In [2]:
```python
#Importing the relevant python libraries and the ols_pwp program

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import ols_pwp
from bokeh.io import output_notebook, show
from bokeh.models import ColumnDataSource, Whisker
from bokeh.plotting import figure
import seaborn as sns
output_notebook()
```

BokehJS 2.4.3 successfully loaded.

Importing the datasets: train, ideal and test.

The next cell shows how the datasets (train, ideal and test) are imported into this notebook to begin the regression analysis using the ordinary least squares method.

In [3]:
```python
df_load = ols_pwp.pwpTasks()

filepath_train = r"C:\Users\tosin\GIT_clones\tosinaa_dlmdspwp01\implement\train.csv"
filepath_ideal = r"C:\Users\tosin\GIT_clones\tosinaa_dlmdspwp01\implement\ideal.csv"
filepath_test = r"C:\Users\tosin\GIT_clones\tosinaa_dlmdspwp01\implement\test.csv"

train = df_load.df_loader(filepath_train)
ideal = df_load.df_loader(filepath_ideal)
test = df_load.df_loader(filepath_test)
```

In [4]:
```python
#Printing the top 5 rows of the train dataset

print(train.head())
```

```
      x         y1         y2        y3         y4
0 -20.0 -8000.4050  10648.215 -0.074365 -16000.222
1 -19.9 -7880.5566  10503.581  0.047023 -15761.586
2 -19.8 -7762.5130  10360.039  0.156072 -15524.712
3 -19.7 -7645.6567  10217.856  0.590901 -15290.346
4 -19.6 -7529.4863  10077.480  0.525232 -15058.604
```

In [5]:
```python
#Printing the top 5 rows of the ideal dataset

print(ideal.head())
```

```
      x        y1        y2        y3        y4        y5        y6        y7  \
0 -20.0 -0.912945  0.408082  9.087055  5.408082 -9.087055  0.912945 -0.839071
1 -19.9 -0.867644  0.497186  9.132356  5.497186 -9.132356  0.867644 -0.865213
2 -19.8 -0.813674  0.581322  9.186326  5.581322 -9.186326  0.813674 -0.889191
3 -19.7 -0.751573  0.659649  9.248426  5.659649 -9.248426  0.751573 -0.910947
4 -19.6 -0.681964  0.731386  9.318036  5.731386 -9.318036  0.681964 -0.930426

         y8        y9  ...        y41        y42       y43       y44  \
0 -0.850919  0.816164  ... -40.456474  40.204040  2.995732 -0.008333
1  0.168518  0.994372  ... -40.233820  40.048590  2.990720 -0.008340
2  0.612391  1.162644  ... -40.006836  39.890660  2.985682 -0.008347
3 -0.994669  1.319299  ... -39.775787  39.729824  2.980619 -0.008354
4  0.774356  1.462772  ... -39.540980  39.565693  2.975530 -0.008361

         y45       y46       y47       y48       y49       y50
0  12.995732  5.298317 -5.298317 -0.186278  0.912945  0.396850
1  12.990720  5.293305 -5.293305 -0.215690  0.867644  0.476954
2  12.985682  5.288267 -5.288267 -0.236503  0.813674  0.549129
3  12.980619  5.283204 -5.283204 -0.247887  0.751573  0.612840
4  12.975530  5.278115 -5.278115 -0.249389  0.681964  0.667902

[5 rows x 51 columns]
```

In [6]:
```python
#Printing the top 5 rows of the test dataset

print(test.head())
```

```
      x             y
0 -17.5  14492.095000
1  19.0      0.480704
2   6.5    274.085500
3  15.9   8039.792500
4 -14.3  -5848.080000
```

Computing the four best ideal functions using the train - ideal datasets pair

In doing this, we shall create an instance of the ols_pwp.pwpOLS() object from the ols_pwp program module. This will grant us access to the following methods:

1. squared_dev(): This method is used for computing the four best functions and returns the output as a list of dictionaries containing the (y-train : y-ideal, min_ssd).
2. idealfour_builder(): This method is used for building the dataset of the four best ideal functions.

In [7]:
```python
# Computing the ideal functions

# Creating the instance of the ols_pwp.pwpOLS object below
SSD = ols_pwp.pwpOLS(train, ideal)

# Computing the ideal4_lst of four functions
ideal4_lst = SSD.squared_dev()
print("Below are the yi-train and the corresponding yi-ideal functions columns (four best) and their minimum SSD values...:")
for idf in ideal4_lst:
    print(idf)
```

```
Below are the yi-train and the corresponding yi-ideal functions columns (four best) and their minimum SSD values...:
{'y1': ['y21', 34.565889914719286]}
{'y2': ['y27', 32.360254907816085]}
{'y3': ['y2', 35.14535429428959]}
{'y4': ['y24', 32.54067830439015]}
```

Validating The Results

To validate the results of the four best ideal functions, other model evaluation metrics shall be used. The metrics are listed below:

1. Mean square error (MSE)
2. Root mean square error (RMSE)
3. Mean absolute error (MAD)
4. LogCosh Loss

In [8]:
```python
# Building a function to compute the root mean square error

def build_rmse(df1, df2):
    pe = ols_pwp.pwpEvaluate()
    mse_train = list()
    for col in df2.columns[1:]:
        mse_train.append(pe.rmse(df1, df2[col]))
    result = (['y' + str(mse_train.index(np.min(mse_train))+1), np.min(mse_train)])
    return result

# Implementing the rmse calculations

print(build_rmse(train.y1, ideal))
print(build_rmse(train.y2, ideal))
print(build_rmse(train.y3, ideal))
print(build_rmse(train.y4, ideal))
```

```
['y21', 0.29]
['y27', 0.28]
['y2', 0.3]
['y24', 0.29]
```

In [9]:
```python
# Building a function to compute the mean square error

def build_mse(df1, df2):
    pe = ols_pwp.pwpEvaluate()
    mse_train = list()
    for col in df2.columns[1:]:
        mse_train.append(pe.mse(df1, df2[col]))
    result = (['y' + str(mse_train.index(np.min(mse_train))+1), np.min(mse_train)])
    return result

# Implementing the mse calculations
```

```
print(build_mse(train.y1, ideal))
print(build_mse(train.y2, ideal))
print(build_mse(train.y3, ideal))
print(build_mse(train.y4, ideal))
```

```
['y21', 0.09]
['y27', 0.08]
['y2', 0.09]
['y24', 0.08]
```

