LAPORAN

MATA KULIAH ANALISIS DATA EKSPLORATIF

"PENYAJIAN DATA MENGGUNAKAN BERBAGAI BENTUK DIAGRAM"

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

PENYELESAIAN STUDI KASUS

1. Daftar Dosen UPN Veteran x Angkatan 2022

Untuk melakukan visualisasi dari data daftar dosen UPN “Veteran” x Angkatan 2022, terdapat beberapa tahapan, yakni melakukan pembuatan *dataframe* dari data dosen UPN “Veteran” x Angkatan 2022, melakukan pengelompokkan data dosen berdasarkan jenjang pendidikan dosen, program studi dosen, dan usia dosen dengan output berupa pembuatan kolom baru.

Selanjutnya terdapat, melakukan penghapusan kolom pendidikan terakhir dosen (karena tidak di gunakan), visualisasi *bar chart* untuk menunjukkan jumlah dosen di per-jenjang pendidikan, visualisasi *line chart* untuk menunjukkan usia rata - rata dosen di jenjang pendidikan tertentu, dan visualisasi *pie c*hart untuk mengetahui berapa banyak proporsi per-jenjang pendidikan dosen.

- Pembuatan Data Frame Dari Data Yang Ada

Melakukan Pembuatan Data Frame Dari Data Yang Ada

```
In [1]: #penggunaan library pandas untuk membuat dataframe
import pandas as pd

#penulisan data untuk dataframe
DataDosen = {"Nomor" : list(range(1, 21)),
             "Tanggal Lahir Dosen" : ["1 Mei 1993", "8 September 1992", "4 Agustus 1978", "1 Januari 1980",
                                      "16 April 1995", "19 Maret 1991", "23 November 1994", "25 Februari 1997",
                                      "20 Oktober 1991", "18 November 1997", "11 Mei 1992", "12 Juni 1982", "11 Juli 1979",
                                      "16 Agustus 1977", "2 November 1970", "8 Desember 1983", "22 Maret 1992", "10 April 1994",
                                      "12 Mei 1996", "11 April 1993"],
          "Pendidikan Terakhir Dosen" : ["S2 Terapan Teknik Informatika, Politeknik Elektronika Negeri Surabaya",
                                         "S2 Teknik Informatika, Institut Teknologi Sepuluh Nopember",
                                         "S3 Biomedical Engineering, University of Rome",
                                         "S3 Informatics, University of WDafashington",
                                         "S3 Biomedik, Institut Teknologi Sepuluh Nopember",
                                         "S3 Informatika, Universitas Telkom",
                                         "S2 Terapan Teknik Elektronika, Politeknik Elektronika Negeri Surabaya",
                                         "S1 Electronical Engineering, University of Manchester",
                                         "S3 Informatics, University of Nigeria", "S1 Biomedik, Universitas Indonesia",
                                         "S3 Teknik Elektro, Institut Teknologi Sepuluh Nopember",
                                         "Post-doctoral Biomedical Engineering, Pusan National University",
                                         "Post-doctoral Electronical Engineering, Hongkong University",
                                         "Post-doctoral Biomedical Engineering, Sydney University",
                                         "Post-doctoral Informatics, Institut Teknologi Bandung",
                                         "S3 Sistem Informasi, Institut Teknologi Sepuluh Nopember",
                                         "S2 Teknik Informatika, Universitas Padjadjaran",
                                         "S2 Teknik Elektro, Universitas Gadjah Mada",
                                         "S2 Teknik Biomedik, Universitas Airlangga",
                                         "S2 Teknik Informatika, Universitas Bina Darma"]}

dfs = pd.DataFrame(DataDosen)
print("Di Tampilkan Data Frame Dari Daftar Data Dosen UPN 'Veteran' :")
dfs
```

```
Di Tampilkan Data Frame Dari Daftar Data Dosen UPN 'Veteran' :
```

Out[1]:

	Nomor	Tanggal Lahir Dosen	Pendidikan Terakhir Dosen
0	1	1 Mei 1993	S2 Terapan Teknik Informatika, Politeknik Elek...
1	2	8 September 1992	S2 Teknik Informatika, Institut Teknologi Sepu...
2	3	4 Agustus 1978	S3 Biomedical Engineering, University of Rome
3	4	1 Januari 1980	S3 Informatics, University of WDafashington
4	5	16 April 1995	S3 Biomedik, Institut Teknologi Sepuluh Nopember
5	6	19 Maret 1991	S3 Informatika, Universitas Telkom
6	7	23 November 1994	S2 Terapan Teknik Elektronika, Politeknik Elek...
7	8	25 Februari 1997	S1 Electronical Engineering, University of Man...
8	9	20 Oktober 1991	S3 Informatics, University of Nigeria
9	10	18 November 1997	S1 Biomedik, Universitas Indonesia

Pada kode script di atas, di lakukan pembuatan *dataframe* dari data-data yang ada pada daftar dosen UPN "Veteran" X Angkatan 2022 menggunakan library pandas yang di misalkan sebagai pd. Terdapat variabel DataDosen yang berisi data-data dan memiliki tiga kolom, yakni nomor (numerik), tanggal lahir dosen (kategorikal), dan pendidikan terakhir dosen (kategorikal). Lalu, data tersebut di simpan ke dalam *dataframe* dfs.

Sehingga, ketika melakukan pemanggilan dfs akan di tampilkan dataframe dari data - data yang ada dan *dataframe* siap di pergunakan untuk analisis lebih lanjut.

- Melakukan Pengelompokkan Data Dosen Dengan Output Berupa Pembuatan Kolom Baru

Melakukan Pengelompokkan Data Dosen Berdasarkan Jenjang Pendidikan Dosen, Program Studi Dosen, Dan Usia Dosen

In [2]:
```
#penggunaan library datetime untuk tanggal lahir dan locale untuk mengatur format
from datetime import datetime
import locale

#mengubah kolom tanggal lahir dosen
locale.setlocale(locale.LC_TIME, "id_ID.UTF8")
dfs["Tanggal Lahir Dosen"] = pd.to_datetime(dfs["Tanggal Lahir Dosen"], format = "%d %B %Y", errors = "coerce")

#menambakan kolom jenjang pendidikan
dfs["Jenjang Pendidikan Dosen"] = dfs["Pendidikan Terakhir Dosen"].str.extract(r'(S\d|Post-doctoral)')[0]

#menambahkan kolom bidang studi dosen
dfs["Program Studi Dosen"] = dfs["Pendidikan Terakhir Dosen"].str.replace(r'(S\d|Post-doctoral)','', regex = False).str.split(','
dfs["Kampus Dosen"] = dfs["Pendidikan Terakhir Dosen"].str.replace(r'(S\d|Post-doctoral)', '', regex = False).str.split(',').str[

#menambahkan kolom usia dosen
dfs["Usia Dosen"] = (datetime.now() - dfs["Tanggal Lahir Dosen"]).astype("<m8[Y]")
dfs["Usia Dosen"] = dfs["Usia Dosen"].astype(int)
dfs
```

Out[2]:

