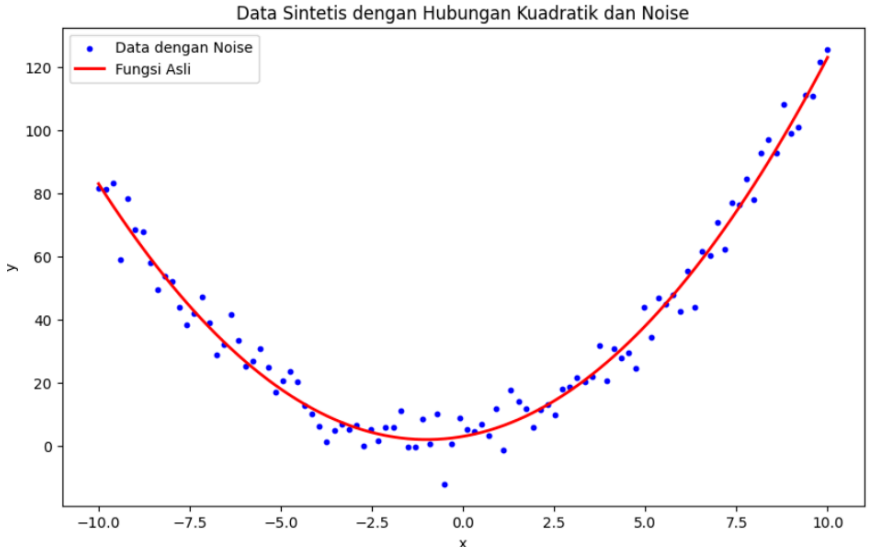
Muhammad Nouval Habibie
2211521020
Akuisisi data - b

1. SymPy
    a. code

```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt

# Mendefinisikan variabel simbolik dan ekspresi matematika
x = sp.symbols('x')  # Membuat variabel simbolik 'x' menggunakan SymPy
expression = x**2 + 2*x + 3  # Mendefinisikan ekspresi kuadratik: x^2 + 2x + 3

# Membuat data sintetis
x_values = np.linspace(-10, 10, 100)  # Menghasilkan 100 titik dari -10 hingga 10 untuk variabel x
y_values = [float(expression.subs(x, val)) for val in x_values]  # Menghitung nilai y berdasarkan
ekspresi matematika untuk setiap nilai x

# Menampilkan nilai x dan y untuk uji coba
print(x_values)  # Mencetak nilai-nilai x
print(y_values)  # Mencetak nilai-nilai y yang dihasilkan dari ekspresi

# Menambahkan noise (gangguan) pada data
noise = np.random.normal(0, 5, len(x_values))  # Membuat noise dengan distribusi normal (mean = 0,
standar deviasi = 5)
y_noisy = y_values + noise  # Menambahkan noise ke nilai y untuk menciptakan data yang bervariasi

# Membuat plot data
plt.figure(figsize=(10, 6))  # Mengatur ukuran gambar plot

# Menampilkan data dengan noise sebagai scatter plot
plt.scatter(x_values, y_noisy, label="Data dengan Noise", color='blue', s=10)  # Plot data dengan noise
dalam bentuk titik

# Menampilkan fungsi asli tanpa noise sebagai garis
plt.plot(x_values, y_values, label="Fungsi Asli", color='red', linewidth=2)  # Plot fungsi asli dalam
bentuk garis merah

# Menambahkan label dan judul pada plot
plt.xlabel("x")  # Label untuk sumbu x
plt.ylabel("y")  # Label untuk sumbu y
plt.legend()  # Menampilkan legenda untuk membedakan antara data dengan noise dan fungsi asli
plt.title("Data Sintetis dengan Hubungan Kuadratik dan Noise")  # Judul plot

# Menampilkan plot
plt.show()  # Menampilkan gambar plot di layar
```