In [10]:
```
# Building a function to compute the mean absolute error

def build_mae(df1, df2):
    pe = ols_pwp.pwpEvaluate()
    mse_train = list()
    for col in df2.columns[1:]:
        mse_train.append(pe.mae(df1, df2[col]))
    result = (['y' + str(mse_train.index(np.min(mse_train))+1), np.min(mse_train)])
    return result

# Implementing the mae calculations

print(build_mae(train.y1, ideal))
print(build_mae(train.y2, ideal))
print(build_mae(train.y3, ideal))
print(build_mae(train.y4, ideal))
```

```
['y21', 0.26]
['y27', 0.24]
['y2', 0.26]
['y24', 0.25]
```

In [33]:
```
# Building a function to compute the logcosh loss

def build_lgcosh(df1, df2):
    pe = ols_pwp.pwpEvaluate()
    mse_train = list()
    for col in df2.columns[1:]:
        mse_train.append(pe.logcosh(df1, df2[col]))
    result = (['y' + str(mse_train.index(np.min(mse_train))+1), round(np.min(mse_train), 2)])
    return result

# Implementing the logcosh calculations

print(build_lgcosh(train.y1, ideal))
print(build_lgcosh(train.y2, ideal))
print(build_lgcosh(train.y3, ideal))
print(build_lgcosh(train.y4, ideal))
```

```
['y21', 16.86]
['y27', 15.79]
['y2', 17.15]
['y24', 15.89]
```

```
C:\Users\tosin\anaconda3\lib\site-packages\pandas\core\arraylike.py:397: RuntimeWarning: overflow encountered in cosh
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Interpreting the Result

The results from the other model evaluation metrics show that the four ideal functions obtained from the sum of square errors is proven to be correct.

In summary, the ideal function pairs are provided below.

1. y1-train, y21-idealfour
2. y2-train, y27-idealfour
3. y3-train, y2-idealfour
4. y4-train, y24-idealfour

In [12]:
```
### Building the idealfour dataset

idealfour = SSD.idealfour_builder()
print(idealfour.head())
```

```
      x       y21        y27        y2        y24
0 -20.0 -8000.000  10648.000  0.408082 -16000.000
1 -19.9 -7880.599  10503.459  0.497186 -15761.198
2 -19.8 -7762.392  10360.232  0.581322 -15524.784
3 -19.7 -7645.373  10218.313  0.659649 -15290.746
4 -19.6 -7529.536  10077.696  0.731386 -15059.072
```

Visualizing The Results

Visualizing the results involves plotting the scatter plots of the dataset pairs to analyze and making inferences based on the findings.

The matches or pairs are summerized below:

1. y1-train, y21-idealfour
2. y2-train, y27-idealfour
3. y3-train, y2-idealfour
4. y4-train, y24-idealfour

The Bokeh visualization library will be used to make the plots to understand the curve fitting patterns of the paired datasets.

In [13]:
```
# Plot of y1-train, y21-idealfour

a = figure(plot_width=650, plot_height=300, title= "Fitting y21-idealfour on y1-train")

a.circle(train.x, train.y1, size=12, color='black', legend_label="y1 - train")
a.square(idealfour.x, idealfour.y21, size=5,color='red', legend_label="y21 - idealfour")

a.legend.title = "Legend."
a.legend.location = "bottom_right"

show(a)
```

In [14]:
```
### Computing the r-squared for this result

pe = ols_pwp.pwpEvaluate()
rs_1 = pe.r_sqr(train['y1'], idealfour['y21'])

print(rs_1)
```

```
1.0
```

In [15]:
```
# Plot of y2-train, y27-idealfour

b = figure(plot_width=650, plot_height=350, title= "Fitting y27-idealfour on y2-train")

b.circle(train.x, train.y2, size=12, color='black', legend_label="y2 - train")
b.square(idealfour.x, idealfour.y27, size=5,color='blue', legend_label="y27 - idealfour")

b.legend.title = "Legend."
b.legend.location = "top_right"

show(b)
```

In [16]:
```
### Computing the r-squared for this result

pe = ols_pwp.pwpEvaluate()
rs_2 = pe.r_sqr(train['y2'], idealfour['y27'])

print(rs_2)
```

```
1.0
```

In [17]:
```
# Plot of y3-train, y2-idealfour

c = figure(title= "Fitting y2-idealfour on y3-train")

c.circle(train.x, train.y3, size=12, color='black', legend_label="y3 - train")
c.square(idealfour.x, idealfour.y2, size=5,color='yellow', legend_label="y2 - idealfour")

c.legend.title = "Legend."
c.legend.location = "bottom_left"
```

```
show(c)
```

In [18]:
```
### Computing the r-squared for this result

pe = ols_pwp.pwpEvaluate()
rs_3 = pe.r_sqr(train['y3'], idealfour['y2'])

print(rs_3)
```

```
0.91
```

In [19]:
```
# Plot of y4-train, y24-idealfour

d = figure(plot_width=650, plot_height=350, title= "Fitting y24-idealfour on y4-train")

d.circle(train.x, train.y4, size=12, color='black', legend_label="y4-train")
d.square(idealfour.x, idealfour.y24, size=5,color='green', legend_label="y24-idealfour")

d.legend.title = "Legend."
d.legend.location = "bottom_right"

show(d)
```

In [20]:
```
### Computing the r-squared for this result

pe = ols_pwp.pwpEvaluate()
rs_4 = pe.r_sqr(train['y4'], idealfour['y24'])

print(rs_4)
```

```
1.0
```

Finding the Best Fit From the Test and IdealFour Datasets

The second task required to find the final fitting function on the test dataset is as follows.

In [21]:
```
# Subsetting the idealfour dataset using the test['x'] to obtain the final idf_in_test dataset.

idf_in_test = idealfour[idealfour['x'].isin(test['x'])]
print(idf_in_test.shape)
```

```
(92, 5)
```

In [22]:
```
# Computing the test and idf_in_test datasets pair maximum deviation and determining which column has the maximum deviation

dev = ols_pwp.pwpDeviation(test,idf_in_test)

test_and_idf = dev.max_dev()
print(test_and_idf)
```

```
[['y24', 20451.356]]
```

In [23]:
```
# Computing the train and ideal datasets pair maximum deviation

dev = ols_pwp.pwpDeviation(train, ideal)
train_ideal = dev.max_dev()
print(train_ideal)
```

```
[['y27', 18648.405], ['y25', 34643.215], ['y25', 23994.92563518], ['y27', 26648.222]]
```

In [24]:
```
# Performing the check for the conditions
# maximum deviation of the test-idealfour datasets pair is less than
# the product of the maximum deviation of the train-ideal datasets pair and the square root of 2.