	Nomor	Tanggal Lahir Dosen	Pendidikan Terakhir Dosen	Jenjang Pendidikan Dosen	Program Studi Dosen	Kampus Dosen	Usia Dosen
0	1	1993-05-01	S2 Terapan Teknik Informatika, Politeknik Elek...	S2	S2 Terapan Teknik Informatika	Politeknik Elektronika Negeri Surabaya	30
1	2	1992-09-08	S2 Teknik Informatika, Institut Teknologi Sepu...	S2	S2 Teknik Informatika	Institut Teknologi Sepuluh Nopember	31
2	3	1978-08-04	S3 Biomedical Engineering, University of Rome	S3	S3 Biomedical Engineering	University of Rome	45
3	4	1980-01-01	S3 Informatics, University of WDafashington	S3	S3 Informatics	University of WDafashington	43
4	5	1995-04-16	S3 Biomedik, Institut Teknologi Sepuluh Nopember	S3	S3 Biomedik	Institut Teknologi Sepuluh Nopember	28
5	6	1991-03-19	S3 Informatika, Universitas Telkom	S3	S3 Informatika	Universitas Telkom	32
6	7	1994-11-23	S2 Terapan Teknik Elektronika, Politeknik Elek...	S2	S2 Terapan Teknik Elektronika	Politeknik Elektronika Negeri Surabaya	28
7	8	1997-02-25	S1 Electronical Engineering, University of Man...	S1	S1 Electronical Engineering	University of Manchester	26
8	9	1991-10-20	S3 Informatics, University of Nigeria	S3	S3 Informatics	University of Nigeria	31
9	10	1997-11-18	S1 Biomedik, Universitas Indonesia	S1	S1 Biomedik	Universitas Indonesia	25

Pada kode script di atas, setelah melakukan pembuatan *dataframe* selanjutnya di lakukan proses analisis dengan menambahkan beberapa informasi yang ada di *dataframe*. Meliputi, pengelompokkan jenjang pendidikan dosen, program studi dosen, dan usia dosen. Penggunaan library datetime dan locale di pergunakan untuk memanipulasi dan melakukan pemformatan pada usia dan tanggal lahir dosen.

Kemudian, di buat kolom jenjang pendidikan dosen (S1, S2, S3, Post-doctoral), program studi dosen, kampus asal dosen (alumni), dan usia dosen yang di peroleh dari perhitungan mengurangkan tanggal lahir dengan tanggal saat ini di mana nilai – nilai pada data ini di peroleh dari mengekstrak nilai yang ada di dalam kolom pendidikan terakhir dosen.

- Melakukan Penghapusan Kolom Pendidikan Terakhir Dosen (Tidak Di Gunakan)

Melakukan Penghapusan Kolom Pendidikan Terakhir Dosen

```
In [3]: dfs = dfs.drop(columns = ["Pendidikan Terakhir Dosen"])
        dfs
```

Out[3]:

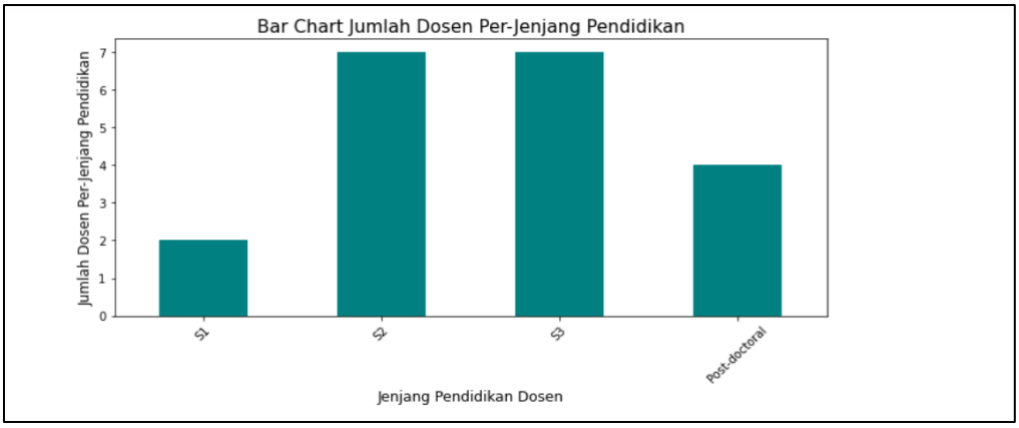
	Nomor	Tanggal Lahir Dosen	Jenjang Pendidikan Dosen	Program Studi Dosen	Kampus Dosen	Usia Dosen
0	1	1993-05-01	S2	S2 Terapan Teknik Informatika	Politeknik Elektronika Negeri Surabaya	30
1	2	1992-09-08	S2	S2 Teknik Informatika	Institut Teknologi Sepuluh Nopember	31
2	3	1978-08-04	S3	S3 Biomedical Engineering	University of Rome	45
3	4	1980-01-01	S3	S3 Informatics	University of WDafashington	43
4	5	1995-04-16	S3	S3 Biomedik	Institut Teknologi Sepuluh Nopember	28
5	6	1991-03-19	S3	S3 Informatika	Universitas Telkom	32
6	7	1994-11-23	S2	S2 Terapan Teknik Elektronika	Politeknik Elektronika Negeri Surabaya	28
7	8	1997-02-25	S1	S1 Electronical Engineering	University of Manchester	26
8	9	1991-10-20	S3	S3 Informatics	University of Nigeria	31
9	10	1997-11-18	S1	S1 Biomedik	Universitas Indonesia	25

Pada kode script di atas, di lakukan proses penghapusan kolom menggunakan dfs.drop pendidikan terakhir dosen, hal ini di karenakan nilai – nilai data yang ada di kolom tersebut, telah di ekstrak dan di pecah ke kolom jenjang pendidikan dosen, program studi dosen, kampus dosen (alumni), dan usia dosen. Kemudian kolom – kolom tersebut siap di visualisasikan untuk mendapatkan *insight.*

- Visualisasi *Bar Chart* Untuk Menunjukkan Jumlah Dosen Di Per-Jenjang Pendidikan

Visualisasi Bar Chart Untuk Menunjukkan Jumlah Dosen Di Per-Jenjang Pendidikan

```
In [6]: #penggunaan library matplotlib untuk visualisasi
        import matplotlib.pyplot as plt

        JumlahDosen = dfs["Jenjang Pendidikan Dosen"].value_counts()
        UrutanDosen = ["S1", "S2", "S3", "Post-doctoral"]
        JumlahDosen = JumlahDosen.loc[UrutanDosen]

        #membuat bar chart
        plt.figure(figsize = (9, 5))
        JumlahDosen.plot(kind = "bar", color = "teal")
        plt.title("Bar Chart Jumlah Dosen Per-Jenjang Pendidikan", fontsize = 15)
        plt.xlabel("Jenjang Pendidikan Dosen", fontsize = 12)
        plt.ylabel("Jumlah Dosen Per-Jenjang Pendidikan", fontsize = 12)
        plt.xticks(rotation = 45, fontsize = 10)
        plt.yticks(fontsize = 10)
        plt.grid(False)

        #visualisasi bar chart
        plt.tight_layout()

        plt.show()
```

