Andy Yang
EECS 118 Term Project
12/07/2024

Introduction

This program is based on the dataset found at: https://www.kaggle.com/datasets/arsalanjamal002/student-sleep-patterns and has ten main applications using five relational queries and five non-relational queries. This dataset describes the sleep patterns of university students using sleep duration, surveyed sleep quality, study hours and other student lifestyle factors. The goal of this program is to understand the patterns between university students' lifestyle choices and their quality and quantity of sleep. The program supports different retrieval functions using relational queries that take in user inputs and filters the relational data when presented to the user. It also provides higher capabilities such as plotting said correlations or creating different machine learning models in order to perform predictions.

The program's main loop begins by first loading the data using the function described in load_data, and prints the main menu as shown in Figure 1. This program supports thirteen total functions. The first three functions are operational and aid the user in loading the specified database, reloading the database/reestablishing the connection to the database, and exiting the program.

```
Loaded Database

-----MAIN MENU-----
Please select a query type by typing its number
Enter 0 to exit at anytime
(0) Exit
(1) Load Database
(2) Reload Database
--- Relational Queries ---
(3) What is the average sleep duration given a gender and university year
(4) Find students' sleep hours who have X caffeinated beverage per day
(5) Find caffeine consumption of students who have XXX minutes of exercise
(6) Find all students per a certain university year who get above 6 hours of sleep
(7) Find students with a sleep quality of above a 7/10 with the sleep quantity of less than 6 hours
--- Non-Relational Queries ---
(8) Use linear regression to plot the relation between study hours and sleep hours
(9) Predict sleep quality given study hours, caffeine consumption, and sleep duration using linear regression
(10) Use linear regression to plot the relation between physical activity and sleep hours
(11) Visualize the decision tree model with the target column being Sleep Quality>7)
(12) Predict sleep duration using linear regression given screen time, caffeine intake, and physical activity
```

Fig. 1. Main Menu and Loaded Database status message

The database was loaded into SQL workbench into the schema “student_sleep”

Loading Functions:

Load_Data:

The load data function implements the connect function in Pymysql and connects to the predefined schema “student_sleep” that is loaded into SQL workbench. It sets the global variable “connection” to be accessed throughout the program.

```
def load_data():
    global connection
    ''' option to use inputted database
    host = input("Enter hostname: ")
    user = input("Enter username: ")
    passwd = input("Enter password: ")
    db = input("Enter database to connect to: ")
    connection = pymysql.connect(
        host=host, user=user, passwd=passwd, db=db
    )
    '''
    connection = pymysql.connect(host='localhost', user='mp', passwd='eecs118', db='student_sleep')
    # test loaded database
    ''' option to test and show all tables in the schema
    cur = connection.cursor()
    cur.execute(f"SHOW TABLES")
    for row in cur.fetchall():
        print(row)
    '''
    print("Loaded Database")
```

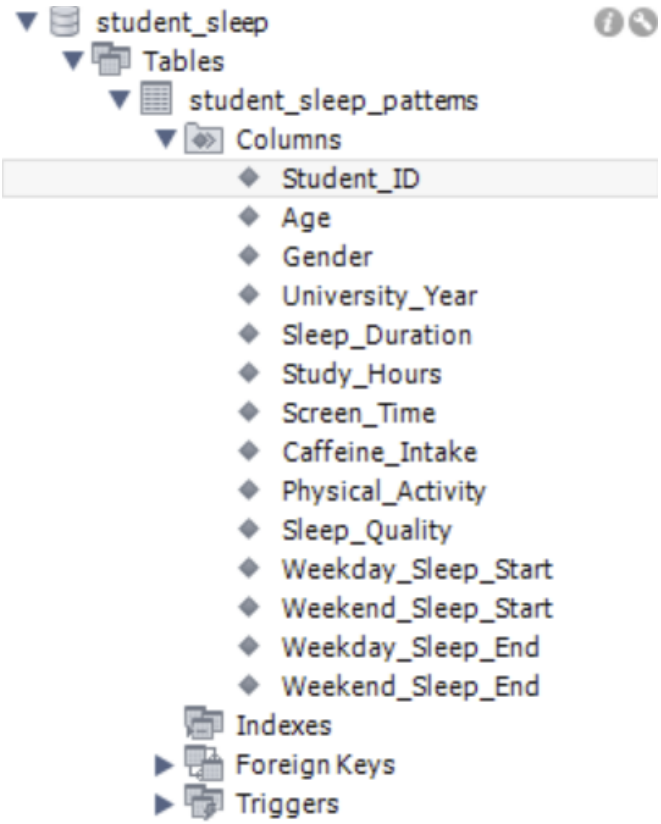

Fig. 2. SQL Workbench schema student_sleep

Reload_Data:

Reload data is a simple function that uses the global variable “connection” and tests if the variable is set. If the connection has been set by load_data, it will close the connection and run the function again.

```
def reload_data():
    global connection
    if connection == None:
        print("No db loaded")
        return
    connection.close()
    print("closed connection")
    load_data()
    print("Reloaded Database")
```

Fig. 3. reload_data() function

Relational Queries:

Function (3): What is the average sleep duration given a gender and university year

Function average() is the first relational query function of the program and takes in two user inputs: Gender, which has three possible answers ‘Male’ ‘Female’ and ‘Other’, and University Year. University year is inputted as an integer and converted back into the string that the database expects. The query then uses the SQL function SELECT AVG on the ‘Sleep Duration’ column while also using conditions. These conditions are the previously discussed user inputs Gender and University_Year. This query is performed to find the average sleep duration in hours for a certain gender and university year.

```
query = f"""
    SELECT AVG(Sleep_Duration)
    FROM student_sleep_patterns
    WHERE Gender = '{Gender}' AND University_Year = '{uni}';
"""
```

Fig. 4. query for average() function

```
3
Enter Gender (Male/Female/Other): Male
Enter university year (1/2/3/4): 3
Average Sleep for Male's and 3rd Year: 6.57 hrs
```

Fig. 5. Example input and output

The query is then executed using the global variable connection and its cursor function.

```python
def average():
    Gender = input("Enter Gender (Male/Female/Other): ")
    University_Year = int(input("Enter university year (1/2/3/4): "))
    if University_Year == 1:
        uni = "1st Year"
    elif University_Year == 2:
        uni = "2nd Year"
    elif University_Year == 3:
        uni = "3rd Year"
    elif University_Year == 4:
        uni = "4th Year"
    query = f"""
        SELECT AVG(Sleep_Duration)
        FROM student_sleep_patterns
        WHERE Gender = '{Gender}' AND University_Year = '{uni}';
    """
    cur = connection.cursor()
    cur.execute(query)
    average = cur.fetchone()
    print(f"Average Sleep for {Gender}'s and {uni}: {average[0]:.2f} hrs")
```

Fig. 6. average() function

Function (4): Find students' sleep hours who have X caffeinated beverage per day

The function caffeine() is the next relational query function and simply uses the SELECT statement from SQL on student id, sleep duration, and caffeine intake columns in conjunction with a conditional statement on caffeine intake. The query must only search in the entries where caffeine intake is higher than the user inputted caffeine intake. This query is then executed using the cursor function. The inspiration for this function comes from wanting to see the amount of caffeinated drinks that students have and which students have the most.

```python
def caffeine():
    caffeine_intake = (input("How many caffeinated drinks (0,1,etc.): "))
    query = f"""
        SELECT Student_ID, Sleep_Duration, Caffeine_Intake FROM student_sleep_patterns
        WHERE Caffeine_Intake = %s;
    """
    cur = connection.cursor()
    cur.execute(query, (caffeine_intake,))
    rows = cur.fetchall()
    print(f"These are the student IDs of students who have {caffeine_intake} caffeinated drinks as well as their amount of sleep")
    for row in rows:
        print(f"Student {row[0]} drank {row[1]} caffeinated drinks")
```

Fig. 7. caffeine() function

```
4
How many caffeinated drinks (0,1,etc.): 2
These are the student IDs of students who have 2 caffeinated drinks as well as their amount of sleep
Student 1 drank 2 caffeinated drinks and slept: 7.7 hours
Student 16 drank 2 caffeinated drinks and slept: 8.8 hours
Student 23 drank 2 caffeinated drinks and slept: 4.3 hours
Student 30 drank 2 caffeinated drinks and slept: 6.4 hours
Student 31 drank 2 caffeinated drinks and slept: 6.8 hours
Student 36 drank 2 caffeinated drinks and slept: 6.2 hours
Student 41 drank 2 caffeinated drinks and slept: 8.3 hours
Student 44 drank 2 caffeinated drinks and slept: 5.4 hours
Student 49 drank 2 caffeinated drinks and slept: 7.6 hours
Student 56 drank 2 caffeinated drinks and slept: 4.5 hours
Student 57 drank 2 caffeinated drinks and slept: 8.0 hours
```

Fig. 8. Example input and output. Output is cut off due to large length

Function (5): Find caffeine consumption of students who have XXX minutes of exercise