    b. output

```
[-10.         -9.7979798  -9.5959596  -9.39393939 -9.19191919
  -8.98989899 -8.78787879 -8.58585859 -8.38383838 -8.18181818
  -7.97979798 -7.77777778 -7.57575758 -7.37373737 -7.17171717
  -6.96969697 -6.76767677 -6.56565657 -6.36363636 -6.16161616
  -5.95959596 -5.75757576 -5.55555556 -5.35353535 -5.15151515
  -4.94949495 -4.74747475 -4.54545455 -4.34343434 -4.14141414
  -3.93939394 -3.73737374 -3.53535354 -3.33333333 -3.13131313
  -2.92929293 -2.72727273 -2.52525253 -2.32323232 -2.12121212
  -1.91919192 -1.71717172 -1.51515152 -1.31313131 -1.11111111
  -0.90909091 -0.70707071 -0.50505051 -0.3030303  -0.1010101
   0.1010101   0.3030303   0.50505051  0.70707071  0.90909091
   1.11111111  1.31313131  1.51515152  1.71717172  1.91919192
   2.12121212  2.32323232  2.52525253  2.72727273  2.92929293
   3.13131313  3.33333333  3.53535354  3.73737374  3.93939394
   4.14141414  4.34343434  4.54545455  4.74747475  4.94949495
   5.15151515  5.35353535  5.55555556  5.75757576  5.95959596
   6.16161616  6.36363636  6.56565657  6.76767677  6.96969697
   7.17171717  7.37373737  7.57575758  7.77777778  7.97979798
   8.18181818  8.38383838  8.58585859  8.78787879  8.98989899
   9.19191919  9.39393939  9.5959596   9.7979798   10.        ]
[83.0, 79.40444852566064, 75.89052137536984, 72.45821854912765, 69.10754004693399, 65.8384858687889, 62.65105601469236, 59.54525048464443, 56.5210692786
4504, 53.57851239669422, 50.71757983879195, 47.938271604938265, 45.24058769513316, 42.62452810937659, 40.0900928476686, 37.637281910009186, 35.266095296
39833, 32.976533006836036, 30.768595041322314, 28.642281399857154, 26.597592082440563, 24.63452708907255, 22.753086419753085, 20.953270074482198, 19.235
07805325987, 17.598510356086116, 16.043566982960092, 14.570247933884302, 13.178553208856238, 11.868482807876749, 10.640036730945820, 9.493214978063465, 8.
42801754922967, 7.444444444444443, 6.542495663707786, 5.722171207019691 5, 4.983471074380166, 4.326395265789204, 3.7509437812468125, 3.257116620752984,
2.844913784307723, 2.5143352719110306, 2.265381083562902, 2.0980512192633407, 2.0123456790123457, 2.008264462809917, 2.0858075706560553, 2.2449750025507
598, 2.4857667584940315, 2.80818283848587, 3.2122232425262713, 3.697887970615242 5, 4.2651770227527805, 4.914090398938885, 5.64462809917355, 6.4567901234
56789, 7.350576471788593, 8.325987144168964, 9.383022140597893, 10.521681461075397, 11.74196510560147, 13.043873074176107, 14.427405366799299, 15.892561
983471069, 17.439342924191408, 19.067748188960312, 20.777777777777782, 22.569431690643803, 24.44270992755841, 26.397612488521577, 28.434139373533316, 3
0.552290582593603, 32.752066115702476, 35.033465972859915, 37.396490154065916, 39.841138659320464, 42.3674114886236, 44.9753086419753, 47.6648301193755
8, 50.43597592082439, 53.28874604632182, 56.22314049586777, 59.23915926946228, 62.336802367105406, 65.51606978879704, 68.77696153453732, 72.119477604326
08, 75.54361799816343, 79.0493827160494, 82.63677175798387, 86.30578512396691, 90.05642281399858, 93.88868482807874, 97.80257116620756, 101.798081828384
85, 105.87521681461072, 110.03397612488521, 114.27435975920824, 118.59636771757987, 123.0]
```