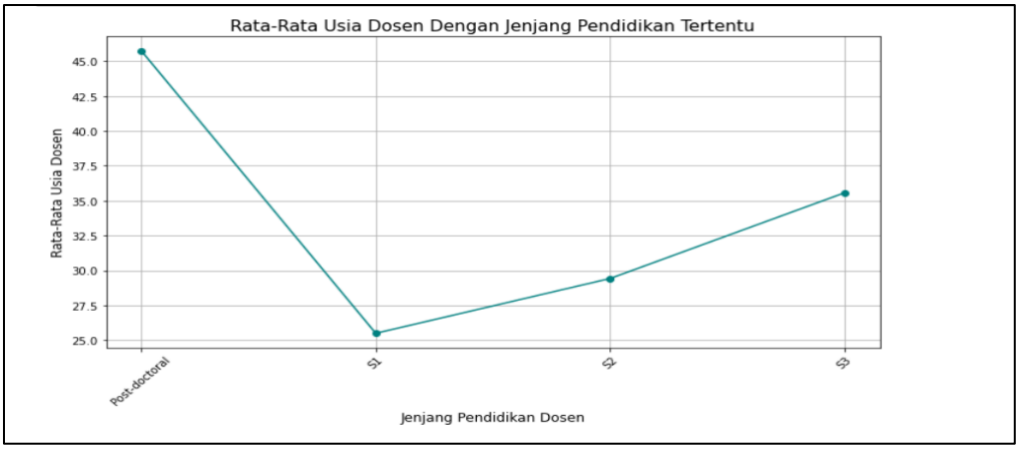


Pada kode script di atas, di lakukan visualisasi dari jenjang pendidikan dosen menggunakan *bar chart* dengan library matplotlib. Lalu, di lakukan perhitungan jumlah dosen perjenjang dengan urutan dosen S1, S2, S3, dan Post-doctoral. Kemudian, akan di tampilkan bar chart dari jumlah dosen per-jenjang pendidikan dengan *insight* pendidikan S1 sebanyak 2, S2 sebanyak 7, S3 sebanyak 7, dan Post-doctoral sebanyak 4.

- Visualisasi *Line Chart* Untuk Menunjukkan Usia Rata - Rata Dosen Di Jenjang Pendidikan Tertentu

Visualisasi Line Chart Untuk Menunjukkan Usia Rata-Rata Dosen Di Jenjang Pendidikan Tertentu

In [7]:
```
#penggunaan library matplotlib untuk visualisasi
import matplotlib.pyplot as plt

#menghitung rata-rata usia untuk setiap jenjang pendidikan
UsiaPerjenjang = dfs.groupby("Jenjang Pendidikan Dosen")["Usia Dosen"].mean()
UrutanDosen = ["S1", "S2", "S3", "Post-doctoral"]
JumlahDosen = UsiaPerjenjang.loc[UrutanDosen]

#usia rata-rata sebagai sumbu x
Jenjang = UsiaPerjenjang.index

#usia rata-rata sebagai sumbu y
UsiaDosen = UsiaPerjenjang.values

#visualisasi line chart
plt.figure(figsize = (10, 6))
plt.plot(Jenjang, UsiaDosen, color = "teal", marker = "o", linestyle = "-")

plt.title("Rata-Rata Usia Dosen Dengan Jenjang Pendidikan Tertentu", fontsize = 15)
plt.xlabel("Jenjang Pendidikan Dosen", fontsize = 12)
plt.ylabel("Rata-Rata Usia Dosen", fontsize = 12)
plt.grid(True)
plt.xticks(rotation = 45)
plt.tight_layout()

plt.show()
```

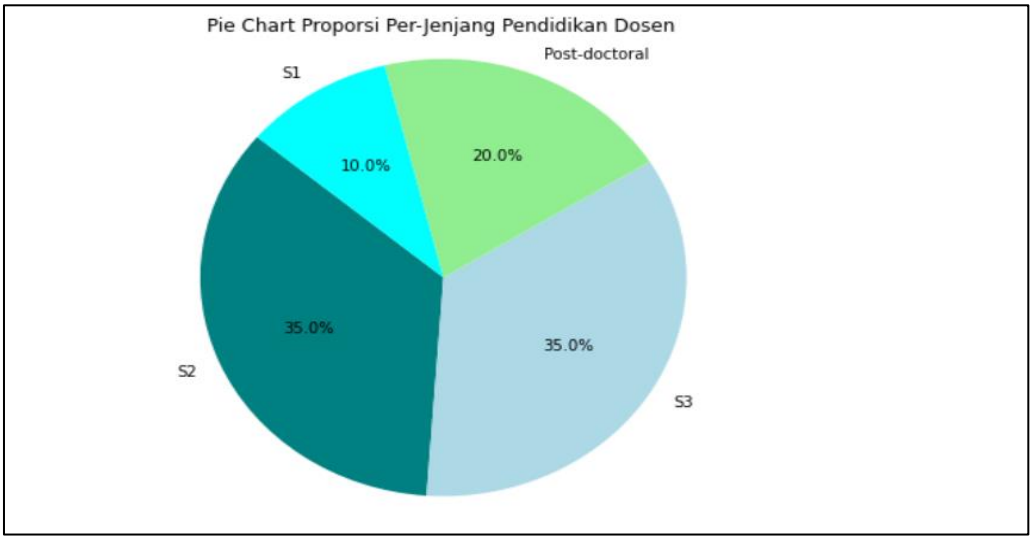


Pada kode script di atas, di lakukan visualisasi dari jenjang pendidikan dosen dengan rata - rata usia dosen menggunakan *line chart* dengan menggunakan library matplotlib. Kemudian, di lakukan perhitungan rata-rata usia per-jenjang pendidikan dosen dan nilainya di simpan dalam variabel UsiaPerjenjang. Di peroleh *insight* bahwa dosen Post-doctoral memiliki rata-rata usia tertinggi di banding dengan lainnya, di ikuti S3 dan S2, sedangkan dosen S1 memiliki rata-rata usia terendah. Dapat di ketahui bahwa, semakin tinggi rata – rata usia dosen maka semakin tinggi pula jenjang pendidikan dosen tersebut.

- Visualisasi *Pie Chart* Untuk Mengetahui Berapa Banyak Proporsi Per-Jenjang Pendidikan Dosen

Visualisasi Pie Chart Untuk Mengetahui Berapa Banyak Proporsi Per-Jenjang Pendidikan Dosen (S1, S2, S3, Dan Pos-doctoral)

```
In [8]: #penggunaan library matplotlib untuk visualisasi
        import matplotlib.pyplot as plt

        #perhitungan presentase dan pemberian warna pie chart untuk proporsi per-jenjang pendidikan dosen
        JumlahDosen = dfs["Jenjang Pendidikan Dosen"].value_counts()
        warna = ["teal", "lightblue", "lightgreen", "cyan", "blue"]
        presentase = JumlahDosen / JumlahDosen.sum()

        #visualisasi pie chart
        plt.figure(figsize = (6, 6))
        plt.pie(JumlahDosen.values, labels = JumlahDosen.index, autopct = "%1.1f%%", startangle = 140, colors = warna)
        plt.title("Pie Chart Proporsi Per-Jenjang Pendidikan Dosen")
        plt.axis("equal")

        plt.show()
```



Pada kode script di atas, di lakukan visualisasi proporsi dari kolom jenjang pendidikan dosen menggunakan *pie chart* dengan presentase persen dengan menggunakan library matplotlib. Kemudian, di lakukan perhitungan presentase dengan menghitung banyak kategori di jenjang pendidikan dosen

kemudian nilai tersebut di simpan dalam variabel JumlahDosen, selanjutnya akan di lakukan perhitungan presentase dengan membagi JumlahDosen/JumlahDosen.sum() dan pemberian warna pie chart menggunakan warna *teal, lightblue, lightgreen, cyan, dan blue*.

