CHS UNIT-3 Task

GROUP MEMBERS

Gurpreet Singh(2021UCS1632)

Jatish Kumar(2021UCS1647)

Raghav Sahay(2021UCS1648)

SUBTASK-1

Aim- Create a table Employee(empid, gender, department, salary, country, year_of_joining)
connect to Employee data file.
Remove missing gender and department values.
Extract year_of_joining column and visualize number of employees w.r.t year of experience in the company.
Perform self-join using Power Query.
Aggregate salary with gender and Visualize using Pie chart.

What is Power BI-

Power BI is a powerful business analytics tool developed by Microsoft. It allows users to visualize and share insights from data in a more interactive and engaging way. It offers a suite of business analytics tools that enable users to connect to a wide range of data sources, transform data, create visualizations, and share insights across their organization. Power BI can be used by business analysts, data scientists, and decision-makers to gain actionable insights from data and make informed decisions. It supports a variety of data sources, including Excel, SQL Server, and cloud-based sources like Google Analytics and Salesforce.

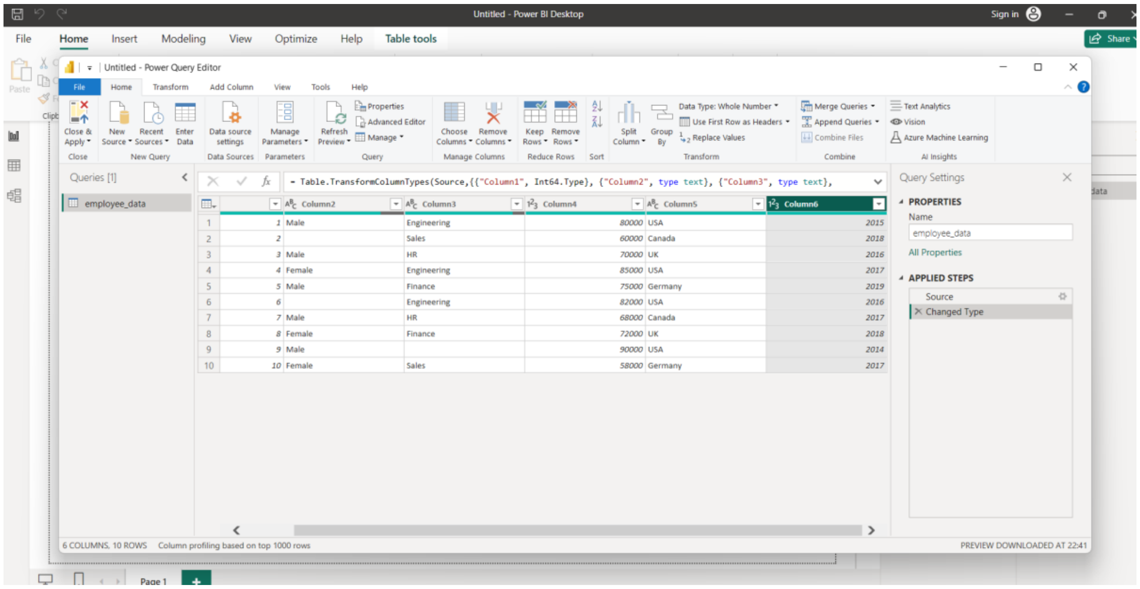
Step 1: Creating a table of employee

->Open Power Query Editor.

->Go to Home > New Source > File > Excel.

->Locate and select your Employee data file.

->Click on "Load" to load the data into Power Query.



```
= Table.TransformColumnTypes(Source,{{"Column1", Int64.Type}, {"Column2", type text}, {"Column3", type text},
```

		Column2	Column3	Column4	Column5	Column6
1	1	Male	Engineering	80000	USA	2015
2	2		Sales	60000	Canada	2018
3	3	Male	HR	70000	UK	2016
4	4	Female	Engineering	85000	USA	2017
5	5	Male	Finance	75000	Germany	2019
6	6		Engineering	82000	USA	2016
7	7	Male	HR	68000	Canada	2017
8	8	Female	Finance	72000	UK	2018
9	9	Male		90000	USA	2014
10	10	Female	Sales	58000	Germany	2017

Query Settings

PROPERTIES

Name: employee_data

All Properties

APPLIED STEPS

Source

Changed Type

6 COLUMNS, 10 ROWS   Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 22:41

Step 2: Removing missing gender and department values

->In Power Query, select the 'Gender' and 'Department' columns.

->Go to Home > Remove Rows > Remove Blank Rows. This will remove rows where either Gender or Department is missing.

Step 3: Extracting year _of _joining column and visualize number of employees w.r.t year of experience in the company.