2. Faker
    a. code

```
from faker import Faker
import pandas as pd

# Inisialisasi Faker untuk membuat data palsu
fake = Faker()

# Menentukan jumlah data yang ingin dibuat
num_records = 100

# Membuat list kosong untuk setiap field (kolom data)
names = []
addresses = []
emails = []
jobs = []

# Menghasilkan data palsu untuk setiap field
for _ in range(num_records):
    names.append(fake.name())          # Menambahkan nama palsu ke list names
```

```
    addresses.append(fake.address())    # Menambahkan alamat palsu ke list addresses
    emails.append(fake.email())         # Menambahkan email palsu ke list emails
    jobs.append(fake.job())             # Menambahkan pekerjaan palsu ke list jobs

# Menggabungkan data yang telah dibuat ke dalam DataFrame
data = pd.DataFrame({
    'Name': names,        # Kolom 'Name' berisi data dari list names
    'Address': addresses, # Kolom 'Address' berisi data dari list addresses
    'Email': emails,      # Kolom 'Email' berisi data dari list emails
    'Job': jobs           # Kolom 'Job' berisi data dari list jobs
})

# Menampilkan beberapa baris pertama dari DataFrame
print(data.head())
```

b. output

```
              Name                                            Address  \
0     Zachary Vargas                   PSC 5565, Box 5870\nAPO AE 00933
1        Yvonne Ford                   Unit 0057 Box 0079\nDPO AP 85341
2     Kenneth Taylor  8836 Richardson Curve Suite 310\nWest Richard,...
3       Tyler Wilson  3747 Gina Junction Apt. 495\nHowardbury, VT 61436
4  Kristopher Fields  706 Wallace Extension Apt. 946\nMillerview, FL...

                         Email                    Job
0  contrerasmelissa@example.org  Engineering geologist
1            john71@example.com      Ceramics designer
2            lisa51@example.org   Electronics engineer
3        bhernandez@example.net        Careers adviser
4        knoxgerald@example.net    Early years teacher
```

3. SDV Single Table & Multi Table
   a. Single Table

```
                    import pandas as pd
from sdv.metadata import SingleTableMetadata
from sdv.single_table import GaussianCopulaSynthesizer

# Membuat contoh tabel pelanggan dalam bentuk DataFrame
customers_data = pd.DataFrame({
    'customer_id': [1, 2, 3, 4, 5],             # ID unik untuk setiap pelanggan
    'name': ['Alice', 'Bob', 'Carol', 'Dave', 'Eve'],  # Nama pelanggan
    'age': [25, 30, 45, 50, 35],                # Umur setiap pelanggan
    'purchase_amount': [100.5, 200.0, 150.75, 300.0, 250.25]  # Jumlah pembelian oleh
pelanggan
})

# Mendefinisikan metadata untuk tabel pelanggan
metadata = SingleTableMetadata()

# Mendeteksi struktur data dari DataFrame customers_data
metadata.detect_from_dataframe(customers_data)
```

```
# Memodifikasi struktur metadata yang terdeteksi untuk menentukan tipe kolom yang lebih spesifik
metadata.update_column(
    column_name='customer_id',
    sdtype='id'  # Menandai kolom 'customer_id' sebagai ID unik
)

metadata.update_column(
    column_name='age',
    sdtype='numerical'  # Menandai kolom 'age' sebagai data numerik (angka)
)

metadata.update_column(
    column_name='purchase_amount',
    sdtype='numerical'  # Menandai kolom 'purchase_amount' sebagai data numerik (angka)
)

# Menginisialisasi synthesizer untuk menghasilkan data sintetis
# menggunakan model Gaussian Copula, dengan metadata yang telah didefinisikan
synthesizer = GaussianCopulaSynthesizer(metadata)

# Melatih synthesizer menggunakan data asli dari tabel pelanggan
synthesizer.fit(customers_data)

# Menghasilkan data sintetis yang memiliki struktur dan pola yang mirip dengan data asli
synthetic_data = synthesizer.sample(num_rows=5)  # Menghasilkan 5 baris data sintetis

# Menampilkan data sintetis yang dihasilkan
print(synthetic_data)
```