Di peroleh *insight* bahwa dosen dengan pendidikan terakhir S1 memiliki proporsi sebesar 10%, S2 sebesar 35%, S3 sebesar 35%, dan Post-doctoral sebesar 20%. Dapat di ketahui bahwa mayoritas pendidikan terakhir dosen dari data dosen UPN "Veteran" adalah di jenjang S2 dan S3.

2. Daftar Pasien Diabetes Berdasarkan Faktor Gula Darah, Berat Badan, Dan Tinggi Badan

Untuk melakukan visualisasi dari data daftar pasien diabetes berdasarkan faktor gula darah, berat badan, dan tinggi badan, terdapat beberapa tahapan, yakni melakukan pembuatan *dataframe* dari data yang ada, pengelompokkan berat badan kurang (*underweight*), normal, berat badan berlebih (*overweight*), berat badan sangat berlebih (obesitas) menurut perhitungan BMI, pengelompokkan gula darah dengan kategori normal, prediabetes, dan diabetes di pembuatan kolom baru.

Visualisasi *bar chart* untuk mengetahui banyaknya jumlah pasien pada kategori berat badan dan kategori gula darah, visualisasi *pie chart* untuk melihat proporsi masing-masing kategori berat badan dan gula darah, dan visualisasi *scatter plot* untuk melihat apakah keduanya memiliki korelasi (hubungan).

- Pembuatan Data Frame Dari Data Yang Ada

Pembuatan Data Frame Dari Data Yang Ada

In [1]:
```
#penggunaan library pandas untuk membuat dataframe
import pandas as pd

#penulisan data untuk data frame
DataPasienDiabetes = {"Nomor" : list(range(1, 21)),
                      "Tinggi Badan (Cm)": ["165", "150", "154", "149", "166", "156", "148", "154", "163", "152", "166",
                                            "151", "150", "146", "163", "151", "146", "156", "160", "147"],
    "Berat Badan (Kg)": ["50", "95", "52", "43", "96", "67", "42", "64", "74", "98", "51", "96", "48", "40",
                         "93", "61", "40", "66", "71", "95"],
    "Gula Darah (Puasa)": ["4.9", "5.7", "4.5", "8.14", "8.05", "4.2", "8.14", "4.89", "5.08", "5.6", "4.7",
                           "5.8", "4.3", "8.19", "8.08", "4.3", "8.10", "4.92", "5.05", "5.3"]}

dfs = pd.DataFrame(DataPasienDiabetes)
print("Ditampilkan Data Frame Data Pasien Diabetes : ")
dfs
```

Ditampilkan Data Frame Data Pasien Diabetes :

Out[1]:

	Nomor	Tinggi Badan (Cm)	Berat Badan (Kg)	Gula Darah (Puasa)
0	1	165	50	4.9
1	2	150	95	5.7
2	3	154	52	4.5
3	4	149	43	8.14
4	5	166	96	8.05
5	6	156	67	4.2
6	7	148	42	8.14
7	8	154	64	4.89
8	9	163	74	5.08
9	10	152	98	5.6

Pada kode script di atas, di lakukan pembuatan *dataframe* dari data - data yang ada pada daftar pasien diabetes menggunakan library pandas yang di misalkan sebagai pd. Terdapat variabel DataPasienDiabetes yang berisi data – data dan memiliki empat kolom, yakni kolom nomor (kategorikal), tinggi badan (numerik), berat badan (numerik), dan gula darah (numerik). Lalu variabel tersebut di simpan sebagai dfs.

Sehingga, ketika melakukan pemanggilan dfs akan di tampilkan *dataframe* dari data - data yang ada dan *dataframe* siap di pergunakan untuk analisis lebih lanjut.

- Pengelompokkan Berat Badan Kurang (*underweight*), Normal, Berat Badan Berlebih (*overweight*), Berat Badan Sangat Berlebih (obesitas) Menurut Perhitungan BMI

Pengelompokkan Berat Badan Kurang (Underweight), Normal, Dan Berat Badan Berlebih (Overweight) Atau Berat Badan Sangat Berlebih (Obesitas) Menurut Perhitungan BMI

In [2]:
```
#menghitung kategori sesuai BMI
dfs["Tinggi Badan (M)"] = dfs["Tinggi Badan (Cm)"].astype(float) / 100
dfs["Berat Badan (Kg)"] = dfs["Berat Badan (Kg)"].astype(float)
dfs["BMI"] = dfs["Berat Badan (Kg)"] / (dfs["Tinggi Badan (M)"] ** 2)

#membuat kategori berat badan
def kategori_bb(bmi):
    if bmi < 18.5:
        return "Kurang (Underweight)"
    elif 18.5 <= bmi < 25:
        return "Normal"
    elif 25 <= bmi < 30:
        return "Berlebih (Overweight)"
    else:
        return "Sangat Berlebih (Obesitas)"

#membuat kolom baru pada data frame
dfs["Kategori Berat Badan"] = dfs["BMI"].apply(kategori_bb)

#menghapus kolom yang tidak di perlukan
dfs = dfs.drop(["Tinggi Badan (M)"], axis = 1)
dfs
```

Out[2]:

	Nomor	Tinggi Badan (Cm)	Berat Badan (Kg)	Gula Darah (Puasa)	BMI	Kategori Berat Badan
0	1	165	50.0	4.9	18.365473	Kurang (Underweight)
1	2	150	95.0	5.7	42.222222	Sangat Berlebih (Obesitas)
2	3	154	52.0	4.5	21.926126	Normal
3	4	149	43.0	8.14	19.368497	Normal
4	5	166	96.0	8.05	34.838148	Sangat Berlebih (Obesitas)
5	6	156	67.0	4.2	27.531229	Berlebih (Overweight)
6	7	148	42.0	8.14	19.174580	Normal
7	8	154	64.0	4.89	26.986001	Berlebih (Overweight)
8	9	163	74.0	5.08	27.852008	Berlebih (Overweight)
9	10	152	98.0	5.6	42.416898	Sangat Berlebih (Obesitas)

Pada kode script di atas, setelah melakukan pembuatan *dataframe* selanjutnya di lakukan proses analisis dengan melakukan pengelompokkan berat badan sesuai dengan perhitungan BMI. Untuk melakukan perhitungan BMI di buat kolom tinggi badan (m) dengan mengambil nilai di kolom tinggi badan (cm) dan membaginya dengan 100. Kemudian, melakukan pengubahan tipe data menjadi float di kolom berat badan (kg). Perhitungan nilai BMI menggunakan pembagian berat badan (kg) di bagi dengan tinggi badan (m) kuadrat. Nilai – nilai yang di peroleh dari perhitungan tersebut akan di simpan di dalam kolom BMI.

