In [ ]:
```
!pip install DataSynthesizer
```

```
Requirement already satisfied: DataSynthesizer in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (0.1.11)
Requirement already satisfied: numpy>=1.18.5 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from DataSynthesizer) (1.26.0)
Requirement already satisfied: pandas>=1.0.5 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from DataSynthesizer) (2.0.1)
Requirement already satisfied: scikit-learn>=0.23.1 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from DataSynthesizer) (1.3.1)
Requirement already satisfied: matplotlib>=3.2.2 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from DataSynthesizer) (3.8.0)
Requirement already satisfied: seaborn>=0.10.1 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from DataSynthesizer) (0.12.2)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=3.2.2->DataSynthesizer) (2.8.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=3.2.2->DataSynthesizer) (1.1.1)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=3.2.2->DataSynthesizer) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=3.2.2->DataSynthesizer) (4.34.4)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=3.2.2->DataSynthesizer) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=3.2.2->DataSynthesizer) (21.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=3.2.2->DataSynthesizer) (9.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from matplotlib>=3.2.2->DataSynthesizer) (3.0.9)
Requirement already satisfied: pytz>=2020.1 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from pandas>=1.0.5->DataSynthesizer) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from pandas>=1.0.5->DataSynthesizer) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from python-dateutil>=2.8.1->matplotlib>=3.2.2->DataSynthesizer) (1.12.0)
Requirement already satisfied: scipy>=1.5.0 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn>=0.23.1->DataSynthesizer) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn>=0.23.1->DataSynthesizer) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from scikit-learn>=0.23.1->DataSynthesizer) (3.1.0)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [ ]:
```
!pip install Faker==2.0.5
```

```
Requirement already satisfied: Faker==2.0.5 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (2.0.5)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: python-dateutil>=2.4 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from Faker==2.0.5) (2.8.2)
Requirement already satisfied: six>=1.10 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from Faker==2.0.5) (1.12.0)
Requirement already satisfied: text-unidecode==1.3 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from Faker==2.0.5) (1.3)
[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [ ]:
```
!pip install mimesis
```

```
Requirement already satisfied: mimesis in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (11.1.0)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
import uuid
```

In [ ]:
```
from mimesis import Address, Code, Datetime, Development, File, Finance, Food, Hardware, Internet, Payment, Person, Science, Text, Transport
import random

# Define a dictionary to map user input to mimesis generators
generator_mapping = {
    # Address
    'address': Address().address,
    'city': Address().city,
    'country': Address().country,
    'state': Address().state,
    'postal_code': Address().postal_code,
    'latitude': Address().latitude,
    'longitude': Address().longitude,

    # Code
    'isbn': Code().isbn,
    'language_code': Code().locale_code,

    # Datetime
    'date': Datetime().date,
    'time': Datetime().time,
    'datetime': Datetime().datetime,
    'timestamp': Datetime().timestamp,

    # Development
    'programming_language': Development().programming_language,
    'os': Development().os,
    'version': Development.version,

    # File
    'file_name': File().file_name,
    'mime_type': File().mime_type,

    # Finance
    'bank': Finance().bank,
    'price': Finance().price,

    # Food
    'spices': Food().spices,
    'dish': Food().dish,
    'fruit': Food().fruit,
    'vegetable': Food().vegetable,

    # Hardware
    'cpu': Hardware().cpu,
    'cpu_codename':Hardware().cpu_codename,
    'cpu_frequency':Hardware().cpu_frequency,
    '':Hardware().graphics,
    # Internet
    'hashtags': Internet().hashtags,
    'username': Internet().user_agent,
    # 'domain': Internet().domain,
    # 'ip': Internet().ip,
    # 'ipv4': Internet().ipv4,
    # 'ipv6': Internet().ipv6,
    'user_agent': Internet().user_agent,
    # 'image_url': Internet().image_url,
    'tld': Internet().tld,

    # Payment
    'credit_card_number': Payment().credit_card_number,
    'credit_card_expiration_date': Payment().credit_card_expiration_date,
    'credit_card_full': Payment().credit_card_owner,
    '':Payment().cvv,

    # Person
    'full_name': Person().full_name,
    'first_name': Person().first_name,
    'last_name': Person().last_name,
    'gender': Person().gender,
    'age': Person().age,
    'title': Person().title,
    'occupation': Person().occupation,
    'telephone': Person().telephone,
    'email':Person().email,
    'height':Person().height,
    'username':Person().username,
    'password':Person().password,
    'weight':Person().weight,

    # Science
    'measure_unit': Science().measure_unit,
    'metric_prefix': Science().metric_prefix,
    # 'gas_element': Science(),
    # 'law': Science().law,
    # 'scientist': Science().scientist,

    # Text
    'word': Text().word,
    'sentence': Text().sentence,
    'color': Text().color,
    'text': Text().text,

    # Transport
    'car': Transport().car,
    'city': Transport().city,
    # 'airplane': Transport().airplane,
    # 'airplane_model': Transport().airplane_model,
    'airport': Transport().airplane,
    'vehicle_registration_code': Transport().vehicle_registration_code,

    # Other
    'boolean': lambda: random.choice([True, False]),
    'uuid': lambda: str(uuid.uuid4()),
    'random': random.random,
}

# Get user input for column name
column_name = input("Enter column name: ")

# Generate data based on user input
# if column_name in generator_mapping:
#     result = generator_mapping[column_name]()
#     print(f"Generated data for {column_name}: {result}")
# else:
#     print(f"Unsupported column name: {column_name}")
```