The function caffeine_exercise takes in an input for physical activity in terms of minutes of exercise. The relational query is used to find caffeine intake for students who have a certain amount of exercise per day in minutes. The inspiration behind this function is to find how much caffeine students intake compared to how much they exercise.

```python
def caffeine_exercise():
    print("Example input string: 240")
    exercise = (input("How many minutes of exercise (XXX): "))
    query = f"""
        SELECT Student_ID, Caffeine_Intake, Physical_Activity FROM student_sleep_patterns
        WHERE Physical_Activity = %s;
    """
    cur = connection.cursor()
    cur.execute(query, (exercise,))
    rows = cur.fetchall()
    print(f"These are the student IDs of students and their caffeine drink consumption if they have {exercise} minutes of exercise: ")
    for row in rows:
        print(f"Student {row[0]} drank {row[1]} drinks")
```

Fig. 9. caffeine_exercise() function

```
5
Example input string: 240
How many minutes of exercise (XXX): 15
These are the student IDs of students and their caffeine drink consumption if they have 15 minutes of exercise:
Student 77 drank 4 drinks
Student 107 drank 1 drinks
Student 250 drank 1 drinks
```

Fig. 10. Example input and output

Function (6): Find all students per a certain university year who get above 6 hours of sleep

The function find_above_six_hours_sleep is a relational query that takes in an int input for university year and performs string operations in order to match the format in the database. The query applies the WHERE clause in order to filter above 6 hours of sleep and the specified university year. physical activity in terms of minutes of exercise. This relational query is used to filter the schema for students who slept more than six hours and are part of a specific university year. This function was inspired by the fact that six hours is generally the recommended amount of sleep and I was curious whether the length of the list of higher classmen would be more due to potentially more sleep than excited freshman.

```
def find_above_six_hours_sleep():
    University_Year = int(input("Enter university year (1/2/3/4): "))
    if University_Year == 1:
        uni = "1st Year"
    elif University_Year == 2:
        uni = "2nd Year"
    elif University_Year == 3:
        uni = "3rd Year"
    elif University_Year == 4:
        uni = "4th Year"
    query = f"""
        SELECT Student_ID, Sleep_Duration FROM student_sleep_patterns
        WHERE Sleep_Duration > 6.0 AND University_Year = '{uni}';
    """
    cur = connection.cursor()
    cur.execute(query)
    rows = cur.fetchall()
    print(f"These are the student IDs of {uni}'s who sleep more than six hours and how much they sleep")
    for row in rows:
        print(f"Student {row[0]} slept {row[1]} hrs")
```

Fig. 11. caffeine_exercise() function

```
6
Enter university year (1/2/3/4): 3
These are the student IDs of 3rd Year's who sleep more than six hours and how much they sleep
Student 14 slept 7.3 hrs
Student 16 slept 8.8 hrs
Student 18 slept 6.1 hrs
Student 20 slept 9.0 hrs
Student 21 slept 6.5 hrs
Student 24 slept 7.7 hrs
Student 33 slept 7.9 hrs
Student 35 slept 6.6 hrs
Student 36 slept 6.2 hrs
```

Fig. 11. Example input and output

Function (7): Find students with a sleep quality of above a 7/10 with the sleep quantity of less than 6 hours")

The function find_above_six_hours_sleep is a relational query that searches for students with specific sleep quality and sleep quantity amounts. The query applies the WHERE clause in order to filter below 6 hours of sleep and above a sleep quality rating of 7. The sleep quality column in this database is an integer value ranging from one to ten to describe how rested they feel.This relational query is used to filter the schema for students who slept less than six hours and still felt their sleep quality was higher than seven. The query also orders the output using the ORDER BY statement. The list is sorted by the highest sleep quality with the least sleep duration.This function was inspired by the fact that six hours is generally the recommended amount of sleep and since sleep quality is a relative rating you can easily query for those who don't need a large quality of sleep to feel good.

```
def find_sleep_qual_quan():
    query = """
        SELECT Student_ID, Sleep_Quality, Sleep_Duration FROM student_sleep_patterns
        WHERE Sleep_Duration < 6.0 AND Sleep_Quality > 7
        ORDER BY Sleep_Quality DESC, Sleep_Duration ASC;
    """
    cur = connection.cursor()
    cur.execute(query)
    rows = cur.fetchall()
    print(f"These are the student IDs of students who sleep more than six hours and how much they sleep")
    for row in rows:
        print(f"Student {row[0]} rated Sleep Quality as a {row[1]}/10 but slept {row[2]} hrs")
```

Fig. 12. Example input and output

```
7
These are the student IDs of students who sleep more than six hours and how much they slee
Student 246 rated Sleep Quality as a 10/10 but slept 4.1 hrs
Student 47 rated Sleep Quality as a 10/10 but slept 4.1 hrs
Student 420 rated Sleep Quality as a 10/10 but slept 4.2 hrs
Student 23 rated Sleep Quality as a 10/10 but slept 4.3 hrs
Student 19 rated Sleep Quality as a 10/10 but slept 4.4 hrs
Student 156 rated Sleep Quality as a 10/10 but slept 4.4 hrs
Student 155 rated Sleep Quality as a 10/10 but slept 4.5 hrs
Student 75 rated Sleep Quality as a 10/10 but slept 4.9 hrs
Student 268 rated Sleep Quality as a 10/10 but slept 4.9 hrs
```