Di buat kondisi *if else* untuk melakukan kategori ke dalam rendah (*underweight*) ketika BMI kurang 18.5, normal ketika BMI di rentang 18.5 – 25, berlebih (*overweight*) ketika BMI di rentang 25 – 30, dan sangat berlebih (obesitas) ketika BMI di rentang lebih dari tersebut. Selanjutnya di lakukan penghapusan kolom tinggi badan (m) karena tidak di gunakan. Kemudian, kolom BMI dan kategori berat badan akan di tambahkan ke dalam *dataframe.*

- Pengelompokkan Gula Darah Dengan Kategori Normal, Prediabetes, Dan Diabetes Di Pembuatan Kolom Baru

Pengelompokkan Gula Darah Dengan Kategori Normal, Prediabetes, Dan Diabetes Di Pembuatan Kolom Baru

In [3]:
```
#membuat kategori gula darah
def kategori_gula(gd):
    gd_float = float(gd)
    if gd_float < 5.6:
        return "Normal"
    elif 5.6 <= gd_float < 6.9:
        return "Prediabetes"
    else:
        return "Diabetes"

#membuat kolom baru pada data frame
dfs["Kategori Gula Darah"] = dfs["Gula Darah (Puasa)"].apply(kategori_gula)
dfs
```

Out[3]:

	Nomor	Tinggi Badan (Cm)	Berat Badan (Kg)	Gula Darah (Puasa)	BMI	Kategori Berat Badan	Kategori Gula Darah
0	1	165	50.0	4.9	18.365473	Kurang (Underweight)	Normal
1	2	150	95.0	5.7	42.222222	Sangat Berlebih (Obesitas)	Prediabetes
2	3	154	52.0	4.5	21.926126	Normal	Normal
3	4	149	43.0	8.14	19.368497	Normal	Diabetes
4	5	166	96.0	8.05	34.838148	Sangat Berlebih (Obesitas)	Diabetes
5	6	156	67.0	4.2	27.531229	Berlebih (Overweight)	Normal
6	7	148	42.0	8.14	19.174580	Normal	Diabetes
7	8	154	64.0	4.89	26.986001	Berlebih (Overweight)	Normal
8	9	163	74.0	5.08	27.852008	Berlebih (Overweight)	Normal
9	10	152	98.0	5.6	42.416898	Sangat Berlebih (Obesitas)	Prediabetes

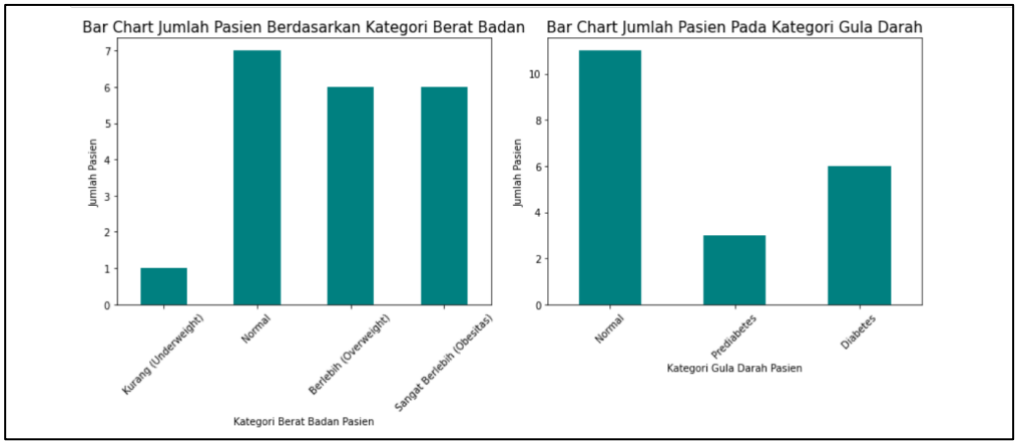
Pada kode script di atas, di lakukan pembuatan kolom kategori gula darah dengan menggunakan *define function* kategori gula. Di buat kondisi *if-else* menggunakan tipe data float. Jika normal nilai gula darah kurang dari 5,6,

jika prediabetes gula darah berada di rentang 5,6 dan kurang dari 6,9, dan jika di luar rentang tersebut maka gula darah adalah kategori diabetes. Kemudian kolom kategori gula darah akan di tambahkan di *dataframe*.

- Visualisasi *Bar Chart* Untuk Mengetahui Banyaknya Jumlah Pasien Pada Kategori Berat Badan Dan Kategori Gula Darah

Visualisasi Bar Chart Untuk Mengetahui Banyaknya Jumlah Pasien Pada Kategori Berat Badan Dan Kategori Gula Darah

In [4]:
```
#penggunaan library matplotlib untuk visualisasi
import matplotlib.pyplot as plt

#jumlah setiap kategori berat badan
JumlahKategoriBB = dfs["Kategori Berat Badan"].value_counts().sort_index()
UrutanBB = ["Kurang (Underweight)", "Normal", "Berlebih (Overweight)", "Sangat Berlebih (Obesitas)"]
JumlahKategoriBB = JumlahKategoriBB.loc[UrutanBB]

#jumlah setiap kategori gula darah
JumlahKategoriGD = dfs["Kategori Gula Darah"].value_counts().sort_index()
UrutanGD = ["Normal", "Prediabetes", "Diabetes"]
JumlahKategoriGD = JumlahKategoriGD.loc[UrutanGD]

#membuat dua subplot
plt.figure(figsize = (12, 6))

#bar chart kategoti berat badan
plt.subplot(1, 2, 1)
JumlahKategoriBB.plot(kind = "bar", color = "teal")
plt.title("Bar Chart Jumlah Pasien Berdasarkan Kategori Berat Badan", fontsize=15)
plt.xlabel("Kategori Berat Badan Pasien")
plt.ylabel("Jumlah Pasien")
plt.xticks(rotation = 45, fontsize = 10)
plt.grid(False)

#bar chart kategori gula darah
plt.subplot(1, 2, 2)
JumlahKategoriGD.plot(kind = "bar", color = "teal")
plt.title("Bar Chart Jumlah Pasien Pada Kategori Gula Darah", fontsize = 15)
plt.xlabel("Kategori Gula Darah Pasien")
plt.ylabel("Jumlah Pasien")
plt.xticks(rotation = 45, fontsize = 10)
plt.grid(False)
plt.tight_layout()
plt.show()
```



Pada kode script di atas, di lakukan visualisasi *bar chart* untuk mengetahui banyaknya jumlah pasien pada kategori berat badan dan kategori gula darah dengan menggunakan library matplotlib. Di lakukan perhitungan jumlah kategori berat badan menggunakan value.counts() kemudian melakukan pengurutan berat badan di mulai dari kurang (*underweight*), normal, berlebih (*overweight*), dan sangat berlebih (*obesitas*) dengan harapan agar saat visualisasi dapat di tampilkan urut.