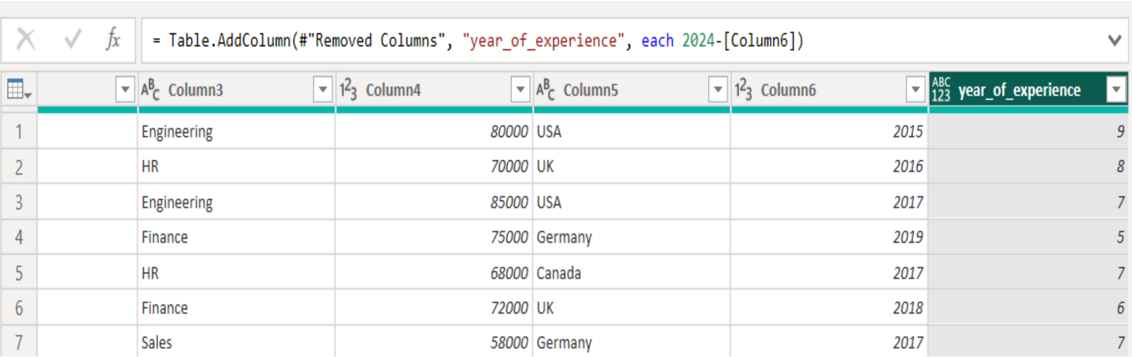
->Select the 'year_of_joining' column.

->Go to Transform > Extract > Year. This will extract the year from the date.

->Select the 'year_of_joining' column.

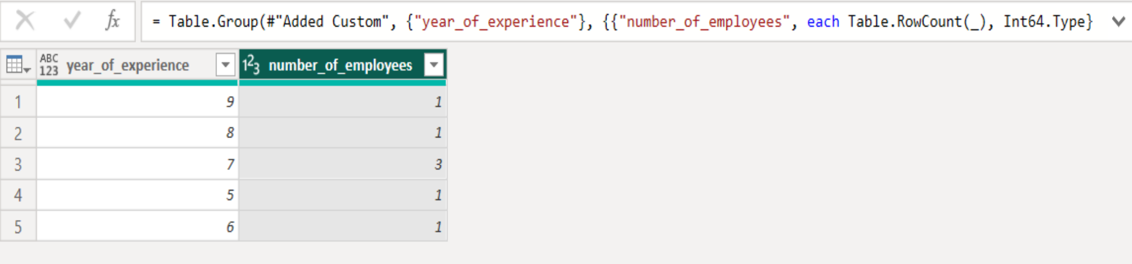
->Go to Home > Group By. Choose 'year_of_joining' as the "Group by" column.

-> Visualize the result.



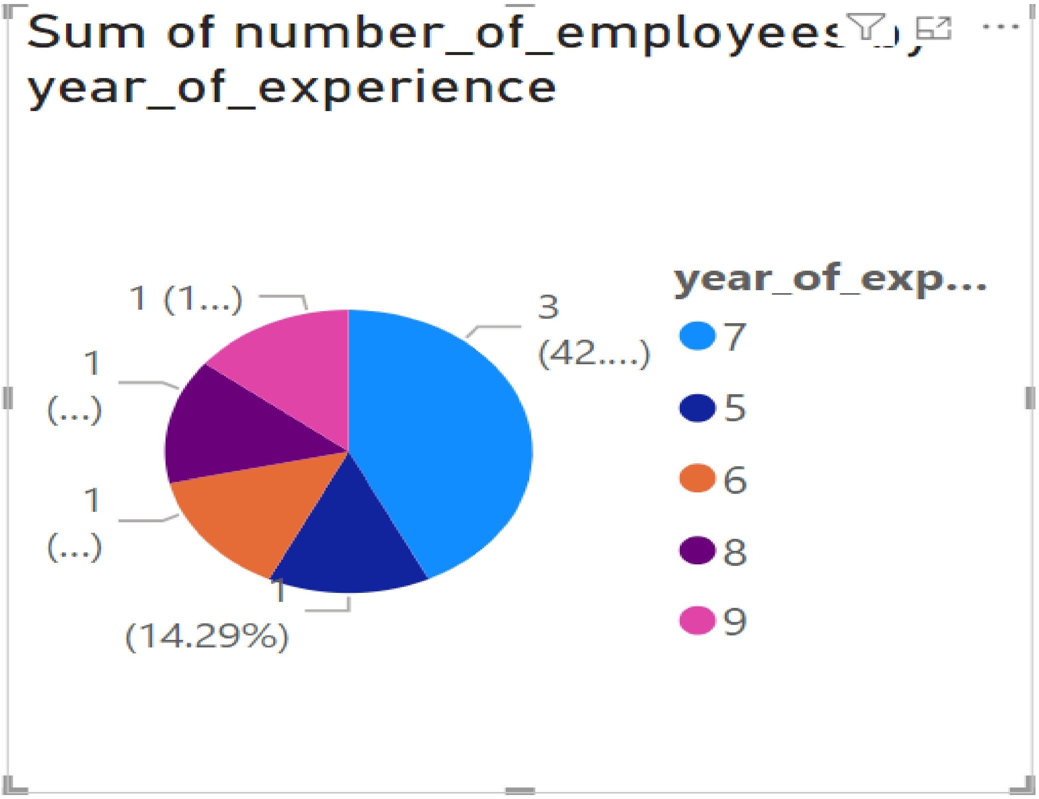
```
= Table.AddColumn(#"Removed Columns", "year_of_experience", each 2024-[Column6])
```

