Final Exam

AI-Lab

FA21-BCS-136

ZAID ASGHAR VIRK

QUESTION NO 1:

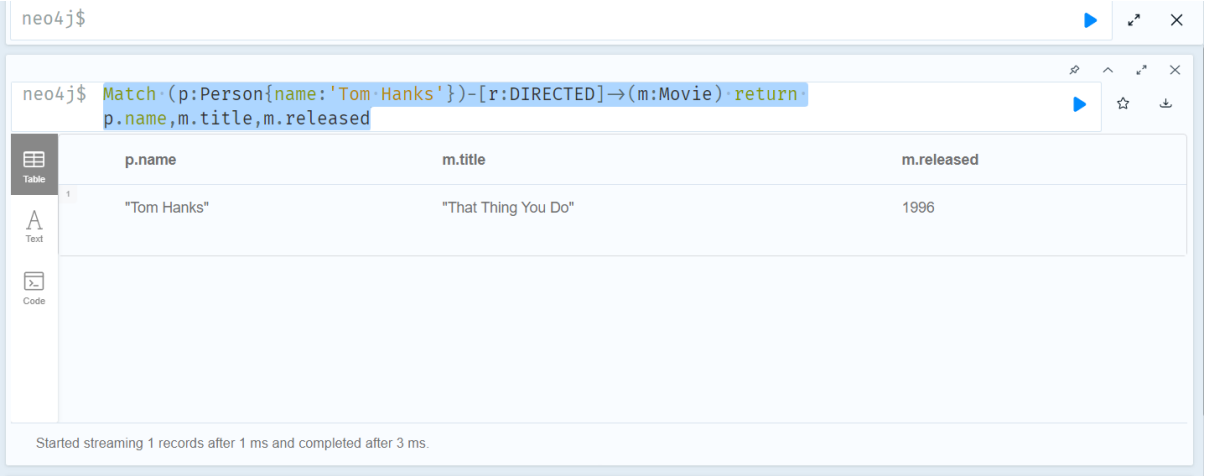
P1.

QUERY:

Match (p:Person{name:'Tom Hanks'})-[r:DIRECTED]->(m:Movie) return p.name,m.title

For the released year too:

Match (p:Person{name:'Tom Hanks'})-[r:DIRECTED]->(m:Movie) return p.name,m.title,m.released



```
neo4j$ Match (p:Person{name:'Lana Wachowski'})-[r:DIRECTED]→(m:Movie) return p.name,m.title
```

	p.name	m.title
1	"Lana Wachowski"	"The Matrix"
2	"Lana Wachowski"	"The Matrix Reloaded"
3	"Lana Wachowski"	"The Matrix Revolutions"
4	"Lana Wachowski"	"Cloud Atlas"
5	"Lana Wachowski"	"Speed Racer"

Started streaming 5 records after 2 ms and completed after 3 ms.

P2:

Query:

```
MATCH (p:Person)-[:DIRECTED]->(m:Movie)
WHERE m.released > 2000
RETURN m.title AS MovieTitle, p.name AS Director
```

Explanation, Multiple Directors against each movie are being displayed.

```
MATCH (p:Person)-[:DIRECTED]→(m:Movie)
WHERE m.released > 2000
RETURN m.title AS MovieTitle, p.name AS Director
```

	MovieTitle	Director
1	"The Matrix Reloaded"	"Lana Wachowski"
2	"The Matrix Reloaded"	"Andy Wachowski"
3	"The Matrix Revolutions"	"Lana Wachowski"
4	"The Matrix Revolutions"	"Andy Wachowski"
5	"RescueDawn"	"Werner Herzog"
6	"Cloud Atlas"	"Tom Tykwer"
7		

Started streaming 17 records after 9 ms and completed after 11 ms.

```
neo4j$ MATCH (p:Person)-[:DIRECTED]->(m:Movie) WHERE m.released > 2000 RETURN m.title AS MovieTitle, p.name AS Director, m.released
```

	MovieTitle	Director	m.released
12	"Speed Racer"	"Lana Wachowski"	2008
13	"Ninja Assassin"	"James Marshall"	2009
14	"Frost/Nixon"	"Ron Howard"	2008
15	"Something's Gotta Give"	"Nancy Meyers"	2003
16	"Charlie Wilson's War"	"Mike Nichols"	2007
17	"The Polar Express"	"Robert Zemeckis"	2004

To Print the name of directors as an array against each movie you can use:

```
MATCH (p:Person)-[:DIRECTED]->(m:Movie)
WHERE m.released > 2000
RETURN m.title AS MovieTitle, collect(p.name) AS Director
```

```
MATCH (p:Person)-[:DIRECTED]->(m:Movie)
WHERE m.released > 2000
RETURN m.title AS MovieTitle, collect(p.name) AS Director
```