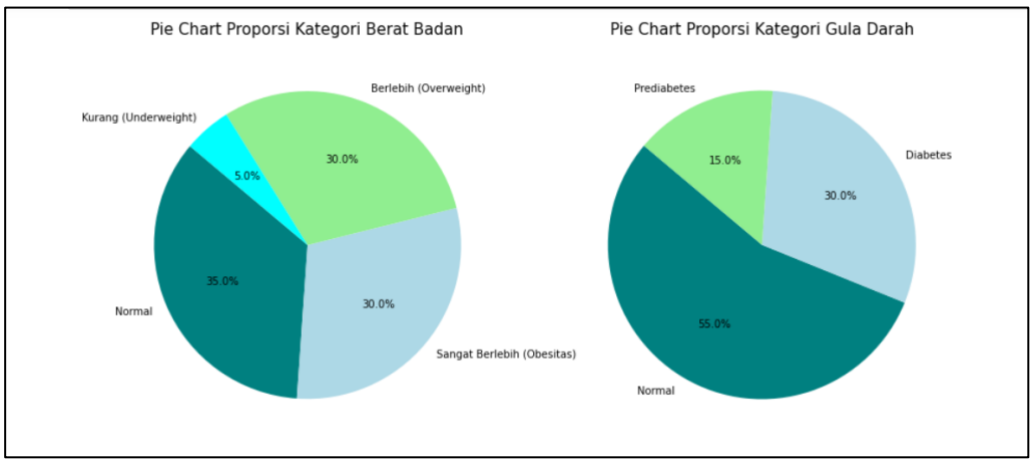
Kemudian di lanjut dengan melakukan perhitungan jumlah kategori gula darah menggunakan value.counts() dan melakukan pengurutan gula darah di mulai dari normal, prediabetes, dan diabetes. Dan di lakukan visualisasi dengan dua *bar chart* yang berdampingan. Untuk *bar chart* pertama terdapat bar chart jumlah pasien berdasarkan kategori berat badan. Dan di peroleh *insight* bahwa terdapat bahwa pasien yang memiliki kategori berat badan kurang (*underweight*) satu orang, normal tujuh orang, berlebih (*overweight*) enam orang, dan sangat berlebih (obesitas) enam orang.

Untuk *bar chart* kedua terdapat *bar chart* kategori jumlah darah pasien. Dan di peroleh *insight* bahwa terdapat pasien yang memiliki kategori gula darah normal 11 orang, prediabetes 3 orang, dan diabetes 6 orang.

- Visualisasi *Pie Chart* Untuk Melihat Proporsi Masing - Masing Kategori Berat Badan Dan Gula Darah

Visualisasi Pie Chart Untuk Melihat Proporsi Masing-Masing Kategori Berat Badan Dan Kategori Gula Darah

In [6]:
```
#penggunaan library matplotlib untuk visualisasi
import matplotlib.pyplot as plt

#perhitungan presentase dan pembuatan warna pie chart kategori berat badan
JumlahKategoriBB = dfs["Kategori Berat Badan"].value_counts()
WarnaBB = ["teal", "lightblue", "lightgreen", "cyan", "blue"]
presentase = JumlahKategoriBB / JumlahKategoriBB.sum()

#perhitungan presentase dan pembuatan warna pie chart untuk kategori gula darah
JumlahKategoriGD = dfs["Kategori Gula Darah"].value_counts()
WarnaGD = ["teal", "lightblue", "lightgreen", "cyan", "blue"]
presentase = JumlahKategoriGD / JumlahKategoriGD.sum()

#membuat dua subplot
plt.figure(figsize = (12, 6))

#pie chart untuk proporsi kategori berat badan
plt.subplot(1, 2, 1)
plt.pie(JumlahKategoriBB.values, labels = JumlahKategoriBB.index, autopct = "%1.1f%%", startangle = 140, colors = WarnaBB)
plt.title("Pie Chart Proporsi Kategori Berat Badan", fontsize = 15)
plt.axis("equal")

#pie chart untuk proporsi kategori berat badan
plt.subplot(1, 2, 2)
plt.pie(JumlahKategoriGD.values, labels = JumlahKategoriGD.index, autopct = "%1.1f%%", startangle = 140, colors = WarnaGD)
plt.title("Pie Chart Proporsi Kategori Gula Darah", fontsize = 15)
plt.axis("equal")

plt.tight_layout()
plt.show()
```



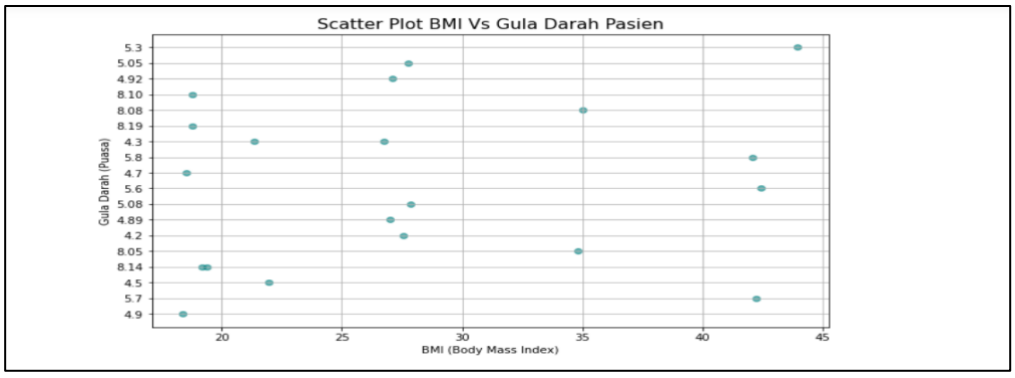
Pada kode script di atas, di lakukan visualisasi *pie chart* untuk melihat proporsi masing – masing kategori berat badan dan gula darah dengan presentase persen dengan menggunakan library matplotlib. Kemudian di lakukan perhitungan presentase dengan perhitungan pertama, yakni menghitung banyak kategori berat badan kemudian nilai tersebut di simpan dalam variabel JumlahKategoriBB dengan perhitungan presentasi JumlahKategoriBB /JumlahKategoriBB.sum() dan pemberian warna *teal, lightblue, cyan, blue*. Untuk perhitungan kedua, yakni menghitung banyak kategori banyak kategori gula darah kemudian nilai tersebut di simpan dalam variabel JumlahKategoriGD dengan perhitungan presentasi JumlahKategoriGD /JumlahKategoriGD.sum() dan pemberian warna *teal, lightblue, cyan, blue*.

Di peroleh insight untuk pie chart proporsi kategori berat bedan, terdapat pasien diabetes yang memiliki berat badan kurang (*underweight*) sebesar 5%, normal 35% , berlebih (*overweight*) sebesar 30%, dan sangat berlebih (obesitas) 30%. Untuk pie chart proporsi kategori gula darah, terdapat pasien diabetes yang memiliki gula darah normal sebesar 55%, prediabetes 15%, dan diabetes 30%. Dapat di katakan, bahwa mayoritas pasien memiliki berat badan normal dan gula darah normal.

- Visualisasi *Scatter Plot* Untuk Kolom BMI Dan Kolom Gula Darah Untuk Melihat Apakah Keduanya Memiliki Korelasi (Hubungan)

Visualisasi Scatter Plot Untuk Kolom BMI Dan Kolom Gula Darah Untuk Melihat Apakah Keduanya Memiliki Korelasi (Hubungan)

```
In [7]: import matplotlib.pyplot as plt
        import numpy as np

        # Visualisasi scatter plot
        plt.figure(figsize=(10, 6))
        plt.scatter(dfs["BMI"], dfs["Gula Darah (Puasa)"], color = "teal", alpha = 0.6)
        plt.title("Scatter Plot BMI Vs Gula Darah Pasien", fontsize = 15)
        plt.xlabel("BMI (Body Mass Index)", fontsize = 10)
        plt.ylabel("Gula Darah (Puasa)", fontsize = 10)
        plt.grid(True)

        plt.show()
```



Pada kode script di atas, di lakukan visualisasi *scatter plot* untuk melihat ada atau tidaknya hubungan (korelasi) antara kolom BMI dan kolom gula darah dengan menggunakan library matplotlib. Di peroleh *insight* tidak ada hubungan linier positif antara BMI dan gula darah, dengan maksud tidak ada yang membuktikan jika BMI semakin tinggi, maka gula darah semakin tinggi (pasien lebih mudah terkena diabetes).

