False
        4      False
               ...
        150    False
        151    False
        152    False
        153    False
        154    False
        Length: 155, dtype: bool
```

Pada kode script di atas, di lakukan proses pengecekan data duplikat yang terdapat dalam *dataframe* hepatitis dataset. Terdapat variabel DataDuplikat2 yang di pergunakan untuk menuliskan dan melakukan pengecekan data duplikat pada baris-baris dataset menggunakan fungsi *duplicated*. Dengan hasil sebuah *series*, yakni serangkaian data yang berisi nilai *boolean*, dengan nilai *true* jika terdapat data duplikat dan nilai *false* jika tidak terdapat data duplikat.

Maka, akan di tampilkan *series* dari DataDuplikat2 yakni *false* yang berarti tidak ada duplikat dari *dataframe* hepatitis dataset.

2) Lakukan Analisis Deskriptif Berapa Banyak Atribut Yang Memiliki Missing Values

Untuk melakukan pengecekan mengenai berapa banyak atribut yang memiliki *missing values*, harus di lakukan beberapa langkah-langkah di antaranya yakni :

2.1) Melakukan Pergantian Nilai "?" Menjadi "Nan" Untuk Menunjukkan Nilai Missing Values

2.1) Melakukan Pergantian Nilai "?" Menjadi "Nan" Untuk Menunjukkan Nilai Missing Values

```
In [11]: HepatitisNew.replace("?", float("nan"), inplace = True)
         HepatitisNew
```

Out[11]:

	Age	Sex	Steroid	Antivirals	Fatique	Malaise	Anorexia	Liver Big	Liver Firm	Spleen Palpable	Speiders	Ascites	Varices	Bilirubin	Alk Phosphate	SGOT	Albumin
0	30	2	1.0	2	2.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	1.0	85.0	18.0	4.0
1	50	1	1.0	2	1.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	0.9	135.0	42.0	3.5
2	78	1	2.0	2	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	96.0	32.0	4.0
3	31	1	NaN	1	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	46.0	52.0	4.0
4	34	1	2.0	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.0	NaN	200.0	4.0

Pada kode script di atas, di lakukan pergantian nilai “?” menjadi “Nan” untuk menunjukkan *missing values*. Hal ini di karenakan “?” bukan merupakan data numerik dan tidak bisa di perhitungkan *missing valuesnya*, oleh karena itu untuk menghitung nilai nan yang terdapat pada hepatitis dataset harus di lakukan pengubahan dari “?” menjadi nilai “Nan” menggunakan float(“nan”) untuk menghasilkan nilai *nan* yang sesuai dengan *inplace = True* di pergunakan untuk mengubah *dataframe* asli.

Maka, akan di tampilkan perubahan data “?” menjadi nilai “Nan”, untuk perubahan ini dapat di perlihatkan dari *index* 4 atribut *alk phosphate* yang awalnya “?” menjadi “NaN”. Kemudian selanjutnya bisa di lakukan ke proses pencarian nilai *missing values* pada masing-masing atribut.

2.2) Nilai Missing Values Pada Masing-Masing Atribut

2.2)Nilai Missing Values Pada Masing-Masing Atribut

```
In [16]: DataMissingValues = HepatitisNew.isna().sum()

         print("Terdapat Missing Values Pada Masing-Masing Atribut : ")
         print(DataMissingValues)
```

```
Terdapat Missing Values Pada Masing-Masing Atribut :
Age                 0
Sex                 0
Steroid             1
Antivirals          0
Fatique             1
Malaise             1
Anorexia            1
Liver Big          10
Liver Firm         11
Spleen Palpable     5
Speiders            5
Ascites             5
Varices             5
Bilirubin           6
Alk Phosphate      29
SGOT                4
Albumin            16
Protime            67
Histology           0
CLASS               0
dtype: int64
```

Pada kode script di atas, di lakukan proses perhitungan *missing values* pada masing-masing atribut yang sebelumnya telah di lakukan proses perubahan nilai “?” menjadi nilai “Nan”. Terdapat variabel DataMissingValues yang di pergunakan untuk melakukan pengecekan nilai *missing values* dari *dataframe* HepatitisNew menggunakan *isna* dan di lakukan perhitungan dengan menggunakan *sum*.

Maka, akan di tampilkan bahwa terdapat *missing values* pada masing-masing atribut dengan rincian, yakni atribut *age, sex, antivirals, histology,* dan *class* memiliki 0 *missing values*. *Steroid, fatique, malaise,* dan *anorexia* memiliki 1 *missing values.* Liver big memiliki 10 *missing values* dan liver firm memiliki 11 *missing values*. *Spleen palpable, speiders, ascites,* dan *varices*

memiliki masing-masing 5 *missing values*. *Bilirubin* memiliki 6 *missing values*, *alk phosphate* memiliki 29 *missing values*, *SGOT* memiliki 4 *missing values*, *albumin* memiliki 16 *missing values*, dan *protime* memiliki 67 *missing values*.

2.3) Nama-Nama Atribut Yang Memiliki Missing Values

2.3) Nama-Nama Atribut Yang Memiliki Missing Values

In [17]:
```
DataMissingValues2 = HepatitisNew.isna().any()
AtributMissing = DataMissingValues2[DataMissingValues2 == True]

print("Nama-Nama Atribut Yang Memiliki Missing Values : ", AtributMissing.index.tolist())
```