	MovieTitle	Director
1	"The Matrix Reloaded"	["Lana Wachowski", "Andy Wachowski"]
2	"The Matrix Revolutions"	["Lana Wachowski", "Andy Wachowski"]
3	"RescueDawn"	["Werner Herzog"]
4	"Cloud Atlas"	["Tom Tykwer", "Lana Wachowski", "Andy Wachowski"]
5	"The Da Vinci Code"	["Ron Howard"]
6	"V for Vendetta"	["James Marshall"]
7		

Started streaming 12 records after 1 ms and completed after 3 ms.

P3:

Query:

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)

WITH p, COUNT(m.title) AS l
WHERE l >=3
RETURN p.name,l
```

```
MATCH (p:Person)-[:ACTED_IN]→(m:Movie)
WITH p, COUNT(m.title) AS l
WHERE l ≥ 3
RETURN p.name,l
```

	p.name	l
1	"Hugo Weaving"	5
2	"Laurence Fishburne"	3
3	"Carrie-Anne Moss"	3
4	"Keanu Reeves"	7
5	"Cuba Gooding Jr."	4
6	"Kevin Bacon"	3

Started streaming 15 records in less than 1 ms and completed after 1 ms.

Also for sorting:

MATCH (p:Person)-[R:ACTED_IN]->(m:Movie) with p, count(m) as number where number>=3 return p.name, number order by number

```
neo4j$ MATCH (p:Person)-[R:ACTED_IN]→(m:Movie) with p, count(m) as number where number ≥ 3
       return p.name, number order by number
```

	p.name	number
1	"Laurence Fishburne"	3
2	"Carrie-Anne Moss"	3
3	"Kevin Bacon"	3
4	"Tom Cruise"	3
5	"Helen Hunt"	3
6	"Robin Williams"	3

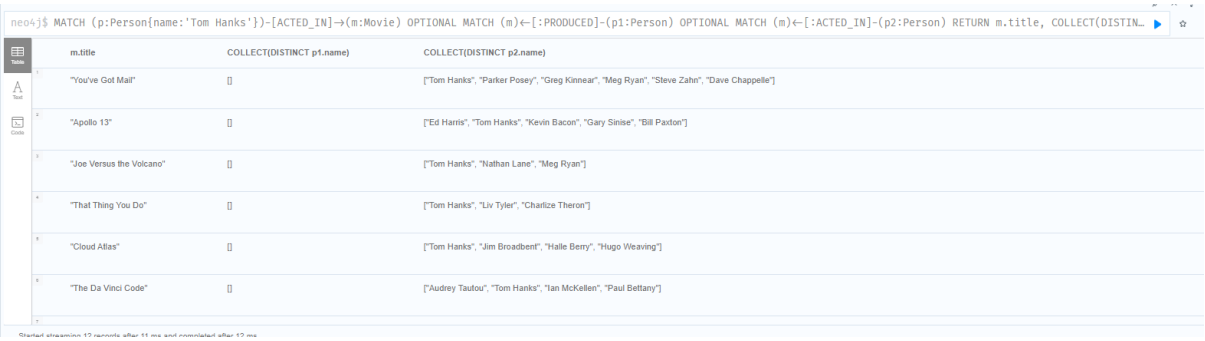
Started streaming 15 records in less than 1 ms and completed after 1 ms.

P4:

Query:

```
MATCH (p:Person{name:'Tom Hanks'})-[ACTED_IN]->(m:Movie)
OPTIONAL MATCH (m)<-[:PRODUCED]-(p1:Person)
OPTIONAL MATCH (m)<-[:ACTED_IN]-(p2:Person)
RETURN m.title, COLLECT(DISTINCT p1.name), COLLECT(DISTINCT p2.name)
```

No producers have been listed against movies tom hanks was actor in.

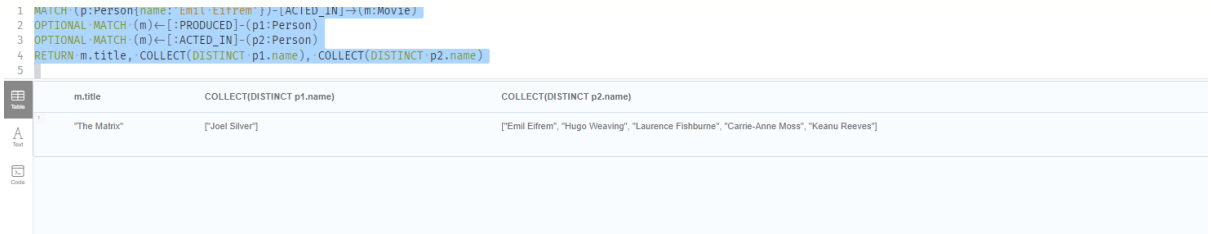


neo4j$ MATCH (p:Person{name:'Tom Hanks'})-[ACTED_IN]→(m:Movie) OPTIONAL MATCH (m)←[:PRODUCED]-(p1:Person) OPTIONAL MATCH (m)←[:ACTED_IN]-(p2:Person) RETURN m.title, COLLECT(DISTIN…

	m.title	COLLECT(DISTINCT p1.name)	COLLECT(DISTINCT p2.name)
1	"You've Got Mail"	[]	["Tom Hanks", "Parker Posey", "Greg Kinnear", "Meg Ryan", "Steve Zahn", "Dave Chappelle"]
2	"Apollo 13"	[]	["Ed Harris", "Tom Hanks", "Kevin Bacon", "Gary Sinise", "Bill Paxton"]
3	"Joe Versus the Volcano"	[]	["Tom Hanks", "Nathan Lane", "Meg Ryan"]
4	"That Thing You Do"	[]	["Tom Hanks", "Liv Tyler", "Charlize Theron"]
5	"Cloud Atlas"	[]	["Tom Hanks", "Jim Broadbent", "Halle Berry", "Hugo Weaving"]
6	"The Da Vinci Code"	[]	["Audrey Tautou", "Tom Hanks", "Ian McKellen", "Paul Bettany"]

TO verify:

```
MATCH (p:Person{name:'Emil Eifrem'})-[ACTED_IN]->(m:Movie)
OPTIONAL MATCH (m)<-[:PRODUCED]-(p1:Person)
OPTIONAL MATCH (m)<-[:ACTED_IN]-(p2:Person)
RETURN m.title, COLLECT(DISTINCT p1.name), COLLECT(DISTINCT p2.name)
```



```
1 MATCH (p:Person{name:'Emil Eifrem'})-[ACTED_IN]->(m:Movie)
2 OPTIONAL MATCH (m)<-[:PRODUCED]-(p1:Person)
3 OPTIONAL MATCH (m)<-[:ACTED_IN]-(p2:Person)
4 RETURN m.title, COLLECT(DISTINCT p1.name), COLLECT(DISTINCT p2.name)
5
```

	m.title	COLLECT(DISTINCT p1.name)	COLLECT(DISTINCT p2.name)
1	"The Matrix"	["Joel Silver"]	["Emil Eifrem", "Hugo Weaving", "Laurence Fishburne", "Carrie-Anne Moss", "Keanu Reeves"]

P4(Without Using Optional)

```
MATCh (n:Person{name:'Tom Hanks'})-[ACTED_IN]->(m:Movie)
MATCH (a:Person)-[:ACTED_IN]->(m)
MATCH (p:Person)-[:PRODUCED]->(m)
return m.title as movietitle, COLLECT(DISTINCT p.name) AS Producers,COLLECT(DISTI
NCT a.name) AS Actors
```

```
MATCh (n:Person{name:'Tom Hanks'})-[ACTED_IN]->(m:Movie)
MATCH (a:Person)-[:ACTED_IN]->(m)
MATCH (p:Person)-[:PRODUCED]->(m)
return m.title as movietitle, COLLECT(DISTINCT p.name) AS Producers,COLLECT(DISTINCT a.name) AS Actors
```

(no changes, no records)

Completed in less than 1 ms.

No producers have been listed against movies tom hanks was actor in.

To verify:

```
MATCh (n:Person)-[ACTED_IN]->(m:Movie)
MATCH (a:Person)-[:ACTED_IN]->(m)
MATCH (p:Person)-[:PRODUCED]->(m)
return m.title as movietitle, COLLECT(DISTINCT p.name) AS Producers,COLLECT(DISTINCT a.name) AS Actors
```