3. Daftar Kasus Pasien Terkonfirmasi Covid-19 Harian

Untuk melakukan visulisasi dari data daftar kasus pasien terkonfirmasi *Covid-19* harian, terdapat beberapa tahapan, yakni melakukan pembuatan *dataframe* dari data yang ada, di lanjutkan dengan melakukan perhitungan pertumbuhan kasus *Covid-19* harian, melakukan visualisasi dari daftar kasus pasien terkonfirmasi *Covid-19* yang ada menggunakan *line chart* (diagram garis), dan visualisasi *scatter plot* unruk kolom jumlah kasus terkonfirmasi *Covid-19* dan pertumbuhan harian kasus *Covid-19* .

- Melakukan Pembuatan Data Frame Dari Data Yang Ada

Melakukan Pembuatan Data Frame Dari Data Yang Ada

In [1]:
```
#penggunaan library pandas untuk membuat data frame
import pandas as pd

#penulisan data untuk data frame
DataPasienCovid = {"Nomor" : list(range(1, 31)),
                   "Tanggal" : ["02-Mar-20", "03-Mar-20", "04-Mar-20", "05-Mar-20", "06-Mar-20", "07-Mar-20", "08-Mar-20",
                                "09-Mar-20", "10-Mar-20", "11-Mar-20", "12-Mar-20", "13-Mar-20", "14-Mar-20", "15-Mar-20",
                                "16-Mar-20", "17-Mar-20", "18-Mar-20", "19-Mar-20", "20-Mar-20", "21-Mar-20", "22-Mar-20",
                                "23-Mar-20", "24-Mar-20", "25-Mar-20", "26-Mar-20", "27-Mar-20", "28-Mar-20", "29-Mar-20",
                                "30-Mar-20", "31-Mar-20"],
                   "Jumlah Kasus Terkonfirmasi" : ["2", "0", "0", "0", "2", "0", "2", "13", "8", "7", "0", "35", "27", "21",
                                                   "17", "38", "55", "82", "60", "81", "64", "65", "106", "105", "103", "153",
                                                   "109", "130", "129", "114"]}

dfs = pd.DataFrame(DataPasienCovid)
print("Ditampilkan Data Frame Data Pasien Kasus Covid-19 : ")
dfs
```

```
Ditampilkan Data Frame Data Pasien Kasus Covid-19 :
```

Out[1]:

	Nomor	Tanggal	Jumlah Kasus Terkonfirmasi
0	1	02-Mar-20	2
1	2	03-Mar-20	0
2	3	04-Mar-20	0
3	4	05-Mar-20	0
4	5	06-Mar-20	2
5	6	07-Mar-20	0
6	7	08-Mar-20	2
7	8	09-Mar-20	13
8	9	10-Mar-20	8
9	10	11-Mar-20	7

Pada kode script di atas, di lakukan pembuatan *dataframe* dari data-data yang ada pada data pasien kasus *Covid-19* harian menggunakan library pandas yang di misalkan sebagai pd. Terdapat variabel DataPasienCovid yang berisi

data-data dan memiliki tiga kolom, yakni kolom nomor (numerik), tanggal (kategorikal) dan kolom jumlah kasus terkonfirmasi (numerik). Lalu, data tersebut di simpan ke dalam *dataframe* dfs.

Sehingga, ketika melakukan pemanggilan dfs akan di tampilkan *dataframe* dari data-data yang ada dan *dataframe* siap di pergunakan untuk proses analisis data dan visualisasi lebih lanjut.

- Melakukan Perhitungan Pertumbuhan Kasus Covid-19 Harian

Melakukan Perhitungan Pertumbuhan Kasus Covid-19 Harian

In [2]:
```
PertumbuhanHarian = [int(dfs["Jumlah Kasus Terkonfirmasi"][0])]
for k in range(1, len(dfs)):
    #perhitungan pertumbuhan harian dengan mengurangkan nilai indeks dari kolom jumlah kasus terkonfirmasi sekarang - sebelumnya
    PertumbuhanHarian.append(int(dfs["Jumlah Kasus Terkonfirmasi"][k]) - int(dfs["Jumlah Kasus Terkonfirmasi"][k - 1]))

dfs["Pertumbuhan Harian Kasus Covid-19"] = PertumbuhanHarian

print("Di Tampilkan Data Frame Dengan Pertumbuhan Harian Covid-19 : ")
dfs
```

```
Di Tampilkan Data Frame Dengan Pertumbuhan Harian Covid-19 :
```

Out[2]:

	Nomor	Tanggal	Jumlah Kasus Terkonfirmasi	Pertumbuhan Harian Kasus Covid-19
0	1	02-Mar-20	2	2
1	2	03-Mar-20	0	-2
2	3	04-Mar-20	0	0
3	4	05-Mar-20	0	0
4	5	06-Mar-20	2	2
5	6	07-Mar-20	0	-2
6	7	08-Mar-20	2	2
7	8	09-Mar-20	13	11
8	9	10-Mar-20	8	-5
9	10	11-Mar-20	7	-1

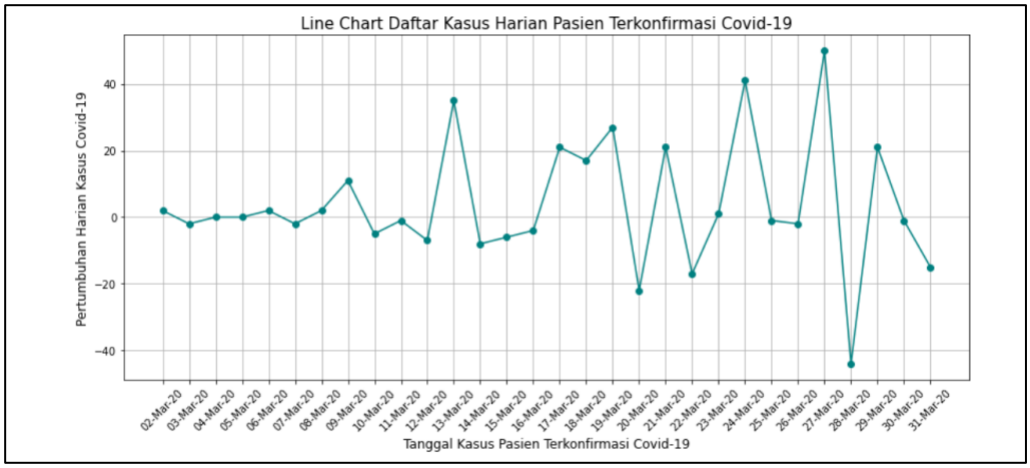
Pada kode script di atas, setelah melakukan pembuatan *dataframe*, selanjutnya akan di lakukan perhitungan pertumbuhan kasus *Covid-19*. Pada kode script di atas, terdapat variabel PertumbuhanHarian yang di inisialisasi dengan nilai pertama dari kolom jumlah kasus terkonfirmasi yang terdapat di indeks ke 0 dengan tipe data menjadi integer. Selanjutnya, di lakukan perulangan dari indeks ke 1 sebanyak jumlah baris *dataframe*.