```
Nama-Nama Atribut Yang Memiliki Missing Values :  ['Steroid', 'Fatique', 'Malaise', 'Anorexia', 'Liver Big', 'Liver Firm', 'Spl
een Palpable', 'Speiders', 'Ascites', 'Varices', 'Bilirubin', 'Alk Phosphate', 'SGOT', 'Albumin', 'Protime']
```

Pada kode script di atas, di lakukan proses pencarian nama-nama atribut yang memiliki nilai *missing values*. Terdapat variabel DataMissingValues2 yang di pergunakan untuk menuliskan pencarian nilai *missing values* menggunakan fungsi *isna* dan *any* untuk membuat *series boolean* yang menunjukkan apakah setiap atribut atau kolom pada *dataframe* HepatitisNew memiliki nilai *missing values* (*true*) atau tidak (*false*).

Selanjutnya, terdapat variabel AtributMissing yang di pergunakan untuk melakukan pemilihan untuk menampilkan nama-nama atribut yang memiliki *missing values*. Maka, akan di tampilkan nama-nama atribut yang memiliki *missing values*, di antaranya yakni *steroid, fatique, anorexia, liver big, liver firm, spleen palpable, speiders, ascites, varices, bilirubin, alk phosphate, SGOT, albumin*, dan *protime*.

3) Lakukan Imputasi Missing Values Dengan Berbagai Teknik Yang Efektif Dan Baik Untuk Mengisi Atribut-Atribut Yang NaN, Misalnya Menggunakan Mean, Median, Modus, Clustering, Regression, Maupun Metode Taksiran Dan Prediksi Lainnya

Untuk melakukan imputasi missing values pada atribut-atribut yang memiliki *missing values* bisa di lakukan beberapa langkah-langkah di antaranya, yakni :

3.1) Mengisi Imputasi Missing Values Kolom Steroid, Fatique, Malaise, Anorexia, Liver Big, Liver Firm, Spleen Palpable, Speiders, Ascites, Varices Menggunakan Nilai Modus Dan Kolom Bilirubin, Alk Phosphate, SGOT, Albumin Menggunakan Nilai Median

3.1) Mengisi Imputasi Missing Values Kolom Steroid, Fatique, Malaise, Anorexia, Liver Big, Liver Firm, Spleen Palpable, Speiders, Ascites, Varices Menggunakan Nilai Modus Dan Kolom Bilirubin, Alk Phosphate, SGOT, Albumin Menggunakan Nilai Median

In [18]:
```
import numpy as np

NilaiModus = ["Steroid", "Fatique", "Malaise", "Anorexia", "Liver Big", "Liver Firm", "Spleen Palpable", "Speiders", "Ascites",
              "Varices"]
NilaiMedian = ["Bilirubin", "Alk Phosphate", "SGOT", "Albumin"]

for kolom in NilaiModus:
    Modus = HepatitisNew[kolom].mode()[0]
    HepatitisNew[kolom].fillna(Modus, inplace=True)

for kolom in NilaiMedian:
    Median = HepatitisNew[kolom].median()
    HepatitisNew[kolom].fillna(Median, inplace=True)

HepatitisNew
```

Out[18]:

	Age	Sex	Steroid	Antivirals	Fatique	Malaise	Anorexia	Liver Big	Liver Firm	Spleen Palpable	Speiders	Ascites	Varices	Bilirubin	Alk Phosphate	SGOT	Albumin
0	30	2	1.0	2	2.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	1.0	85.0	18.0	4.0
1	50	1	1.0	2	1.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	0.9	135.0	42.0	3.5
2	78	1	2.0	2	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	96.0	32.0	4.0
3	31	1	2.0	1	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	46.0	52.0	4.0
4	34	1	2.0	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.0	85.0	200.0	4.0

Pada kode script di atas, di lakukan proses mengisi imputasi *missing values* dengan menggunakan nilai modus dan nilai median pada masing-masing

atribut yang memiliki *missing values*. Di pergunakan *library* numpy untuk menggunakan fungsi bawaan numpy untuk menghitung nilai modus dan median. Terdapat variabel NilaiModus yang berisi list atribut (kolom) *steroid, fatique, malaise, liver big, liver firm, spleen palpable, speiders, ascites,* dan *varices* di mana nama-nama atribut ini nantinya akan di isi dengan nilai modus (nilai yang paling sering muncul) dari atribut tersebut. Pengisian nilai modus ini dengan mempertimbangkan bahwa pada kolom-kolom tersebut berisi nilai 1 dan 2 sama seperti nilai *boolean true* dan *false*, oleh karena itu nilai modus cocok di gunakan untuk melakukan dan imputasi untuk menanggani *missing values* pada atribut tersebut.

Selanjutnya, terdapat variabel NilaiMedian yang berisi list atribut (kolom) *bilirubin, alk phosphat, SGOT,* dan *albumin*, di mana nama-nama atribut ini nantinya akan di isi dengan nilai median (nilai tengah) dari atribut tersebut. Pengisian nilai median ini dengan mempertimbangkan bahwa pada kolom-kolom tersebut berisi nilai-nilai dengan rentang seperti 1,0 dan 0,9 oleh karena itu nilai median cocok di gunakan untuk melakukan dan imputasi untuk menanggani *missing values* pada atribut tersebut.

Terdapat iterasi *looping for* yang di pergunakan untuk mengisi nilai modus pada list atribut NilaiModus dengan menggunakan fungsi bawaan numpy, yakni *mode* kemudian *fillna* di pergunakan untuk mengganti nilai *missing values* menjadi nilai modus yang di berikan. Terdapat pula iterasi *looping for* kedua yang di pergunakan untuk mengisi nilai median pada list atribut NilaiMedian dengan menggunakan fungsi bawaan numpy, yakni median kemudian fillna di pergunakan untuk mengganti *missing values* menjadi nilai median yang di berikan.

Maka, akan di tampilkan *dataframe* dari HepatitisNew dengan pengisian nilai modus dan median dari atribut yang di tentukan. Dapat di perlihatkan bahwa, *streoid index* 3 yang sebelumnya nilai *NaN* di ganti dengan nilai modus sebesar 1.0 dan *alk phosphate index* 4 yang sebelumnya nilai *NaN* di ganti dengan nilai median 85.0.

3.2) Mengisi Imputasi Missing Values Kolom Protime Menggunakan Nilai Regresi

3.2) Mengisi Imputasi Missing Values Kolom Protime Menggunakan Nilai Regresi

In [20]:
```
from sklearn.linear_model import LinearRegression

