Student Spendings CLI Documentation

Yatharth Shah

CMPSC 431W – SP24

GitHub Link –

<https://github.com/yms5350/ymsCMPSC431W-Final-Project>

Video Recording Link –

<https://psu.zoom.us/rec/share/8n8Q2cLQnS26iibKS3A7TEcTdrlj-nlVUEBmYvAmcMQ8gyloq6ibdN-z0jyfkBUx.CIt5QpS1WRMTSeHq?startTime=1713998020000>

Overview:

This project creates a powerful database management system capable of handling large amounts of student data across multiple dimensions, such as demographics, academics, expenses, finances, and payment methods. The system uses PostgreSQL to provide a comprehensive set of functionalities that are accessible via a Command Line Interface (CLI). This solution is designed for educational institutions that need a flexible and efficient way to dynamically manage and analyze student information.

Implementation:

1. Libraries Required:

Before deploying and utilizing the database system, ensure that the following software and libraries are installed:

PostgreSQL: a powerful open-source object-relational database system that uses and extends the SQL language, as well as many features for safely storing and scaling even the most complex data workloads.

Python: an interpreted, high-level, and all-purpose programming language. Python's design philosophy emphasizes code readability, as evidenced by its extensive use of whitespace.

psycopg2: Python's PostgreSQL database adapter. It is designed to efficiently handle large amounts of data as well as smaller operations.

To install psycopg2, use pip, Python's package installer. Run the commands below in your terminal:

* pip install psycopg2

1. Database Setup:

Before running the Python scripts, set up your PostgreSQL database.

* CREATE DATABASE students\_db; ( students\_db is just an example, you can name it whatever you want to.)

1. Table Creation:

To create the necessary tables within students\_db, execute the following SQL commands:

CREATE TABLE StudentDemographics (

Student\_ID INT PRIMARY KEY,

First\_Name VARCHAR(50),

Last\_Name VARCHAR(50),

Age INT,

Gender VARCHAR(50)

);  
  
CREATE TABLE StudentAcademics (

Student\_ID INT PRIMARY KEY,

First\_Name VARCHAR(50),

Last\_Name VARCHAR(50),

Major VARCHAR(50),

Year\_In\_School VARCHAR(50),

FOREIGN KEY (Student\_ID) REFERENCES StudentDemographics(Student\_ID)

);  
  
CREATE TABLE StudentFinances (

Student\_ID INT PRIMARY KEY,

Monthly\_Income INT,

Financial\_Aid INT,

Tuition INT,

FOREIGN KEY (Student\_ID) REFERENCES StudentDemographics(Student\_ID)

);

CREATE TABLE StudentExpenses (

Student\_ID INT PRIMARY KEY,

Housing INT,

Food INT,

Transportation INT,

Books\_Supplies INT,

Entertainment INT,

Personal\_Care INT,

Technology INT,

Health\_Wellness INT,

Miscellaneous INT,

FOREIGN KEY (Student\_ID) REFERENCES StudentDemographics(Student\_ID)

);

CREATE TABLE StudentPaymentMethods (

Student\_ID INT PRIMARY KEY,

Preferred\_Payment\_Method VARCHAR(50),

FOREIGN KEY (Student\_ID) REFERENCES StudentDemographics(Student\_ID)

);

Remember, these are the commands I used for my database. You can change it however you want to.

1. Python Implementation:

The connect\_to\_db function sets up a connection to the PostgreSQL server. Update the connection parameters to match your PostgreSQL configuration:

* import psycopg2

def connect\_to\_db():

try:

conn = psycopg2.connect(

dbname="students\_db",

user="your\_username",

password="your\_password",

host="localhost"

)

return conn

except Exception as error:

print(f"Error: {error}")

return None

CLI Functionality:

The student database management system's command line interface (CLI) is designed to allow users to interact directly with the database using a variety of menu-driven options. Each choice corresponds to a specific database operation or a set of operations that handle data input, changes, and retrieval. Below is an explanation of each menu choice and its accompanying usefulness.