Lalu, di lakukan perhitungan pertumbuhan harian kasus *Covid-19* dengan cara mengurangkan nilai indeks dari kolom jumlah kasus terkonfirmasi sekarang – nilai indeks dari kolom jumlah kasus terkonfirmasi sebelumnya, dan hasil tersebut akan di simpan dalam variabel PertumbuhanHarian. Selanjutnya, di buat kolom baru pada *dataframe* dengan nama pertumbuhan harian kasus *Covid-19* yang berisi nilai-nilai yang terdapat pada variabel PertumbuhanHarian. Kemudian, di tampilkan *dataframe* dengan 3 kolom, yakni tanggal, jumlah kasus terkonfirmasi, dan pertumbuhan harian kasus *Covid-19*.

- Melakukan Visualisasi *Line Chart* Untuk Daftar Kasus Pasien Terkonfirmasi Covid-19

Melakukan Visualisasi Line Chart (Diagram Garis) Untuk Daftar Kasus Pasien Terkonfirmasi Covid-19

```
In [3]: #penggunaan library matplotlib untuk visualisasi
        import matplotlib.pyplot as plt

        #visualisasi line chart
        plt.figure(figsize = (12, 6))
        plt.plot(dfs["Tanggal"], dfs["Pertumbuhan Harian Kasus Covid-19"], marker = "o", color = "teal")
        plt.title("Line Chart Daftar Kasus Harian Pasien Terkonfirmasi Covid-19", fontsize = 15)
        plt.xlabel("Tanggal Kasus Pasien Terkonfirmasi Covid-19", fontsize = 12)
        plt.ylabel("Pertumbuhan Harian Kasus Covid-19", fontsize = 12)
        plt.xticks(rotation = 45, fontsize = 10)
        plt.yticks(fontsize = 10)
        plt.grid(True)

        plt.tight_layout()
        plt.show()
```



Pada kode script di atas, setelah mendapatkan nilai-nilai pertumbuhan harian kasus *Covid-19* selanjutnya akan di lakukan visualisasi dari data yang ada. Dalam visualisasi kali ini di pergunakan kolom tanggal dan pertumbuhan harian kasus Covid-19 dalam sebuah *line chart* (diagram garis). Dengan tanggal sebagai sumbu x dan pertumbuhan harian kasus *Covid-19* sebagai sumbu y. Dari *line chart* (diagram garis) tersebut bisa di peroleh insight mengenai bagaimana pertumbuhan awal, puncak pertumbuhan, tren umum, dan periode kritis dari data pertumbuhan harian kasus *Covid-19* di bulan maret tahun 2020.

Dapat di perlihatkan bahwa terdapat pola penurunan dan peningkatan dari pertumbuhan harian kasus *Covid-19*. Di mulai dengan pada tanggal 02 maret 2020 hingga 11 maret 2020 (minggu pertama) terjadi penurunan dan peningkatan di rentang -2 sampai dengan 2. Selanjutnya, pada tanggal 12 maret 2020 hingga 18 Maret 2020 (minggu kedua) terjadi pola peningkatan secara tajam, yakni tepatnya di tanggal 13 maret 2020 terjadi peningkatan sebanyak 35 dan di tanggal 17 maret terjadi peningkatan sebanyak 21.

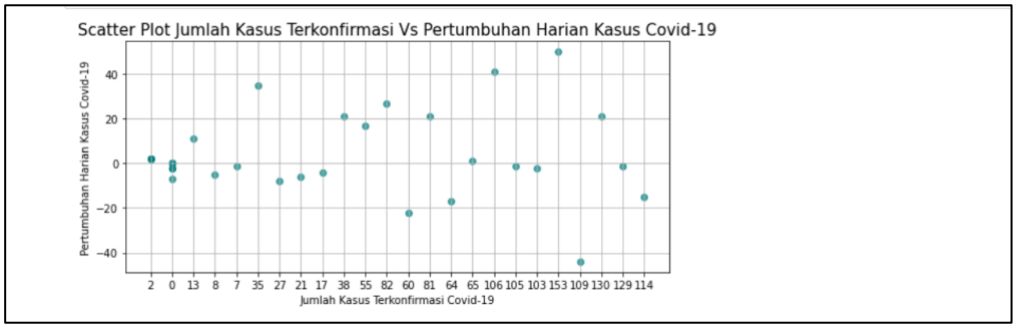
Di tanggal 19 maret 2020 hingga 25 maret 2020 terjadi pola peningkatan dan penurunan secara tajam, yakni peningkatan di tanggal 19 maret 2020 sebesar 27 dan di tanggal 24 maret 2020 (minggu ketiga) sebesar 41. Dan yang terakhir di tanggal 25 maret 2020 hingga 31 Maret 2020 (minggu keempat) terjadi peningkatan dan penurunan secara tajam, yakni peningkatan di tanggal 27 maret 2020 sebesar 50 dan penurunan tajam di tanggal 28 maret 2020 sebesar 44.

Sehingga dapat di simpulkan bahwa selama bulan maret 2020, terjadi pola tren peningkatan secara tajam di 27 maret 2020 dan penurunan secara tajam di 28 maret 2020.

- Visualisasi *Scatter Plot* Untuk Kolom Jumlah Kasus Terkonfirmasi Covid-19 Dan Pertumbuhan Harian Kasus Covid-19

Visualisasi Scatter Plot Untuk Kolom Jumlah Kasus Terkonfirmasi Covid-19 Dan Pertumbuhan Harian Kasus Covid-19

In [11]:
```
#penggunaan library matplotlib untuk visualisasi
import matplotlib.pyplot as plt

#visualisasi scatter plot
plt.figure(figsize = (8, 4))
plt.scatter(dfs["Jumlah Kasus Terkonfirmasi"], dfs["Pertumbuhan Harian Kasus Covid-19"], color = "teal", alpha = 0.7)
plt.title("Scatter Plot Jumlah Kasus Terkonfirmasi Vs Pertumbuhan Harian Kasus Covid-19", fontsize = 15)
plt.xlabel("Jumlah Kasus Terkonfirmasi Covid-19", fontsize = 10)
plt.ylabel("Pertumbuhan Harian Kasus Covid-19", fontsize = 10)
plt.grid(True)
plt.tight_layout()

plt.show()
```



Pada kode script di atas, di lakukan visualisasi *scatter plot* untuk melihat ada atau tidaknya hubungan (korelasi) antara kolom jumlah kasus terkonfirmasi *Covid-19* dengan kolom pertumbuhan harian kasus *Covid-19* dengan menggunakan library matplotlib. Di peroleh insight bahwa terdapat hubungan positif antar kedua kolom tersebut, bisa di perlihatkan jika jumlah kasus terkonfirmasi *Covid-19* meningkat, maka pertumbuhan harian kasus *Covid-19* juga ikut meningkat.