



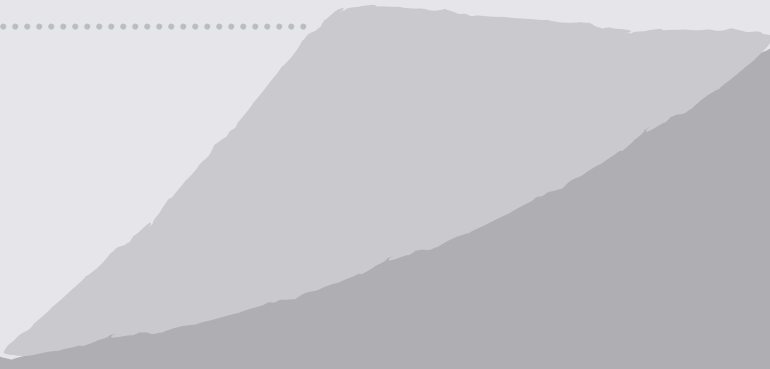
IMPROVING STUDENT WELLBEING TOGETHER

# STUDENT MENTAL HEALTH

Group - 20

---

Vinyas Naidu Karri (NUID-002485274)  
Sri Sai Taru Vemu (NUID-002840565)



# Project Overview

## Objectives:

- Enhance mental health insights
- Provide preventive measures and early care
- Ensure sustainability of initiatives

## Web Application Features:

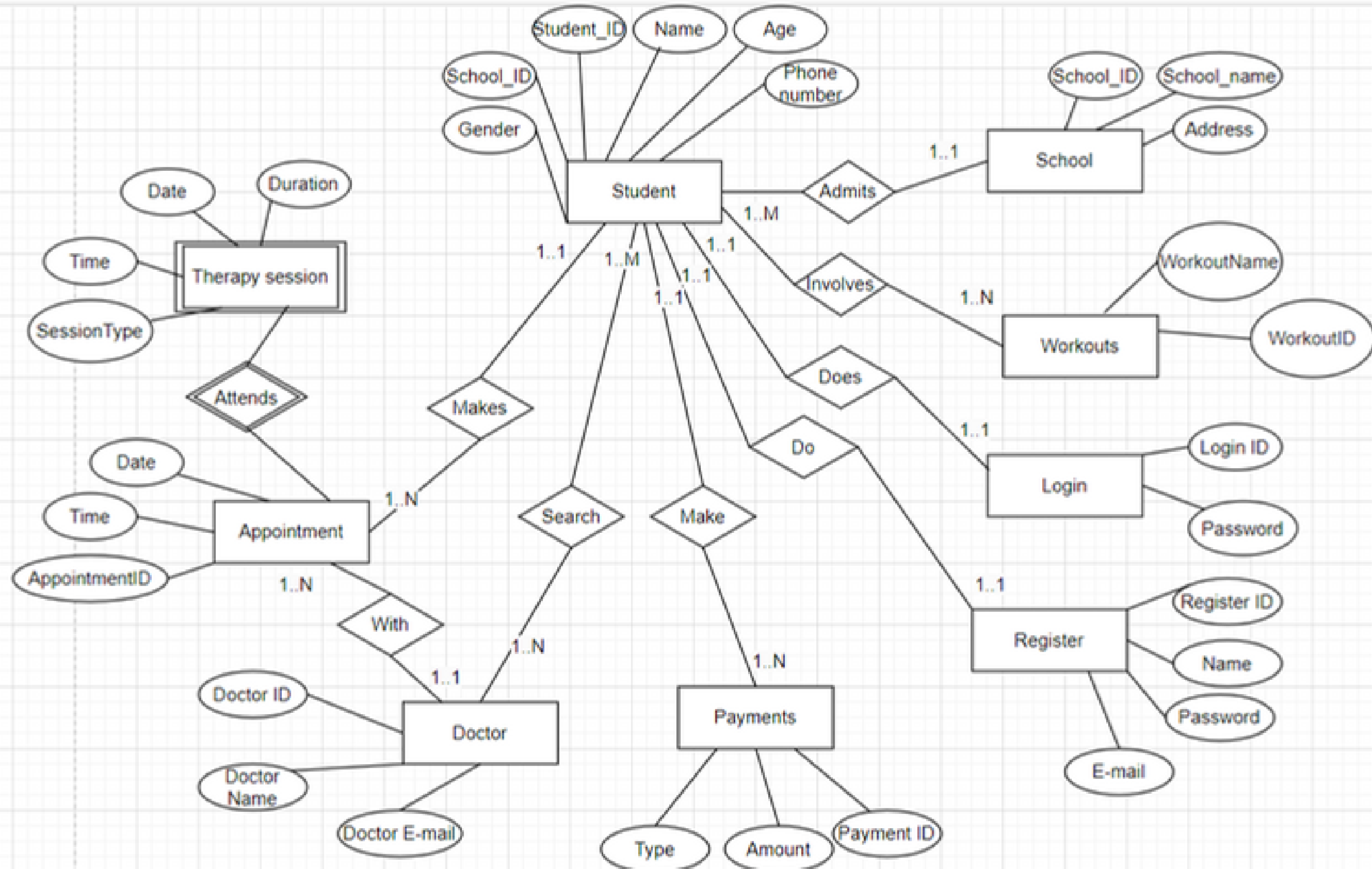
- Centralized student information storage
- Data collection on:
  - Student details
  - Mental health indicators
  - Lifestyle factors
  - Personal details
  - Activities
  - Financial stress and healthcare

## Benefits:

- Insight generation through data analysis
- Targeted interventions based on trends
- Collaborative care with professionals
- Personalized well-being support for students



# ERR Diagram



# Relational Database

**STUDENT:** StudentID (PK), SchoolID\_FK (FK to SCHOOL), UserID\_FK (FK to LOGIN), Name, Age, PhoneNumber, Gender, Course, CreditLoad, GPA.

**SCHOOL:** SchoolID (PK), Name, Address.

**WORKOUT:** WorkoutID (PK), WorkoutName.

**STUDENT\_WORKOUTS:** StudentID\_FK (FK to STUDENT), WorkoutID\_FK (FK to WORKOUT).

**LOGIN:** UserID (PK), PasswordHash.

**REGISTER\_C:** RegisterID (PK), StudentID\_FK (FK to STUDENT), Name, Email.

**PAYMENTS:** PaymentID (PK), StudentID\_FK (FK to STUDENT), Amount, Type.

**DOCTORS:** DoctorID (PK), DoctorName, DoctorEmail.

**STUDENT\_DOCTOR:** StudentID\_FK (FK to STUDENT), DoctorID\_FK (FK to DOCTORS).

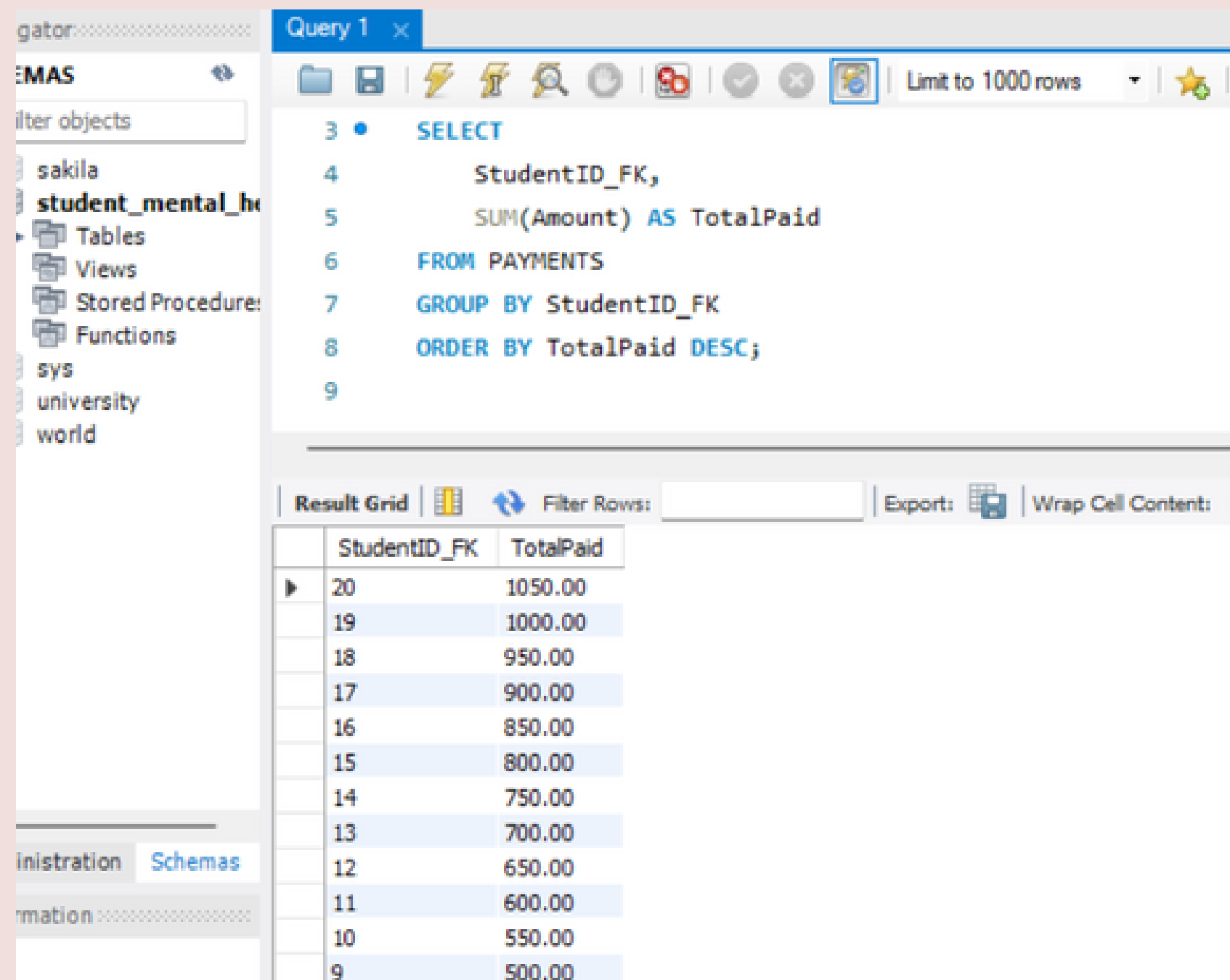
**APPOINTMENTS:** AppointmentID (PK), StudentID\_FK (FK to STUDENT), DoctorID\_FK (FK to DOCTORS), Date, Time.

**THERAPY\_SESSIONS:** AppointmentID\_FK (FK to APPOINTMENTS), Date, Duration, Time, SessionType.

# MY SQL IMPLEMENTATION

Query 1 – how much total amount each student has paid, and order the students by the total amount paid in descending order

```
SELECT
    StudentID_FK,
    SUM(Amount) AS TotalPaid
FROM
    PAYMENTS
GROUP BY
    StudentID_FK
ORDER BY
    TotalPaid DESC;
```



The screenshot shows a MySQL query editor window titled "Query 1". The query is as follows:

```
3 SELECT
4     StudentID_FK,
5     SUM(Amount) AS TotalPaid
6 FROM PAYMENTS
7 GROUP BY StudentID_FK
8 ORDER BY TotalPaid DESC;
9
```

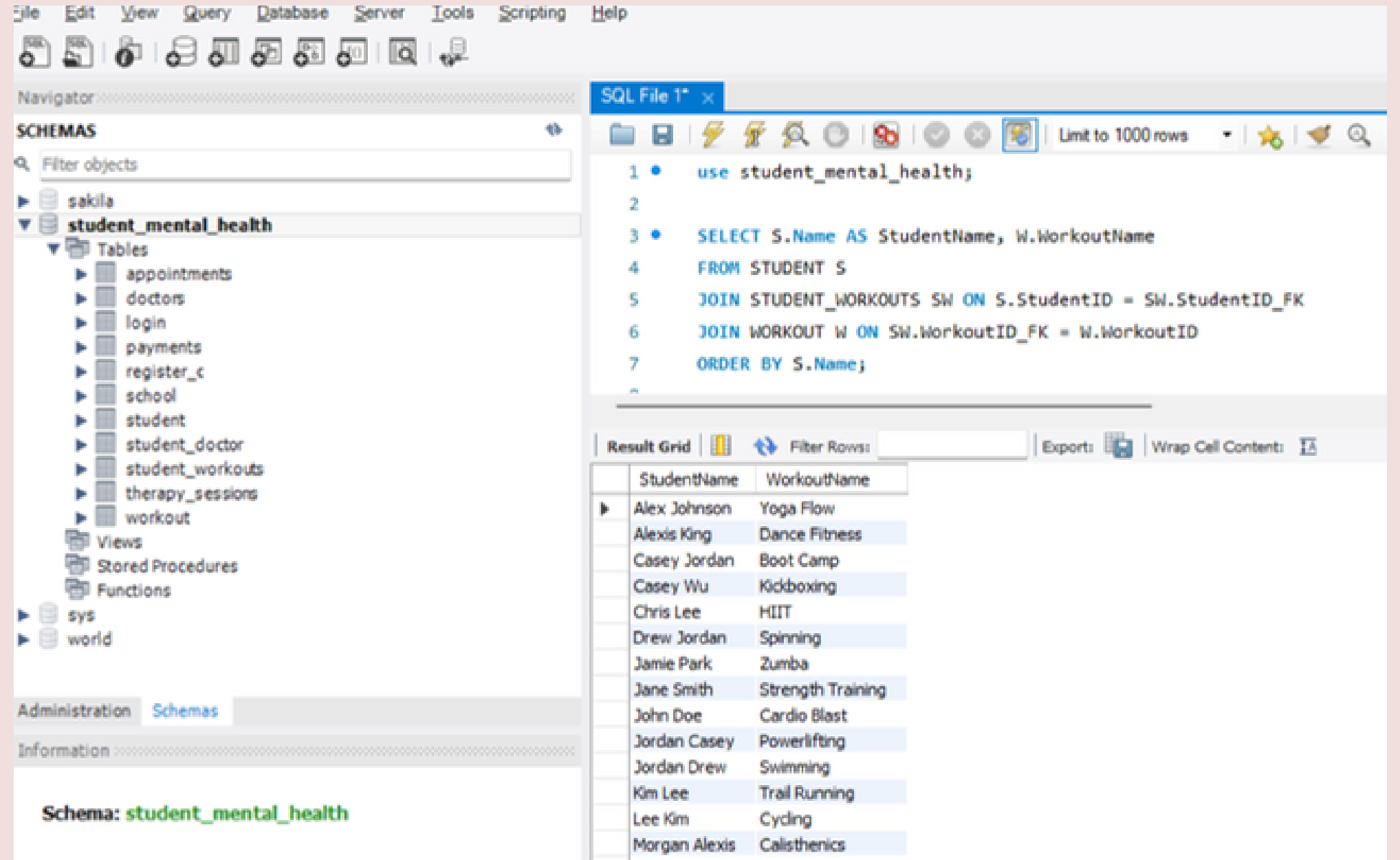
The results are displayed in a table with the following data:

StudentID_FK	TotalPaid
20	1050.00
19	1000.00
18	950.00
17	900.00
16	850.00
15	800.00
14	750.00
13	700.00
12	650.00
11	600.00
10	550.00
9	500.00

# MY SQL IMPLEMENTATION

Query 2 -.List all students with their workouts

```
SELECT S.Name AS StudentName,  
W.WorkoutName  
FROM STUDENT S  
JOIN STUDENT_WORKOUTS SW ON  
S.StudentID = SW.StudentID_FK  
JOIN WORKOUT W ON SW.WorkoutID_FK =  
W.WorkoutID  
ORDER BY S.Name;
```



The screenshot shows a MySQL IDE interface. On the left, the 'Navigator' pane displays the 'SCHEMAS' tree with 'sakila' and 'student\_mental\_health' expanded. The 'student\_mental\_health' schema contains tables like 'appointments', 'doctors', 'login', 'payments', 'register\_c', 'school', 'student', 'student\_doctor', 'student\_workouts', 'therapy\_sessions', and 'workout'. The main editor window shows a SQL query file with the following code:

```
1 • use student_mental_health;  
2  
3 • SELECT S.Name AS StudentName, W.WorkoutName  
4 FROM STUDENT S  
5 JOIN STUDENT_WORKOUTS SW ON S.StudentID = SW.StudentID_FK  
6 JOIN WORKOUT W ON SW.WorkoutID_FK = W.WorkoutID  
7 ORDER BY S.Name;
```

Below the query editor, the 'Result Grid' pane displays the query results in a table format:

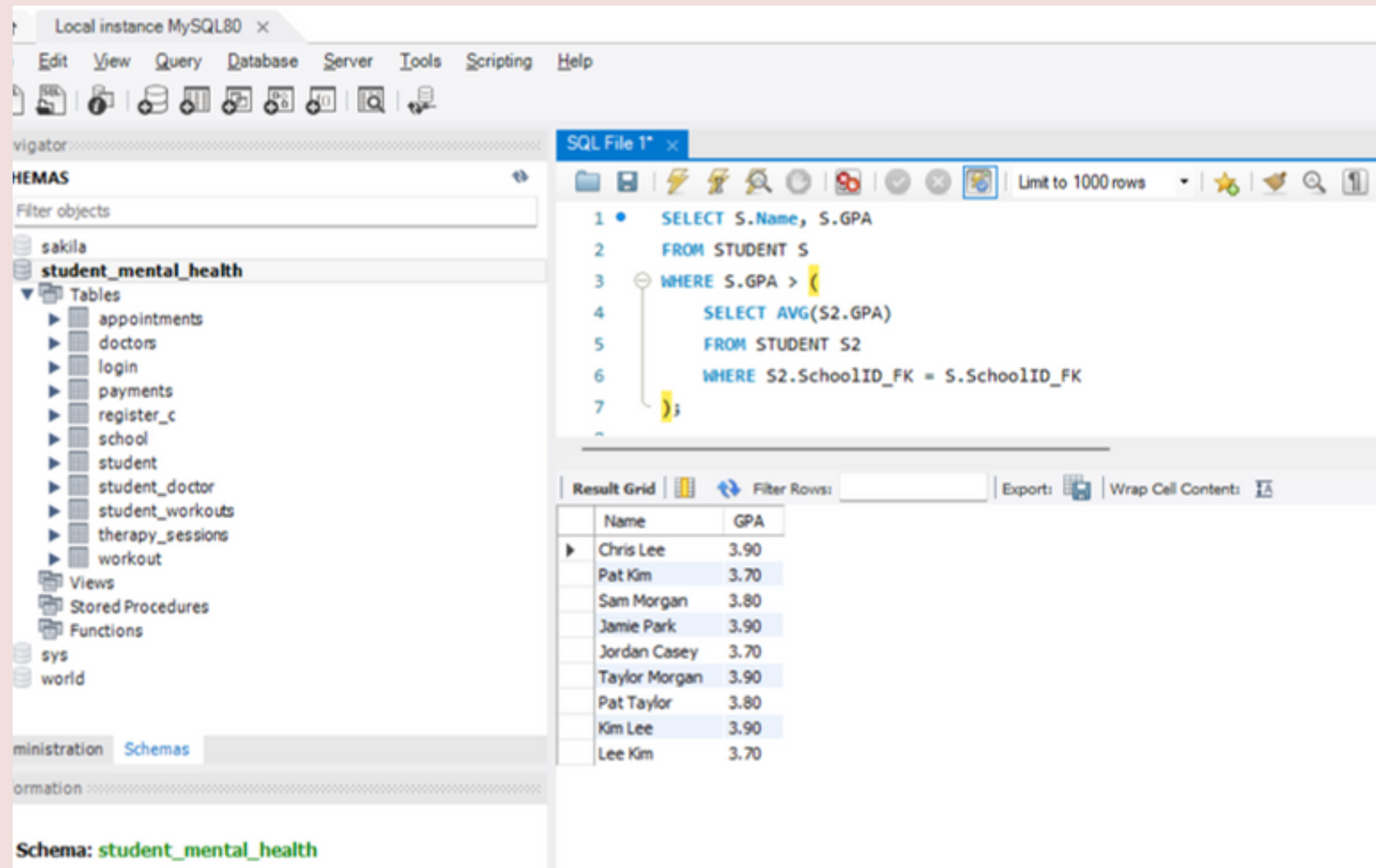
StudentName	WorkoutName
Alex Johnson	Yoga Flow
Alexis King	Dance Fitness
Casey Jordan	Boot Camp
Casey Wu	Kickboxing
Chris Lee	HIIT
Drew Jordan	Spinning
Jamie Park	Zumba
Jane Smith	Strength Training
John Doe	Cardio Blast
Jordan Casey	Powerlifting
Jordan Drew	Swimming
Kim Lee	Trail Running
Lee Kim	Cycling
Morgan Alexis	Calisthenics



# MY SQL IMPLEMENTATION

Query 3 List students and their GPAs if they are above the average GPA of their school

```
SELECT S.Name, S.GPA
FROM STUDENT S
WHERE S.GPA > (
  SELECT AVG(S2.GPA)
  FROM STUDENT S2
  WHERE S2.SchoolID_FK = S.SchoolID_FK
);
```



The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tab is active, displaying a tree view of databases including 'sakila' and 'student\_mental\_health'. The 'student\_mental\_health' database is selected, showing its tables and views. On the right, the 'SQL File 1\*' window displays the following SQL query:

```
1 SELECT S.Name, S.GPA
2 FROM STUDENT S
3 WHERE S.GPA > (
4     SELECT AVG(S2.GPA)
5     FROM STUDENT S2
6     WHERE S2.SchoolID_FK = S.SchoolID_FK
7 );
```

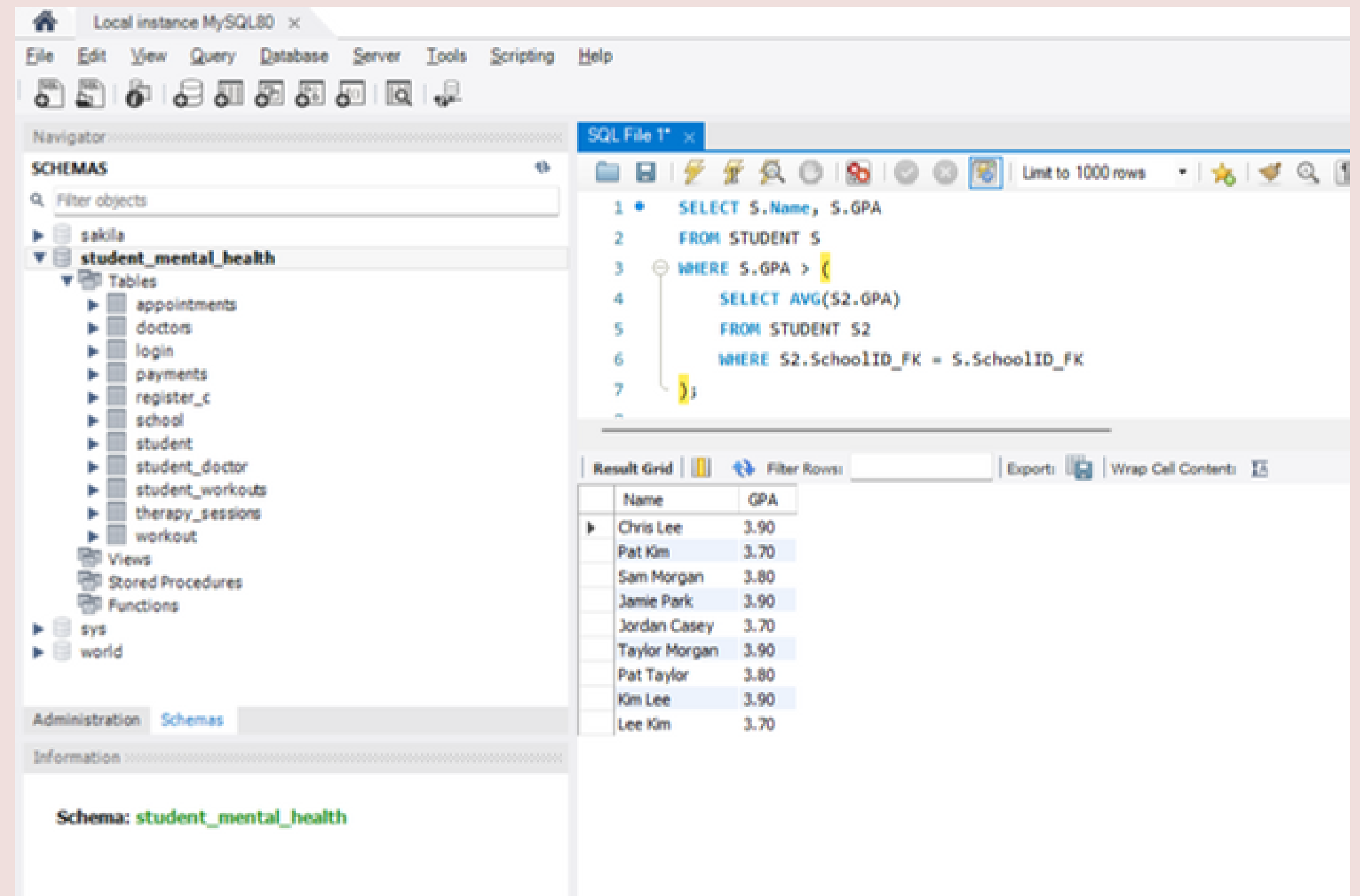
Below the query, the 'Result Grid' shows the output of the query, listing students and their GPAs. The results are as follows:

Name	GPA
Chris Lee	3.90
Pat Kim	3.70
Sam Morgan	3.80
Jamie Park	3.90
Jordan Casey	3.70
Taylor Morgan	3.90
Pat Taylor	3.80
Kim Lee	3.90
Lee Kim	3.70

# MY SQL IMPLEMENTATION

Query 4 Find workouts that have been attended by at least one student

```
SELECT W.WorkoutName
FROM WORKOUT W
WHERE EXISTS (
  SELECT StudentID
  FROM STUDENT_WORKOUTS SW
  WHERE SW.WorkoutID_FK = W.WorkoutID );
```



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'sakila' and 'student\_mental\_health' expanded. The 'student\_mental\_health' schema contains tables like 'appointments', 'doctors', 'login', 'payments', 'register\_c', 'school', 'student', 'student\_doctor', 'student\_workouts', 'therapy\_sessions', and 'workout'. The main window shows a SQL query in 'SQL File 1\*' that selects the name and GPA of students from the 'STUDENT' table, filtered by a subquery that checks for their presence in the 'STUDENT\_WORKOUTS' table. The 'Result Grid' at the bottom displays the query results.

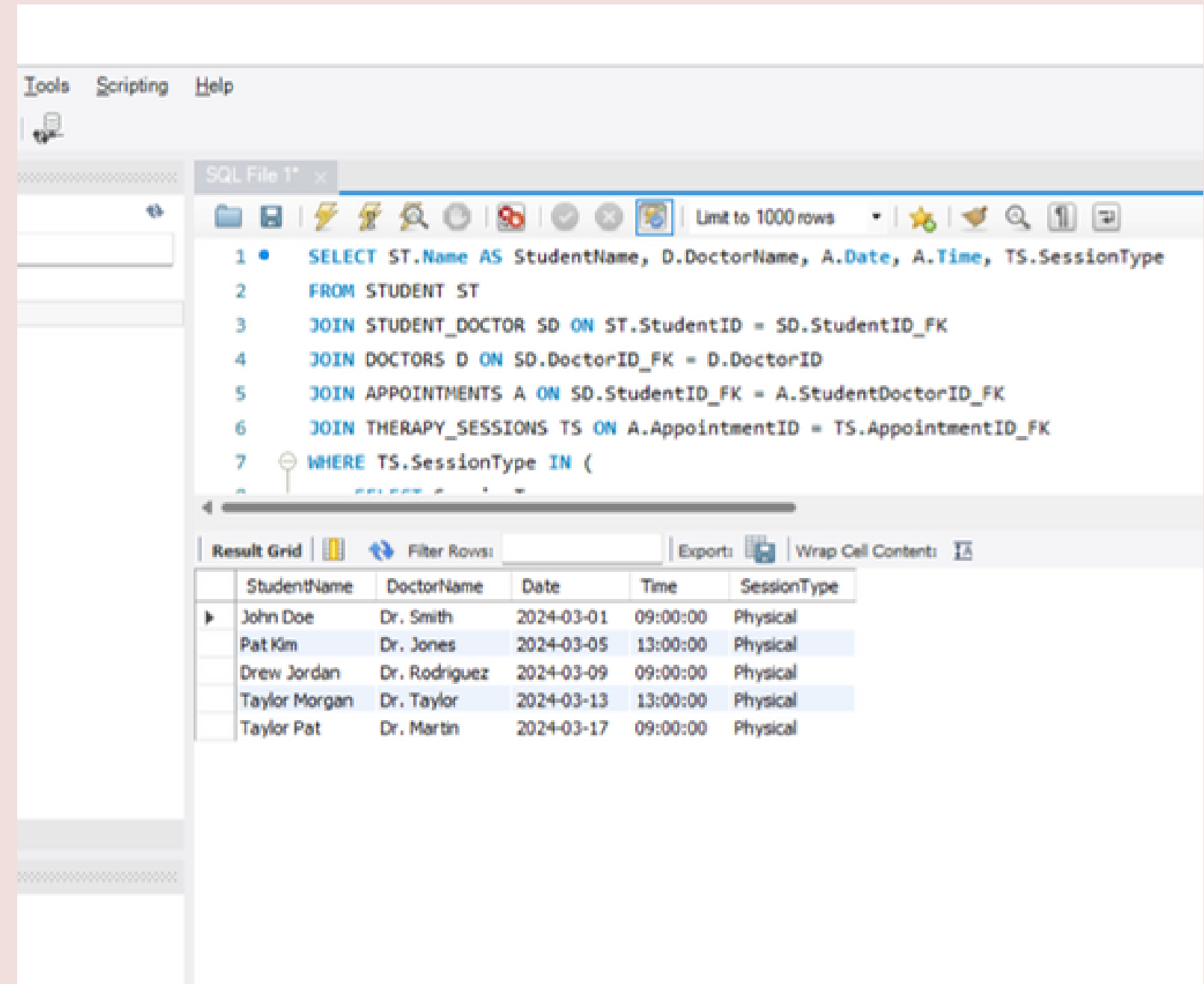
Name	GPA
Chris Lee	3.90
Pat Kim	3.70
Sam Morgan	3.80
Jamie Park	3.90
Jordan Casey	3.70
Taylor Morgan	3.90
Pat Taylor	3.80
Kim Lee	3.90
Lee Kim	3.70



# MY SQL IMPLEMENTATION

Query 5 List students, their doctors, and appointment details for a specific type of therapy session

```
SELECT ST.Name AS StudentName, D.DoctorName, A.Date, A.Time,
TS.SessionType
FROM STUDENT ST
JOIN STUDENT_DOCTOR SD ON ST.StudentID = SD.StudentID_FK
JOIN DOCTORS D ON SD.DoctorID_FK = D.DoctorID
JOIN APPOINTMENTS A ON SD.StudentID_FK =
A.StudentDoctorID_FK
JOIN THERAPY_SESSIONS TS ON A.AppointmentID =
TS.AppointmentID_FK
WHERE TS.SessionType IN (
SELECT SessionType
FROM THERAPY_SESSIONS
WHERE SessionType = 'Physical' )
ORDER BY A.Date, A.Time;
```



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query is a complex JOIN statement that filters for 'Physical' therapy sessions. The result grid displays 6 rows of data, including student names, doctor names, dates, times, and session types.

	StudentName	DoctorName	Date	Time	SessionType
▶	John Doe	Dr. Smith	2024-03-01	09:00:00	Physical
	Pat Kim	Dr. Jones	2024-03-05	13:00:00	Physical
	Drew Jordan	Dr. Rodriguez	2024-03-09	09:00:00	Physical
	Taylor Morgan	Dr. Taylor	2024-03-13	13:00:00	Physical
	Taylor Pat	Dr. Martin	2024-03-17	09:00:00	Physical

# PYTHON IMPLEMENTATION

```
In [3]: ➤ import mysql.connector  
        from mysql.connector import Error
```

```
In [4]: ➤ connection = mysql.connector.connect(host=' localhost',  
                                              database='student_mental_health',  
                                              user='root',  
                                              password='new_password',  
                                              auth_plugin='mysql_native_password')
```

```
In [5]: ➤ connection
```

```
Out[5]: <mysql.connector.connection_cext.CMySQLConnection at 0x20d7b450970>
```

```
In [8]: ➤ if connection.is_connected():  
        db_Info = connection.get_server_info()  
        print("Connected to MySQL Server version ", db_Info)  
        cursor = connection.cursor()  
        cursor.execute("select database();")  
        record = cursor.fetchone()  
        print("Your connected to database: ", record)
```

```
Connected to MySQL Server version 8.0.23  
Your connected to database: ('student_mental_health',)
```

# Retrieving tables from DataBase

```
query = "SELECT * FROM student"
cursor.execute(query)

# Fetch all rows from the result set
data = cursor.fetchall()

# Create a DataFrame from the fetched data
columns = [desc[0] for desc in cursor.description]
df = pd.DataFrame(data, columns=columns)

print(df)
```

<IPython.core.display.Javascript object>

	StudentID	SchoolID_FK	UserID_FK	Name
0	1	1	1	John Doe
1	2	2	2	Jane Smith
2	3	3	3	Alex Johnson
3	4	4	4	Chris Lee
4	5	5	5	Pat Kim
5	6	1	6	Sam Morgan
6	7	2	7	Jamie Park
7	8	3	8	Casey Wu
8	9	4	9	Drew Jordan
9	10	5	10	Robin Taylor
10	11	1	11	Jordan Casey

```
def select_query():
    sql_select_Query = "select * from student"
    cursor = connection.cursor()
    cursor.execute(sql_select_Query)
    records = cursor.fetchall()
    print("Students in the garace list and details will be :\n")
    for row in records:
        print('Students_table each row =', row, "\n")
```

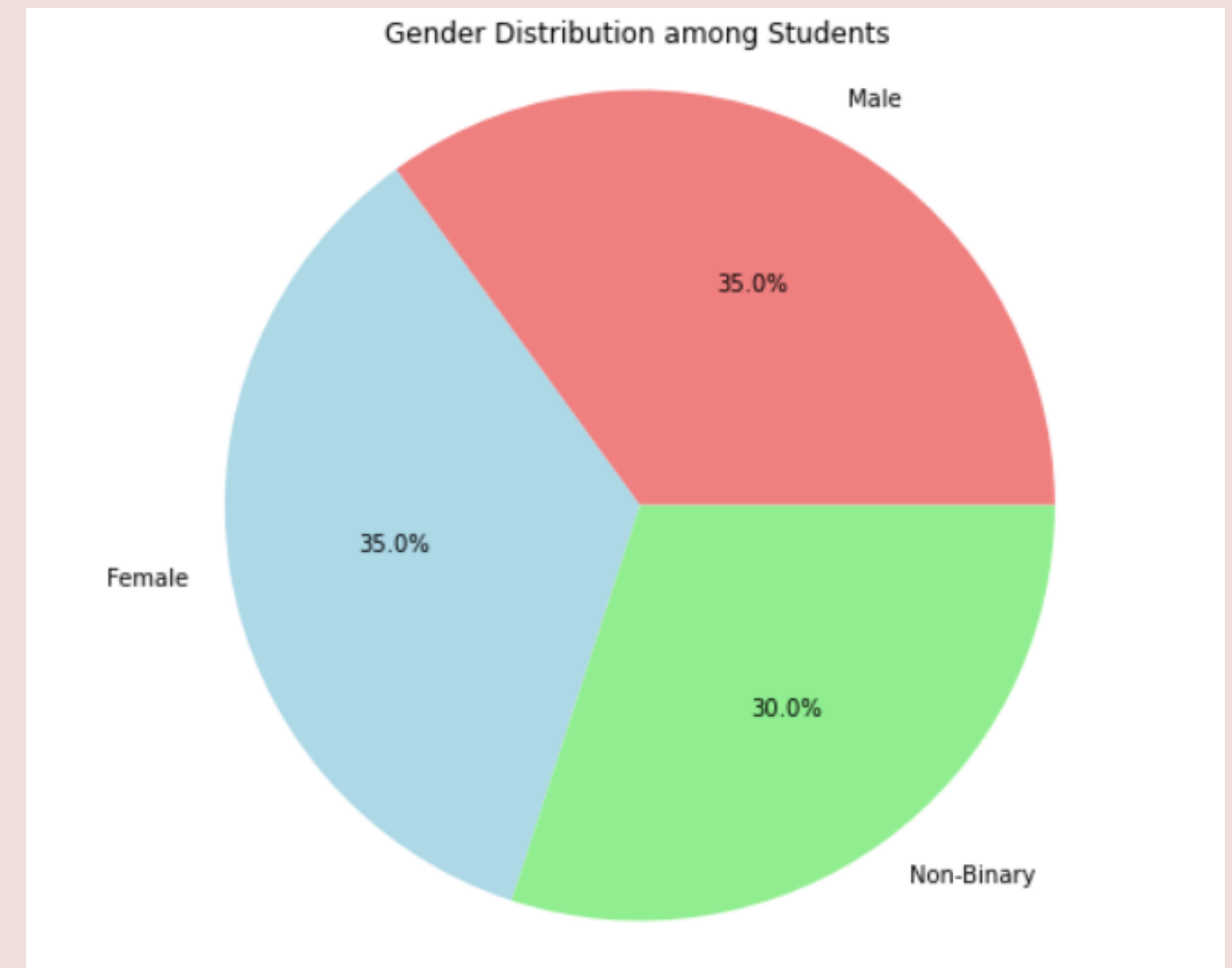
select\_query()

Students in the garace list and details will be :

Students\_table each row = (1, 1, 1, 'John Doe', 20, '555-0001', 'Male', '')

# Gender Distribution among Students

```
# Pie Chart
gender_counts = df['Gender'].value_counts()
plt.figure(figsize=(8, 6))
plt.pie(gender_counts, labels=gender_counts.index,
        autopct='%1.1f%%',
        colors=['lightcoral', 'lightblue', 'lightgreen'])
plt.title('Gender Distribution among Students')
plt.axis('equal')
plt.tight_layout()
plt.show()
```

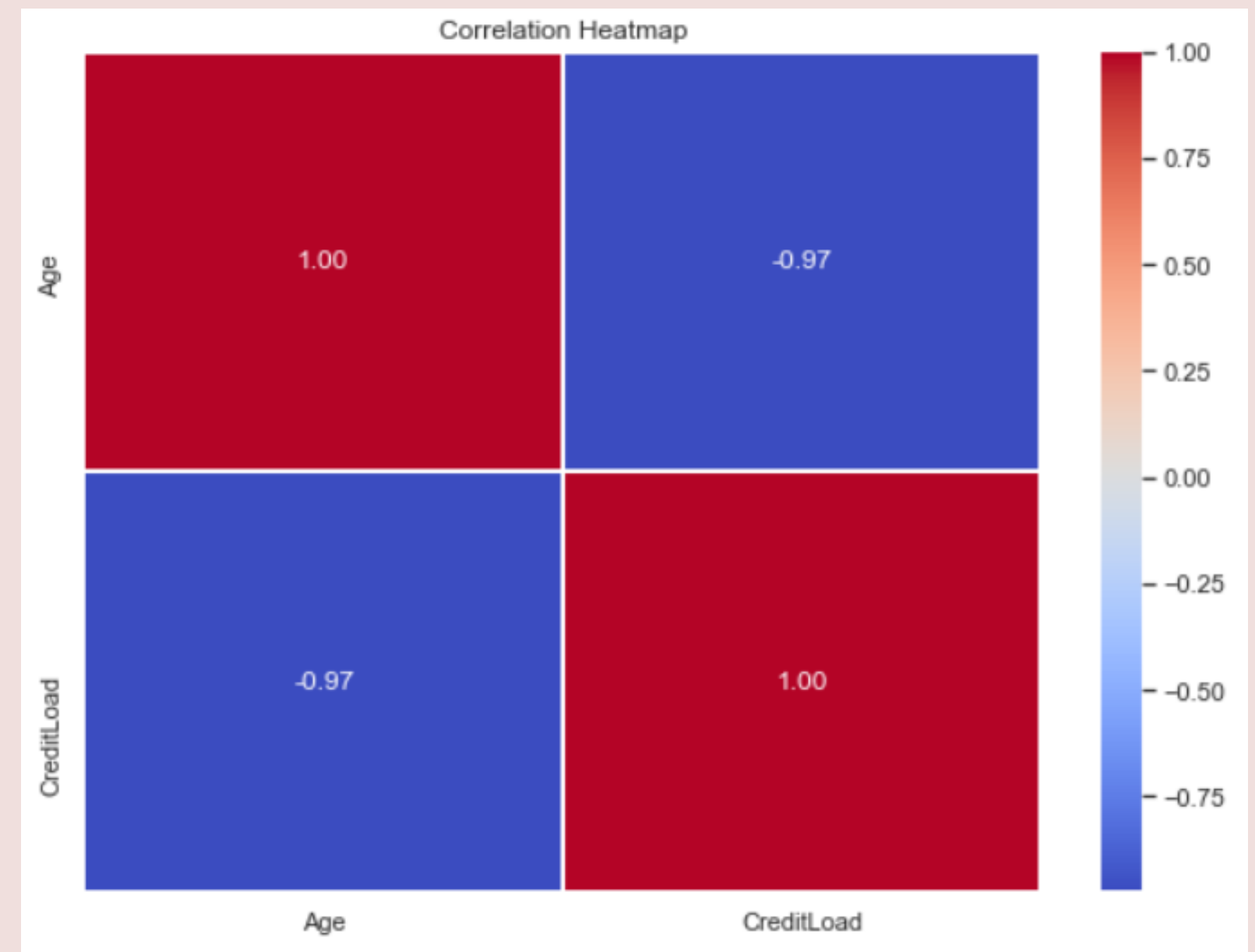


# Correlation Heatmap

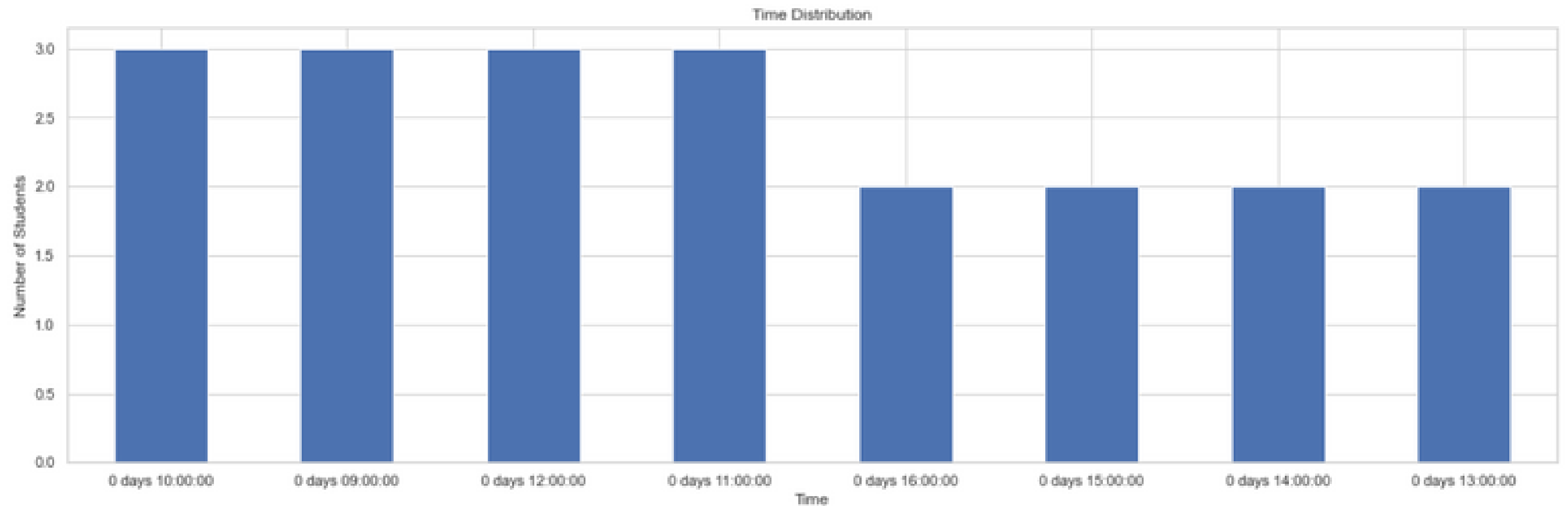
```
import seaborn as sns

# Calculate correlation matrix
correlation_matrix = df[
    ['Age', 'CreditLoad', 'GPA']].corr()

# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True,
            cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.show()
```



# Time Distribution



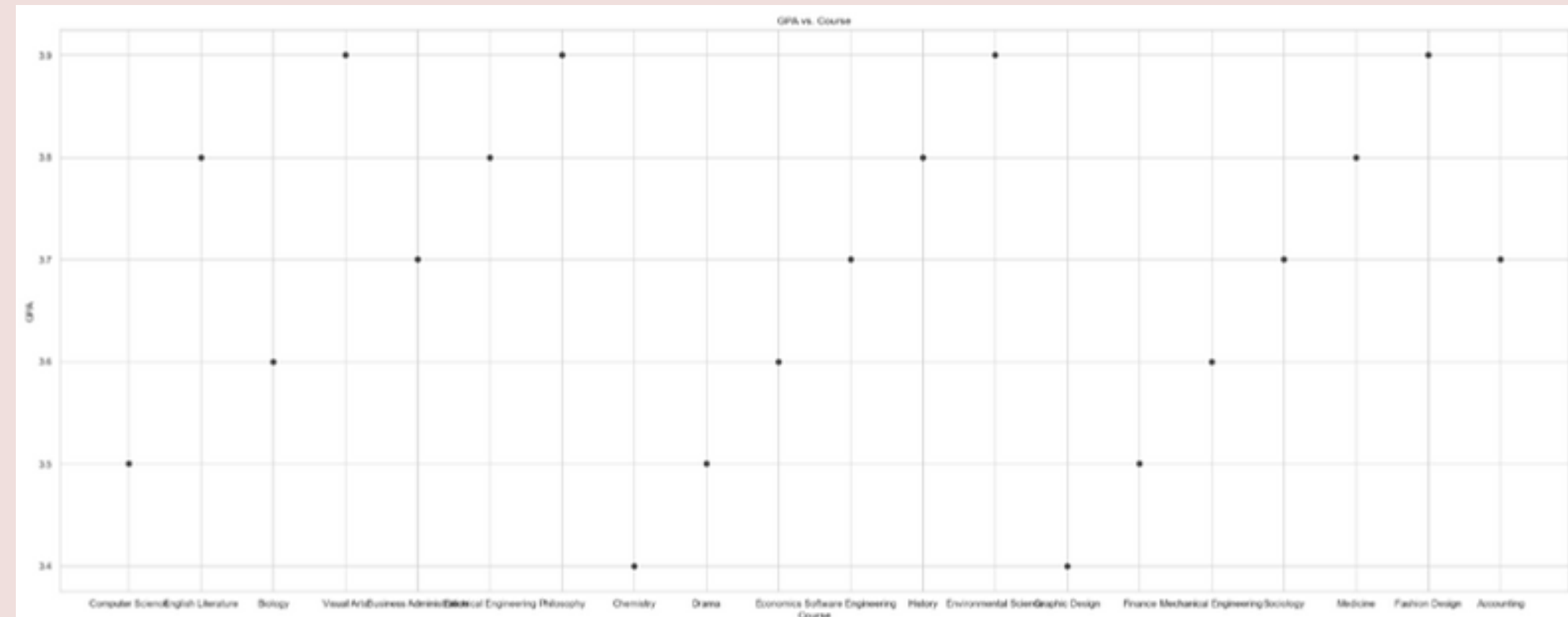
```
plt.figure(figsize=(20, 6))
gender_counts = df2['Time'].value_counts()
gender_counts.plot(kind='bar')
plt.title('Time Distribution')
plt.xlabel('Time')
plt.ylabel('Number of Students')
plt.xticks(rotation=0)
```



# GPA vs. Course

```
: # Scatter Plot
plt.figure(figsize=(25, 10))
plt.scatter(df['Course'], df['GPA'],
            color='black', alpha=0.7)

plt.xlabel('Course')
plt.ylabel('GPA')
plt.title('GPA vs. Course')
plt.grid(True)
plt.tight_layout()
plt.show()
```



# NOSQL IMPLEMENTATION

KVN'S ORG - 2024-04-17 > PROJECT 0

## Overview

### Database Deployments

Create deployment ...


Cluster0


Connect

View info

Edit configuration

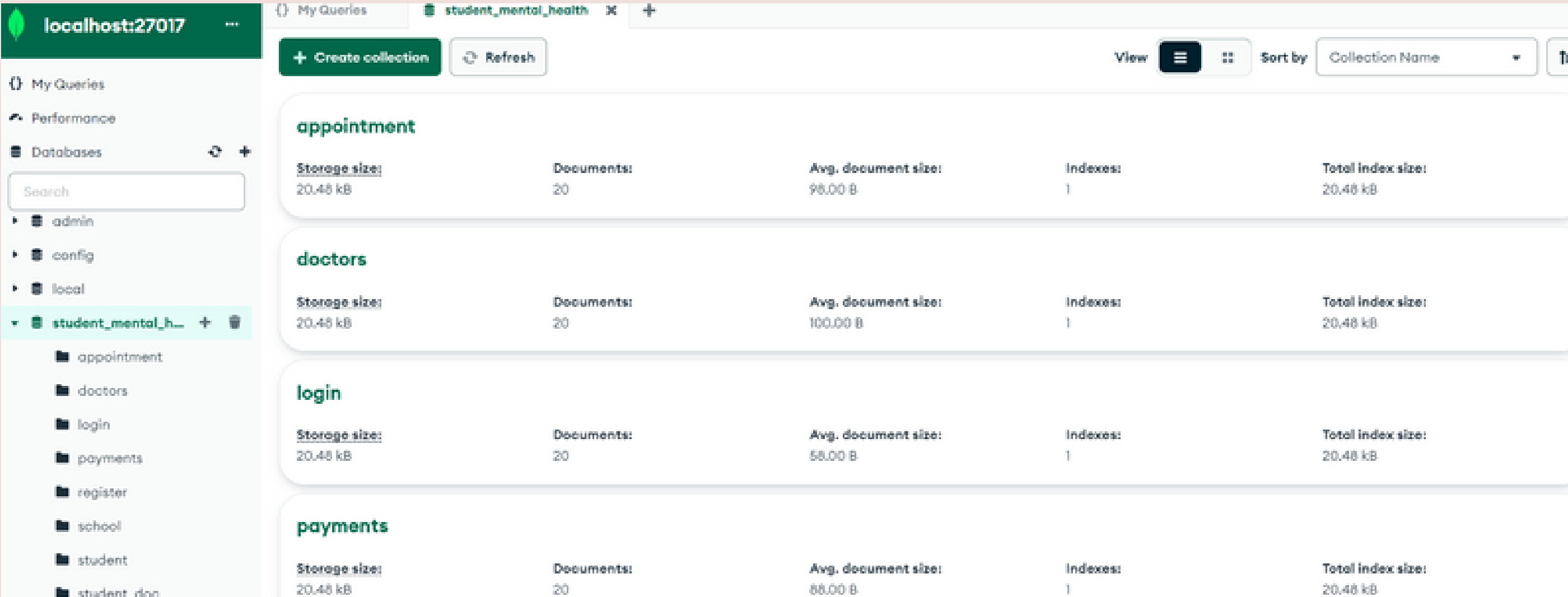
Data Size: 134.42 MB

 Browse collections →

 View monitoring →

+ Add Tag

# NoSQL Collections created on MongoDB for all the MySQL tables as shown



The screenshot shows the MongoDB Compass interface. On the left, a sidebar lists databases: 'admin', 'config', 'local', and 'student\_mental\_h...'. The 'student\_mental\_h...' database is selected, showing a list of collections: 'appointment', 'doctors', 'login', 'payments', 'register', 'school', 'student', and 'student\_doc'. The main panel displays details for four collections: 'appointment', 'doctors', 'login', and 'payments'. Each collection's details are shown in a card format with a table of statistics.

Collection Name	Storage size	Documents	Avg. document size	Indexes	Total index size
appointment	20.48 kB	20	98.00 B	1	20.48 kB
doctors	20.48 kB	20	100.00 B	1	20.48 kB
login	20.48 kB	20	58.00 B	1	20.48 kB
payments	20.48 kB	20	88.00 B	1	20.48 kB

# Finding Male Students

```
> db.student.find({"Gender": "Male"});
< {
  _id: ObjectId("661ca5f3389e0c978178b79c"),
  StudentID: 1,
  SchoolID_FK: 1,
  UserID_FK: 1,
  Name: 'John Doe',
  Age: 20,
  PhoneNumber: '555-0001',
  Gender: 'Male',
  Course: 'Computer Science',
  CreditLoad: 15,
  GPA: 3.5
}
{
  _id: ObjectId("661ca5f3389e0c978178b79f"),
  StudentID: 4,
  SchoolID_FK: 4,
  UserID_FK: 4,
  Name: 'Chris Lee',
```

# Finding students whose age is above 20

```
> db.student.find({ "Age": { $gt:20 } });  
< {  
  _id: ObjectId("661ca5f3389e0c978178b79d"),  
  StudentID: 2,  
  SchoolID_FK: 2,  
  UserID_FK: 2,  
  Name: 'Jane Smith',  
  Age: 22,  
  PhoneNumber: '555-0002',  
  Gender: 'Female',  
  Course: 'English Literature',  
  CreditLoad: 12,  
  GPA: 3.8  
}  
{  
  _id: ObjectId("661ca5f3389e0c978178b79f"),  
  StudentID: 4,  
  SchoolID_FK: 4,  
  UserID_FK: 4,  
  Name: 'Chris Lee',  
  Age: 21,  
  PhoneNumber: '555-0004',  
  Gender: 'Male',  
  Course: 'Visual Arts',  
  CreditLoad: 14,  
  GPA: 3.9  
}
```

# Finding students whose GPA is above 3.5

```
> db.student.find({ "GPA": { $gt:3.5 } });  
< {  
  _id: ObjectId("661ca5f3389e0c978178b79d"),  
  StudentID: 2,  
  SchoolID_FK: 2,  
  UserID_FK: 2,  
  Name: 'Jane Smith',  
  Age: 22,  
  PhoneNumber: '555-0002',  
  Gender: 'Female',  
  Course: 'English Literature',  
  CreditLoad: 12,  
  GPA: 3.8  
}  
{  
  _id: ObjectId("661ca5f3389e0c978178b79e"),  
  StudentID: 3,  
  SchoolID_FK: 3,  
  UserID_FK: 3,  
  Name: 'Alex Johnson',  
  Age: 19,  
  PhoneNumber: '555-0003',  
  Gender: 'Non-Binary',  
  Course: 'Biology',  
  CreditLoad: 16,  
  GPA: 3.6
```

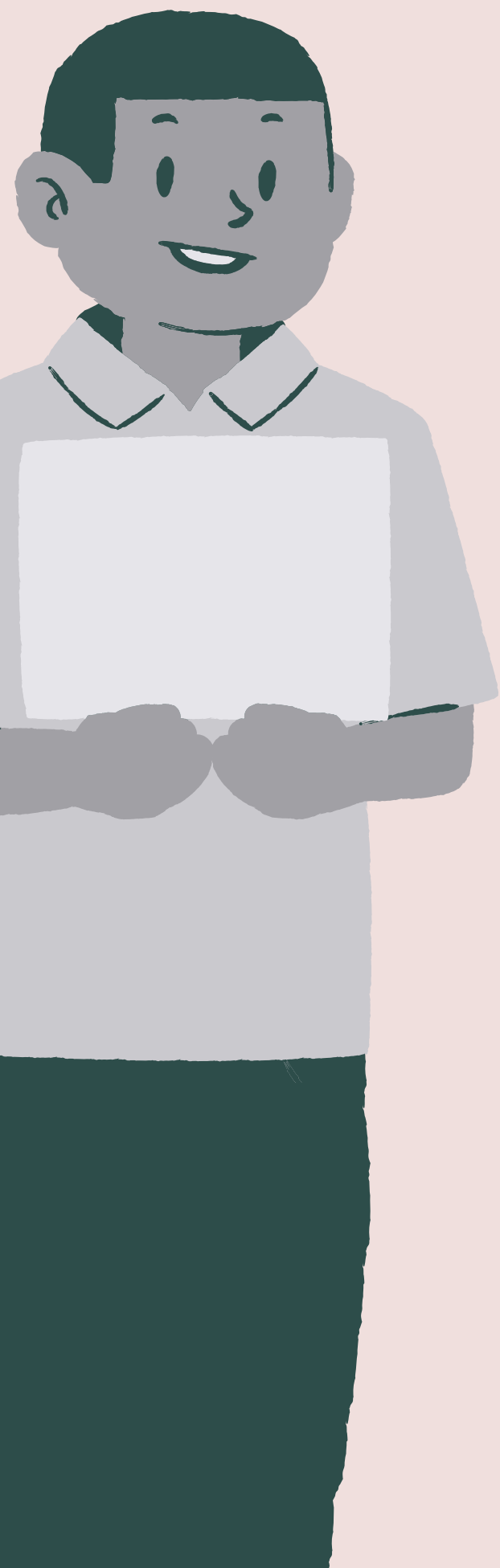


# Counting number of students based on their gender

```
> db.student.aggregate([ { $group: { _id: "$Gender", total: { $sum: 1 } } } ] )
< {
  _id: 'Male',
  total: 7
}
{
  _id: 'Female',
  total: 7
}
{
  _id: 'Non-Binary',
  total: 6
}
```

# Students whose name starts with A

```
> db.student.find({Name: /^A/ })
< {
  _id: ObjectId("661ca5f3389e0c978178b79e"),
  StudentID: 3,
  SchoolID_FK: 3,
  UserID_FK: 3,
  Name: 'Alex Johnson',
  Age: 19,
  PhoneNumber: '555-0003',
  Gender: 'Non-Binary',
  Course: 'Biology',
  CreditLoad: 16,
  GPA: 3.6
}
{
  _id: ObjectId("661ca5f3389e0c978178b7a7"),
  StudentID: 12,
  SchoolID_FK: 2,
  UserID_FK: 12,
  Name: 'Alexis King',
  Age: 22,
  PhoneNumber: '555-0012',
  Gender: 'Non-Binary',
  Course: 'History',
  CreditLoad: 12,
  GPA: 3.8
}
```



# THANK YOU

Your support is truly appreciated. Goodbye!