print(test_and_idf[0][1] < (train_ideal[1][1] * np.sqrt(2)))
```

```
True
```

In [25]:
```
# Building the fit dataset derived from the max deviation calculation
# using the test and idealfour datasets pair

fit = idf_in_test[["x", "y24"]]
print(fit.head())
```

```
       x        y24
7  -19.3 -14378.114
8  -19.2 -14155.776
13 -18.7 -13078.406
14 -18.6 -12869.712
25 -17.5 -10718.750
```

In [26]:
```
# Conclusively, we can plot the fit dataset (idf_in_test[[x, y24]]) on the test dataset to study the curve fit.

e = figure(plot_width=650, plot_height=350, title= "Fitting y24-fit on y-test")

e.circle(test.x, test.y, size=12, color='black', legend_label="y - test")
e.square(fit.x, fit.y24, size=7,color='green', legend_label="y24 - fit")

e.legend.title = "Legend."
e.legend.location = "bottom_right"

show(e)
```

In [27]:
```
### Computing the r-squared for this result

pe = ols_pwp.pwpEvaluate()
rs_5 = pe.r_sqr(test['y'], idf_in_test['y24'])

print(rs_5)
```

```
1.08
```

Conclusion

Therefore, the fitting function is that contained in the fit dataset.

Creating the Database and Writing the Datasets into Database Tables

At this stage, it is necessary to write all the DataFrame datasets into an sql database. So this will involve the 3 datasets provided for the project and the additional two datasets created while working on the project.

To begin this next step, importing the important libraries is very important.

In [40]:
```
# Importing the libraries and creating the connection object

import sqlite3
from sqlalchemy import create_engine, text
engine = create_engine('sqlite:///pwpDatasets', echo=True, future=True)
conn = engine.connect()
```

In [ ]:
```
# Creating the pwpDatasets Database

query1 = "CREATE DATABASE IF NOT EXISTS pwpDatasets;"
conn.execute(text(query1))
```

Creating the Tables inside the pwpDatasets Database:

1. train
2. ideal
3. test
4. idealfour
5. fit

In [44]:
```
# Creating the train table inside the pwpDatasets Database

query2 = "CREATE TABLE IF NOT EXISTS train(x DECIMAL, y1 DECIMAL, y2 DECIMAL, y3 DECIMAL, y4 DECIMAL);"
result2 = conn.execute(text(query2))
conn.commit()
```

```
2023-04-20 12:15:39,435 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:39,436 INFO sqlalchemy.engine.Engine CREATE TABLE IF NOT EXISTS train(x DECIMAL, y1 DECIMAL, y2 DECIMAL, y3 DECIMAL, y4 DECIMAL);
2023-04-20 12:15:39,437 INFO sqlalchemy.engine.Engine [cached since 6.18s ago] ()
2023-04-20 12:15:39,438 INFO sqlalchemy.engine.Engine COMMIT
```

In [45]:

```python
# Creating the ideal table inside the pwpDatasets Database

query3 = "CREATE TABLE IF NOT EXISTS ideal(x DECIMAL, y1 DECIMAL, y2 DECIMAL, y3 DECIMAL, y4 DECIMAL, y5 DECIMAL, y6 DECIMAL, y7 DECIMAL, y8 DECIMAL, y9 DECIMAL, y10 DECIMAL, y11 DECIMAL, y12 DECIMAL, y13 DECIMAL, y14 DEC
result3 = conn.execute(text(query3))
conn.commit()
```

```
2023-04-20 12:15:40,242 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:40,243 INFO sqlalchemy.engine.Engine CREATE TABLE IF NOT EXISTS ideal(x DECIMAL, y1 DECIMAL, y2 DECIMAL, y3 DECIMAL, y4 DECIMAL, y5 DECIMAL, y6 DECIMAL, y7 DECIMAL, y8 DECIMAL, y9 DECIMAL, y10 DECIMAL, y1
1 DECIMAL, y12 DECIMAL, y13 DECIMAL, y14 DECIMAL, y15 DECIMAL, y16 DECIMAL, y17 DECIMAL, y18    DECIMAL, y19 DECIMAL, y20 DECIMAL, y21 DECIMAL, y22 DECIMAL, y23 DECIMAL, y24 DECIMAL, y25      DECIMAL, y26 DECIMAL, y27 DEC
IMAL, y28 DECIMAL, y29 DECIMAL, y30 DECIMAL, y31 DECIMAL, y32 DECIMAL, y33 DECIMAL, y34 DECIMAL, y35 DECIMAL, y36 DECIMAL, y37 DECIMAL, y38 DECIMAL, y39 DECIMAL, y40 DECIMAL, y41 DECIMAL, y42 DECIMAL, y43 DECIMAL, y44 DEC
IMAL, y45       DECIMAL, y46 DECIMAL, y47 DECIMAL, y48 DECIMAL, y49 DECIMAL, y50 DECIMAL);
2023-04-20 12:15:40,244 INFO sqlalchemy.engine.Engine [generated in 0.00203s] ()
2023-04-20 12:15:40,374 INFO sqlalchemy.engine.Engine COMMIT
```

In [46]:

```python
# Creating the test table inside the pwpDatasets Database

query5 = "CREATE TABLE IF NOT EXISTS test(x DECIMAL, y DECIMAL);"
result5 = conn.execute(text(query5))
conn.commit()
```

```
2023-04-20 12:15:41,329 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:41,330 INFO sqlalchemy.engine.Engine CREATE TABLE IF NOT EXISTS test(x DECIMAL, y DECIMAL);
2023-04-20 12:15:41,331 INFO sqlalchemy.engine.Engine [generated in 0.00230s] ()
2023-04-20 12:15:41,426 INFO sqlalchemy.engine.Engine COMMIT
```

In [47]:

```python
# Creating the idealfour table inside the pwpDatasets Database

query4 = "CREATE TABLE IF NOT EXISTS idealfour(x DECIMAL, y1 DECIMAL, y2 DECIMAL, y3 DECIMAL, y4 DECIMAL);"
result4 = conn.execute(text(query4))
conn.commit()
```

```
2023-04-20 12:15:41,761 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:41,762 INFO sqlalchemy.engine.Engine CREATE TABLE IF NOT EXISTS idealfour(x DECIMAL, y1 DECIMAL, y2 DECIMAL, y3 DECIMAL, y4 DECIMAL);
2023-04-20 12:15:41,762 INFO sqlalchemy.engine.Engine [generated in 0.00160s] ()
2023-04-20 12:15:41,764 INFO sqlalchemy.engine.Engine COMMIT
```

In [48]:

```python
# Creating the fit table inside the pwpDatasets Database

query6 = "CREATE TABLE IF NOT EXISTS fit(x DECIMAL, y24 DECIMAL);"
result6 = conn.execute(text(query6))
conn.commit()
```