DataLengkap = HepatitisNew.dropna(subset = ["Protime", "Bilirubin", "Alk Phosphate", "SGOT"])
DataImputasi = HepatitisNew[HepatitisNew["Protime"].isnull()]

ModelRegresi = LinearRegression()
ModelRegresi.fit(DataLengkap[["Bilirubin", "Alk Phosphate", "SGOT"]], DataLengkap["Protime"])

NilaiPrediksi = ModelRegresi.predict(DataImputasi[["Bilirubin", "Alk Phosphate", "SGOT"]])

DataImputasi["Protime"] = NilaiPrediksi.round(1)
HepatitisNew.loc[DataImputasi.index] = DataImputasi

HepatitisNew
```

Out[20]:

roid	Antivirals	Fatique	Malaise	Anorexia	Liver Big	Liver Firm	Spleen Palpable	Speiders	Ascites	Varices	Bilirubin	Alk Phosphate	SGOT	Albumin	Protime	Histology	CLASS
1.0	2	2.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	1.0	85.0	18.0	4.0	66.0	1	2
1.0	2	1.0	2.0	2.0	1.0	2.0	2.0	2.0	2.0	2.0	0.9	135.0	42.0	3.5	62.7	1	2
2.0	2	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	96.0	32.0	4.0	66.5	1	2
2.0	1	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	0.7	46.0	52.0	4.0	80.0	1	2
2.0	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.0	85.0	200.0	4.0	62.5	1	2

Pada kode script di atas, di lakukan proses imputasi untuk menanggani *missing values* pada atribut *protime* menggunakan regresi. Dengan menggunakan *library* scikit-learn dan menggunakan model regresi linier. Terdapat variabel DataLengkap yang di pergunakan untuk menghapus baris yang memiliki nilai *nan* dari atribut *bilirubin, alk phosphate,* dan *SGOT* yang

di pergunakan untuk mempersiapkan data yang di gunakan dalam model regresi, namun nilai *nan* dari atribut-atribut ini sudah di lakukan imputasi dengan pengisian nilai sebelumnya. Kemudian, terdapat variabel DataImputasi yang di pergunakan untuk menyimpan nilai *nan* pada atribut atau kolom *protime* menggunakan *isnull* atau *isna* yang akan di pergunakan untuk memperkirakan nilai dari atribut protime yang hilang.

Selanjutnya, terdapat variabel ModelRegresi yang di pergunakan untuk membuat objek model regresi linier. Terdapat ModelRegresi.fit untuk melakukan pemrosesan model regresi linier dengan variabel independen/prediktor/x berupa atribut atau kolom *bilirubin, alk phosphate*, dan *SGOT* dengan variabel dependen/respon/y berupa atribut *protime*. Kemudian, di lanjutkan melakukan prediksi nilai regresi untuk mengisi nilai *nan* pada kolom *protime* dengan tipe data *float* dan terdapat 1 angka di belakang koma.

Terdapat HepatitisNew.loc[DataImputasi.index] yang di pergunakan untuk menggantikan nilai yang hilang dalam *dataframe* menggunakan nilai yang telah di imputasi/di prediksi. Penggunaan imputasi menggunakan regresi ini di karenakan dalam dataset hepatitis terdapat hubungan antara protime dengan bilirubin, alk phosphate, dan SGOT untuk memahami tingkat kerusakan hati dan perkembangan hepatitis.

Selain itu, kolom *bilirubin, alk phosphate*, dan *SGOT* bisa mempengaruhi waktu pembekuan darah, yakni *protime*. Maka, akan di tampilkan hasil imputasi atau pengisian nilai di kolom atau atribut *protime* yang bisa di perlihatkan pada index 0 yang sebelumnya *nan* menjadi 66,0.

3.3) Mengecek Keberadaan Missing Values Setelah Di Lakukan Imputasi

```
3.3) Mengecek Keberadaan Missing Values Setelah Melakukan Imputasi

In [22]: DataMissingValues3 = HepatitisNew.isna().sum()

         print("Terdapat Missing Values Pada Masing-Masing Atribut Setelah Di Lakukan Imputasi : ")
         print(DataMissingValues3)

         Terdapat Missing Values Pada Masing-Masing Atribut Setelah Di Lakukan Imputasi :
         Age                0
         Sex                0
         Steroid            0
         Antivirals         0
         Fatique            0
         Malaise            0
         Anorexia           0
         Liver Big          0
         Liver Firm         0
         Spleen Palpable    0
         Speiders           0
         Ascites            0
         Varices            0
         Bilirubin          0
         Alk Phosphate      0
         SGOT               0
         Albumin            0
         Protime            0
         Histology          0
         CLASS              0
         dtype: int64
```

Pada kode script di atas, di lakukan proses pengecekan apakah masih terdapat nilai *missing values* pada masing-masing atribut yang telah di lakukan imputasi. Terdapat variabel DataMissingValues3 yang di pergunakan untuk melakukan pengecekan dengan menggunakan fungsi *isna* dan *sum* untuk melakukan penjumlahan *missing values*. Setelah di lakukan proses imputasi, maka akan di tampilkan bahwa semua atribut dalam *dataframe* HepatitisNew yang berisi dataset hepatitis sudah tidak ada lagi yang memiliki *missing values*.

4. DETEKSI *OUTLIER* : *INTERQUARTILE RANGE* DAN *BOXPLOT*

Sebelum melakukan deteksi *outlier* dalam *ATM-Transaction* dataset, harus di lakukan langkah-langkah awal untuk *meload* (membaca) dataset sebagai berikut.

- **Load Dataset *ATM-Transaction***

- Load Dataset ATM-Transaction

In [1]:
```
import pandas as pd