		Column3	Column4	Column5	Column6	year_of_experience
1		Engineering	80000	USA	2015	9
2		HR	70000	UK	2016	8
3		Engineering	85000	USA	2017	7
4		Finance	75000	Germany	2019	5
5		HR	68000	Canada	2017	7
6		Finance	72000	UK	2018	6
7		Sales	58000	Germany	2017	7



```
= Table.Group(#"Added Custom", {"year_of_experience"}, {{"number_of_employees", each Table.RowCount(_), Int64.Type}
```

	year_of_experience	number_of_employees
1	9	1
2	8	1
3	7	3
4	5	1
5	6	1



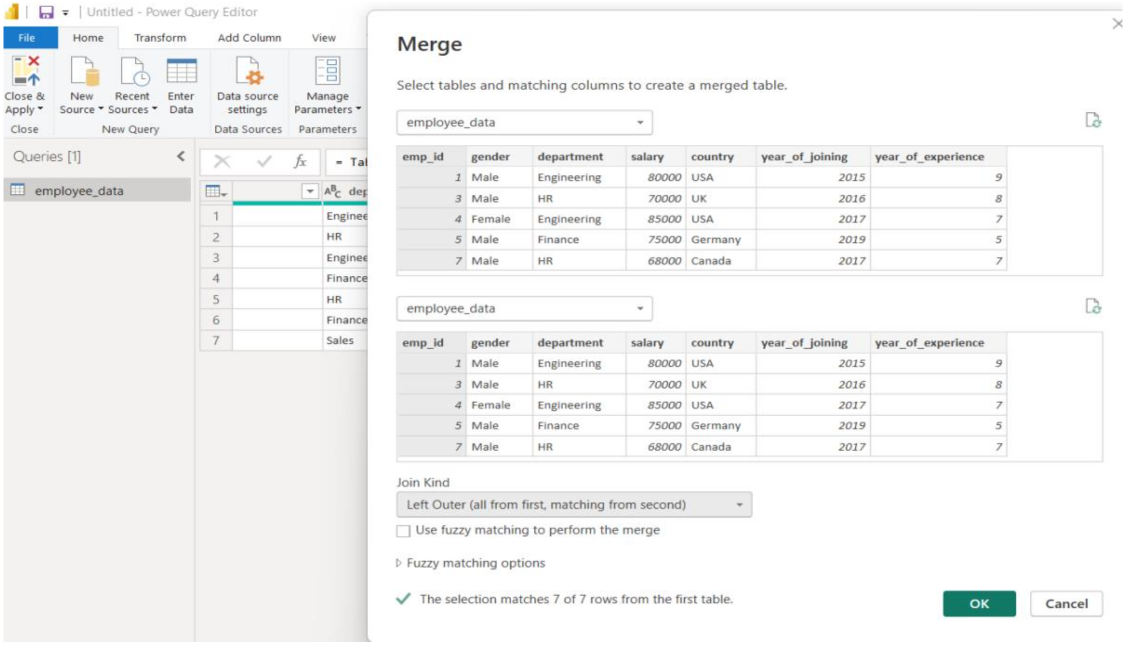
Step 4: Performing self-join

->You can perform a self-join by merging the data with itself based on a common column (e.g., 'emp_id').

->Go to Home > Merge Queries > Merge Queries as New. Select the 'Employee' table as the second table.

->Choose 'emp_id' as the common column. Choose "Left Outer" as the "Join Kind".

->Click OK. This will create a new table with the merged data.



Merge

Select tables and matching columns to create a merged table.

employee_data

emp_id	gender	department	salary	country	year_of_joining	year_of_experience
1	Male	Engineering	80000	USA	2015	9
3	Male	HR	70000	UK	2016	8
4	Female	Engineering	85000	USA	2017	7
5	Male	Finance	75000	Germany	2019	5
7	Male	HR	68000	Canada	2017	7

employee_data

emp_id	gender	department	salary	country	year_of_joining	year_of_experience
1	Male	Engineering	80000	USA	2015	9
3	Male	HR	70000	UK	2016	8
4	Female	Engineering	85000	USA	2017	7
5	Male	Finance	75000	Germany	2019	5
7	Male	HR	68000	Canada	2017	7

Join Kind

Left Outer (all from first, matching from second)

□ Use fuzzy matching to perform the merge

▷ Fuzzy matching options

✓ The selection matches 7 of 7 rows from the first table.