```
2023-04-20 12:15:42,092 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:42,093 INFO sqlalchemy.engine.Engine CREATE TABLE IF NOT EXISTS fit(x DECIMAL, y24 DECIMAL);
2023-04-20 12:15:42,094 INFO sqlalchemy.engine.Engine [generated in 0.00150s] ()
2023-04-20 12:15:42,221 INFO sqlalchemy.engine.Engine COMMIT
```

Ingesting the datasets into the respective tables:

1. train --> train dataset
2. ideal --> ideal dataset
3. test --> test dataset
4. idealfour --> idealfour dataset
5. fit --> fit dataset

In [50]:

```python
# Ingesting the train Dataframe into the train sql table.

train.to_sql("train", engine, if_exists="replace", index=False)
```

```
2023-04-20 12:15:42,834 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:42,835 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("train")
2023-04-20 12:15:42,835 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,837 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:42,837 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:42,838 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("train")
2023-04-20 12:15:42,839 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,840 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:42,840 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:42,841 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER BY name
2023-04-20 12:15:42,842 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,843 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_temp_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER BY name
2023-04-20 12:15:42,843 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,844 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("train")
2023-04-20 12:15:42,845 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,846 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:42,847 INFO sqlalchemy.engine.Engine [raw sql] ('train',)
2023-04-20 12:15:42,848 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list("train")
2023-04-20 12:15:42,848 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,849 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list("train")
2023-04-20 12:15:42,849 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,851 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:42,851 INFO sqlalchemy.engine.Engine [raw sql] ('train',)
2023-04-20 12:15:42,854 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("train")
2023-04-20 12:15:42,856 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,857 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("train")
2023-04-20 12:15:42,858 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,859 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("train")
2023-04-20 12:15:42,859 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,860 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("train")
2023-04-20 12:15:42,861 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,861 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("train")
2023-04-20 12:15:42,862 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,863 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("train")
2023-04-20 12:15:42,864 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:42,865 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:42,867 INFO sqlalchemy.engine.Engine [raw sql] ('train',)
2023-04-20 12:15:42,871 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:42,872 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:42,873 INFO sqlalchemy.engine.Engine
DROP TABLE train
2023-04-20 12:15:42,874 INFO sqlalchemy.engine.Engine [no key 0.00065s] ()
2023-04-20 12:15:43,041 INFO sqlalchemy.engine.Engine COMMIT
2023-04-20 12:15:43,044 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,048 INFO sqlalchemy.engine.Engine
CREATE TABLE train (
	x FLOAT,
	y1 FLOAT,
	y2 FLOAT,
	y3 FLOAT,
	y4 FLOAT
)


2023-04-20 12:15:43,049 INFO sqlalchemy.engine.Engine [no key 0.00143s] ()
2023-04-20 12:15:43,136 INFO sqlalchemy.engine.Engine COMMIT
2023-04-20 12:15:43,137 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,142 INFO sqlalchemy.engine.Engine INSERT INTO train (x, y1, y2, y3, y4) VALUES (?, ?, ?, ?, ?)
2023-04-20 12:15:43,143 INFO sqlalchemy.engine.Engine [generated in 0.00422s] [(-20.0, -8000.405, 10648.215, -0.07436482, -16000.222), (-19.9, -7880.5566, 10503.581, 0.047023416, -15761.586), (-19.8, -7762.513, 10360.039,
0.15607175, -15524.712), (-19.7, -7645.6567, 10217.856, 0.59090126, -15290.346), (-19.6, -7529.4863, 10077.48, 0.52523214, -15058.604), (-19.5, -7415.344, 9938.282, 1.195719, -14829.303), (-19.4, -7301.1904, 9800.395, 1.0
92391, -14603.181), (-19.3, -7188.988, 9663.686, 1.0499092, -14378.3955)  ... displaying 10 of 400 total bound parameter sets ...  (19.8, 7761.9194, -5639.603, 0.39675143, 15524.904), (19.9, 7880.9478, -5735.1235, 0.65514
016, 15761.235)]
2023-04-20 12:15:43,148 INFO sqlalchemy.engine.Engine COMMIT
```

Out[50]:

```
400
```

In [51]:

```python
# Querying the train table from the pwpDatasets database

query7 = "SELECT * FROM train LIMIT 5"
result7 = conn.execute(text(query7))
for res7 in result7:
    print(res7)
```

```
2023-04-20 12:15:43,256 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,258 INFO sqlalchemy.engine.Engine SELECT * FROM train LIMIT 5
2023-04-20 12:15:43,259 INFO sqlalchemy.engine.Engine [generated in 0.00274s] ()
(-20.0, -8000.405, 10648.215, -0.07436482, -16000.222)
(-19.9, -7880.5566, 10503.581, 0.047023416, -15761.586)
(-19.8, -7762.513, 10360.039, 0.15607175, -15524.712)
(-19.7, -7645.6567, 10217.856, 0.59090126, -15290.346)
(-19.6, -7529.4863, 10077.48, 0.52523214, -15058.604)
```

In [52]:

```python
# Ingesting the ideal Dataframe into the train sql table.