Fig. 13. Example input and output

Non-relational queries():

Function (8): Use linear regression to plot the relation between study hours and sleep hours

The function lr_on_sleep_hours() uses the SQL query to select the data “Study Hours” and “Sleep Duration”, and then it uses pandas to create a dataframe using the rows and columns it fetches. Using sklearn the data is then split into testing and training and a linear regression model is fit on the training data. The program outputs the R score and coefficient variables in order to see the regression model accuracy and correlation. The scatterplot of the data is then displayed to the screen along with the linear regression line. The plot is labelled with Study Hours as the X axis and Sleep Duration as the feature. This plot is then saved to the python executable directory location. The inspiration of this function was to see if there was a linear or any obvious correlation in the data.

```
def lr_on_study_sleep_hours():
    query = """
        SELECT Study_Hours, Sleep_Duration
        FROM student_sleep_patterns
    """
    cur = connection.cursor()
    cur.execute(query)
    rows = cur.fetchall()
    col = ['Study_Hours', 'Sleep_Duration']
    sleep_data = pd.DataFrame(rows, columns=col)

    X_train = None
    Y_train = None
    X_test = None
    Y_test = None
    X = sleep_data[['Study_Hours']] # needs to be 2d array so use double brackets
    Y = sleep_data['Sleep_Duration']
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
    model = linear_model.LinearRegression()
    model.fit(X_train, Y_train)
    print("Linear Regression Model R score and line variables:")
    print("r_score = ", model.score(X_test, Y_test))
    print("a = ", model.coef_) # correlation
    print("b = ", model.intercept_)
    x_range = np.arange(X.min()[0], X.max()[0]+1, 1)
    y_range = (model.coef_) * x_range + model.intercept_
    plt.scatter(sleep_data['Study_Hours'], sleep_data['Sleep_Duration'], color='b')
    plt.plot(y_range, color='r')
    plt.xlabel('Study Hours')
    plt.ylabel('Sleep_Duration')
    plt.title('Study Hours vs. Sleep_Duration with linear regression')
    plt.show()
    plt.savefig("Study Hours vs. Sleep_Duration.png")
    print("saved linear regression Study Hours vs. Sleep_Duration.png")
    plt.clf()
```

Fig. 13. Function lr_on_study_sleep_hours()

```
Select a choice: 8
Linear Regression Model R score and line variables:
r_score =  -0.0039689369326847 41
a =  [-0.00500538]
b =  6.531389930671879
```

Fig. 13. Example input and output with very low R score and slope coefficient “a”