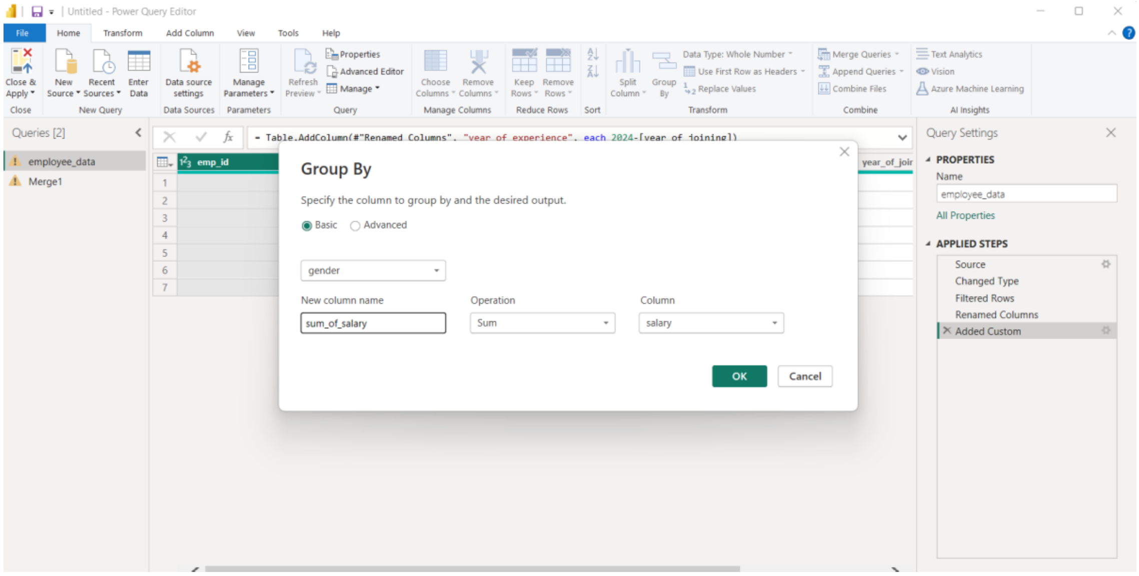
Step 5: Aggregating salary with gender

->Select the 'gender' and 'salary' columns.

->Go to Home > Group By. Choose 'gender' as the "Group by" column.

->Choose "Sum" as the "Aggregate column" and choose 'Salary' as the "Column to sum". Click OK. This will give you the total salary for each gender.

->Now visualize using the pie chart.



```
= Table.AddColumn(#"Renamed Columns", "year of experience", each 2024-[year of joining])
```

Group By

Specify the column to group by and the desired output.

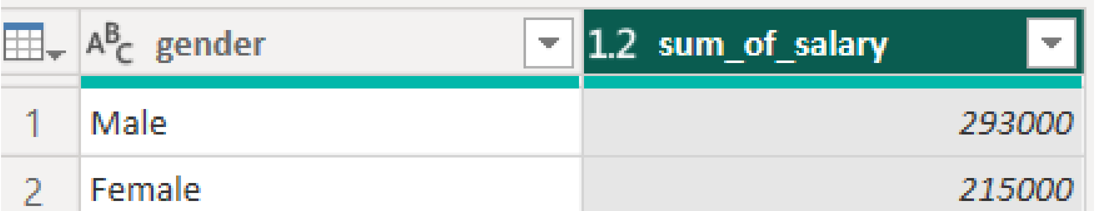
● Basic ○ Advanced

gender

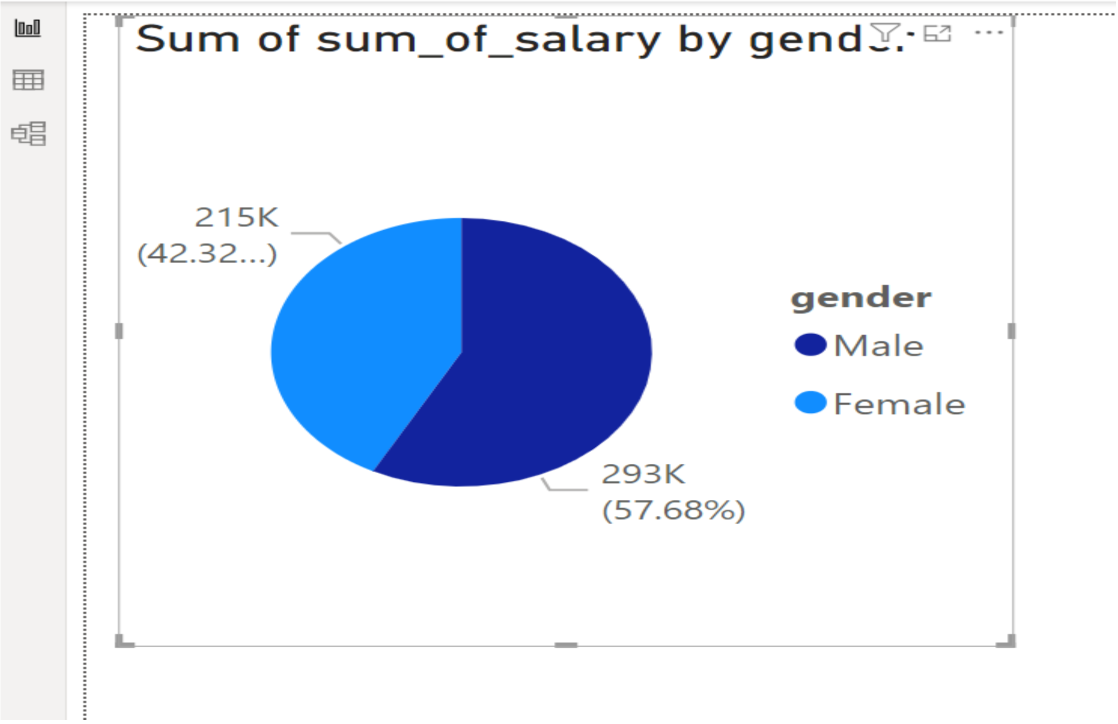
New column name: sum_of_salary

Operation: Sum

Column: salary



	gender	sum_of_salary
1	Male	293000
2	Female	215000



SUBTASK-2

Aim- Visualize the result of any Machine Learning algorithm on any dataset of your choice in PowerBI.