AtmTransaction = pd.read_csv("ATM-Transaction.csv", sep = ";")
AtmTransaction
```

Out[1]:

	ATM Name	Transaction Date	No Of Withdrawals	Total amount Withdrawn (Rupee)	Total amount Withdrawn (Rupiah)	Weekday
0	Big Street ATM	01/01/11	50	123800	23136982	Saturday
1	Mount Road ATM	01/01/11	253	767900	143512831	Saturday
2	Airport ATM	01/01/11	98	503400	94080426	Saturday
3	KK Nagar ATM	01/01/11	265	945300	176667117	Saturday
4	Christ College ATM	01/01/11	74	287700	53768253	Saturday
...	...	...	...	...	...	...
11584	Big Street ATM	29/09/17	137	468800	87614032	FRIDAY
11585	Mount Road ATM	29/09/17	79	305100	57020139	FRIDAY
11586	Airport ATM	29/09/17	117	709900	132673211	FRIDAY
11587	KK Nagar ATM	29/09/17	76	408700	76381943	FRIDAY
11588	Christ College ATM	29/09/17	143	700400	130897756	FRIDAY

11589 rows × 6 columns

Pada kode di atas, di lakukan load dan pembacaan ATM-*Transaction* dataset menggunakan *library* pandas dengan jenis file csv. Kemudian, ATM-*Transaction* dataset tersebut di simpan ke dalam variabel AtmTransaction dengan 11589 baris dan 6 kolom.

- **Memeriksa Dan Menampilkan Nama-Nama Kolom Pada ATM-Transaction Dataset**

- Memeriksa Dan Menampilkan Nama-Nama Kolom Pada ATM-Transaction Dataset

In [2]:
```
AtmKolom = AtmTransaction.columns

print("Di Tampilkan Nama-Nama Kolom Yang Ada Pada ATM-Transaction Dataset : ")
for kolom in AtmKolom:
    print(kolom)
```

```
Di Tampilkan Nama-Nama Kolom Yang Ada Pada ATM-Transaction Dataset :
ATM Name
Transaction Date
No Of Withdrawals
Total amount Withdrawn (Rupee)
Total amount Withdrawn (Rupiah)
Weekday
```

Pada kode di atas, di lakukan proses pemeriksaan dan pembacaan nama-nama kolom yang terdapat pada ATM-*Transaction* dataset untuk di lakukan penjelasan lebih lanjut mengenai informasi yang terdapat dalam kolom atau atribut yang terdapat dalam dataset tersebut yang akan di jelaskan lebih lanjut seperti di bawah. Terdapat nama kolom *ATM Name, Transaction Date, No Of Withdrawn, Total Amount Withdrawn (Rupee), Total Amount Withdrawn (Rupiah),* dan *Weekday.*

- **Menampilkan Statistika Deskriptif Dari ATM-Transaction Dataset**

- Menampilkan Statistika Deskriptif Dari ATM-Transaction Dataset

In [3]:
```
AtmDescribe = AtmTransaction.describe()

print("Di Tampilkan Statistika Deskriptif Dari ATM-Transaction : ")
AtmDescribe
```

```
Di Tampilkan Statistika Deskriptif Dari ATM-Transaction :
```

Out[3]:

	No Of Withdrawals	Total amount Withdrawn (Rupee)	Total amount Withdrawn (Rupiah)
count	11589.000000	1.158900e+04	1.158900e+04
mean	123.341099	5.223059e+05	9.761375e+07
std	67.315288	3.248167e+05	6.070499e+07
min	1.000000	1.000000e+02	1.868900e+04
25%	79.000000	3.057000e+05	5.713227e+07
50%	115.000000	4.700000e+05	8.783830e+07
75%	158.000000	6.716000e+05	1.255153e+08
max	491.000000	2.549800e+06	4.765321e+08

Pada kode di atas, di lakukan proses penampilan statistika deskriptif dari ATM-Transaction dataset menggunakan fungsi *describe*. Maka, akan di tampilkan *count, mean, std*, Q1, Q2, Q3, dan nilai *max* dari masing-masing kolom atau atribut dari ATM-Transaction dataset.

Selanjutnya, bisa di lanjutkan dengan menjawab pertanyaan-pertanyaan dan melakukan deteksi outlier lebih lanjut pada *ATM-Transaction* dataset sebagai berikut.

A. Deskripsikan Definisi Masing-Masing Atribut Pada Dataset ATM Transcation Berdasarkan Pengertian Di Dalam Domain Keilmuan Banking Dan Referensi Yang Dapat Di Pertanggungjawabkan.

Terdapat beberapa atribut pada dataset ATM-Transaction, dengan penjelasan lebih lanjut sebagai berikut.

- ATM *Name* : Merupakan atribut untuk merepresentasikan informasi mengenai nama ATM tertentu sebagai identifikasi uniknya, dalam kolom ini nama atm di tuliskan, seperti Big Street ATM dan Mount Road ATM.
- *Transcaction Date* : Merupakan atribut untuk merepresentasikan informasi mengenai tanggal transaksi yang di lakukan pada ATM, dalam kolom ini tanggal transaksi di tuliskan dengan format tanggal-bulan-tahun, seperti 01/01/11.
- *No Of Withdrawals* : Merupakan atribut untuk merepresentasikan informasi mengenai jumlah penarikan uang yang di lakukan pada tanggal tertentu dan di mesin ATM tertentu, dalam kolom ini jumlah penarikan uang di tuliskan, seperti 50 dan 253.
- *Total Amount Withdrawn (In Rupee)* : Merupakan atribut untuk merepresentasikan total jumlah penarikan uang yang di lakukan pada tanggal dan hari tersebut dalam mata uang rupee, dalam kolom ini jumlah penarikan uang di tuliskan, seperti 123800 dan 767900.
- *Total Amount Withdrawn (In Rupiah)* : Merupakan atribut untuk merepresentasikan total jumlah penarikan uang yang di lakukan pada tanggal dan hari tersebut dalam mata uang rupiah, dalam kolom ini jumlah penarikan uang di tuliskan, seperti 23136982 dan 143512831.
- *Weekday* : Merupakan atribut untuk merepresentasikan hari di tanggal transaksi ATM tersebut, dalam kolom ini hari di tuliskan seperti saturday dan sunday.

B. Lakukan Analisis Deskiriptif Dan Deteksi Outlier Menggunakan Python Dan Menjawab Pertanyaan Berikut.

1) Bank Mana Yang Teramai Dan Tersepi Bagi Nasabah Untuk Melakukan Penarikan Uang?

1) Bank Mana Yang Teramai Dan Tersepi Bagi Nasabah Untuk Melakukan Penarikan Uang?

In [9]:
```
KategoriPenarikan = AtmTransaction.groupby("ATM Name")["No Of Withdrawals"].sum()