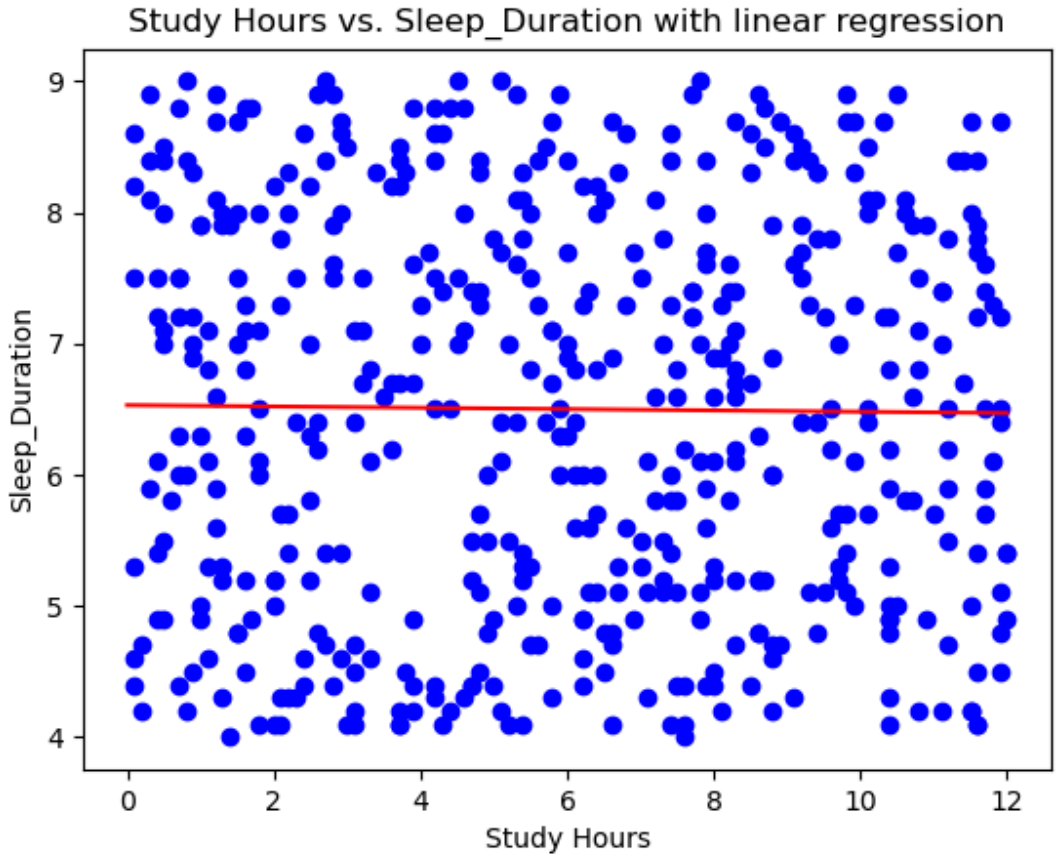

Fig. 13. Scatter plot of data entries for sleep duration and study hours as well as linear regression model line.

As can be seen in both the code output as well as the figure representing the linear regression, there is a very weak correlation within the data between study hours and sleep duration in this specific dataset.

Function (9): Predict sleep quality given study hours, caffeine consumption, and sleep duration using linear regression

This function creates a linear regression model based on the columns Sleep Quality, Study Hours, Caffeine Intake and Sleep Duration. The program then creates a linear regression model using these input features. The output of the program shows you the mean squared and root mean squared errors of the model. The lower the mean squared error the more accurate the models predictions are. As can be seen in the output this model has a fairly high mean square error, again due to the randomness of the database. The program then takes in three inputs for the study hours, caffeine intake and sleep duration. These variables are used to create a dataframe that represents what the model is currently trained on. We then feed this dataframe into the model in order to get the prediction based on the user inputs. The program will output a float value of sleep quality out of ten. The inspiration of this function is to attempt to predict a sleep quality value based on certain lifestyle factors like study hours and caffeine intake, since many students may sacrifice their sleep quality for better grades.

```python
# sql side
query = """
    SELECT Sleep_Quality, Study_Hours, Caffeine_Intake, Sleep_Duration
    FROM student_sleep_patterns
"""
cur = connection.cursor()
cur.execute(query)
rows = cur.fetchall()
#pd/sklearn create dataframe and create model
col = ['Sleep_Quality', 'Study_Hours', 'Caffeine_Intake', 'Sleep_Duration']
sleep_data = pd.DataFrame(rows, columns=col)
X_train = None
Y_train = None
X_test = None
Y_test = None
X = sleep_data[['Study_Hours', 'Caffeine_Intake', 'Sleep_Duration']]
Y = sleep_data['Sleep_Quality']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
model = linear_model.LinearRegression()
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
# error scores
print("Model error scores: ")
mean_sq_error = mean_squared_error(Y_test, Y_pred)
root_mean_sq_error = math.sqrt(mean_squared_error(Y_test, y_pred=Y_pred))
print("mean sq error = ", mean_sq_error)
print("root mean sq error = ", root_mean_sq_error)
# user input prediction
study_hours = float(input("Please enter study hours with input form (X.X): "))
caffeine_intake = int(input("Please enter caffeine drink intake per day (Integer input): "))
sleep_duration = float(input("Enter sleep duration in hours as a float (X.X): "))
user_answer = pd.DataFrame([[study_hours, caffeine_intake, sleep_duration]], columns=['Study_Hours', 'Caffeine_Intake', 'Sleep_Duration'])
user_prediction = model.predict(user_answer)
print(f"The model predicts you had a sleep quality: {user_prediction[0]:.2f} / 10")
```

Fig. 14. predict_sleep_quality() function

```
Select a choice: 9
Model error scores:
mean sq error =  9.36538903910269
root mean sq error =  3.060292312688886
Please enter study hours with input form (X.X): 4.0
Please enter caffeine drink intake per day (Integer input): 3
Enter sleep duration in hours as a float (X.X): 4
The model predicts you had a sleep quality: 5.40 / 10
```

Fig. 15. predict_sleep_quality() function output with accuracy metrics and user inputs as well as predicted outcome

Function (10): Use linear regression to plot the relation between physical activity and sleep hours