ideal.to_sql("ideal", engine, if_exists="replace", index=False)
```

```
2023-04-20 12:15:43,348 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,350 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("ideal")
2023-04-20 12:15:43,350 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,352 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:43,353 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,353 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("ideal")
2023-04-20 12:15:43,355 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,356 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:43,357 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,357 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER BY name
2023-04-20 12:15:43,358 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,359 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_temp_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER BY name
2023-04-20 12:15:43,360 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,360 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("ideal")
2023-04-20 12:15:43,361 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,362 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:43,363 INFO sqlalchemy.engine.Engine [raw sql] ('ideal',)
2023-04-20 12:15:43,365 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list("ideal")
2023-04-20 12:15:43,366 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,366 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list("ideal")
2023-04-20 12:15:43,368 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,370 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:43,371 INFO sqlalchemy.engine.Engine [raw sql] ('ideal',)
2023-04-20 12:15:43,374 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("ideal")
2023-04-20 12:15:43,375 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,376 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("ideal")
2023-04-20 12:15:43,376 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,377 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("ideal")
2023-04-20 12:15:43,378 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,379 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("ideal")
2023-04-20 12:15:43,380 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,381 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("ideal")
2023-04-20 12:15:43,381 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,383 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("ideal")
2023-04-20 12:15:43,383 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,384 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:43,385 INFO sqlalchemy.engine.Engine [raw sql] ('ideal',)
2023-04-20 12:15:43,389 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:43,390 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,391 INFO sqlalchemy.engine.Engine 
DROP TABLE ideal
2023-04-20 12:15:43,392 INFO sqlalchemy.engine.Engine [no key 0.00116s] ()
2023-04-20 12:15:43,497 INFO sqlalchemy.engine.Engine COMMIT
2023-04-20 12:15:43,504 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,506 INFO sqlalchemy.engine.Engine 
CREATE TABLE ideal (
	x FLOAT, 
	y1 FLOAT, 
	y2 FLOAT, 
	y3 FLOAT, 
	y4 FLOAT, 
	y5 FLOAT, 
	y6 FLOAT, 
	y7 FLOAT, 
	y8 FLOAT, 
	y9 FLOAT, 
	y10 FLOAT, 
	y11 FLOAT, 
	y12 FLOAT, 
	y13 FLOAT, 
	y14 FLOAT, 
	y15 FLOAT, 
	y16 FLOAT, 
	y17 FLOAT, 
	y18 FLOAT, 
	y19 FLOAT, 
	y20 FLOAT, 
	y21 FLOAT, 
	y22 FLOAT, 
	y23 FLOAT, 
	y24 FLOAT, 
	y25 FLOAT, 
	y26 FLOAT, 
	y27 FLOAT, 
	y28 FLOAT, 
	y29 FLOAT, 
	y30 FLOAT, 
	y31 FLOAT, 
	y32 FLOAT, 
	y33 FLOAT, 
	y34 FLOAT, 
	y35 FLOAT, 
	y36 FLOAT, 
	y37 FLOAT, 
	y38 FLOAT, 
	y39 FLOAT, 
	y40 FLOAT, 
	y41 FLOAT, 
	y42 FLOAT, 
	y43 FLOAT, 
	y44 FLOAT, 
	y45 FLOAT, 
	y46 FLOAT, 
	y47 FLOAT, 
	y48 FLOAT, 
	y49 FLOAT, 
	y50 FLOAT
)


2023-04-20 12:15:43,507 INFO sqlalchemy.engine.Engine [no key 0.00098s] ()
2023-04-20 12:15:43,592 INFO sqlalchemy.engine.Engine COMMIT
2023-04-20 12:15:43,596 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,614 INFO sqlalchemy.engine.Engine INSERT INTO ideal (x, y1, y2, y3, y4, y5, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, y16, y17, y18, y19, y20, y21, y22, y23, y24, y25, y26, y27, y28, y29, y30, y31, 
y32, y33, y34, y35, y36, y37, y38, y39, y40, y41, y42, y43, y44, y45, y46, y47, y48, y49, y50) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, 
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
2023-04-20 12:15:43,615 INFO sqlalchemy.engine.Engine [generated in 0.01198s] [(-20.0, -0.9129453, 0.40808207, 9.087055, 5.408082, -9.087055, 0.9129453, -0.8390715, -0.85091937, 0.81616414, 18.258905, -20.0, -58.0, -45.0, 
20.0, 13.0, 400.0, -400.0, 800.0, 410.0, 289.0, -8000.0, 8000.0, 8000.0, -16000.0, -23995.0, -5832.0, 10648.0, -8020.0, -7600.0, -8795.0, 20.0, 4.472136, 20.12461, -0.7464143, 10.0, 100.0, -20.0, -1.3210273, 399.08707, 89
9.5919, -40.456474, 40.20404, 2.9957323, -0.008333334, 12.995732, 5.2983174, -5.2983174, -0.18627828, 0.9129453, 0.3968496), (-19.9, -0.8676441, 0.4971858, 9.132356, 5.4971857, -9.132356, 0.8676441, -0.8652126, 0.1685176
8, 0.9943716, 17.266117, -19.9, -57.7, -44.8, 19.9, 12.95, 396.01, -396.01, 792.02, 406.01, 285.61, -7880.599, 7880.599, 7880.599, -15761.198, -23636.797, -5735.339, 10503.459, -7900.499, -7484.589, -8667.619, 19.9, 4.460
942, 20.025234, -0.6204504, 9.9, 99.5, -19.9, -1.3648299, 395.14236, 893.5128, -40.23382, 40.04859, 2.9907198, -0.008340283, 12.99072, 5.293305, -5.293305, -0.21569017, 0.8676441, 0.47695395), (-19.8, -0.81367373, 0.58132
184, 9.186326, 5.5813217, -9.186326, 0.81367373, -0.88919115, 0.6123911, 1.1626437, 16.11074, -19.8, -57.4, -44.6, 19.8, 12.9, 392.04, -392.04, 784.08, 402.04, 282.24, -7762.392, 7762.392, 7762.392, -15524.784, -23282.17
6, -5639.752, 10360.232, -7782.192, -7370.352, -8541.472, 19.8, 4.449719, 19.925863, -0.47573867, 9.8, 99.0, -19.8, -1.3949956, 391.22632, 887.4587, -40.006836, 39.89066, 2.985682, -0.008347246, 12.985682, 5.288267, -5.28
8267, -0.23650314, 0.81367373, 0.5491291), (-19.7, -0.75157344, 0.65964943, 9.248426, 5.6596494, -9.248426, 0.75157344, -0.91094714, -0.99466854, 1.3192989, 14.805996, -19.7, -57.1, -44.4, 19.7, 12.85, 388.09, -388.09, 77
6.18, 398.09, 278.89, -7645.373, 7645.373, 7645.373, -15290.746, -22931.12, -5545.233, 10218.313, -7665.073, -7257.283, -8416.553, 19.7, 4.438468, 19.826498, -0.31643602, 9.7, 98.5, -19.7, -1.4112228, 387.33844, 881.4303
6, -39.775787, 39.729824, 2.9806187, -0.008354219, 12.9806185, 5.2832036, -5.2832036, -0.24788749, 0.75157344, 0.6128399), (-19.6, -0.6819636, 0.7313861, 9.318036, 5.731386, -9.318036, 0.6819636, -0.9304263, 0.7743557, 1.
4627723, 13.366487, -19.6, -56.8, -44.2, 19.6, 12.8, 384.16, -384.16, 768.32, 394.16, 275.56, -7529.536, 7529.536, 7529.536, -15059.072, -22583.607, -5451.776, 10077.696, -7549.136, -7145.376, -8292.856, 19.6, 4.427189, 1
9.727139, -0.147038, 9.6, 98.0, -19.6, -1.4133497, 383.47803, 875.4286, -39.54098, 39.565693, 2.9755297, -0.008361204, 12.97553, 5.278115, -5.278115, -0.24938935, 0.6819636, 0.6679019), (-19.5, -0.60553986, 0.795815, 9.39
446, 5.795815, -9.39446, 0.60553986, -0.9475798, -0.11702018, 1.59163, 11.808027, -19.5, -56.5, -44.0, 19.5, 12.75, 380.25, -380.25, 760.5, 390.25, 272.25, -7414.875, 7414.875, 7414.875, -14829.75, -22239.625, -5359.375, 
9938.375, -7434.375, -7034.625, -8170.375, 19.5, 4.41588, 19.627787, 0.027781596, 9.5, 97.5, -19.5, -1.4013548, 379.64447, 869.45416, -39.30277, 39.397907, 2.9704144, -0.008368201, 12.970414, 5.273, -5.273, -0.24094884, 
0.60553986, 0.7144341), (-19.4, -0.52306575, 0.8522923, 9.476934, 5.8522925, -9.476934, 0.52306575, -0.96236485, -0.59004813, 1.7045846, 10.147476, -19.4, -56.2, -43.8, 19.4, 12.7, 376.36, -376.36, 752.72, 386.36, 268.96, 
-7301.384, 7301.384, 7301.384, -14602.768, -21899.152, -5268.024, 9800.344, -7320.784, -6925.024, -8049.104, 19.4, 4.404543, 19.52844, 0.20333645, 9.4, 97.0, -19.4, -1.3753581, 375.83694, 863.5077, -39.06153, 39.226147, 
2.9652731, -0.00837521, 12.965273, 5.267858, -5.267858, -0.22290246, 0.52306575, 0.75279135), (-19.3, -0.43536535, 0.90025383, 9.564634, 5.900254, -9.564634, 0.43536535, -0.97474456, 0.97776526, 1.8005077, 8.402552, -19.
3, -55.9, -43.6, 19.3, 12.65, 372.49, -372.49, 744.98, 382.49, 265.69, -7189.057, 7189.057, 7189.057, -14378.114, -21562.172, -5177.717, 9663.597, -7208.357, -6816.567, -7929.037, 19.3, 4.3931766, 19.429102, 0.37509164, 
9.3, 96.5, -19.3, -1.3356192, 372.05463, 857.5897, -38.817684, 39.050125, 2.9601052, -0.00838223, 12.960105, 5.26269, -5.26269, -0.19596967, 0.43536535, 0.7834847)  ... displaying 10 of 400 total bound parameter sets ... 
(19.8, 0.81367373, 0.58132184, 10.813674, 5.5813217, -10.813674, -0.81367373, -0.88919115, 0.6123911, 1.1626437, 16.11074, 19.8, 61.4, 34.6, -19.8, -6.9, 392.04, -392.04, 784.08, 402.04, 519.84, 7762.392, 7762.392, -7762.
392, 15524.784, 23292.176, 10360.232, -5639.752, 7782.192, 8154.432, 6983.312, 19.8, 4.449719, 19.925863, 1.1516088, 9.8, 99.0, -19.8, 0.23235193, 392.85367, 95.45868, 40.006836, -39.309338, 2.985682, -0.012468828, 12.985
682, 5.288267, -5.288267, 0.23650314, 0.81367373, 0.5491291), (19.9, 0.8676441, 0.4971858, 10.867644, 5.4971857, -10.867644, -0.8676441, -0.8652126, 0.16851768, 0.9943716, 17.266117, 19.9, 61.7, 34.8, -19.9, -6.95, 396.0
1, -396.01, 792.02, 406.01, 524.41, 7880.599, 7880.599, -7880.599, 15761.198, 23646.797, 10503.459, -5735.339, 7900.499, 8276.609, 7093.579, 19.9, 4.460942, 20.025234, 1.1148378, 9.9, 99.5, -19.9, 0.3704583, 396.87766, 9
7.51282, 40.23382, -39.551407, 2.9907198, -0.012484395, 12.99072, 5.293305, -5.293305, 0.21569017, 0.8676441, 0.47695395)]
2023-04-20 12:15:43,622 INFO sqlalchemy.engine.Engine COMMIT
```

Out[52]: 400

In [53]:

```python
# Querying the ideal table from the database

query8 = "SELECT * FROM ideal LIMIT 5"
result8 = conn.execute(text(query8))
for res8 in result8:
    print(res8)
```

```
2023-04-20 12:15:43,736 INFO sqlalchemy.engine.Engine SELECT * FROM ideal LIMIT 5
2023-04-20 12:15:43,737 INFO sqlalchemy.engine.Engine [generated in 0.00121s] ()
(-20.0, -0.9129453, 0.40808207, 9.087055, 5.408082, -9.087055, 0.9129453, -0.8390715, -0.85091937, 0.81616414, 18.258905, -20.0, -58.0, -45.0, 20.0, 13.0, 400.0, -400.0, 800.0, 410.0, 289.0, -8000.0, 8000.0, 8000.0, -1600
0.0, -23995.0, -5832.0, 10648.0, -8020.0, -7600.0, -8795.0, 20.0, 4.472136, 20.12461, -0.7464143, 10.0, 100.0, -20.0, -1.3210273, 399.08707, 899.5919, -40.456474, 40.20404, 2.9957323, -0.008333334, 12.995732, 5.2983174, -
5.2983174, -0.18627828, 0.9129453, 0.3968496)
(-19.9, -0.8676441, 0.4971858, 9.132356, 5.4971857, -9.132356, 0.8676441, -0.8652126, 0.16851768, 0.9943716, 17.266117, -19.9, -57.7, -44.8, 19.9, 12.95, 396.01, -396.01, 792.02, 406.01, 285.61, -7880.599, 7880.599, 7880.
599, -15761.198, -23636.797, -5735.339, 10503.459, -7900.499, -7484.589, -8667.619, 19.9, 4.460942, 20.025234, -0.6204504, 9.9, 99.5, -19.9, -1.3648299, 395.14236, 893.5128, -40.23382, 40.04859, 2.9907198, -0.008340283, 1
2.99072, 5.293305, -5.293305, -0.21569017, 0.8676441, 0.47695395)
(-19.8, -0.81367373, 0.58132184, 9.186326, 5.5813217, -9.186326, 0.81367373, -0.88919115, 0.6123911, 1.1626437, 16.11074, -19.8, -57.4, -44.6, 19.8, 12.9, 392.04, -392.04, 784.08, 402.04, 282.24, -7762.392, 7762.392, 776
2.392, -15524.784, -23282.176, -5639.752, 10360.232, -7782.192, -7370.352, -8541.472, 19.8, 4.449719, 19.925863, -0.47573867, 9.8, 99.0, -19.8, -1.3949956, 391.22632, 887.4587, -40.006836, 39.89066, 2.985682, -0.00834724
6, 12.985682, 5.288267, -5.288267, -0.23650314, 0.81367373, 0.5491291)
(-19.7, -0.75157344, 0.65964943, 9.248426, 5.6596494, -9.248426, 0.75157344, -0.91094714, -0.99466854, 1.3192989, 14.805996, -19.7, -57.1, -44.4, 19.7, 12.85, 388.09, -388.09, 776.18, 398.09, 278.89, -7645.373, 7645.373,
7645.373, -15290.746, -22931.12, -5545.233, 10218.313, -7665.073, -7257.283, -8416.553, 19.7, 4.438468, 19.826498, -0.31643602, 9.7, 98.5, -19.7, -1.4112228, 387.33844, 881.43036, -39.775787, 39.729824, 2.9806187, -0.0083
54219, 12.9806185, 5.2832036, -5.2832036, -0.24788749, 0.75157344, 0.6128399)
(-19.6, -0.6819636, 0.7313861, 9.318036, 5.731386, -9.318036, 0.6819636, -0.9304263, 0.7743557, 1.4627723, 13.366487, -19.6, -56.8, -44.2, 19.6, 12.8, 384.16, -384.16, 768.32, 394.16, 275.56, -7529.536, 7529.536, 7529.53
6, -15059.072, -22583.607, -5451.776, 10077.696, -7549.136, -7145.376, -8292.856, 19.6, 4.427189, 19.727139, -0.147038, 9.6, 98.0, -19.6, -1.4133497, 383.47803, 875.4286, -39.54098, 39.565693, 2.9755297, -0.008361204, 12.
97553, 5.278115, -5.278115, -0.24938935, 0.6819636, 0.6679019)
```

In [54]:
```python
# Ingesting the test Dataframe into the train sql table.