PenarikanTeramai = KategoriPenarikan.idxmax()
PenarikanTersepi = KategoriPenarikan.idxmin()

JumlahPenarikanTeramai = KategoriPenarikan.max()
JumlahPenarikanTersepi = KategoriPenarikan.min()

print("Di Peroleh Nama Bank Dengan Penarikan Uang Teramai :", PenarikanTeramai, "Dengan Jumlah Penarikan :", \
      JumlahPenarikanTeramai)
print("Di Peroleh Nama Bank Dengan Penarikan Uang Tersepi :", PenarikanTersepi, "Dengan Jumlah Penarikan :", \
      JumlahPenarikanTersepi)
```

```
Di Peroleh Nama Bank Dengan Penarikan Uang Teramai : KK Nagar ATM Dengan Jumlah Penarikan : 401858
Di Peroleh Nama Bank Dengan Penarikan Uang Tersepi : Airport ATM Dengan Jumlah Penarikan : 204709
```

Pada kode script di atas, di lakukan proses pencarian nama bank yang teramai dan tersepi bagi nasabah untuk melakukan penarikan uang. Terdapat variabel KategoriPenarikan yang di pergunakan untuk menuliskan proses pengelompokkan menggunakan *group by* dari *dataframe* AtmTransaction dengan atribut atau kolom dari baris *ATM Name* dan menggabungkannya dengan kolom *No Of Withdrawals* kemudian di jumlahkan menggunakan *sum*. Terdapat variabel PenarikanTeramai yang di pergunakan untuk mencari nama bank teramai dari KategoriPenarikan menggunakan *idxmax* dan variabel PenarikanTersepi yang di pergunakan untuk mencari nama bank tersepi dari KategoriPenarian menggunakan *idxmin*.

Kemudian, terdapat variabel JumlahPenarikanTeramai yang di pergunakan untuk mencari nilai maksimum dan menampilkan jumlah penarikan dari nama bank teramai menggunakan *max* dan variabel JumlahPenarikanTersepi yang di pergunakan untuk mencari nilai minimum dan menampilkan jumlah penarikan dari nama bank tersepi menggunakan *min*. Maka, akan di tampilkan nama bank dengan penarikan uang teramai adalah KK Nagar Atm dengan jumlah penarikan 401858 dan nama bank dengan penarikan tersepi adalah Airport Atm dengan jumlah penarikan 204709.

1.1) Nama Bank Dengan Masing-Masing Jumlah Penarikan

1.1) Nama Bank Dengan Masing-Masing Jumlah Penarikan

In [13]:
```
for AtmBank, JumlahPenarikan in KategoriPenarikan.items():
    print("Nama Bank : \n", AtmBank, "Dengan Jumlah Penarikan", JumlahPenarikan)
```

```
Nama Bank :
 Airport ATM Dengan Jumlah Penarikan 204709
Nama Bank :
 Big Street ATM Dengan Jumlah Penarikan 207062
Nama Bank :
 Christ College ATM Dengan Jumlah Penarikan 291207
Nama Bank :
 KK Nagar ATM Dengan Jumlah Penarikan 401858
Nama Bank :
 Mount Road ATM Dengan Jumlah Penarikan 324564
```

Pada kode script di atas, di lakukan proses pencarian nama bank dengan masing-masing jumlah penarikan yang di pergunakan untuk membuktikan bahwa jawaban di nomor sebelumnya sudah benar. Terdapat proses *looping for* dari variabel AtmBank dan JumlahPenarikan dari KategoriPenarikan.items yang di pergunakan untuk menyimpan nilai-nilai dari setiap pasangan nama bank dan jumlah penarikan masing-masing bank.

Maka, akan di tampilkan nama bank Airport ATM dengan jumlah penarikan 204709, Big Street ATM dengan jumlah penarikan 207062, Christ College ATM dengan jumlah panrikan 291207, KK Nagar ATM dengan jumlah penarikan 401858, dan Mount Road ATM dengan jumlah penarikan 324564.

2) Pada Hari Apa ATM Paling Sering Dan Pada Hari Apa Yang Paling Jarang Di Kunjungi Oleh Nasabah Untuk Melakukan Penarikan Uang?

2.1) Melakukan Rename Pada Kolom Nama Weekday (Hari)

2.1) Melakukan Rename Pada Kolom Nama Weekday (Hari)

In [16]:
```
RenameHari = {"SUNDAY" : "Sunday", "MONDAY" : "Monday", "TUESDAY" : "Tuesday", "WEDNESDAY" : "Wednesday", \
              "THURSDAY" : "Thursday", "FRIDAY" : "Friday", "SATURDAY" : "Saturday"}