The function lr_on_sleep_hours() uses the SQL query to select the data “Physical Activity” and “Sleep Duration”, and then it uses pandas to create a dataframe using the rows and columns it fetches. The data is then split into testing and training sections and a linear regression model is fit on the training data. The program outputs the R score and coefficient variables in order to see the regression model accuracy and correlation. The scatterplot of the data is then displayed to the screen along with the linear regression line. The plot is labelled with Physical Activity as the X axis and Sleep Duration as the featured outcome. This plot is then saved to the python executable directory location. The inspiration of this function was to see if there was a linear or any obvious

correlation in the data for physical activity and sleep duration, in order to see if students who were more active slept any longer or shorter.

```
def physactivity_vs_sleep_duration():
    query = """
        SELECT Physical_Activity, Sleep_Duration
        FROM student_sleep_patterns
    """
    cur = connection.cursor()
    cur.execute(query)
    rows = cur.fetchall()
    col = ['Physical_Activity', 'Sleep_Duration']
    sleep_data = pd.DataFrame(rows, columns=col)

    X_train = None
    Y_train = None
    X_test = None
    Y_test = None
    X = sleep_data[['Physical_Activity']] # needs to be 2d array so use double brackets
    Y = sleep_data['Sleep_Duration']
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
    model = linear_model.LinearRegression()
    model.fit(X_train, Y_train)
    print("r_score = ", model.score(X_test, Y_test))
    print("a = ", model.coef_) # correlation
    print("b = ", model.intercept_)
    x_range = np.arange(X.min()[0], X.max()[0]+1, 1)
    y_range = (model.coef_) * x_range + model.intercept_
    plt.scatter(sleep_data['Physical_Activity'], sleep_data['Sleep_Duration'], color='b')
    plt.plot(y_range, color='r')
    plt.xlabel('University_Year')
    plt.ylabel('Sleep_Duration')
    plt.title('University_Year vs. Sleep_Duration with linear regression')
    plt.savefig("University_Year vs. Sleep_Duration.png")
    plt.show()
    print("saved linear regression University_Year vs. Sleep_Duration.png")
    plt.clf()
```

Fig. 16. physactivity_vs_sleep_duration() function output with accuracy metrics and user inputs as well as predicted outcome

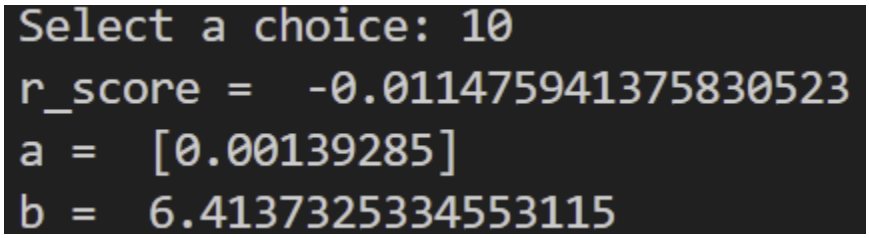

Fig. 16. physactivity_vs_sleep_duration() function output with accuracy metrics and user inputs as well as predicted outcome