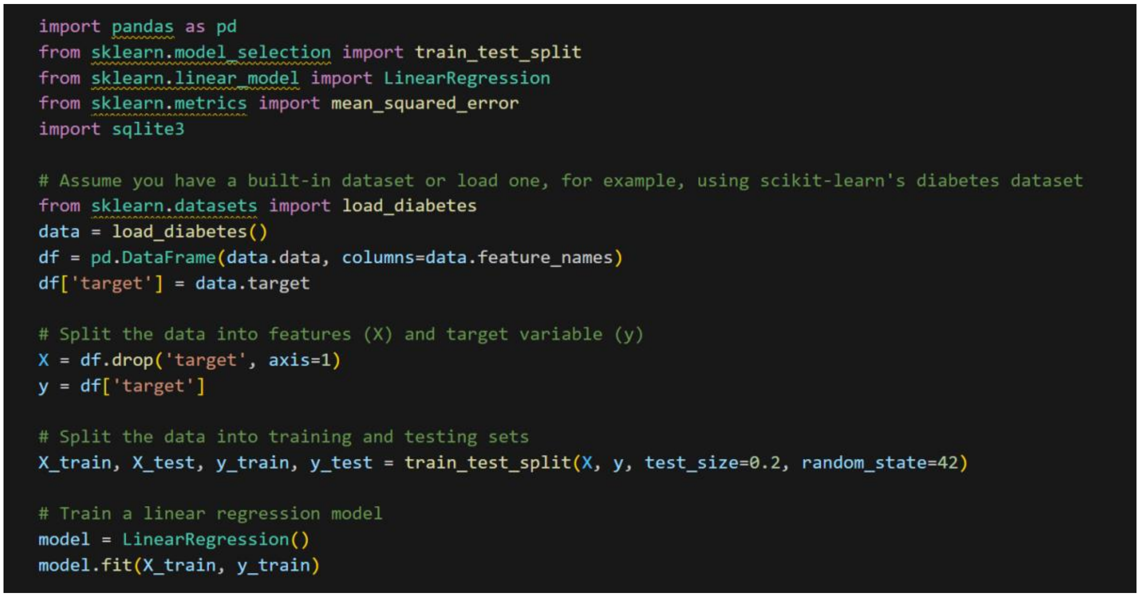
Steps:

->Used a simple linear regression model and its result is put into an excel file.

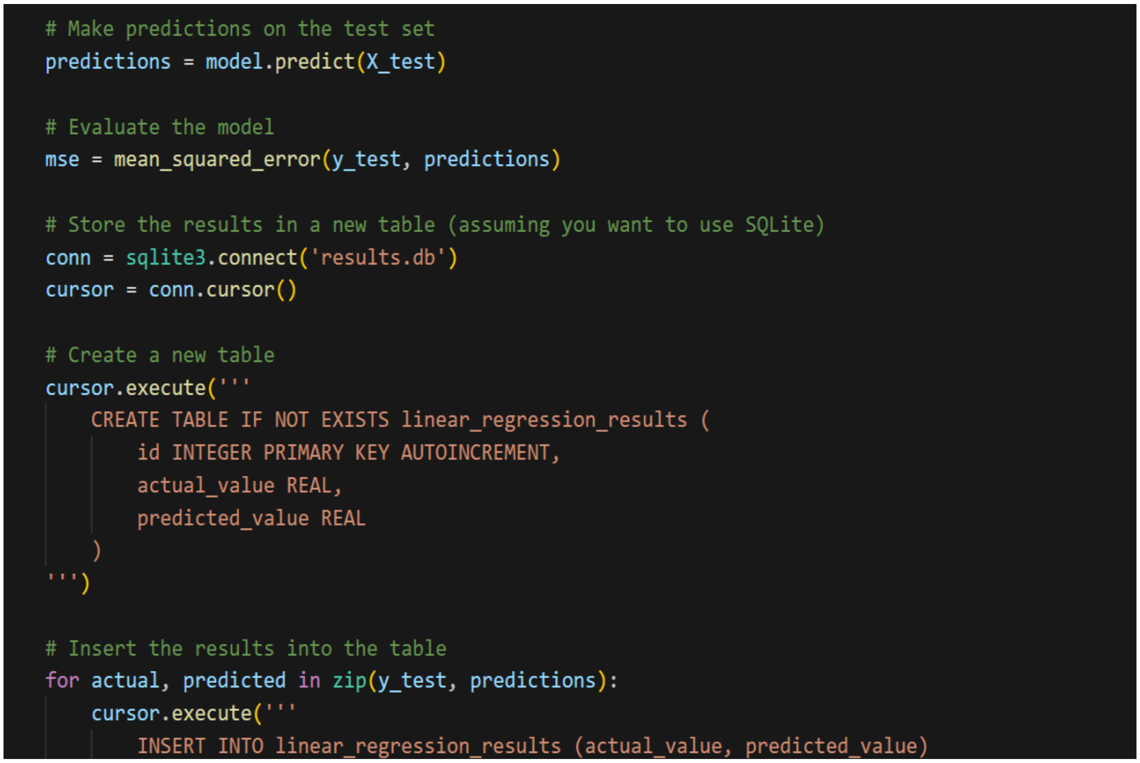
->The excel file is then loaded into power BI.