AtmTransaction["Weekday"].replace(RenameHari, inplace = True)
AtmTransaction
```

Out[16]:

	ATM Name	Transaction Date	No Of Withdrawals	Total amount Withdrawn (Rupee)	Total amount Withdrawn (Rupiah)	Weekday
0	Big Street ATM	01/01/11	50	123800	23136982	Saturday
1	Mount Road ATM	01/01/11	253	767900	143512831	Saturday
2	Airport ATM	01/01/11	98	503400	94080426	Saturday
3	KK Nagar ATM	01/01/11	265	945300	176667117	Saturday
4	Christ College ATM	01/01/11	74	287700	53768253	Saturday

Pada kode script di atas, di lakukan proses *rename* (pengubahan) nama dari atribut atau kolom weekday (hari), hal ini di karenakan terdapat perbedaan penulisan pada nama hari di kolom tersebut. Terdapat nama hari yang di tuliskan dengan huruf kapital semua (*upper case*) dan terdapat hari yang di tuliskan dengan huruf kapital di awal (*title case*) hal ini bisa mempengaruhi proses pencarian dalam hal kekonsistenan jumlah hari dengan kunjungan teramai dan tersepi dari bank untuk melakukan penarikan.

Terdapat variabel RenameHari yang di pergunakan sebagai *directory* untukk menuliskan nama-nama hari yang ingin di ganti, seperti "SUNDAY" menjadi "Sunday". Kemudian, di lakukan proses *replace* atau pencarian nama-nama yang cocok dari dataframe kolom *weekday* di AtmTransaction. Maka, akan di tampilkan rename di kolom *weekday, seperti Sunday, Monday, Tuesday, Wednesday, Thursday, Friday,* dan *Saturday*.

2.2) Mencari Nama Hari Dengan Kunjungan Teramai Dan Tersepi Untuk Melakukan Penarikan Dari Bank

2.2) Mencari Nama Hari Dengan Kunjungan Teramai Dan Tersepi Untuk Melakukan Penarikan Dari Bank

```
In [19]: KategoriHari = AtmTransaction.groupby("Weekday")["No Of Withdrawals"].sum()

         KategoriHariTeramai = KategoriHari.sort_values(ascending = False)
         KategoriHariTersepi = KategoriHari.sort_values(ascending = True)

         HariTeramai = KategoriHariTeramai.index[0]
         HariTersepi = KategoriHariTersepi.index[0]

         JumlahKunjunganTeramai = KategoriHariTeramai.iloc[0]
         JumlahKunjunganTersepi = KategoriHariTersepi.iloc[0]

         print("Di Peroleh Hari Dengan Kunjungan Teramai :", HariTeramai, "Dengan Jumlah Kunjungan Untuk Penarikan :", \
               JumlahKunjunganTeramai)
         print("Di Peroleh Hari Dengan Kunjungan Tersepi :", HariTersepi, "Dengan Jumlah Kunjungan Untuk Penarikan :", \
               JumlahKunjunganTersepi)

         Di Peroleh Hari Dengan Kunjungan Teramai : Sunday Dengan Jumlah Kunjungan Untuk Penarikan : 311304
         Di Peroleh Hari Dengan Kunjungan Tersepi : Friday Dengan Jumlah Kunjungan Untuk Penarikan : 167463
```

Pada kode script di atas, di lakukan proses pencarian jumlah hari dengan kunjungan teramai dan tersepi untuk melakukan penarikan dari bank. Terdapat variabel KategoriHari yang di pergunakan untuk menuliskan proses pengelompokkan menggunakan *group by* dari *dataframe* AtmTransaction dengan atribut atau kolom dari baris *weekday* dan menggabungkannya dengan kolom *No Of Withdrawals* kemudian di jumlahkan menggunakan *sum*. Terdapat variabel KategoriHariTeramai yang di pergunakan untuk mengurutkan kolom *No Of Withdrawals* dengan urutan *descending* (terbesar ke terkecil) dan variabel KategoriHariTersepi yang di pergunakan untuk mengurutkan kolom *No Of Withdrawals* dengan urutan *ascending* (terkecil ke terbesar).

Kemudian, terdapat variabel HariTeramai yang berisi dan menyimpan nilai *index* ke 0 yang berupa hasil dari variabel KategoriHariTeramai yang berisi nama hari teramai dan variabel HariTersepi yang berisi dan menyimpan nilai *index* ke 0 yang berupa hasil dari variabel KategoriHariTersepi yang berisi nama hari tersepi. Selanjutnya, terdapat variabel JumlahKunjunganTeramai yang di pergunakan untuk mengambil baris pertama dari data yang telah di urutkan pada KategoriHariTeramai yang berisi jumlah penarikan dari hari teramai dan variabel KategoriHariTersepi yang berisi jumlah penarikan dari hari tersepi. Maka, akan di tampilkan hari dengan kunjungan teramai adalah *Sunday* dengan jumlah kunjungan untuk penarikan 311304 dan hari dengan kunjungan tersepi adalah *Friday* dengan jumlah kunjungan untuk penarikan 167463.

2.3) Mencari Nama Hari Dengan Kunjungan Dari Masing-Masing Bank

2.3) Mencari Nama Hari Dengan Kunjungan Untuk Penarikan Uang Dari Masing-Masing Bank

```
In [20]: for AtmBank, JumlahKunjungan in KategoriHari.items():
             print("Hari : \n", AtmBank, "Dengan Jumlah Kunjungan Untuk Penarikan", JumlahKunjungan)