test.to_sql("test", engine, if_exists="replace", index=False)
```

```
2023-04-20 12:15:43,824 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,825 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("test")
2023-04-20 12:15:43,826 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,828 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:43,828 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,829 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("test")
2023-04-20 12:15:43,830 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,831 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:43,832 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,832 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER BY name
2023-04-20 12:15:43,833 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,834 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_temp_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER BY name
2023-04-20 12:15:43,835 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,836 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("test")
2023-04-20 12:15:43,836 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,838 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:43,839 INFO sqlalchemy.engine.Engine [raw sql] ('test',)
2023-04-20 12:15:43,840 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list("test")
2023-04-20 12:15:43,841 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,842 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list("test")
2023-04-20 12:15:43,842 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,843 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:43,844 INFO sqlalchemy.engine.Engine [raw sql] ('test',)
2023-04-20 12:15:43,846 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("test")
2023-04-20 12:15:43,848 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,849 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("test")
2023-04-20 12:15:43,849 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,850 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("test")
2023-04-20 12:15:43,851 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,852 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("test")
2023-04-20 12:15:43,853 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,854 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("test")
2023-04-20 12:15:43,854 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,856 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("test")
2023-04-20 12:15:43,856 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:43,858 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:43,858 INFO sqlalchemy.engine.Engine [raw sql] ('test',)
2023-04-20 12:15:43,860 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:43,861 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,863 INFO sqlalchemy.engine.Engine
DROP TABLE test
2023-04-20 12:15:43,864 INFO sqlalchemy.engine.Engine [no key 0.00069s] ()
2023-04-20 12:15:43,951 INFO sqlalchemy.engine.Engine COMMIT
2023-04-20 12:15:43,954 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:43,955 INFO sqlalchemy.engine.Engine
CREATE TABLE test (
	x FLOAT,
	y FLOAT
)


2023-04-20 12:15:43,955 INFO sqlalchemy.engine.Engine [no key 0.00052s] ()
2023-04-20 12:15:44,036 INFO sqlalchemy.engine.Engine COMMIT
2023-04-20 12:15:44,037 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,039 INFO sqlalchemy.engine.Engine INSERT INTO test (x, y) VALUES (?, ?)
2023-04-20 12:15:44,040 INFO sqlalchemy.engine.Engine [generated in 0.00131s] [(-17.5, 14492.095), (19.0, 0.4807038), (6.5, 274.0855), (15.9, 8039.7925), (-14.3, -5848.08), (0.3, 4.4066796), (0.1, -1.2131777), (4.8, 109.8
1799)  ... displaying 10 of 100 total bound parameter sets ...  (6.7, 601.3089), (3.9, 60.296154)]
2023-04-20 12:15:44,043 INFO sqlalchemy.engine.Engine COMMIT
```

Out[54]:
```
100
```

In [55]:
```python
# Querying the test table from the database

query9 = "SELECT * FROM test LIMIT 5"
result9 = conn.execute(text(query9))
for res9 in result9:
    print(res9)
```

```
2023-04-20 12:15:44,144 INFO sqlalchemy.engine.Engine SELECT * FROM test LIMIT 5
2023-04-20 12:15:44,145 INFO sqlalchemy.engine.Engine [generated in 0.00113s] ()
(-17.5, 14492.095)
(19.0, 0.4807038)
(6.5, 274.0855)
(15.9, 8039.7925)
(-14.3, -5848.08)
```

In [56]:
```python
# Ingesting the idealfour Dataframe into the train sql table.

idealfour.to_sql("idealfour", engine, if_exists="replace", index=False)
```

```
2023-04-20 12:15:44,256 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,257 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("idealfour")
2023-04-20 12:15:44,258 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,259 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:44,260 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,262 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("idealfour")
2023-04-20 12:15:44,262 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,263 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:44,264 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,266 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER BY name
2023-04-20 12:15:44,266 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,268 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_temp_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER BY name
2023-04-20 12:15:44,268 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,268 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("idealfour")
2023-04-20 12:15:44,270 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,271 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:44,271 INFO sqlalchemy.engine.Engine [raw sql] ('idealfour',)
2023-04-20 12:15:44,273 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list("idealfour")
2023-04-20 12:15:44,273 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,274 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list("idealfour")
2023-04-20 12:15:44,275 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,276 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:44,277 INFO sqlalchemy.engine.Engine [raw sql] ('idealfour',)
2023-04-20 12:15:44,279 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("idealfour")
2023-04-20 12:15:44,279 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,282 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("idealfour")
2023-04-20 12:15:44,283 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,284 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("idealfour")
2023-04-20 12:15:44,286 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,288 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("idealfour")
2023-04-20 12:15:44,289 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,290 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("idealfour")
2023-04-20 12:15:44,290 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,291 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("idealfour")
2023-04-20 12:15:44,293 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,296 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:44,297 INFO sqlalchemy.engine.Engine [raw sql] ('idealfour',)
2023-04-20 12:15:44,299 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:44,300 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,302 INFO sqlalchemy.engine.Engine 
DROP TABLE idealfour
2023-04-20 12:15:44,304 INFO sqlalchemy.engine.Engine [no key 0.00187s] ()
2023-04-20 12:15:44,386 INFO sqlalchemy.engine.Engine COMMIT
2023-04-20 12:15:44,387 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,388 INFO sqlalchemy.engine.Engine 
CREATE TABLE idealfour (
        x FLOAT, 
        y21 FLOAT, 
        y27 FLOAT, 
        y2 FLOAT, 
        y24 FLOAT
)