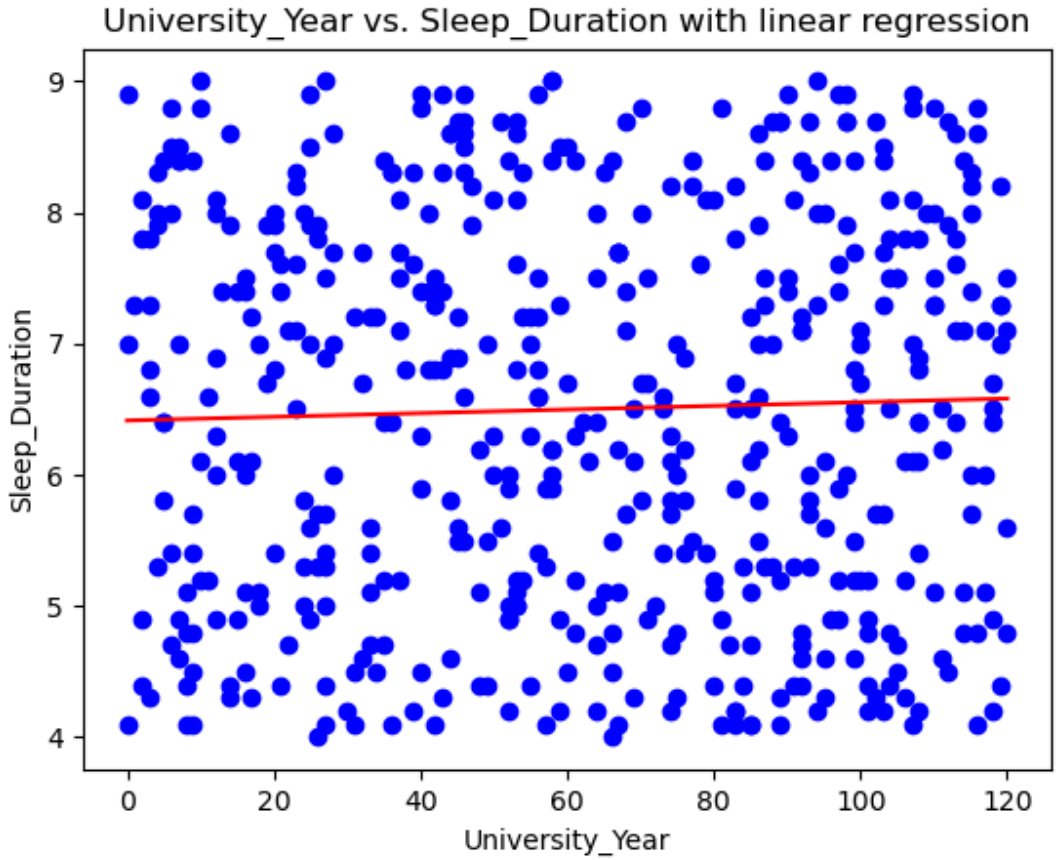

Fig. 17. Scatter plot of data entries for sleep duration and university year as well as linear regression model line.

As can be seen in the scatterplot as well as the output, there is a very weak correlation among the data. Again, this is due to the nature of the database and the variability among the data. There is no obvious linear correlation in the data, but potentially other models could show a stronger correlation. These two variables may also just be completely independent of one another.

Function (11): Visualize the decision tree model with the target column being Sleep Quality>7)

The function show_decision_tree() uses the SQL statement select in order to select all the columns that have a numerical value. The function then creates a dataframe using these columns. The column Sleep_Quality takes some more data processing because it is not a binary value and the target column is a binary result. The binary tree has two class names “Slept real good” and “Slept not good” and the Sleep Quality target must match this result. The Sleep_Quality_Status is the post processed result. The program applies a condition using the apply function from

pandas and a condition to assign a result of 1 if the column result is above 7. After the data processing step, the data is split into testing and training data and a model is generated using the decision tree classifier. This model is then fit using training data. The program outputs the model score as well as generates the pdf of the decision tree graph and displays it. The inspiration behind this application is to be able to visualize how the model operates on different thresholds in order to reach the final classification of “Slept good” or “Slept not good”.

```
def show_decision_tree():
    query = """
        SELECT Age, Sleep_Duration, Study_Hours, Screen_Time, Caffeine_Intake, Physical_Activity, Sleep_Quality
        FROM student_sleep_patterns
    """
    cur = connection.cursor()
    cur.execute(query)
    rows = cur.fetchall()
    col = ['Age', 'Sleep_Duration', 'Study_Hours', 'Screen_Time', 'Caffeine_Intake', 'Physical_Activity', 'Sleep_Quality']

    all_sleep_data = pd.DataFrame(rows, columns=col)
    all_sleep_data['Sleep_Quality_Status'] = all_sleep_data['Sleep_Quality'].apply(lambda x: 1 if x>7 else 0)
    X_train = None
    Y_train = None
    X_test = None
    Y_test = None
    X = all_sleep_data[['Age', 'Sleep_Duration', 'Study_Hours', 'Screen_Time', 'Caffeine_Intake', 'Physical_Activity']]
    Y = all_sleep_data['Sleep_Quality_Status']
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
    model = tree.DecisionTreeClassifier(random_state=42)
    model = model.fit(X_train, Y_train)
    print("model score is: ", model.score(X_test, Y_test))

    dot = tree.export_graphviz(model, feature_names=X.columns, class_names=['Slept good', 'Slept not good'])
    graph = graphviz.Source(dot)
    graph.render('decision_tree_graph')
    graph.view()
```

Fig. 18. show_decision_tree() function

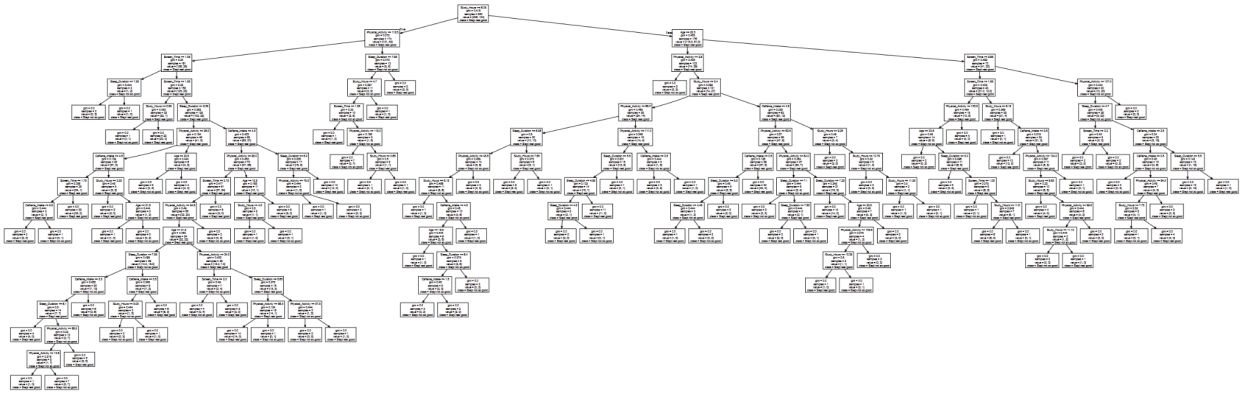

Fig. 19. Decision tree for the model using input features described in the columns and output feature as Sleep_Quality_Status