```
MATCh (n:Person)-[ACTED_IN]->(m:Movie)
MATCH (a:Person)-[:ACTED_IN]->(m)
MATCH (p:Person)-[:PRODUCED]->(m)
return m.title as movietitle, COLLECT(DISTINCT p.name) AS Producers,COLLECT(DISTINCT a.name) AS Actors
```

	movietitle	Producers	Actors
1	"The Matrix"	["Joel Silver"]	["Emil Eifrem", "Hugo Weaving", "Laurence Fishburne", "Carrie-Anne Moss", "Keanu Reeves"]
2	"The Matrix Reloaded"	["Joel Silver"]	["Hugo Weaving", "Laurence Fishburne", "Carrie-Anne Moss", "Keanu Reeves"]
3	"The Matrix Revolutions"	["Joel Silver"]	["Hugo Weaving", "Laurence Fishburne", "Carrie-Anne Moss", "Keanu Reeves"]
4	"Jerry Maguire"	["Cameron Crowe"]	["Jerry O'Connell", "Bonnie Hunt", "Jay Mohr", "Cuba Gooding Jr.", "Jonathan Lipnicki", "Renee Zellweger", "Kelly Preston", "Regina King", "Tom Cruise"]
5	"When Harry Met Sally"	["Rob Reiner", "Nora Ephron"]	["Carrie Fisher", "Billy Crystal", "Bruno Kirby", "Meg Ryan"]
6	"V for Vendetta"	["Andy Wachowski", "Lana Wachowski", "Joel Silver"]	["Ben Miles", "Natalie Portman", "John Hurt", "Stephen Rea", "Hugo Weaving"]

Started streaming 9 records after 1 ms and completed after 2 ms.

```
MATCh (n:Person{name:'Emil Eifrem'})-[ACTED_IN]->(m:Movie)
MATCH (a:Person)-[:ACTED_IN]->(m)
MATCH (p:Person)-[:PRODUCED]->(m)
return m.title as movietitle, COLLECT(DISTINCT p.name) AS Producers,COLLECT(DISTINCT a.name) AS Actors
```

```
MATCh (n:Person{name:'Emil Eifrem'})-[ACTED_IN]->(m:Movie)
MATCH (a:Person)-[:ACTED_IN]->(m)
MATCH (p:Person)-[:PRODUCED]->(m)
return m.title as movietitle, COLLECT(DISTINCT p.name) AS Producers,COLLECT(DISTINCT a.name) AS Actors
```

	movietitle	Producers	Actors
1	"The Matrix"	["Joel Silver"]	["Emil Eifrem", "Hugo Weaving", "Laurence Fishburne", "Carrie-Anne Moss", "Keanu Reeves"]

P5:

```
MATCH (m:Movie)
DETACH DELETE m
```

Question#2:

Code:

```
import random

##########################
Size_of_Population = 70
Mutation_Rate = 0.01
GENERATIONS = 1000
TARGET = 50
Error_Thresh_Hold = 2

##########################
def initialize_population1(size):
    population = []
    for _ in range(size):
        x = random.randint(0, TARGET)
        y = random.randint(0, TARGET)
        population.append((x, y))
    return population
#A better alternate using listcomprehension,
def initialize_population(size):
    return [(random.randint(0, TARGET), random.randint(0, TARGET)) for _ in
range(size)]

def fitness(individual):
    x, y = individual
    return abs(3*x + 2*y - TARGET)

def parent_selection(population):
    tournament_size = 5
    selected = random.sample(population, tournament_size)
    selected.sort(key=fitness)
    return selected[0], selected[1]

#Crossover
def crossover(parent1, parent2):
    x1, y1 = parent1
    x2, y2 = parent2
    child1 = (x1, y2)
    child2 = (x2, y1)
    return child1, child2

def mutaion(individual):
```

```python
    x, y = individual
    if random.random() < Mutation_Rate:
        x = random.randint(0, 50)
    if random.random() < Mutation_Rate:
        y = random.randint(0, 50)
    return (x, y)

def next_generation(current_population):
    new_pop = []
    for _ in range(Size_of_Population // 2):
        parent1, parent2 = parent_selection(current_population)
        child1, child2 = crossover(parent1, parent2)
        child1 = mutaion(child1)
        child2 = mutaion(child2)
        new_pop.append(child1)
        new_pop.append(child2)
    return new_pop
#genteic algorithm
def geneti_algo():
    population = initialize_population(Size_of_Population)
    for generation in range(GENERATIONS):
        population.sort(key=fitness)
        best_individual = population[0]
        best_fitness = fitness(best_individual)
        ###Question2,last part Conversion check###
        if best_fitness < Error_Thresh_Hold:
            return best_individual, generation
        population = next_generation(population)
    population.sort(key=fitness)
    return population[0], GENERATIONS

solution, generations = geneti_algo()
print(f"Solution: x = {solution[0]}, y = {solution[1]} after {generations}
generations.")
```

OUTPUT:

GeneticAlgorithm.py > ...