Main Menu :

1. Insert data: Selecting this option takes the user to the Insert Data Menu, where they can create new entries in categories such as student demographics, academics, expenses, finances, and payment methods. This brings us to the INSERT MENU :
   1. Insert Student Demographics: Collects and stores information like student ID, first name, last name, age, and gender in the studentdemographics table.
   2. Insert Student Academics: Inputs academic details such as student ID (linked to demographics), first name, last name, major, and year in school into the studentacademics table.
   3. Insert Student Expenses: Enters various expenses (housing, food, transportation, etc.) linked by student ID into the studentexpenses table.
   4. Insert Student Finances: Adds data regarding a student's finances, including monthly income, financial aid, and tuition fees into the studentfinances table.
   5. Insert Student Payment Methods: Stores information about preferred payment methods for each student in the studentpaymentmethods table.
2. Update data: This option opens the Update Data Menu, which allows you to update existing records in the database across multiple tables based on student IDs. This brings us to the UPDATE MENU :
   1. Update Student Demographics: Updates demographic details like first name, last name, age, or gender for a specified student ID in the studentdemographics table.
   2. Update Student Academics: Allows modification of academic records such as major or year in school for a given student ID in the studentacademics table.
   3. Update Student Expenses: Users can update expense categories for a student, reflecting changes in costs like housing or food in the studentexpenses table.
   4. Update Student Finances: Updates financial records, including changes in monthly income, financial aid, or tuition fees in the studentfinances table.
   5. Update Student Payment Methods: Modifies the preferred payment methods recorded for a student in the studentpaymentmethods table.
3. Delete data: Users can delete records from the database. This function is sensitive because it permanently deletes student records and associated data from all tables.
4. List all data: Displays an exhaustive list of all student data, bringing together several tables to provide a cohesive picture of the data housed in the database.
5. Search Data: Users can search for specific student data based on their ID, displaying relevant data from multiple linked databases.
6. Sort data: This function sorts several student data types, such as demographics and academics, based on provided fields. This brings us to the SORT MENU :
   1. Sort Student Demographics: Sorts demographic data by specified fields such as age or gender.
   2. Sort Student Academics: Sorts academic data by fields like major or year in school.
   3. Sort Student Expenses: Sorts expense data by categories such as housing or food.
   4. Sort Student Finances: Sorts financial data by fields such as monthly income or financial aid.
   5. Sort Student Payment Methods: Sorts payment method data by the preferred method.
7. Group Data: Users can aggregate data based on specified features (for example, gender in demographics or major in academics) to obtain aggregated views such as counts or sums. This brings us to the GROUP DATA MENU :
   1. Group Student Demographics by Gender: Displays the count of students grouped by gender.
   2. Group Student Academics by Major: Shows the number of students grouped by their major.
   3. Group Student Expenses by Category: Aggregates total expenses by each category like housing or food.
   4. Group Student Finances by Income Range: Groups students by income ranges and shows the count for each range.
   5. Group Student Payment Methods by Type: Counts the number of students by each type of payment method.
8. Students above-average expenses: Finds and identifies all students with higher-than-average costs in several categories, which aids in financial analysis and budgeting.
9. Aggregate Student Finances: This option generates and displays aggregate financial data for all students, including average monthly income and total financial aid.

CLI Input Instructions:

Insert:

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Sub-Option | User Input | Definition |
| 1 | 1 – Student Demographics | Student ID | Unique Student ID |
|  |  | First Name | Given name of the student |
|  |  | Last Name | Family name of the student |
|  |  | Age | Age of the student |
|  |  | Gender | Gender of the Student |
|  | 2 – Student Academics | Student ID | Unique Student ID (matches Demographics) |
|  |  | Major | Main field of study of the student |
|  |  | Year in School | Current academic year of the student |
|  | 3 – Student Expenses | Student ID | Unique Student ID (matches Demographics) |
|  |  | Housing | Annual Housing expenses |
|  |  | Food | Annual food expenses |
|  |  | Transportation | Annual transportation expense |
|  |  | Books & Supplies | Annual cost of books and supplies |
|  |  | Entertainment | Annual entertainment expenses |
|  |  | Personal Care | Annual personal care expenses |
|  |  | Technology | Annual technology expenses |
|  |  | Health and Wellness | Annual cost of medicines , health and wellness products or treatments |
|  |  | Miscellaneous | Annual miscellaneous expenses |
|  | 4 – Student Finances | Student ID | Unique Student ID (matches Demographics) |
|  |  | Monthly Income | Monthly income of the student |
|  |  | Financial Aid | Annual financial aid received |
|  |  | Tuition | Annual Tuition fees |
|  | 5 – Student Payment Methods | Student ID | Unique Student ID (matches Demographics) |
|  |  | Preferred Payment Method | Preferred method for payments ( eg, Credit Card , Cash etc.) |