Hari :
 Friday Dengan Jumlah Kunjungan Untuk Penarikan 167463
Hari :
 Monday Dengan Jumlah Kunjungan Untuk Penarikan 181986
Hari :
 Saturday Dengan Jumlah Kunjungan Untuk Penarikan 224705
Hari :
 Sunday Dengan Jumlah Kunjungan Untuk Penarikan 311304
Hari :
 Thursday Dengan Jumlah Kunjungan Untuk Penarikan 174770
Hari :
 Tuesday Dengan Jumlah Kunjungan Untuk Penarikan 185715
Hari :
 Wednesday Dengan Jumlah Kunjungan Untuk Penarikan 183457
```

Pada kode script di atas, di lakukan proses pencarian nama hari dengan masing-masing jumlah kunjungan untuk penarikan yang di pergunakan untuk membuktikan bahwa jawaban di nomor sebelumnya sudah benar. Terdapat proses *looping for* dari variabel AtmBank dan JumlahKunjungan dari KategoriHari.items yang di pergunakan untuk menyimpan nilai-nilai dari setiap pasangan nama hari dan jumlah kunjungan hari untuk penarikan dari masing-masing bank.

Maka, akan di tampilkan hari *Friday* dengan jumlah kunjungan untuk penarikan 167463, *Monday* dengan jumlah kunjungan untuk penarikan 181986, *Saturday* dengan jumlah kunjungan untuk penarikan 224705, *Sunday* dengan jumlah kunjungan untuk penarikan 311304, *Thursday* dengan jumlah kunjungan untuk penarikan 174770, *Tuesday* dengan jumlah kunjungan untuk penarikan 185715, dan *Wednesday* dengan jumlah kunjungan untuk penarikan 183457.

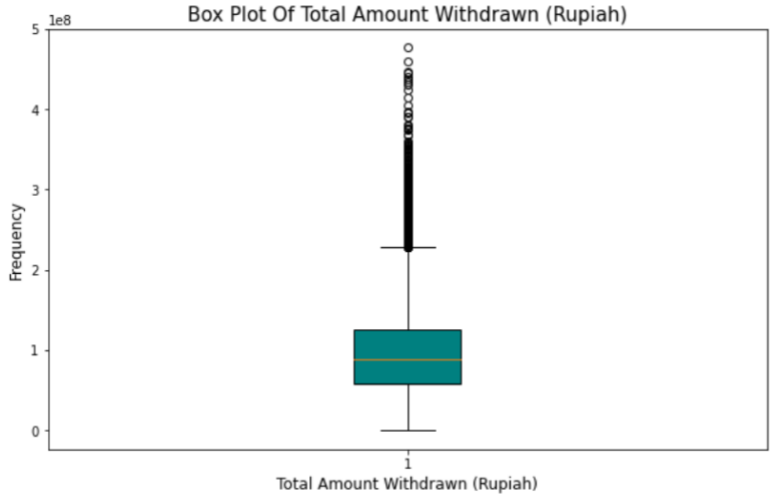
3) Lakukan Deteksi Outlier Untuk Melihat Potensi Adanya Fraud Berdasarkan Atribut Total Amount Withdrawn (In Rupiah) Pada Dataset ATM Transaction Menggunakan Metode InterQuartileRange Dan Visualisasikan Menggunakan BoxPlot Serta Sebutkan Nama No Of Withdrawals Dan Nama ATM-nya Yang Terdeteksi Fraud.

Untuk melakukan deteksi outlier dan melihat potensi adanya fraud berdasarkan atribut total amount withdrawn (in rupiah) menggunakan metode InterQuartileRange dan memvisualisasikannya menggunakan BoxPlot, serta menyebutkan nama No Of Withdrawals dengan nama ATM yang terderteksi fraud, bisa di lakukan serangkaian langkah-langkah sebagai berikut :

3.1) Melakukan Visualisasi Menggunakan BoxPlot Dan *Menyebutkan No Of Withdrawals* Dan Nama ATM-nya Yang Terdeteksi *Fraud*

3.1) Melakukan Visualisasi Menggunakan BoxPlot Dan Menyebutkan No Of Withdrawals Dan Nama ATM-nya Yang Terdeteksi Fraud

In [28]:
```
import matplotlib.pyplot as plt

#perhitungan interquartile range
Q1 = AtmTransaction["Total amount Withdrawn (Rupiah)"].quantile(0.25)
Q3 = AtmTransaction["Total amount Withdrawn (Rupiah)"].quantile(0.75)
IQR = Q3 - Q1

#perhitungan bb dan ba untuk mendeteksi outlier
BB = Q1 - 1.5 * IQR
BA = Q3 + 1.5 * IQR

#melakukan filter data untuk outlier
Outliers = AtmTransaction[(AtmTransaction["Total amount Withdrawn (Rupiah)"] < BB) | \
                          (AtmTransaction["Total amount Withdrawn (Rupiah)"] > BA)]

#visualisasi menggunakan boxplot
plt.figure(figsize = (10, 6))
plt.boxplot(AtmTransaction["Total amount Withdrawn (Rupiah)"], patch_artist = True, boxprops = {"facecolor" : "teal"})
plt.title("Box Plot Of Total Amount Withdrawn (Rupiah)", size = 15)
plt.xlabel("Total Amount Withdrawn (Rupiah)", size = 12)
plt.ylabel("Frequency", size = 12)

plt.show()

#perhitungan no of withdrawals dan nama atm yang terdeteksi fraud
KolomOutliers = ["No Of Withdrawals", "ATM Name"]
DataOutliers = Outliers[KolomOutliers]

#penampilan hasil no of withdrawals dan nama atm yang terdeteksi fraud
print("No Of Withdrawals Dan Nama ATM Yang Terdeteksi Fraud : ")
DataOutliers
```