->Scatter plot is used to visualize the the result of linear regression model

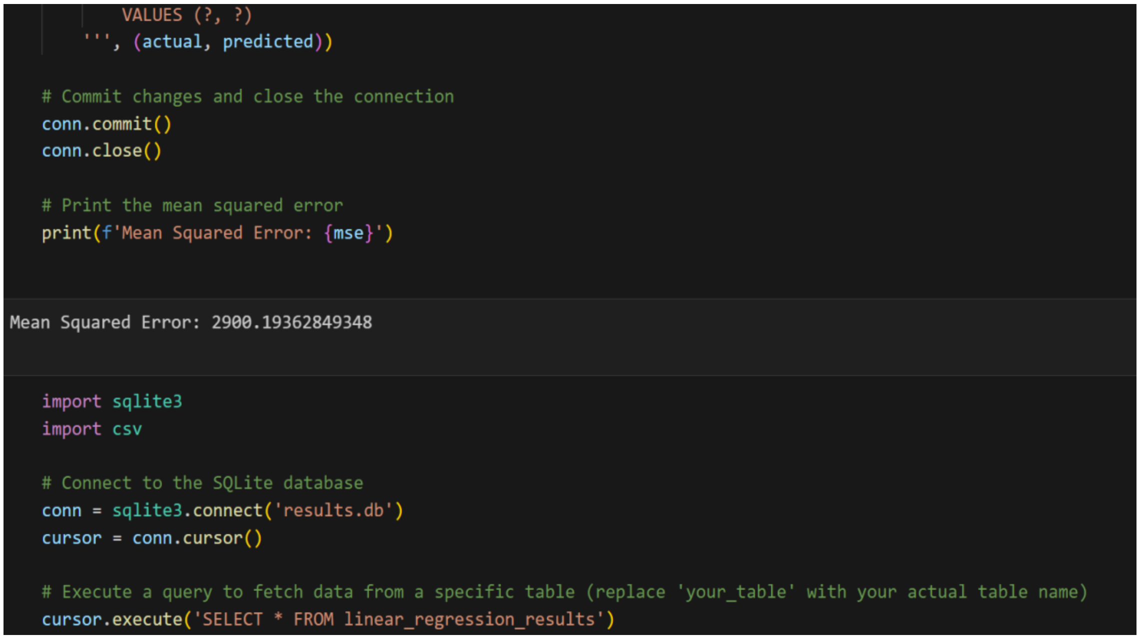
Code:



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import sqlite3

# Assume you have a built-in dataset or load one, for example, using scikit-learn's diabetes dataset
from sklearn.datasets import load_diabetes
data = load_diabetes()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

# Split the data into features (X) and target variable (y)
X = df.drop('target', axis=1)
y = df['target']

# 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)

# Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```



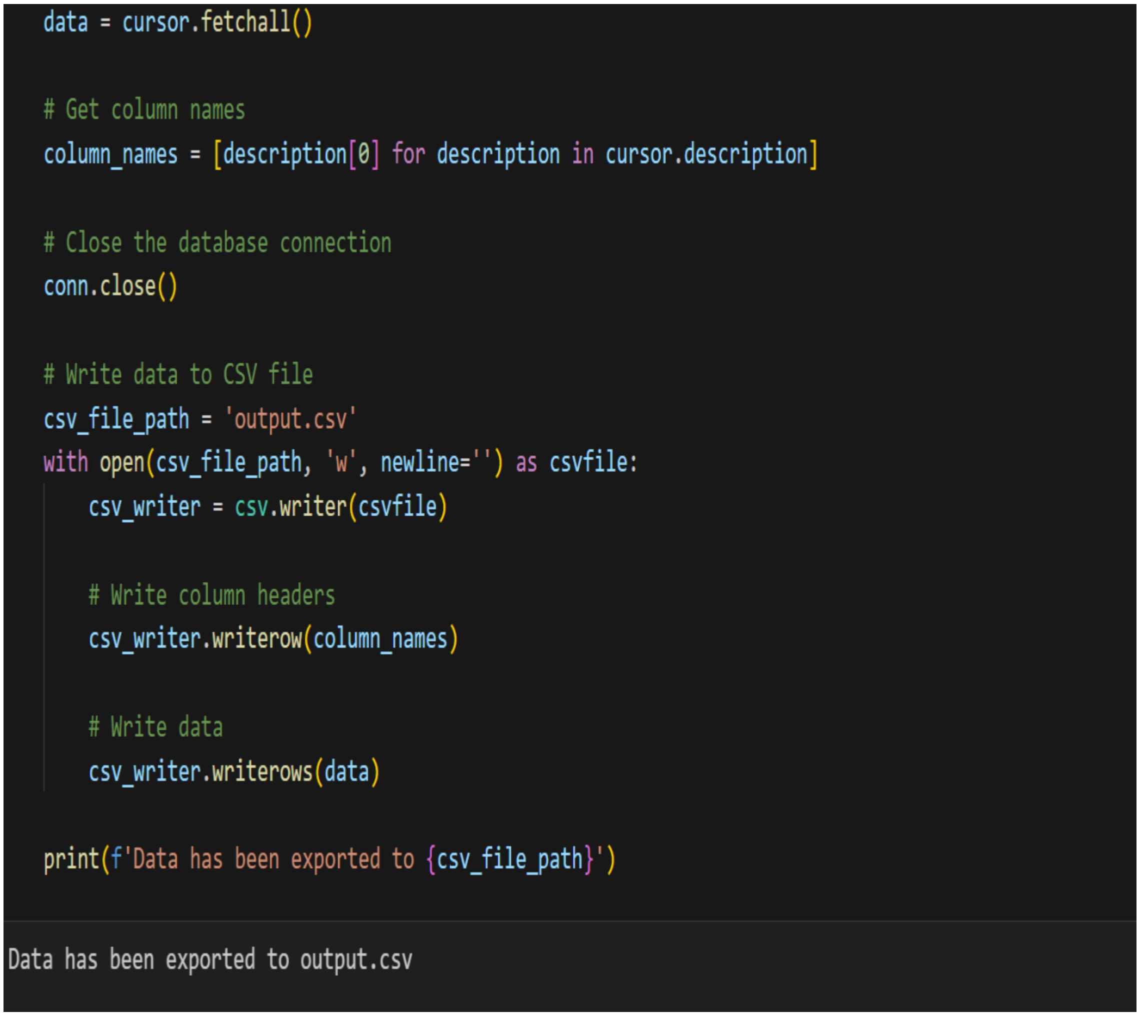
```python
# Make predictions on the test set
predictions = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, predictions)

# Store the results in a new table (assuming you want to use SQLite)
conn = sqlite3.connect('results.db')
cursor = conn.cursor()

# Create a new table
cursor.execute('''
    CREATE TABLE IF NOT EXISTS linear_regression_results (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        actual_value REAL,
        predicted_value REAL
    )
''')

# Insert the results into the table
for actual, predicted in zip(y_test, predictions):
    cursor.execute('''
        INSERT INTO linear_regression_results (actual_value, predicted_value)
```



```
        VALUES (?, ?)
    ''', (actual, predicted))

# Commit changes and close the connection
conn.commit()
conn.close()

# Print the mean squared error
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 2900.19362849348
```

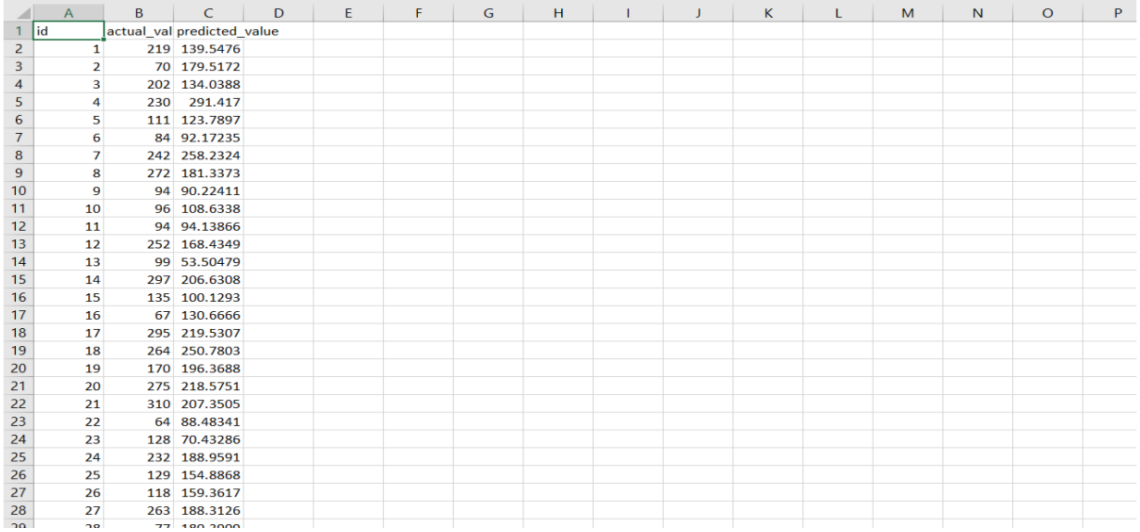
```
import sqlite3
import csv

# Connect to the SQLite database
conn = sqlite3.connect('results.db')
cursor = conn.cursor()

# Execute a query to fetch data from a specific table (replace 'your_table' with your actual table name)
cursor.execute('SELECT * FROM linear_regression_results')
```



```
data = cursor.fetchall()

# Get column names
column_names = [description[0] for description in cursor.description]

# Close the database connection
conn.close()

# Write data to CSV file
csv_file_path = 'output.csv'
with open(csv_file_path, 'w', newline='') as csvfile:
    csv_writer = csv.writer(csvfile)

    # Write column headers
    csv_writer.writerow(column_names)

    # Write data
    csv_writer.writerows(data)

print(f'Data has been exported to {csv_file_path}')
```

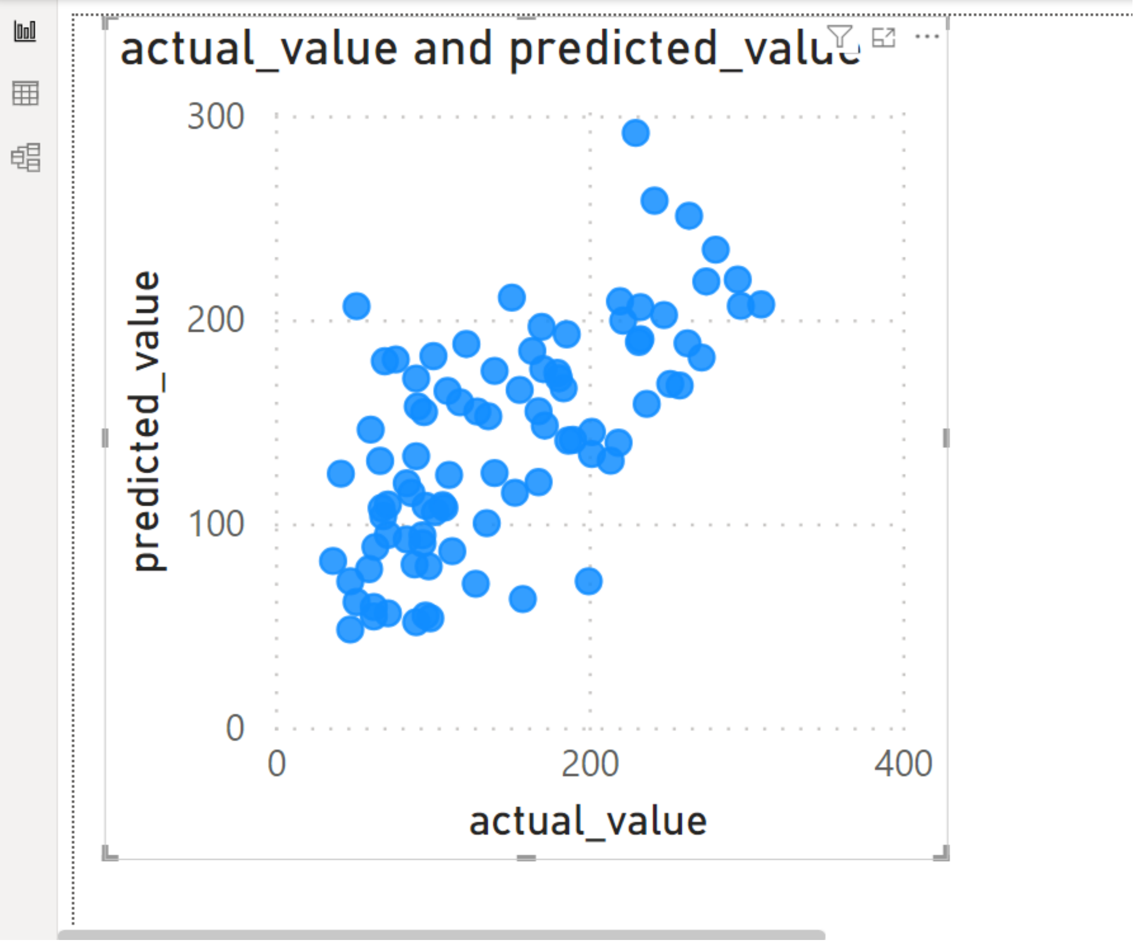
```
Data has been exported to output.csv
```

Glimpse of output excel file:



id	actual_val	predicted_value
1	219	139.5476
2	70	179.5172
3	202	134.0388
4	230	291.417
5	111	123.7897
6	84	92.17235
7	242	258.2324
8	272	181.3373
9	94	90.22411
10	96	108.6338
11	94	94.13866
12	252	168.4349
13	99	53.50479
14	297	206.6308
15	135	100.1293
16	67	130.6666
17	295	219.5307
18	264	250.7803
19	170	196.3688
20	275	218.5751
21	310	207.3505
22	64	88.48341
23	128	70.43286
24	232	188.9591
25	129	154.8868
26	118	159.3617
27	263	188.3126

Visualization :


actual_value and predicted_value
predicted_value
actual_value