Expected Output on Successful :

* “ Record Added Successfully. “

Update :

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Sub-Option | User Input | Definition |
| 2 | 1 – Student Demographics | Student ID | Unique Student ID |
|  |  | First Name ( Optional ) | Updated Given name of the student |
|  |  | Last Name ( Optional ) | Updated Family name of the student |
|  |  | Age ( Optional ) | Updated Age of the student |
|  |  | Gender ( Optional ) | Updated Gender of the Student |
|  | 2 – Student Academics | Student ID | Unique Student ID (matches Demographics) |
|  |  | Major ( Optional ) | Updated Main field of study of the student |
|  |  | Year in School ( Optional ) | Updated Current academic year of the student |
|  | 3 – Student Expenses | Student ID | Unique Student ID (matches Demographics) |
|  |  | Housing ( Optional ) | Updated Annual Housing expenses |
|  |  | Food ( Optional ) | Updated Annual food expenses |
|  |  | Transportation ( Optional ) | Updated Annual transportation expense |
|  |  | Books & Supplies ( Optional ) | Updated Annual cost of books and supplies |
|  |  | Entertainment ( Optional ) | Updated Annual entertainment expenses |
|  |  | Personal Care ( Optional ) | Updated Annual personal care expenses |
|  |  | Technology ( Optional ) | Updated Annual technology expenses |
|  |  | Health and Wellness ( Optional ) | Updated Annual cost of medicines , health and wellness products or treatments |
|  |  | Miscellaneous ( Optional ) | Updated Annual miscellaneous expenses |
|  | 4 – Student Finances | Student ID | Unique Student ID (matches Demographics) |
|  |  | Monthly Income ( Optional ) | Updated Monthly income of the student |
|  |  | Financial Aid ( Optional ) | Updated Annual financial aid received |
|  |  | Tuition ( Optional ) | Updated Annual Tuition fees |
|  | 5 – Student Payment Methods | Student ID | Unique Student ID (matches Demographics) |
|  |  | Preferred Payment Method ( Optional ) | Updated Preferred method for payments ( eg, Credit Card , Cash etc.) |

Note : If you don’t want to update a particular Input ( For eg: You want to update the Year in School but not the Major), just leave it blank for no change.

Expected Output on Successful Update :

* “ Record Updated Successfully”

Delete :

|  |  |  |
| --- | --- | --- |
| Option | User Input | Definition |
| 3 | Student ID | Unique Student ID to identify the record to be deleted |

Expected Output on Successful Deletion :

* “ Record Deleted Successfully”

List All Records:

|  |  |  |
| --- | --- | --- |
| Option | User Input | Definition |
| 4 | - | Lists all student records in a unified view |

Expected Output:

* “ Unified Student Data:

Student ID First Name Last Name Age Gender Major Year in School Housing Food Transportation Books Supplies Entertainment Personal Care Technology Health Wellness Miscellaneous Monthly Income Financial Aid Tuition Preferred Payment Method

* 0 John Doe 19 Non-binary Psychology Freshman 709 296 123 188 41 78 134 127 72 958 270 5939 Credit/Debit Card”

( Just an example if you had one dataset. Original has many data points, so I didn’t include all of them) .

Search for a specific Record :

|  |  |  |
| --- | --- | --- |
| Option | User Input | Definition |
| 5 | Student ID | Unique Student ID to search for specific student information. |

Expected Output:

* Enter student ID to search for: 12

Detailed Information for Student ID: 12

Student ID First Name Last Name Age Gender Major Year in School Housing Food Transportation Books Supplies Entertainment Personal Care Technology Health Wellness Miscellaneous Monthly Income Financial Aid Tuition Preferred Payment Method

12 Bukayo Saka 21 Male Economics Sophomore 894 280 120 126 41 50 78 113 114 719 540 4863 Credit/Debit Card

Sort :