```
No Of Withdrawals Dan Nama ATM Yang Terdeteksi Fraud :
```

Out[28]:

	No Of Withdrawals	ATM Name
13	309	KK Nagar ATM
153	302	KK Nagar ATM
338	286	KK Nagar ATM
608	244	KK Nagar ATM
758	252	KK Nagar ATM
...	...	...
11469	222	Christ College ATM
11474	244	Christ College ATM
11479	261	Christ College ATM
11499	251	Christ College ATM
11558	257	Christ College ATM

461 rows × 2 columns

Pada kode script di atas, di lakukan proses visualisasi boxplot untuk mendeteksi outlier untuk melihat potensi adanya total jumlah penarikan uang pada bank yang terdeteksi *fraud* dan penyembutan *no of withdrawals* beserta nama atm yang terdeteksi *fraud*. Di pergunakan *library* matplotlib untuk melakukan visualisasi *boxplot*. Terdapat variabel Q1 yang di pergunakan untuk

menuliskan *quartile* pertama dari atribut atau kolom *total amount withdrawn* (rupiah) di kali dengan 0,25 dan variabel Q3 yang di pergunakan untuk menuliskan *quartile* ketiga dari atribut atau kolom *total amount withdrawn* (rupiah) di kali dengan 0,75 kemudian nilai hasil dari variabel Q1 dan Q3 di proses dalam variabel IQR yang berisi selisih antara hasil Q3 di kurang dengan Q1 yang di pergunakan untuk mengukur sebaran data dalam *boxplot*.

Selanjutnya, terdapat perhitungan batas bawah dan batas atas untuk mendeteksi outlier yang di tuliskan dalam variabel BB dan BA. Terdapat variabel BB yang di pergunakan untuk menuliskan batas bawah dengan perhitungan nilai Q1 di kurang dengan 1,5 dan di kali dengan nilai IQR dan variabel BA yang di pergunakan untuk menuliskan batas atas dengan perhitungan Q3 di kurang dengan 1,5 dan di kali dengan nilai IQR.

Kemudian, terdapat proses filter data untuk mendeteksi outlier yang di tuliskan dalam variabel Outliers, yakni data dari dataframe AtmTransasction dengan kolom *total amount withdrawn* (rupiah) kurang dari BB dan menggunakan operator | (or/atau) lebih dari BA akan di simpan ke dalam variabel Outliers. Setelah melakukan serangkaian proses perhitungan tersebut, selanjutnya akan di lakukan proses visualisasi menggunakan boxplot dengan ukuran 10 di kali 6 dari kolom *total amount withdrawn* (rupiah) dengan warna *teal*.

Dari visualisasi boxplot tersebut, dapat di perlihatkan bahwa terdapat komponen dalam *boxplot*, di antaranya yakni *box* (kotak), *whisker* (garis bawah dan garis atas), dan titik-titik yang berada di luar *whisker*. Di dalam box tersebut terdapat quartil pertama (Q1) dan quartil ketiga (Q3) yang menggambarkan interquartile range (IQR). Dapat di peroleh interpretasi bahwa data yang berada dalam dataset ATM-Transaction memiliki distribusi normal, hal ini di buktikan dengan adanya ukuran *box* yang sama antara atas dan bawah setelah di bagi dengan median. Terdapat banyak *whisker* yang terletak di atas *box* hal ini menunjukkan bahwa terdapat beberapa nilai *outliers* di batas atas dari dataset ATM Transaction.

Selanjutnya, di lakukan proses pencarian *no of withdrawals* (transaksi penarikan uang) dan nama ATM yang terdeteksi *fraud*. Terdapat variabel KolomOutliers yang di pergunakan untuk menuliskan *list* yang berisi kolom *no of withdrawals* dan *ATM name*. Kemudian, terdapat variabel DataOutliers yang di pergunakan untuk menuliskan dan penggambilan kolom-kolom yang tercantum dalam KolomOutliers dengan data dari variabel Outliers yang di anggap *fraud* dan berisi data outlier dari kolom-kolom tersebut.

Maka, akan di tampilkan *no of withdrawals* (transaksi penarikan uang) dengan nama ATM yang terdeteksi *fraud* berupa *dataframe* dan bisa di perlihatkan bahwa sebanyak 461 transaksi yang terdeteksi fraud. Seperti no of withdrawals (transaksi penarikan uang) 309 dengan nama atm KK Nagar ATM.

3.3) Menampilkan Semua Data No Of Withdrawals Dengan Nama ATM Yang Terdeteksi Fraud

3.3) Menampilkan Semua Data No Of Withdrawals Dan Nama Yang Terdeteksi Fraud

```
In [52]: for index, row in DataOutliers.iterrows():
             print("No Of Withdrawals :", {row["No Of Withdrawals"]}, "Dengan Nama ATM Yang Terdeteksi Fraud :", {row["ATM Name"]})
```

```
No Of Withdrawals : {309} Dengan Nama ATM Yang Terdeteksi Fraud : {'KK Nagar ATM'}
No Of Withdrawals : {302} Dengan Nama ATM Yang Terdeteksi Fraud : {'KK Nagar ATM'}
No Of Withdrawals : {286} Dengan Nama ATM Yang Terdeteksi Fraud : {'KK Nagar ATM'}
No Of Withdrawals : {244} Dengan Nama ATM Yang Terdeteksi Fraud : {'KK Nagar ATM'}
No Of Withdrawals : {252} Dengan Nama ATM Yang Terdeteksi Fraud : {'KK Nagar ATM'}
```

Pada kode script di atas, di lakukan proses penampilan semua data dari *no of withdrawals* (transaksi penarikan uang) dengan nama ATM yang

terdeteksi fraud. Di lakukan proses looping for dengan variabel *index* yang mengiterasi semua baris dalam dataframe DataOutliers dan *row* yang berisi seluruh data dari dataset tersebut. Maka, akan di tampilkan semua informasi dari *no of withdrawals* dengan nama ATM yang terdeteksi fraud dalam bentuk list.