In [ ]:
```
from mimesis.schema import Field, Schema
```

In [ ]:
```
num_columns = int(input("Enter the number of columns: "))

# Get user input for column names and their corresponding generators
column_mappings = {}
for _ in range(num_columns):
    column_name = input("Enter column name: ")
    if column_name in generator_mapping:
        column_mappings[column_name] = generator_mapping[column_name]
    else:
        print(f"Unsupported column name: {column_name}")

# Define the description function dynamically based on user input
def description():
    data = {}
    for column_name, generator in column_mappings.items():
        data[column_name] = generator()
    return data

schema = Schema(schema=description)


# Generate data
# result = schema.create()
# print(result)
```

In [ ]:
```
results = [{col_name: generator() for col_name, generator in column_mappings.items()} for _ in range(100)]
results
```

Out[ ]:
```
[{'age': 19, 'gender': 'Male'},
 {'age': 64, 'gender': 'Other'},
 {'age': 54, 'gender': 'Male'},
 {'age': 60, 'gender': 'Male'},
 {'age': 36, 'gender': 'Female'},
 {'age': 38, 'gender': 'Female'},
 {'age': 47, 'gender': 'Female'},
 {'age': 54, 'gender': 'Male'},
 {'age': 50, 'gender': 'Female'},
 {'age': 55, 'gender': 'Other'},
 {'age': 44, 'gender': 'Male'},
 {'age': 44, 'gender': 'Female'},
 {'age': 28, 'gender': 'Female'},
 {'age': 46, 'gender': 'Female'},
 {'age': 64, 'gender': 'Female'},
 {'age': 24, 'gender': 'Female'},
 {'age': 37, 'gender': 'Female'},
 {'age': 31, 'gender': 'Female'},
 {'age': 66, 'gender': 'Female'},
 {'age': 60, 'gender': 'Male'},
 {'age': 49, 'gender': 'Other'},
 {'age': 57, 'gender': 'Female'},
 {'age': 36, 'gender': 'Other'},
 {'age': 32, 'gender': 'Male'},
 {'age': 21, 'gender': 'Female'},
 {'age': 46, 'gender': 'Male'},
 {'age': 56, 'gender': 'Other'},
 {'age': 56, 'gender': 'Other'},
 {'age': 23, 'gender': 'Male'},
 {'age': 43, 'gender': 'Male'},
 {'age': 65, 'gender': 'Other'},
 {'age': 44, 'gender': 'Other'},
 {'age': 27, 'gender': 'Female'},
 {'age': 53, 'gender': 'Female'},
 {'age': 27, 'gender': 'Female'},
 {'age': 45, 'gender': 'Female'},
 {'age': 51, 'gender': 'Female'},
 {'age': 41, 'gender': 'Female'},
 {'age': 30, 'gender': 'Female'},
 {'age': 58, 'gender': 'Female'},
 {'age': 24, 'gender': 'Female'},
 {'age': 62, 'gender': 'Female'},
 {'age': 48, 'gender': 'Female'},
 {'age': 27, 'gender': 'Male'},
 {'age': 27, 'gender': 'Male'},
 {'age': 61, 'gender': 'Female'},
 {'age': 19, 'gender': 'Other'},
 {'age': 55, 'gender': 'Other'},
 {'age': 18, 'gender': 'Male'},
 {'age': 32, 'gender': 'Other'},
 {'age': 32, 'gender': 'Female'},
 {'age': 33, 'gender': 'Other'},
 {'age': 18, 'gender': 'Female'},
 {'age': 58, 'gender': 'Male'},
 {'age': 66, 'gender': 'Other'},
 {'age': 32, 'gender': 'Other'},
 {'age': 42, 'gender': 'Other'},
 {'age': 27, 'gender': 'Other'},
 {'age': 63, 'gender': 'Female'},
 {'age': 21, 'gender': 'Female'},
 {'age': 23, 'gender': 'Other'},
 {'age': 17, 'gender': 'Female'},
 {'age': 35, 'gender': 'Female'},
 {'age': 25, 'gender': 'Female'},
 {'age': 57, 'gender': 'Other'},
 {'age': 34, 'gender': 'Female'},
 {'age': 33, 'gender': 'Other'},
 {'age': 51, 'gender': 'Other'},
 {'age': 26, 'gender': 'Male'},
 {'age': 24, 'gender': 'Male'},
 {'age': 24, 'gender': 'Female'},
 {'age': 65, 'gender': 'Other'},
 {'age': 22, 'gender': 'Male'},
 {'age': 18, 'gender': 'Other'},
 {'age': 26, 'gender': 'Male'},
 {'age': 63, 'gender': 'Male'},
 {'age': 44, 'gender': 'Other'},
 {'age': 51, 'gender': 'Other'},
 {'age': 61, 'gender': 'Male'},
 {'age': 61, 'gender': 'Male'},
 {'age': 57, 'gender': 'Female'},
 {'age': 16, 'gender': 'Male'},
 {'age': 57, 'gender': 'Other'},
 {'age': 43, 'gender': 'Male'},
 {'age': 38, 'gender': 'Other'},
 {'age': 23, 'gender': 'Male'},
 {'age': 29, 'gender': 'Female'},
 {'age': 29, 'gender': 'Male'},
 {'age': 55, 'gender': 'Male'},
 {'age': 44, 'gender': 'Female'},
 {'age': 21, 'gender': 'Female'},
 {'age': 25, 'gender': 'Male'},
 {'age': 56, 'gender': 'Male'},
 {'age': 34, 'gender': 'Female'},
 {'age': 35, 'gender': 'Female'},
 {'age': 65, 'gender': 'Female'},
 {'age': 45, 'gender': 'Female'},
 {'age': 27, 'gender': 'Male'},
 {'age': 54, 'gender': 'Female'},
 {'age': 46, 'gender': 'Other'}]
```