```python
import random

#########################
Size_of_Population = 70
Mutation_Rate = 0.01
GENERATIONS = 1000
TARGET = 50
Error_Thresh_Hold = 2

#########################
def initialize_population1(size):
    population = []
    for _ in range(size):
        x = random.randint(0, TARGET)
        y = random.randint(0, TARGET)
        population.append((x, y))
    return population
#A better alternate using listcomprehension,
def initialize_population(size):
    return [(random.randint(0, TARGET), random.randint(0, TARGET)) for   in range(size)]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER Python

```
PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 15, y = 2 after 0 generations.
PS C:\Users\Zaid\Desktop\final-exam>
```

Welcome | GeneticAlgorithm.py | Linear Regression.py

GeneticAlgorithm.py > ...

```python
import random

#########################
Size_of_Population = 70
Mutation_Rate = 0.01
GENERATIONS = 1000
TARGET = 50
Error_Thresh_Hold = 2

#########################
def initialize_population1(size):
    population = []
    for _ in range(size):
        x = random.randint(0, TARGET)
        y = random.randint(0, TARGET)
        population.append((x, y))
    return population
#A better alternate using listcomprehension,
def initialize_population(size):
    return [(random.randint(0, TARGET), random.randint(0, TARGET)) for   in range(size)]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER Python

```
PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 15, y = 2 after 0 generations.
PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 13, y = 6 after 0 generations.
PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 10, y = 10 after 1 generations.
PS C:\Users\Zaid\Desktop\final-exam>
```

Changing Parameters 1:

```
Size_of_Population = 10
Mutation_Rate = 0.01
GENERATIONS = 100
TARGET = 50
Error_Thresh_Hold = 2
```

OUTPUT:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   JUPYTER                    Python

PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 3, y = 25 after 100 generations.
PS C:\Users\Zaid\Desktop\final-exam>
```

```
##########################
def initialize_population1(size):
    population = []
    for _ in range(size):
        x = random.randint(0, TARGET)
        y = random.randint(0, TARGET)
        population.append((x, y))
    return population
#A better alternate using listcomprehension,
def initialize_population(size):
    return [(random.randint(0, TARGET), random.randint(0, TARGET)) for   in range(size)]
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   JUPYTER                    Python

PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 3, y = 25 after 100 generations.
PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 8, y = 13 after 1 generations.
PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 3, y = 22 after 100 generations.
PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 18, y = 0 after 100 generations.
PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 11, y = 8 after 65 generations.
PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 13, y = 5 after 92 generations.
PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" c:/Users/Zaid/Desktop/final-exam/GeneticAlgorithm.py
Solution: x = 9, y = 12 after 1 generations.
PS C:\Users\Zaid\Desktop\final-exam>
```

Question NO 3:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

df = pd.read_csv('./Datasetfile/FuelConsumptionCo2.csv')

print(df.head())

X = df[['ENGINESIZE']]
y = df['CO2EMISSIONS']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
intercept = model.intercept_
coefficients = model.coef_
print(f'Intercept: {intercept}')
print(f'Coefficients: {coefficients}')
enginecc=2.5
engine_size = pd.DataFrame([[enginecc]], columns=['ENGINESIZE'])
predicted_CO2 = model.predict(engine_size)
print(f'Predicted CO2 emissions for a vehicle with a {enginecc}-liter engine:
{predicted_CO2[0]}')
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    JUPYTER                                    Python

PS C:\Users\Zaid\Desktop\final-exam> & "C:/Program Files/Python312/python.exe" "c:/Users/Zaid/Desktop/final-exam/Linear Regression.py"
   MODELYEAR   MAKE       MODEL VEHICLECLASS  ...  FUELCONSUMPTION_HWY  FUELCONSUMPTION_COMB FUELCONSUMPTION_COMB_MPG CO2EMISSIONS
0       2014  ACURA         ILX      COMPACT  ...                  6.7                   8.5                       33          196
1       2014  ACURA         ILX      COMPACT  ...                  7.7                   9.6                       29          221
2       2014  ACURA  ILX HYBRID      COMPACT  ...                  5.8                   5.9                       48          136
3       2014  ACURA     MDX 4WD  SUV - SMALL  ...                  9.1                  11.1                       25          255
4       2014  ACURA     RDX AWD  SUV - SMALL  ...                  8.7                  10.6                       27          244

[5 rows x 13 columns]
Intercept: 126.28970217408721
Coefficients: [38.99297872]
Predicted CO2 emissions for a vehicle with a 2.5-liter engine: 223.7721489851724
Mean Absolute Error (MAE): 24.09725741170784
Mean Squared Error (MSE): 985.9381692274999
R-squared (R2): 0.7615595731934373
PS C:\Users\Zaid\Desktop\final-exam>
```