Function (12): Predict sleep duration using linear regression given screen time, caffeine intake, and physical activity

The function predict_based_on_fitness uses the same SQL Select statement in order to obtain the data from the database of the columns “Sleep Duration” “Screen Time” “Caffeine Intake” and “Physical Activity”. The function then creates a dataframe using the rows that it pulled and the columns provided, and then it generates a linear regression model and predicts the accuracy of this linear regression model by feeding the X_test variable to it, and then calculating the mean squared error and root mean squared error. The function then asks for user inputs for the Screen Time, Caffeine intake and physical activity variables. The function then generates a dataframe using these user inputted variables, and then feeds this dataframe to the model in order to predict the sleep duration amount. This output is formatted to be up to two decimal place values.

```
def predict_based_on_fitness():
    # sql side
    query = """
        SELECT Sleep_Duration, Screen_Time, Caffeine_Intake, Physical_Activity
        FROM student_sleep_patterns
    """
    cur = connection.cursor()
    cur.execute(query)
    rows = cur.fetchall()
    #pd/sklearn create dataframe and create model
    col = ['Sleep_Duration', 'Screen_Time', 'Caffeine_Intake', 'Physical_Activity']
    sleep_data = pd.DataFrame(rows, columns=col)
    X_train = None
    Y_train = None
    X_test = None
    Y_test = None
    X = sleep_data[['Screen_Time', 'Caffeine_Intake', 'Physical_Activity']]
    Y = sleep_data['Sleep_Duration']
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
    model = linear_model.LinearRegression()
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    # error scores
    mean_sq_error = mean_squared_error(Y_test, Y_pred)
    root_mean_sq_error = math.sqrt(mean_squared_error(Y_test, y_pred=Y_pred))
    print(f"Model error scores: mean square {mean_sq_error} and root msq: {root_mean_sq_error}")
    # user input prediction
    screen_hours = float(input("Please enter screen many hours per day input form (X.X): "))
    caffeine_intake = int(input("Please enter caffeine drink intake per day (Integer input): "))
    phys_activity = float(input("Enter physical activity duration in hours as a float (X.X): "))
    user_answer = pd.DataFrame([[screen_hours, caffeine_intake, phys_activity]], columns=['Screen_Time', 'Caffeine_Intake', 'Physical_Activity'])
    user_prediction = model.predict(user_answer)
    print(f"The model predicts you slept: {user_prediction[0]:.2f} hours based on your fitness inputs")
```

Fig. 20. Predict_based_on_fitness() function

```
Select a choice: 12
Model error scores: mean square 2.3053702844559307 and root msq: 1.5183445868629197
Please enter screen many hours per day input form (X.X): 3.0
Please enter caffeine drink intake per day (Integer input): 4
Enter physical activity duration in hours as a float (X.X): 9
The model predicts you slept: 6.47 hours based on your fitness inputs
```

Fig. 21. Input options for the function and outputs including model error scores and total prediction based on user input.

Conclusion:

This program is designed to demonstrate the potential of both relational and non-relational queries in order to perform data analysis and understand the correlation in a given dataset. This program has ten main functions, as well as three operative functions. The five relational queries operate on the given data using solely SQL statements and are used mainly to retrieve information that is in the relational form of the database. These functions operate in order to

search and filter different requested variables in the database. The five non relational queries are to be used beyond SQL in order to generate more in depth conclusions about correlations in the data or to use trained models in order to predict a given column.

An observation can be made that the metrics and quality scores of the model are very low. The R score of the linear regression should be as close to 1 as possible for a strong model, and the mean square error should be as close to zero as possible for a strongly correlated model. The model score should also be as close to 100 for strong model accuracies. Many of my accuracy metrics display my model as inaccurate or non correlated.
Upon further analysis I discovered the dataset was synthetically generated by the kaggle owner. However, none of the data is unrealistic. This can just show that there is sometimes no desired correlation in your studies, potentially depending on the relatedness of the variables or how random the answers can be. The implementation of the functions are completely correct, from the retrieval of the data to machine learning applications that are performed. The error scores are also displayed to the user when generating any of the models so that the user will not be misled by database accuracy.