2023-04-20 12:15:44,389 INFO sqlalchemy.engine.Engine [no key 0.00062s] ()
2023-04-20 12:15:44,470 INFO sqlalchemy.engine.Engine COMMIT
2023-04-20 12:15:44,471 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,473 INFO sqlalchemy.engine.Engine INSERT INTO idealfour (x, y21, y27, y2, y24) VALUES (?, ?, ?, ?, ?)
2023-04-20 12:15:44,473 INFO sqlalchemy.engine.Engine [generated in 0.00176s] [(-20.0, -8000.0, 10648.0, 0.40808207, -16000.0), (-19.9, -7880.599, 10503.459, 0.4971858, -15761.198), (-19.8, -7762.392, 10360.232, 0.5813218
4, -15524.784), (-19.7, -7645.373, 10218.313, 0.65964943, -15290.746), (-19.6, -7529.536, 10077.696, 0.7313861, -15059.072), (-19.5, -7414.875, 9938.375, 0.795815, -14829.75), (-19.4, -7301.384, 9800.344, 0.8522923, -1460
2.768), (-19.3, -7189.057, 9663.597, 0.90025383, -14378.114)  ... displaying 10 of 400 total bound parameter sets ...  (19.8, 7762.392, -5639.752, 0.58132184, 15524.784), (19.9, 7880.599, -5735.339, 0.4971858, 15761.198)]
2023-04-20 12:15:44,477 INFO sqlalchemy.engine.Engine COMMIT
```

Out[56]:

```
400
```

In [57]:

```python
# Querying the idealfour table from the database

query10 = "SELECT * FROM test LIMIT 5"
result10 = conn.execute(text(query10))
for res10 in result10:
    print(res10)
```

```
2023-04-20 12:15:44,567 INFO sqlalchemy.engine.Engine SELECT * FROM test LIMIT 5
2023-04-20 12:15:44,568 INFO sqlalchemy.engine.Engine [cached since 0.4245s ago] ()
(-17.5, 14492.095)
(19.0, 0.4807038)
(6.5, 274.0855)
(15.9, 8039.7925)
(-14.3, -5848.08)
```

In [58]:

```python
# Ingesting the fit Dataframe into the train sql table.

fit.to_sql("fit", engine, if_exists="replace", index=False)
```

```
2023-04-20 12:15:44,747 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,749 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("fit")
2023-04-20 12:15:44,751 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,753 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:44,755 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,756 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("fit")
2023-04-20 12:15:44,757 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,758 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:44,759 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,760 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER BY name
2023-04-20 12:15:44,760 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,764 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_temp_master WHERE type='table' AND name NOT LIKE 'sqlite~_%' ESCAPE '~' ORDER BY name
2023-04-20 12:15:44,766 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,767 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("fit")
2023-04-20 12:15:44,768 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,769 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:44,770 INFO sqlalchemy.engine.Engine [raw sql] ('fit',)
2023-04-20 12:15:44,772 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list("fit")
2023-04-20 12:15:44,773 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,774 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list("fit")
2023-04-20 12:15:44,775 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,777 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:44,778 INFO sqlalchemy.engine.Engine [raw sql] ('fit',)
2023-04-20 12:15:44,779 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("fit")
2023-04-20 12:15:44,780 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,781 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("fit")
2023-04-20 12:15:44,781 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,782 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("fit")
2023-04-20 12:15:44,783 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,784 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("fit")
2023-04-20 12:15:44,785 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,785 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("fit")
2023-04-20 12:15:44,786 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,787 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("fit")
2023-04-20 12:15:44,788 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-04-20 12:15:44,789 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * FROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND type in ('table', 'view')
2023-04-20 12:15:44,790 INFO sqlalchemy.engine.Engine [raw sql] ('fit',)
2023-04-20 12:15:44,793 INFO sqlalchemy.engine.Engine ROLLBACK
2023-04-20 12:15:44,794 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,795 INFO sqlalchemy.engine.Engine 
DROP TABLE fit
2023-04-20 12:15:44,796 INFO sqlalchemy.engine.Engine [no key 0.00170s] ()
2023-04-20 12:15:44,877 INFO sqlalchemy.engine.Engine COMMIT
2023-04-20 12:15:44,879 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,880 INFO sqlalchemy.engine.Engine 
CREATE TABLE fit (
        x FLOAT, 
        y24 FLOAT
)


2023-04-20 12:15:44,881 INFO sqlalchemy.engine.Engine [no key 0.00094s] ()
2023-04-20 12:15:44,961 INFO sqlalchemy.engine.Engine COMMIT
2023-04-20 12:15:44,962 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-04-20 12:15:44,963 INFO sqlalchemy.engine.Engine INSERT INTO fit (x, y24) VALUES (?, ?)
2023-04-20 12:15:44,964 INFO sqlalchemy.engine.Engine [generated in 0.00114s] [(-19.3, -14378.114), (-19.2, -14155.776), (-18.7, -13078.406), (-18.6, -12869.712), (-17.5, -10718.75), (-17.4, -10536.048), (-17.0, -9826.0), 
(-16.6, -9148.592)  ... displaying 10 of 92 total bound parameter sets ...  (19.4, 14602.768), (19.6, 15059.072)]
2023-04-20 12:15:44,967 INFO sqlalchemy.engine.Engine COMMIT
```

Out[58]:

```
92
```

In [59]:

```python
# Querying the idealfour table from the database

query11 = "SELECT * FROM test LIMIT 5"
result11 = conn.execute(text(query11))
for res11 in result11:
    print(res11)
```

```
2023-04-20 12:15:45,067 INFO sqlalchemy.engine.Engine SELECT * FROM test LIMIT 5
2023-04-20 12:15:45,067 INFO sqlalchemy.engine.Engine [cached since 0.9237s ago] ()
(-17.5, 14492.095)
(19.0, 0.4807038)
(6.5, 274.0855)
(15.9, 8039.7925)
(-14.3, -5848.08)
```