	customer_id	name	age	purchase_amount
0	906487240	Eve	44	268.50
1	553966664	Bob	36	240.58
2	422235102	Dave	30	213.18
3	772062382	Carol	28	218.76
4	560297202	Dave	44	280.08

b. Multi Table

```
import pandas as pd
from sdv.metadata import MultiTableMetadata

# Membuat tabel "users" yang berisi data pengguna
users_data = pd.DataFrame({
    'user_id': [1, 2, 3],             # ID unik untuk setiap pengguna
    'username': ['Alice', 'Bob', 'Carol'],  # Nama pengguna
    'age': [24, 35, 45]               # Umur setiap pengguna
})

# Membuat tabel "orders" yang berisi data pesanan
orders_data = pd.DataFrame({
    'order_id': [101, 102, 103, 104],  # ID unik untuk setiap pesanan
```

```
    'user_id': [1, 2, 3, 1],          # ID pengguna, sebagai foreign key yang mengacu ke tabel "users"
    'product': ['Laptop', 'Phone', 'Tablet', 'Monitor'],  # Produk yang dibeli
    'amount': [1000, 500, 300, 150]    # Jumlah pembelian dalam satuan tertentu (misalnya dolar)
})

# Membuat instance MultiTableMetadata untuk mendefinisikan metadata pada beberapa tabel
metadata = MultiTableMetadata()

# Menambahkan metadata untuk tabel "users"
metadata.add_table('users')  # Menambahkan tabel "users" ke metadata
metadata.add_column(table_name='users', column_name='user_id', sdtype='id')  # Menandai "user_id" sebagai ID unik pengguna
metadata.add_column(table_name='users', column_name='username', sdtype='categorical')  # Menandai "username" sebagai data kategorikal
metadata.add_column(table_name='users', column_name='age', sdtype='numerical')  # Menandai "age" sebagai data numerik
metadata.set_primary_key(table_name='users', column_name='user_id')  # Menentukan "user_id" sebagai primary key untuk tabel "users"

# Menambahkan metadata untuk tabel "orders"
metadata.add_table('orders')  # Menambahkan tabel "orders" ke metadata
metadata.add_column(table_name='orders', column_name='order_id', sdtype='id')  # Menandai "order_id" sebagai ID unik untuk pesanan
metadata.add_column(table_name='orders', column_name='user_id', sdtype='id')  # Menandai "user_id" sebagai foreign key yang mengacu ke tabel "users"
metadata.add_column(table_name='orders', column_name='product', sdtype='categorical')  # Menandai "product" sebagai data kategorikal
metadata.add_column(table_name='orders', column_name='amount', sdtype='numerical')  # Menandai "amount" sebagai data numerik yang menunjukkan jumlah pesanan
metadata.set_primary_key(table_name='orders', column_name='order_id')  # Menentukan "order_id" sebagai primary key untuk tabel "orders"

# Mendefinisikan hubungan antara tabel "users" dan "orders" menggunakan foreign key
metadata.add_relationship(
    parent_table_name='users',          # Tabel induk (tabel "users")
    child_table_name='orders',          # Tabel anak (tabel "orders")
    parent_primary_key='user_id',       # Kolom primary key pada tabel induk ("user_id" di "users")
    child_foreign_key='user_id'         # Kolom foreign key pada tabel anak ("user_id" di "orders")
)
```

c. initial the syntizher

```
from sdv.multi_table import HMASynthesizer

# Inisialisasi HMASynthesizer dengan metadata
# `metadata` berisi definisi struktur atau skema untuk tabel-tabel terkait (users dan orders)
synthesizer = HMASynthesizer(metadata)

# Menyiapkan data input sebagai dictionary dari tabel-tabel
```

```
# Dictionary ini memetakan nama tabel ke data DataFrame yang sesuai
tables = {
    'users': users_data,    # Tabel pengguna asli
    'orders': orders_data   # Tabel pesanan asli
}

# Melatih synthesizer pada data
# HMASynthesizer akan mempelajari pola dan distribusi data dari tabel `users` dan `orders`
# berdasarkan metadata yang sudah ditentukan
synthesizer.fit(tables)
```

```
C:\Users\user\AppData\Local\Programs\Python\Python312\Lib\site-packages\sdv\multi_table\base.py:110: FutureWarning: The 'MultiTableMetadata' is deprecated. Please use the new 'Metadata' class for synthesizers.
  warnings.warn(DEPRECATION_MSG, FutureWarning)
C:\Users\user\AppData\Local\Programs\Python\Python312\Lib\site-packages\sdv\multi_table\base.py:102: UserWarning: We strongly recommend saving the metadata using 'save_to_json' for replicability in future SDV versions.
  warnings.warn(
Preprocess Tables: 100%|| 2/2 [00:00<00:00,  7.41it/s]

Learning relationships:
(1/1) Tables 'users' and 'orders' ('user_id'): 100%|| 3/3 [00:00<00:00,  7.14it/s]

Modeling Tables: 100%|| 2/2 [00:00<00:00,  2.78it/s]
```

d. generate synthetic data

```
# Data sintetis akan memiliki struktur dan pola yang mirip dengan data asli
synthetic_data = synthesizer.sample()

# Menampilkan data sintetis untuk tabel "Users"
print("Synthetic Users Table")
print(synthetic_data['users'])    # Menampilkan data sintetis untuk tabel `users`

# Menampilkan data sintetis untuk tabel "Orders"
print("\nSynthetic Orders Table")
print(synthetic_data['orders'])   # Menampilkan data sintetis untuk tabel `orders`
```

```
Synthetic Users Table
     user_id username  age
0  450136792      Bob   40
1  558009640      Bob   43
2  539105334    Carol   45

Synthetic Orders Table
    order_id    user_id  product  amount
0  865756271  450136792   Tablet    1000
1     913090  450136792   Tablet    1000
2  817638140  558009640   Tablet    1000
3  364504858  539105334  Monitor    1000
```

4. SDV: Time Series Data
   a. code

```
import pandas as pd
```

```
import numpy as np
from sdv.metadata import SingleTableMetadata
from sdv.sequential import PARSynthesizer

# Membuat DataFrame sampel deret waktu
data = {
   'data_id': np.arange(10),  # ID unik untuk setiap baris, mulai dari 0 hingga 9
   'timestamp': pd.date_range(start='2024-01-01', periods=10, freq='D'),  # Tanggal harian mulai 1 Januari 2024
   'sales': [100, 120, 130, 150, 180, 200, 210, 230, 240, 260]  # Nilai penjualan harian
}
sales_data = pd.DataFrame(data)  # Mengonversi data ke DataFrame `sales_data`

# Membuat metadata untuk dataset
# Metadata ini menjelaskan tipe data dari setiap kolom dalam DataFrame
metadata = SingleTableMetadata()
metadata.add_column(column_name='data_id', sdtype='id')        # Kolom 'data_id' ditandai sebagai ID unik
metadata.add_column(column_name='timestamp', sdtype='datetime') # Kolom 'timestamp' ditandai sebagai tipe datetime
metadata.add_column(column_name='sales', sdtype='numerical')    # Kolom 'sales' ditandai sebagai tipe numerik
metadata.set_sequence_key(column_name='data_id')               # Menentukan 'data_id' sebagai kunci urutan data

# Inisialisasi synthesizer
# `PARSynthesizer` digunakan untuk mensintesis data deret waktu,
# sehingga data sintetis mempertahankan pola dan urutan temporal dari data asli
synthesizer = PARSynthesizer(metadata)

# Melatih synthesizer pada data penjualan
# Synthesizer akan mempelajari pola, distribusi, dan tren dalam `sales_data`
synthesizer.fit(sales_data)

# Menghasilkan data sintetis
# Parameter `10` menunjukkan jumlah sampel data sintetis yang akan dihasilkan
synthetic_sales_data = synthesizer.sample(10)

# Menampilkan data sintetis
print("\nSynthetic Sales Data:")
```

```
print(synthetic_sales_data)
```