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Sub-Option | User Input | Definition |
| 6 | 1 – Sort Student Demographics | Sort Field | The field name in student demographics to sort by |
|  |  | Order | Sort order ( ASC for ascending or DESC for descending) |
|  | 2 – Sort Student Academics | Sort Field | The field name in student academics to sort by |
|  |  | Order | Sort order ( ASC for ascending or DESC for descending) |
|  | 3 – Sort Student Expenses | Sort Field | The field name in student expenses to sort by |
|  |  | Order | Sort order ( ASC for ascending or DESC for descending) |
|  | 4 – Sort Student Finances | Sort Field | The field name in student finances to sort by |
|  |  | Order | Sort order ( ASC for ascending or DESC for descending) |
|  | 3 – Sort Student Payment Methods | Sort Field | The field name in student payment methods to sort by |
|  |  | Order | Sort order ( ASC for ascending or DESC for descending) |

Expected Output:

* Enter choice: 1

Sorting Student Demographics...

Enter column to sort by (e.g., age, gender): age

Choose order (ASC for ascending, DESC for descending): ASC

(9, 'Selina', 'Gomez', 18, 'Female')

(0, 'John', 'Doe', 19, 'Non-binary')

(13, 'Maria', 'Lopez', 19, 'Female')

(18, 'Isha', 'Ambani', 19, 'Female')

(20, 'Wayne', 'Rooney', 19, 'Male')

(4, 'Anushka', 'Kohli', 20, 'Female')

(19, 'Nita', 'Merchant', 20, 'Female')

(12, 'Bukayo', 'Saka', 21, 'Male')

(17, 'Surya', 'Yadav', 21, 'Female')

(15, 'Mario', 'Luiz', 22, 'Non-binary')

(8, 'Lewis', 'Hamilton', 22, 'Non-binary')

(10, 'Arjun', 'Kapoor', 23, 'Male')

(7, 'Jack', 'Grealish', 23, 'Female')

(6, 'Phil', 'Foden', 23, 'Female')

(3, 'Lionel', 'Messi', 23, 'Male')

(16, 'Lamin', 'Yamal', 23, 'Female')

(21, 'Suneel', 'Chettri', 24, 'Male')

(1, 'Garry', 'Potter', 24, 'Male')

(2, 'Cristiano', 'Ronaldo', 24, 'Male')

(14, 'Axel', 'Santino', 24, 'Non-binary')

(11, 'Yatharth', 'Shah', 25, 'Male')

(5, 'Jude', 'Bellingham', 25, 'Non-binary')

Group :

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Sub-Option | User Input | Definition |
| 7 | 1 – Group Student Demographics by Gender | - | - |
|  | 2 – Group Student Academics by Major | - | - |
|  | 3 – Group Student Expenses by Category | - | - |
|  | 4 – Group Student Finances by Income Range | - | - |
|  | 5 – Group Student Payment Methods by Type | - | - |

Expected Output:

* Enter choice: 1

Grouping Student Demographics by Gender...

Gender: Non-binary, Count: 5

Gender: Female, Count: 9

Gender: Male, Count: 8

Students Above Average Expenses :

|  |  |  |
| --- | --- | --- |
| Option | No User Input Needed | Definition |
| 8 | - | Identifies students with expenses above average |

Expected Output:

* Enter choice: 8

Student ID: 1, Total Expenses: 1848

Student ID: 4, Total Expenses: 1997

Student ID: 6, Total Expenses: 1905

Student ID: 8, Total Expenses: 1882

Student ID: 9, Total Expenses: 1858

Student ID: 11, Total Expenses: 2067

Student ID: 15, Total Expenses: 2097

Student ID: 16, Total Expenses: 2229

Student ID: 18, Total Expenses: 1880

Student ID: 19, Total Expenses: 2002

Student ID: 21, Total Expenses: 1976

Aggregate Student Finances :

|  |  |  |
| --- | --- | --- |
| Option | No User Input Needed | Definition |
| 9 | - | Computes financial aggregates such as average income and total financial aid |

Expected Output:

* Enter choice: 9

Average Monthly Income: 1050.86

Total Financial Aid: 10004.00

Exit :

|  |  |  |
| --- | --- | --- |
| Option | No User Input Needed | Definition |
| 0 | - | Exits the program safely |

NOTE : To insert data, I am uploading my CSV file and my project – 2 documentations on my GitHub. Look at all the instructions in the insert paragraph in the documentation and csv file for the data points . I did this because I cannot extract the sql file because of some issue in my laptop. I have confirmed it with the TA if I can do that ( TA – Teja Parasa , Wednesday 5 – 8 pm)