In [ ]:
```
res_df = pd.DataFrame(results)
```

In [ ]:
```
res_df.shape
```

Out[ ]:
```
(100, 2)
```

In [ ]:
```
res_df.head()
```

Out[ ]:

	age	gender
0	19	1
1	64	0
2	54	1
3	60	1
4	36	0

In [ ]:
```
import pandas as pd

# Assuming 'df' is your DataFrame
res_df['gender'] = res_df['gender'].replace({'Male': 1, 'Female': 0,'Other':0})

# Now, 'Male' will be replaced with 1 and 'Female' with 0
```

In [ ]:
```
res_df.head()
```

Out[ ]:

	age	gender
0	19	1
1	64	0
2	54	1
3	60	1
4	36	0

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Assuming 'res_df' contains the generated data
# Perform data preprocessing (e.g., encoding categorical variables)

# Define features (X) and target (y)
X = res_df.drop(columns=['age']).  # Replace 'target_column' with the actual target column name
y = res_df['age']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate model performance
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")

# Optionally, you can fine-tune hyperparameters and try different models for better performance
```

```
Mean Squared Error: 275.06820196048474
```

Model Evaluation:

The MSE value should be interpreted in the context of the range and distribution of 'age' values in your dataset. For instance, if the range of 'age' is very large, an MSE of 275 may be considered good. However, if the range is relatively small, it might indicate a need for improvement.

mean squared error (MSE) of 0.0131 suggests that the model is making relatively small errors when predicting the target variable. This is a good sign, indicating that the model's predictions are close to the actual values in the test set.

Keep in mind that the interpretation of MSE depends on the scale of your target variable. If the target variable has a specific unit (e.g., dollars, kilograms, etc.), the MSE value should be interpreted in the context of that unit. Lower MSE values indicate better model performance.

In summary, a MSE of 0.0131 is an encouraging result, but it's important to also consider the context of your specific problem and whether this level of error is acceptable for your application. If necessary, further optimization or trying different models can be explored to potentially improve performance.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: