# Introduction

The purpose of the database is to store students information based on persona details , class attendance and affiliation , financial records and hostels assigned. The database is managed using the MySQL and manipulated though python programming language. The database contains records on students names, contacts, residence, classes, hostels and fees balance. The different tables are linked using foreign keys and attributes. in the students table, the registration number is the primary key. it all so acts as the foreign key in all other tables. The tables contains an average of five records which were initially contained on manual spreadsheets and will be manually entered into the database. The data is intended to be used by administrative, financial and hostel management staff.

# The database creation process uses MySQL work bench

Create a new schema called students

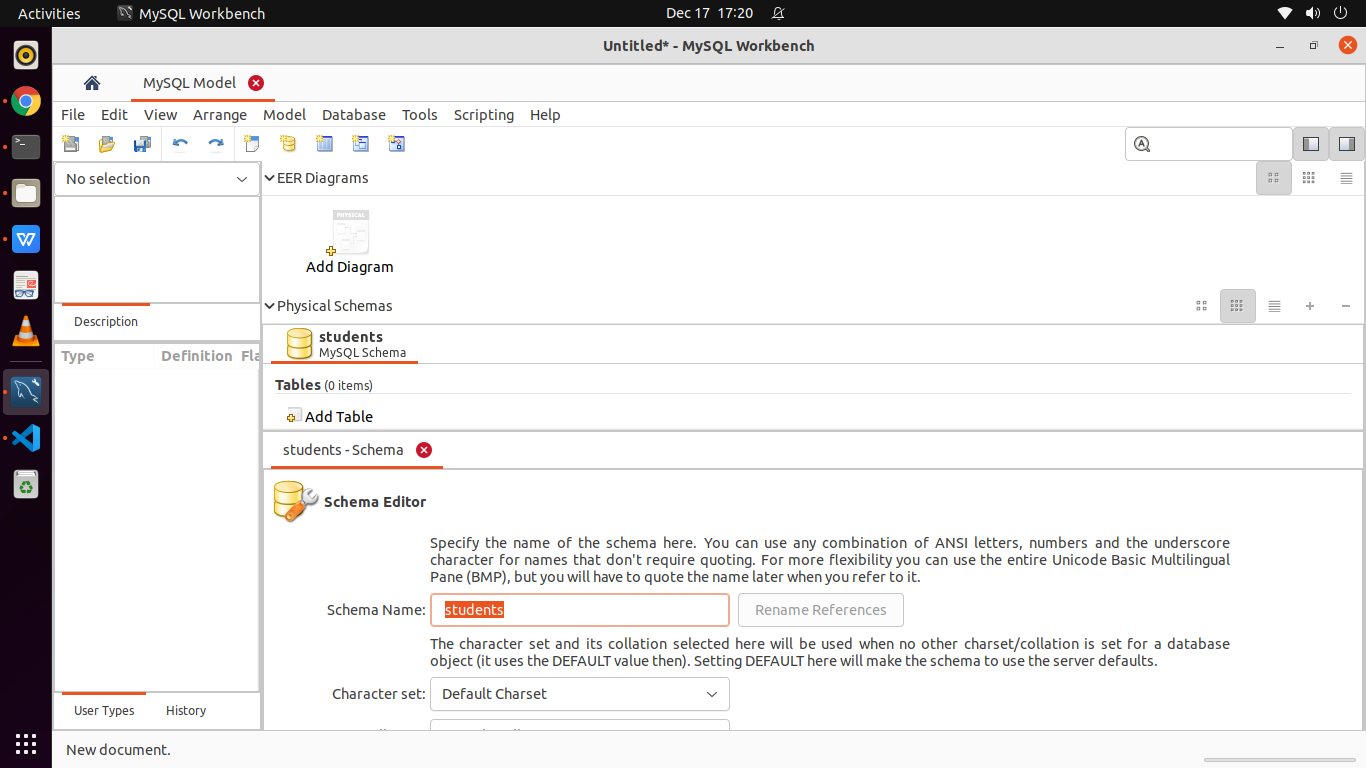
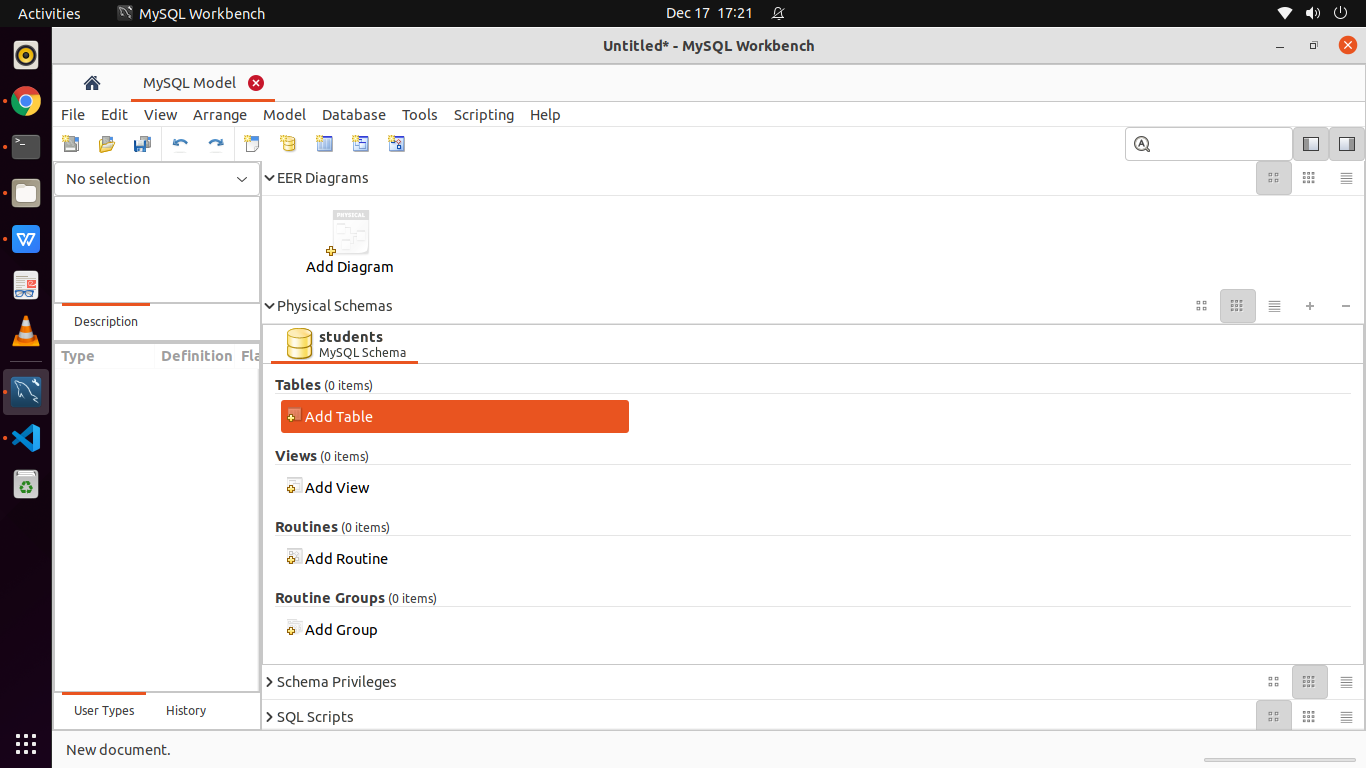


Figure 1 Create students schema

Click add table button and provide the required attributes



Provide attribute. repeat the process for all tables.

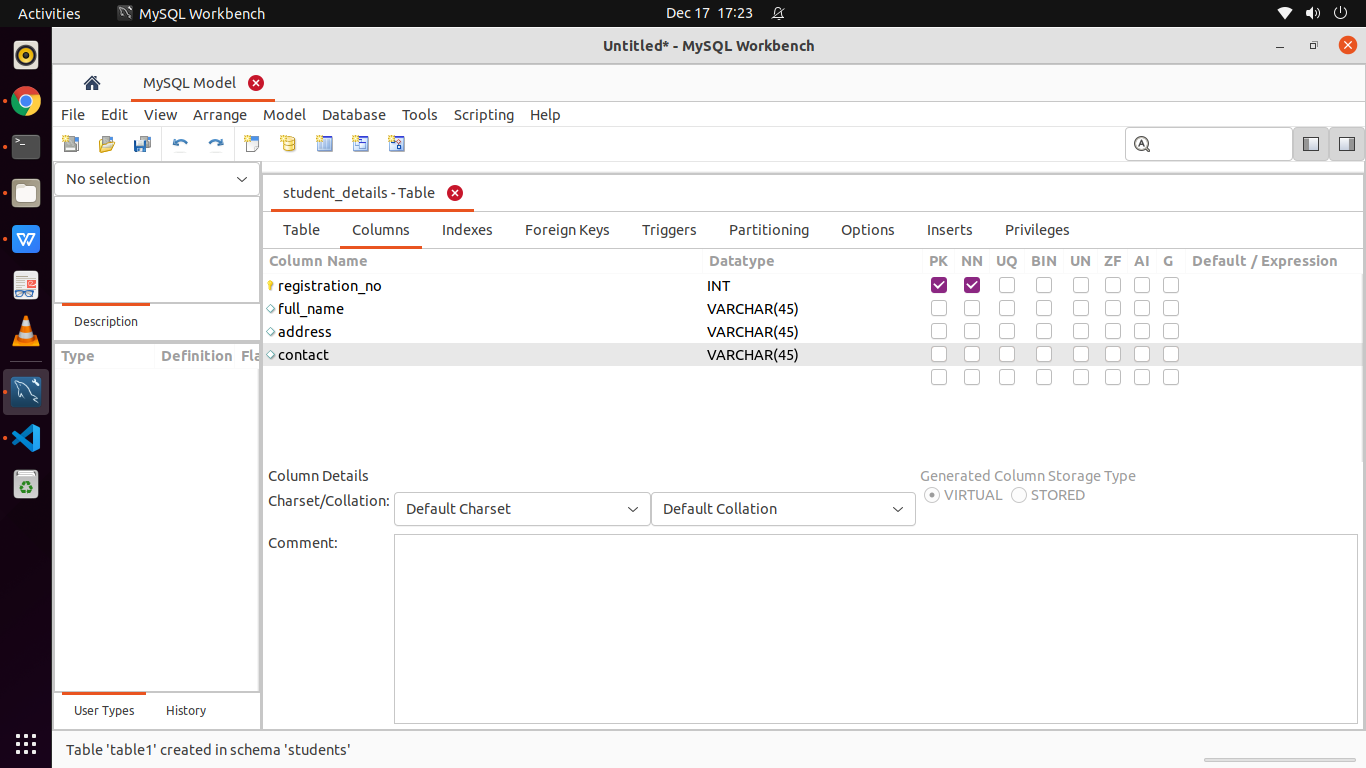


Figure 2 Add table attributes

# EER diagram

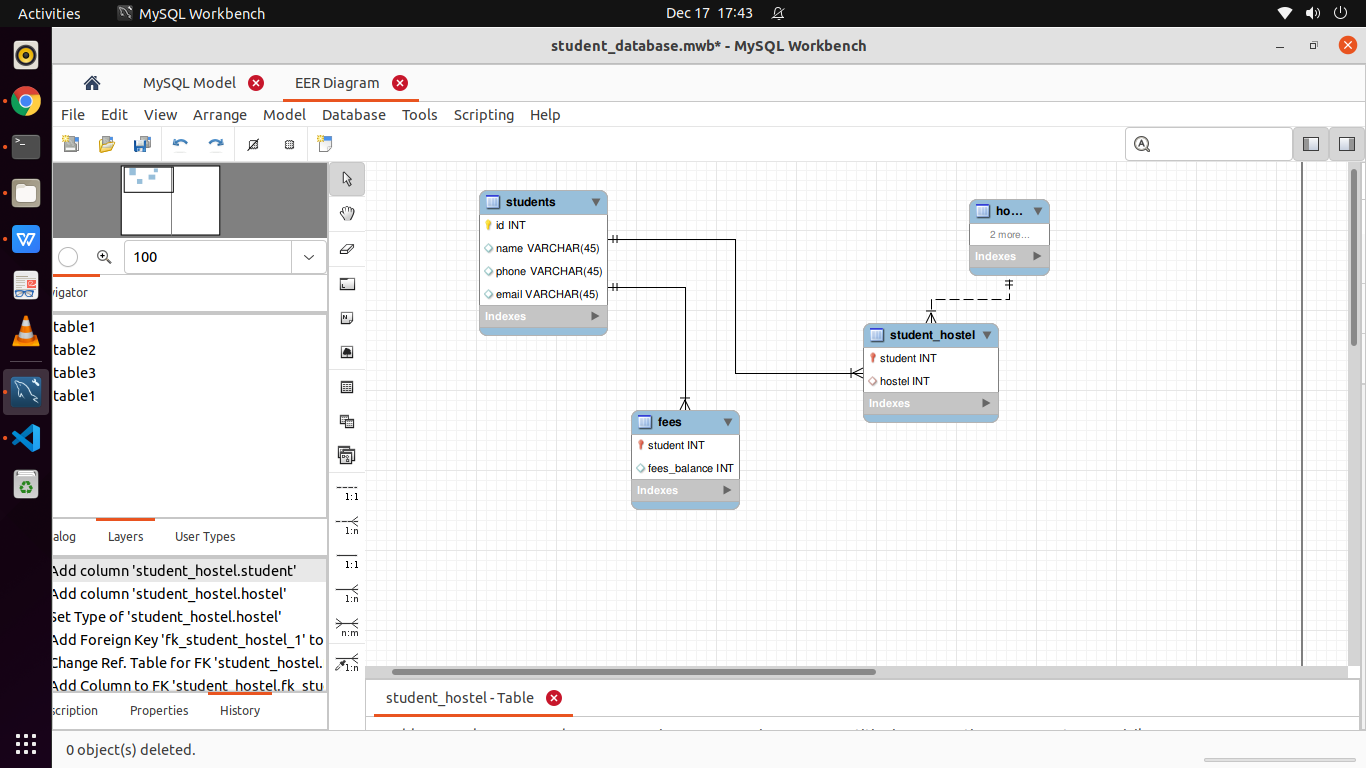


Figure 3 EER diagram showing table relations

# Database programming with python

The program is developed in Ubuntu version 21.10, the latest version as of December 2021. The process begins by installing MySQL-connector driver which facilitates connection between python and MySQL. The process is shown in the screenshot below.

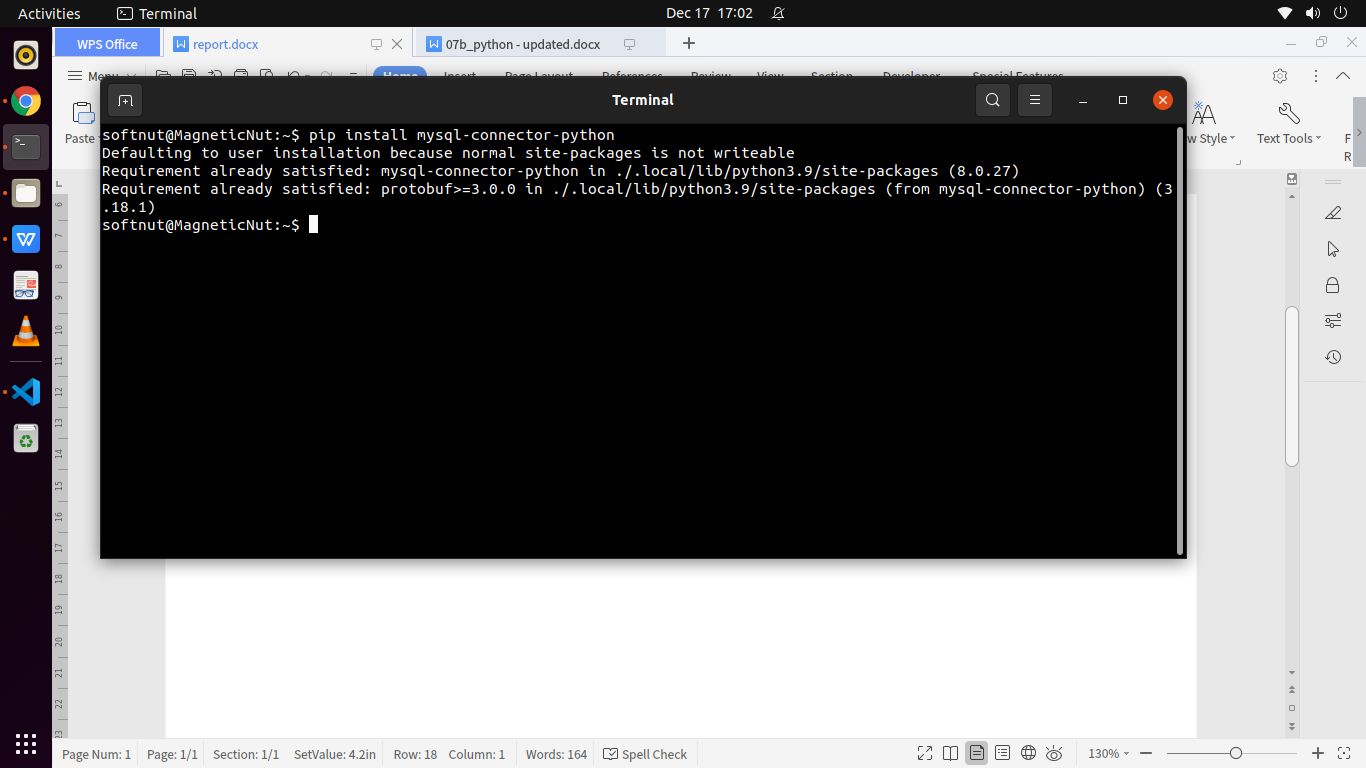


Figure 4 Install python MySQL connector module

There are several ways of adding modules to the development environment, however, using pip has proven to be the easiest of all and was my favorite choice in this project.

# Open MySQL on the Ubuntu terminal

To open MySQL on the terminal, use the command sudo MySQL -u root. However, if the MySQL settings require password provision, one may need to provide it as required.

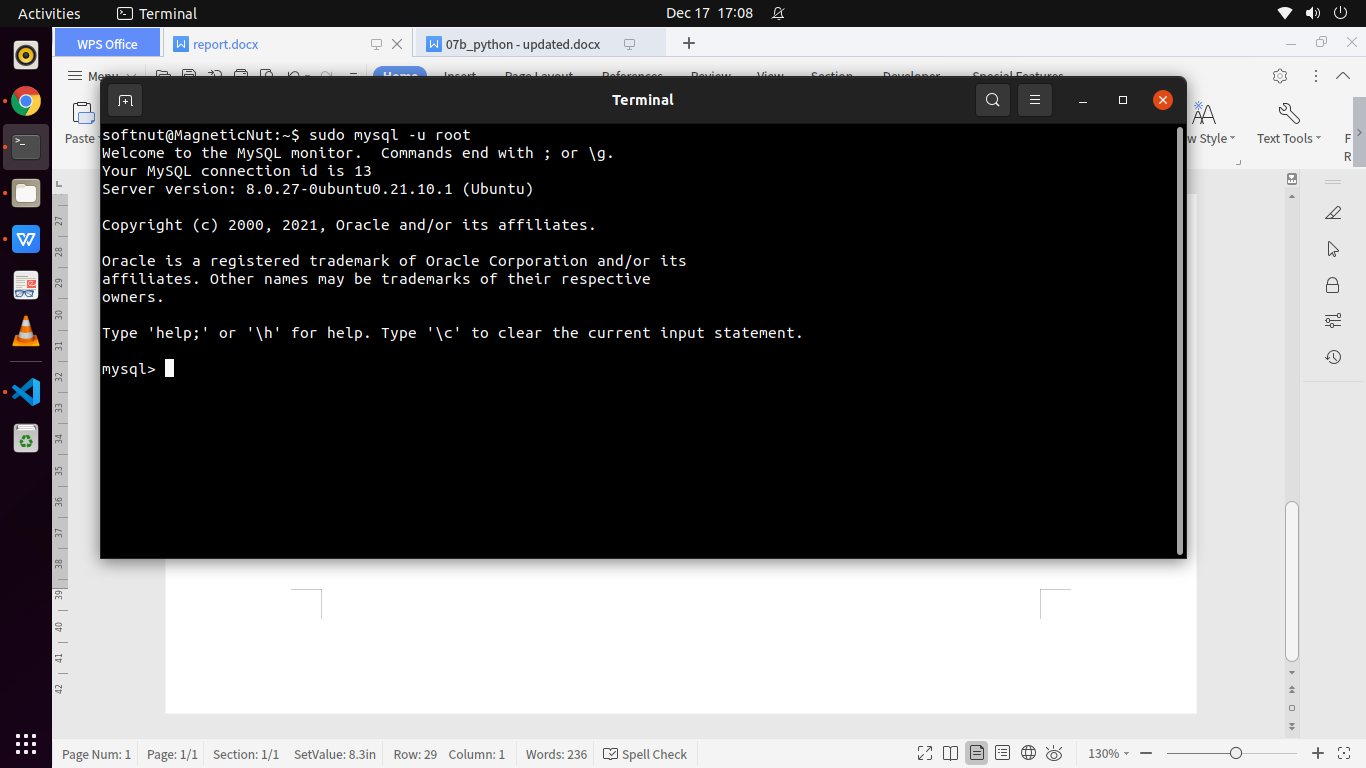


Figure 5 Open MySQL on Ubuntu terminal

In order to access database and carry out CRUD operations, we need to create a user and grant all the required privileges.

CREATE USER 'new\_user'@'localhost' IDENTIFIED BY 'password';

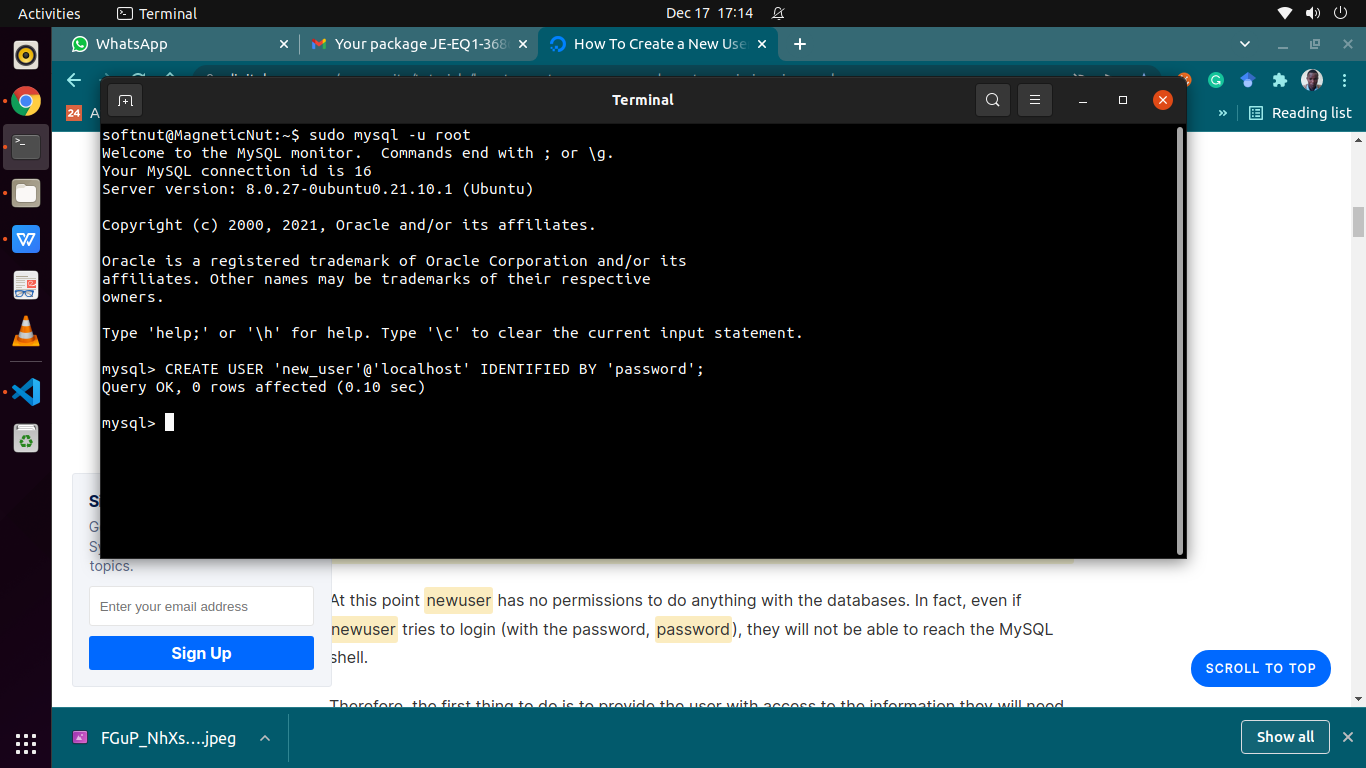
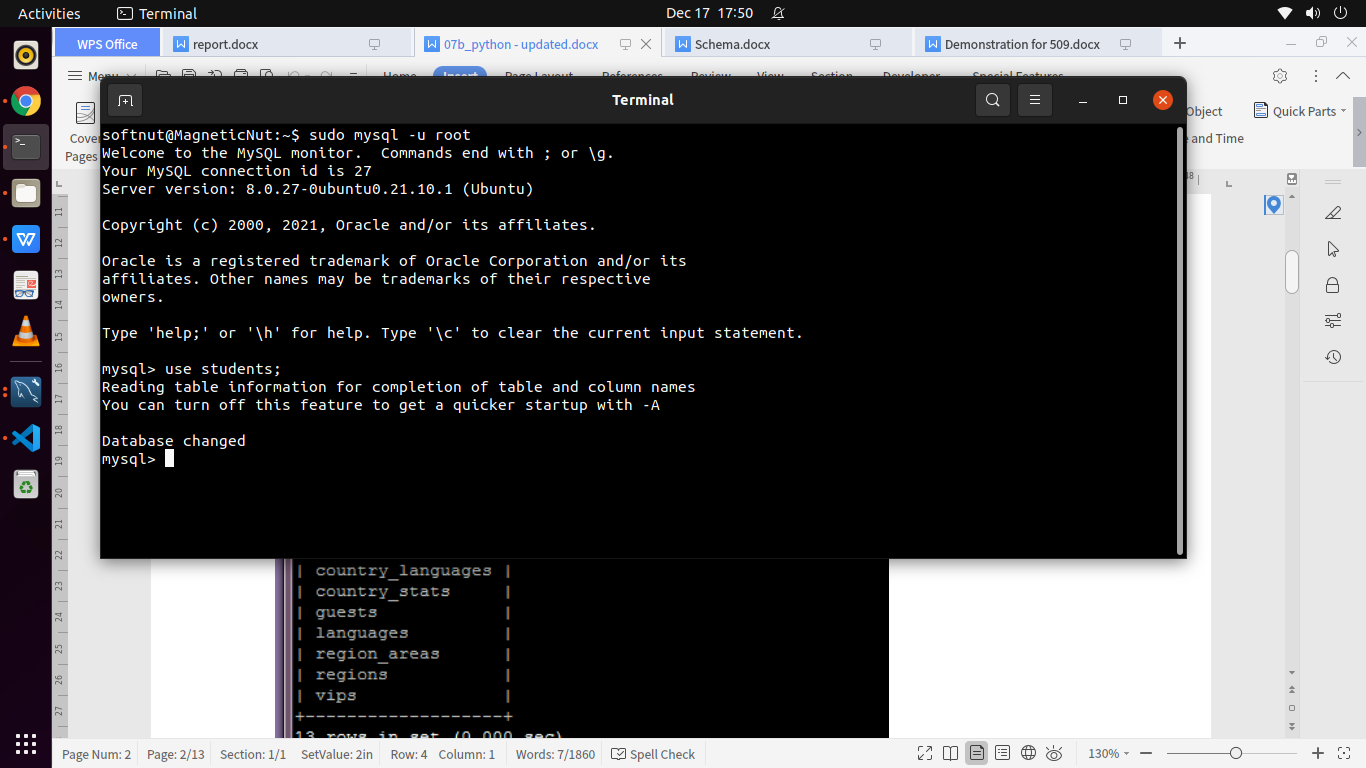
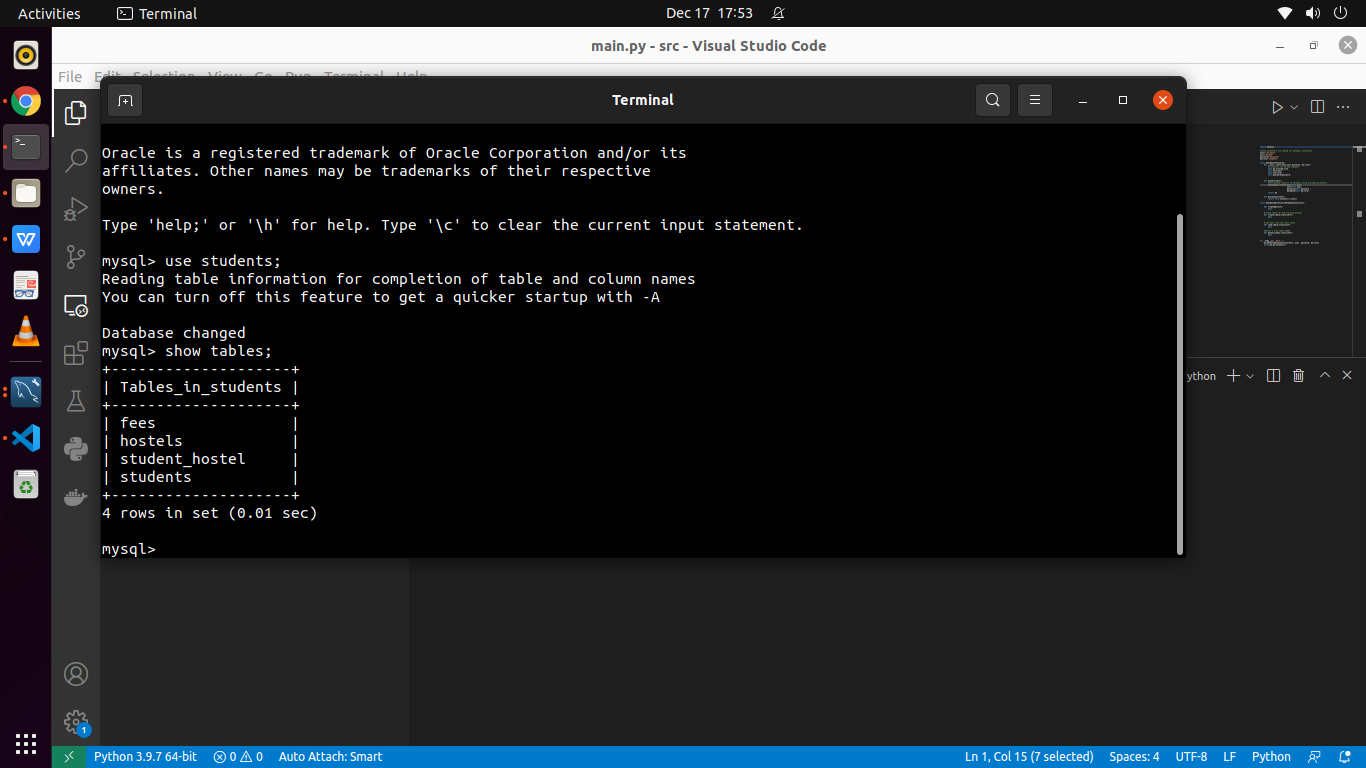


Figure 6 CREATE USER 'new\_user'@'localhost' IDENTIFIED BY 'password';

Now select students table for the next steps



The database tables are shown in the figure below.



Now i grant all privileges to the new user as shown below.

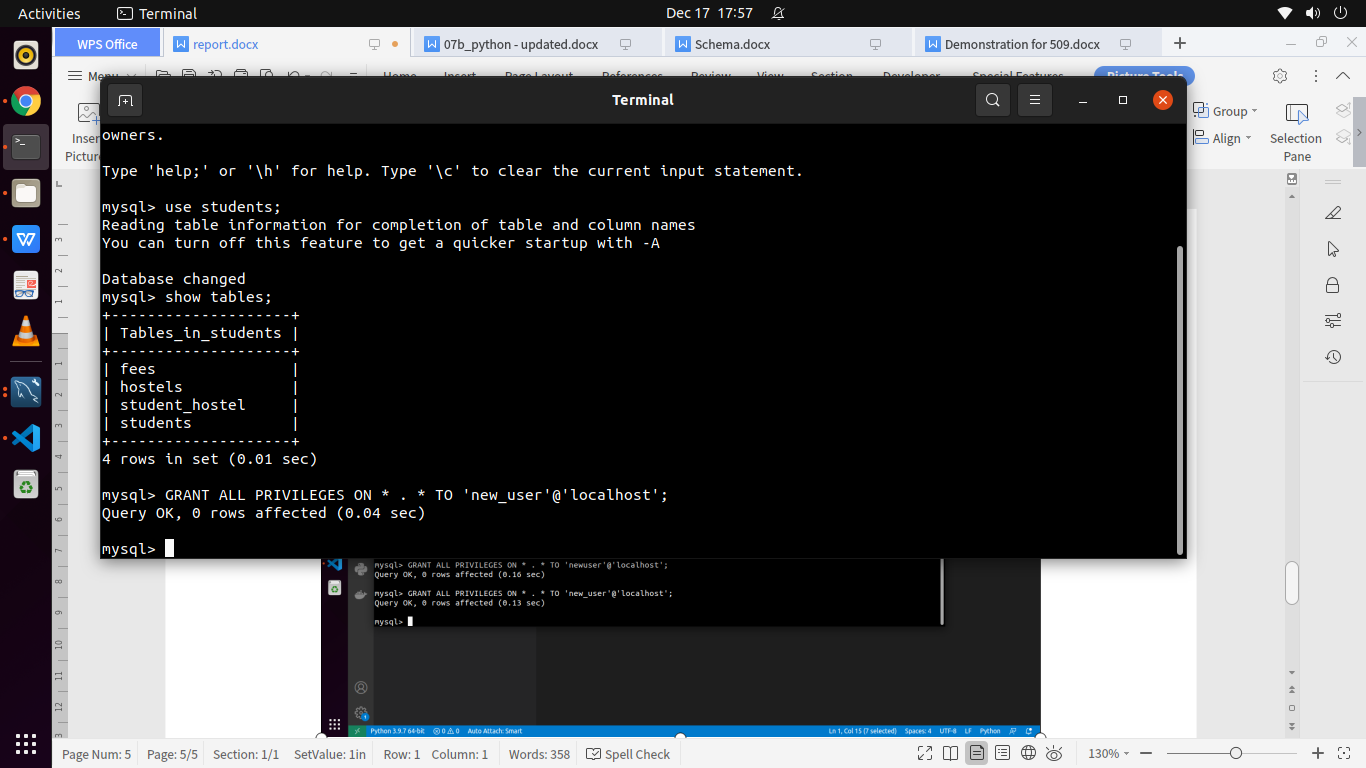


Figure 7 Grant privileges to new\_user

# Python programming Section

the python program is contained in a project called project. The source files are located in the src sub folder. the program begins by importing the MySQLdb module which is required for database connection.

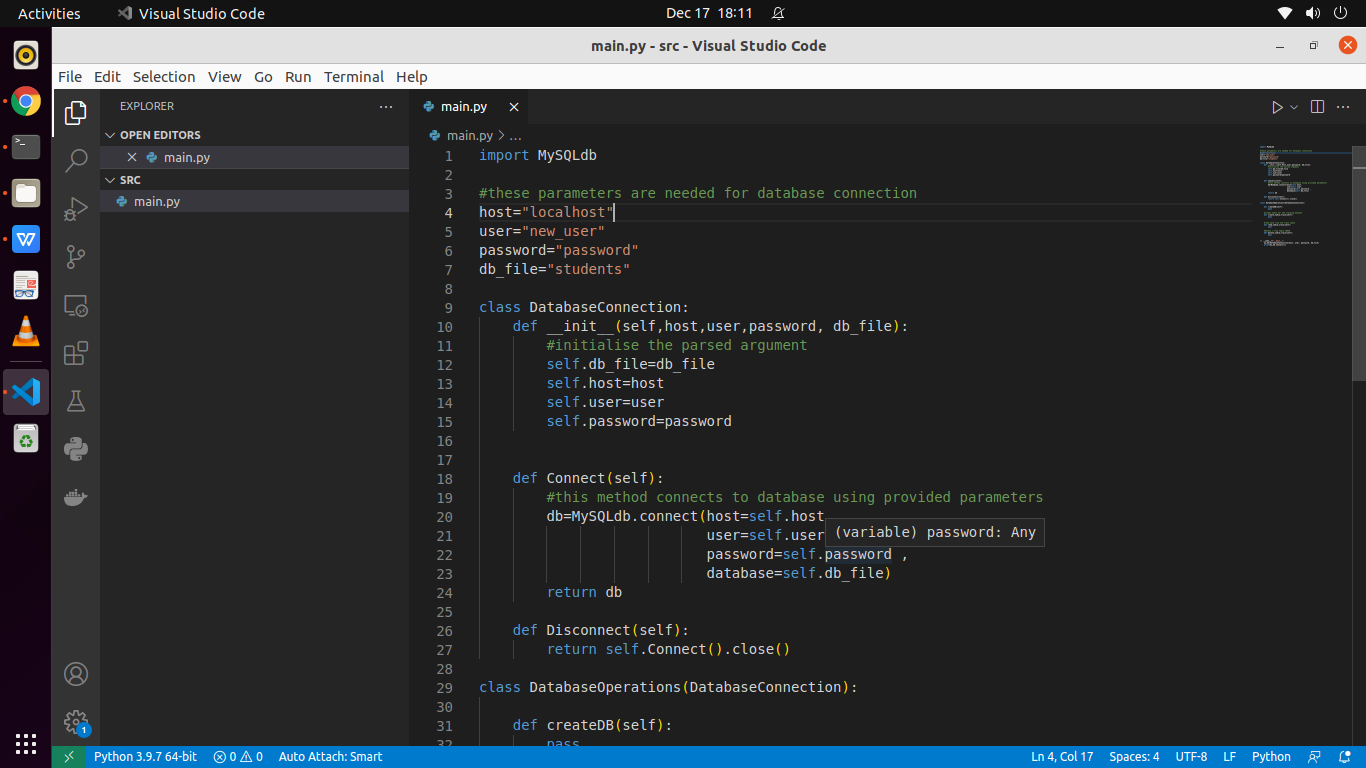
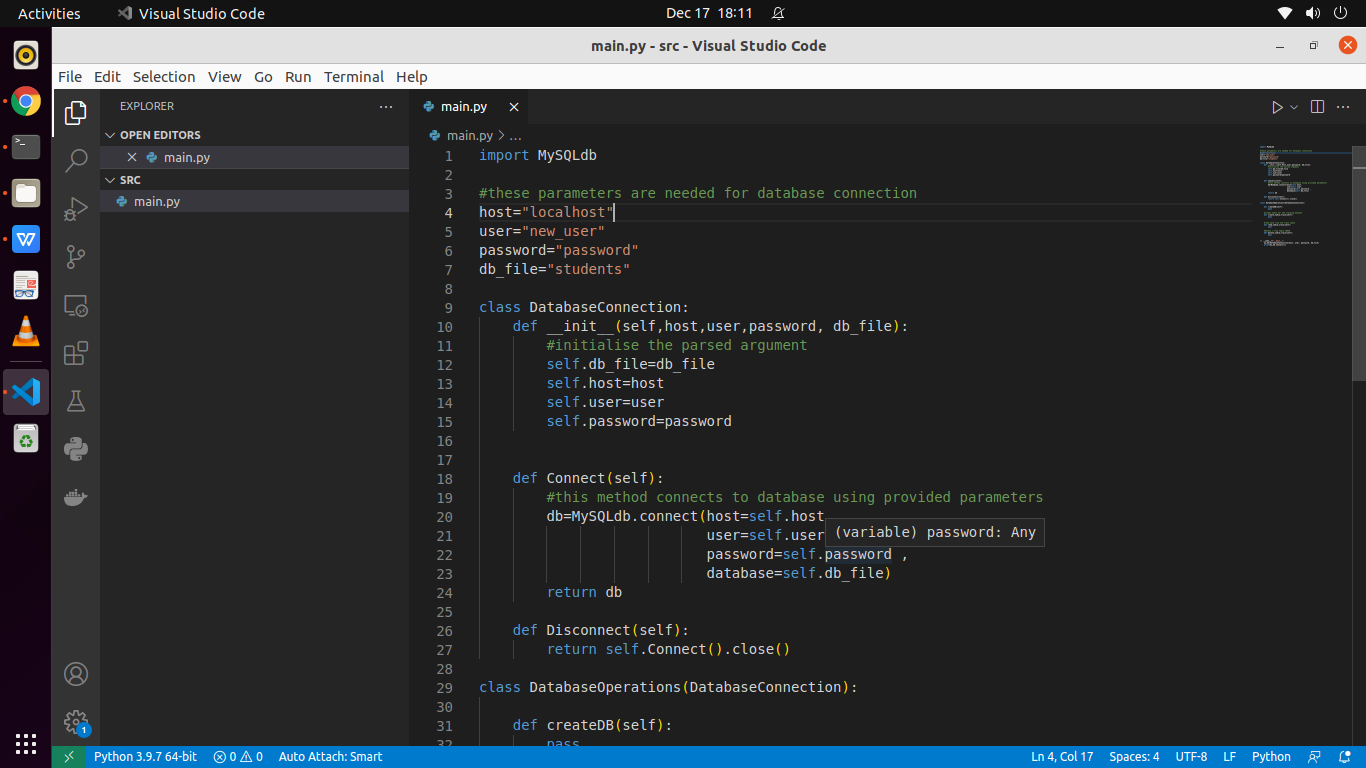


Figure 8 Importing required modules and database connection module

The actual coding was done in Visual Studio Code, a cross platform editor for windows, Linux and Mac. Developers are free to choose other editors such as IDLE, sublime text and pycharm.

Database Connection

The code provided in the screenshot below present the database connection class and methods. The class can be inherited to facilitate encapsulated database connection. It also provides an easy way of changing the connection parameters without changing the source code. The source code is provided in the appendix.



# Database Cursor

Database cursors queries SQL databases and stores the results for further processing, visualization or presentation. they are usually database specific and are created using the connection.cursor() method function.

# Data manipulation scripts

### Selecting table records

In this case, each table operation(create, read, update or delete is defined as an independent function.) as a result, i will only touch on the operations of one table, in this case, the students table.

Lets begin by selecting all the records of students table.

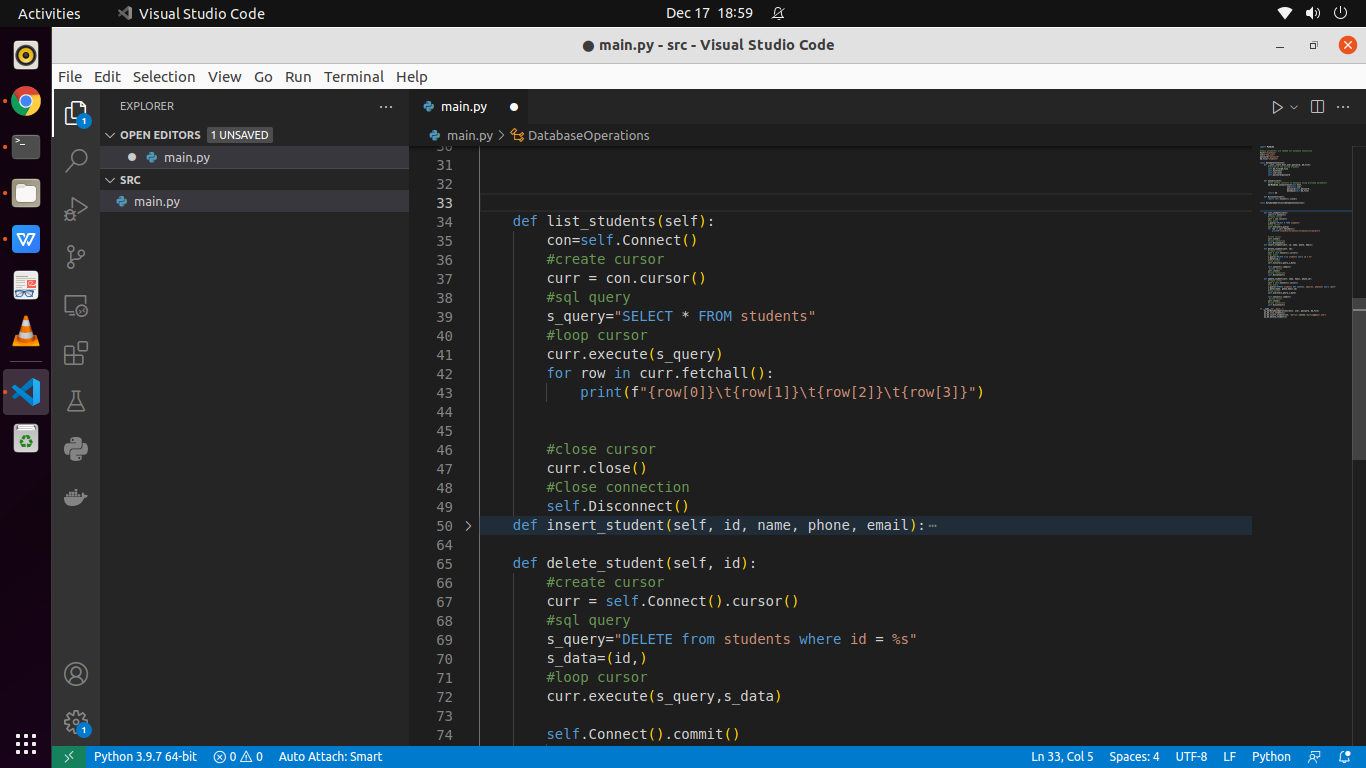
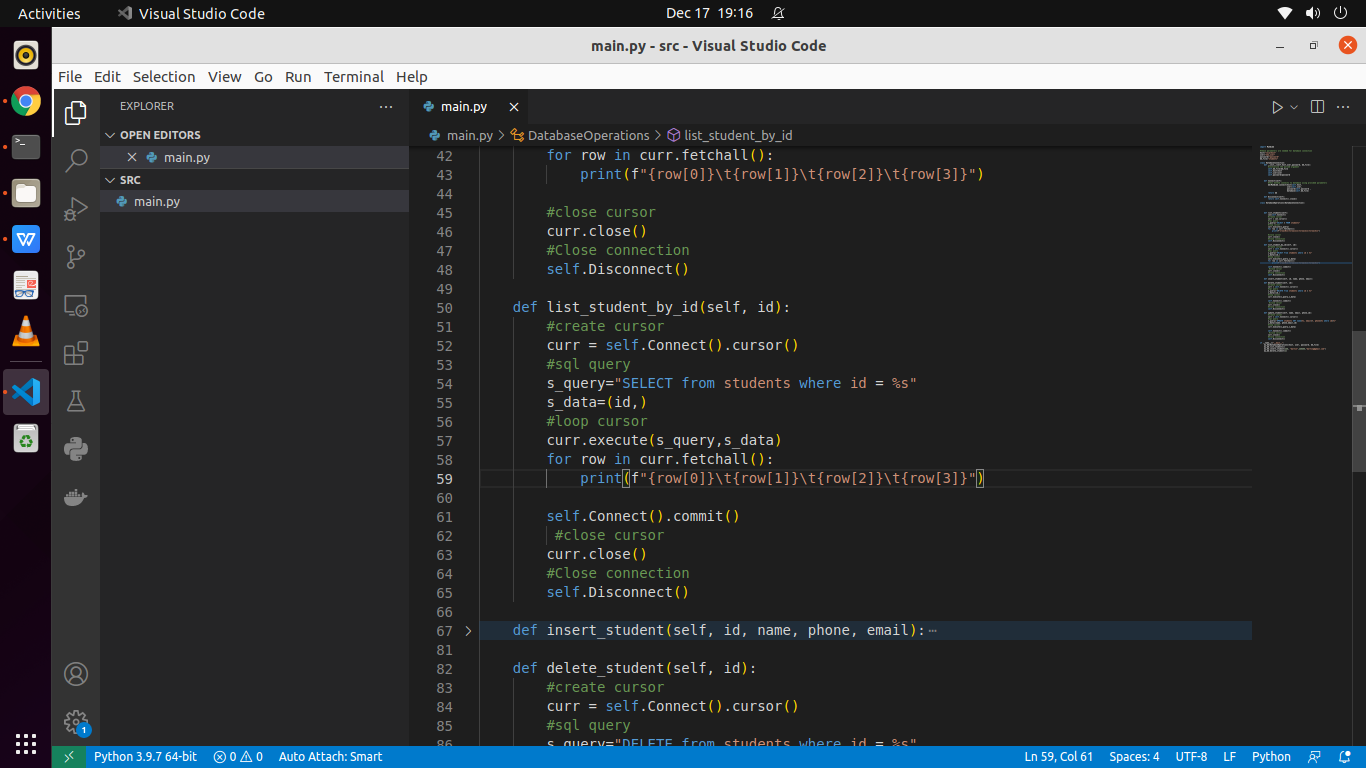


Figure 9 Select all table records

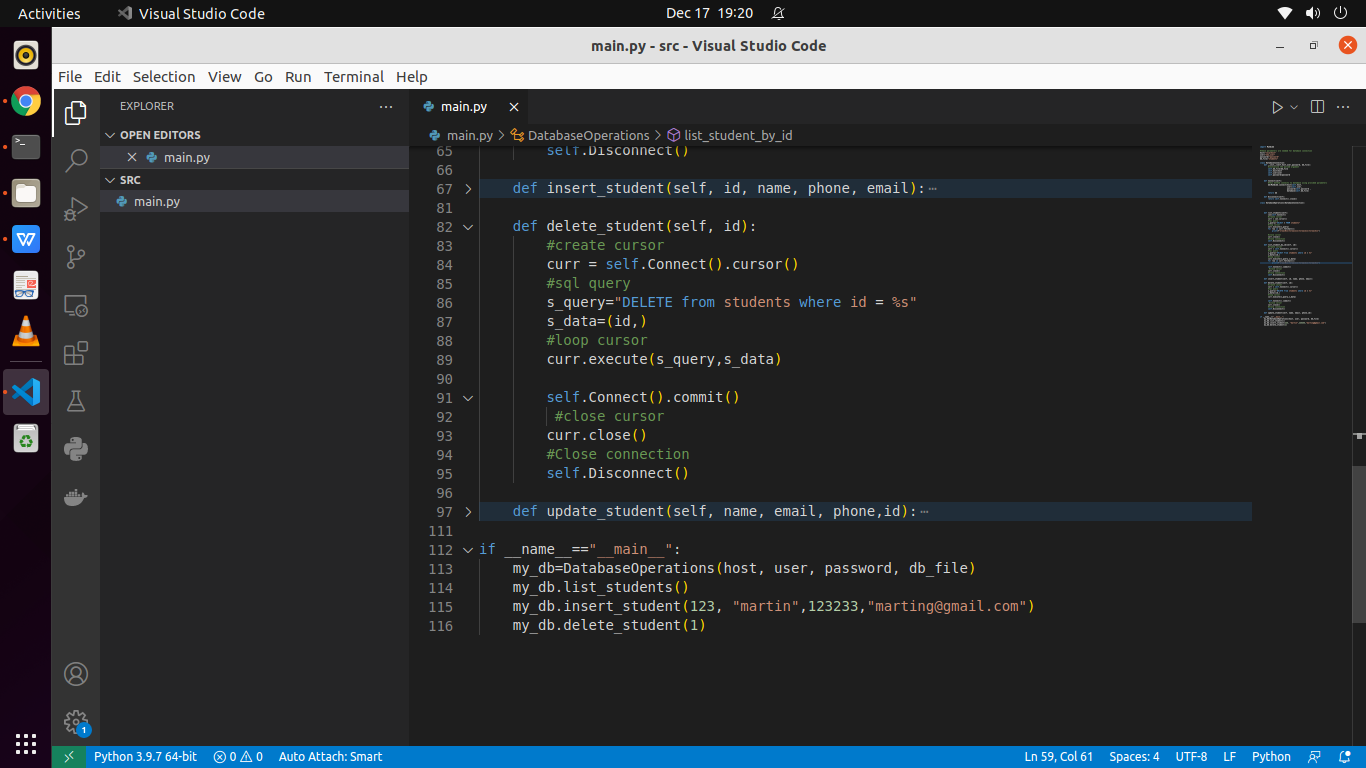
The function list\_students lists all records in the table students. It creates a cursor the connection created earlier and executes the query shown there in. The for loop prints all results extracted from the table. The last steps entail closing the cursor as well as the connection.

### Selecting specific record

The different between this function and the one above is the query part alone. The function first creates a cursor using the self.Connect().cursor(). The cursor is used to execute the provided query and parsed student id.



### Deleting record student table



### Inserting student records

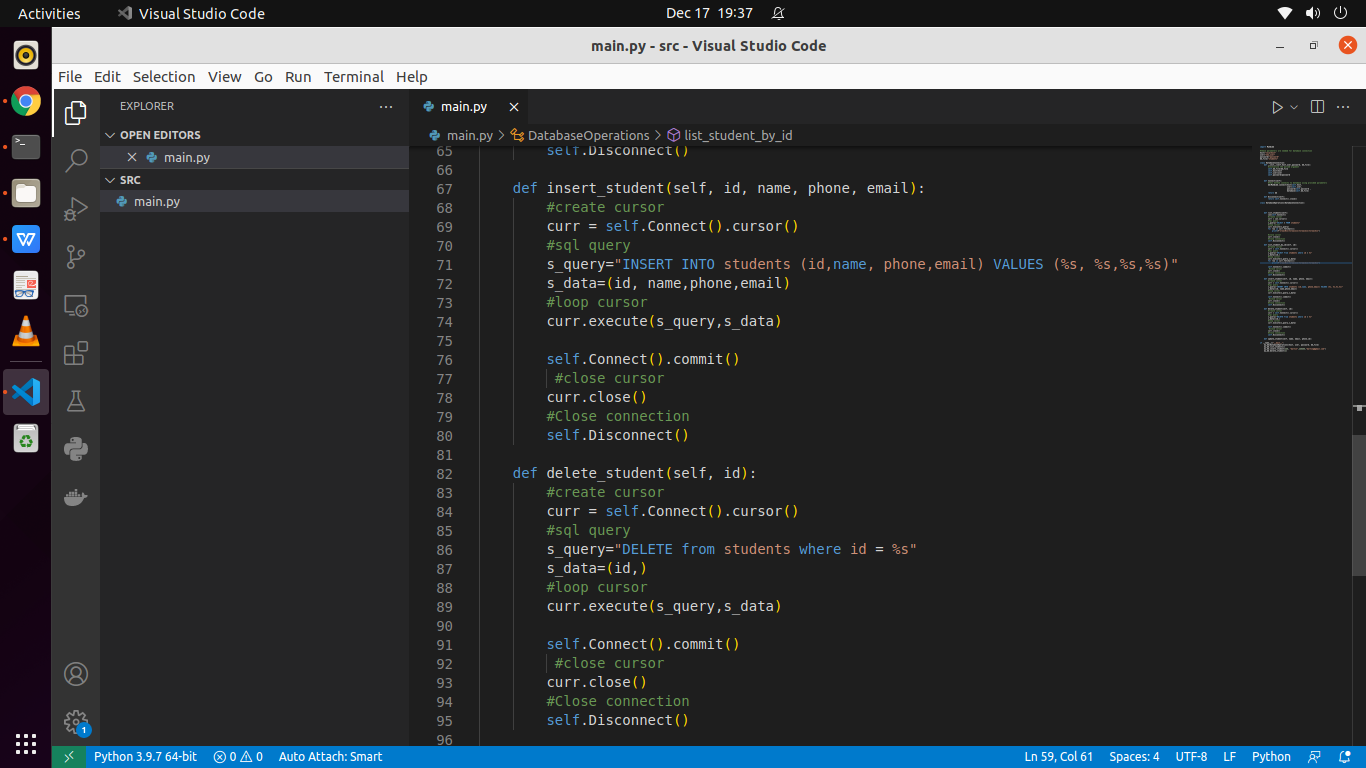


Figure 10 Insert records into students table

### Update students tables

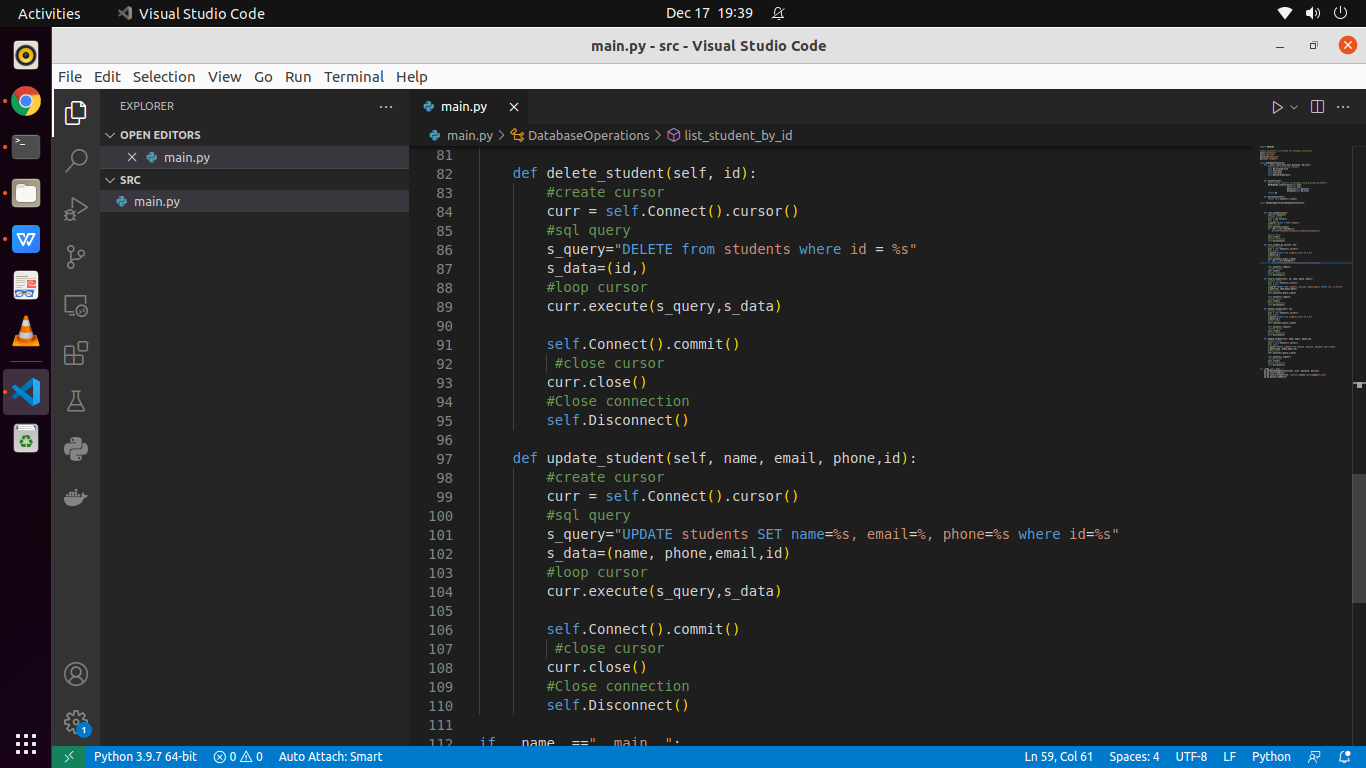


Figure 11 Record update function

# Appendix

## Project source code

|  |
| --- |
| import MySQLdb  #these parameters are needed for database connection  host="localhost"  user="new\_user"  password="password"  db\_file="students"  class DatabaseConnection:  def \_\_init\_\_(self,host,user,password, db\_file):  #initialise the parsed argument  self.db\_file=db\_file  self.host=host  self.user=user  self.password=password  def Connect(self):  #this method connects to database using provided parameters  db=MySQLdb.connect(host=self.host,  user=self.user,  password=self.password ,  database=self.db\_file)  return db  def Disconnect(self):  return self.Connect().close()  class DatabaseOperations(DatabaseConnection):  def list\_students(self):  con=self.Connect()  #create cursor  curr = con.cursor()  #SQL query  s\_query="SELECT \* FROM students"  #loop cursor  curr.execute(s\_query)  for row in curr.fetchall():  print(f"{row[0]}\t{row[1]}\t{row[2]}\t{row[3]}")  #close cursor  curr.close()  #Close connection  self.Disconnect()  def list\_student\_by\_id(self, id):  #create cursor  curr = self.Connect().cursor()  #SQL query  s\_query="SELECT from students where id = %s"  s\_data=(id,)  #loop cursor  curr.execute(s\_query,s\_data)  for row in curr.fetchall():  print(f"{row[0]}\t{row[1]}\t{row[2]}\t{row[3]}")  self.Connect().commit()  #close cursor  curr.close()  #Close connection  self.Disconnect()  def insert\_student(self, id, name, phone, email):  #create cursor  curr = self.Connect().cursor()  #SQL query  s\_query="INSERT INTO students (id,name, phone,email) VALUES (%s, %s,%s,%s)"  s\_data=(id, name,phone,email)  #loop cursor  curr.execute(s\_query,s\_data)  self.Connect().commit()  #close cursor  curr.close()  #Close connection  self.Disconnect()  def delete\_student(self, id):  #create cursor  curr = self.Connect().cursor()  #SQL query  s\_query="DELETE from students where id = %s"  s\_data=(id,)  #loop cursor  curr.execute(s\_query,s\_data)  self.Connect().commit()  #close cursor  curr.close()  #Close connection  self.Disconnect()  def update\_student(self, name, email, phone,id):  #create cursor  curr = self.Connect().cursor()  #SQL query  s\_query="UPDATE students SET name=%s, email=%, phone=%s where id=%s"  s\_data=(name, phone,email,id)  #loop cursor  curr.execute(s\_query,s\_data)  self.Connect().commit()  #close cursor  curr.close()  #Close connection  self.Disconnect()  if \_\_name\_\_=="\_\_main\_\_":  my\_db=DatabaseOperations(host, user, password, db\_file)  my\_db.list\_students()  my\_db.insert\_student(123, "martin",123233,"marting@gmail.com")  my\_db.delete\_student(1) |

SQL SCRIPTS

|  |
| --- |
| SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;  SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;  SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';  -- -----------------------------------------------------  -- Schema students  -- -----------------------------------------------------  -- -----------------------------------------------------  -- Schema students  -- -----------------------------------------------------  CREATE SCHEMA IF NOT EXISTS `students` ;  USE `students` ;  -- -----------------------------------------------------  -- Table `students`.`students`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `students`.`students` (  `id` INT NOT NULL,  `name` VARCHAR(45) NULL,  `phone` VARCHAR(45) NULL,  `email` VARCHAR(45) NULL,  PRIMARY KEY (`id`))  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `students`.`hostels`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `students`.`hostels` (  `id` INT NOT NULL,  `name` VARCHAR(45) NULL,  PRIMARY KEY (`id`))  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `students`.`fees`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `students`.`fees` (  `student` INT NOT NULL,  `fees\_balance` INT NULL,  PRIMARY KEY (`student`),  CONSTRAINT `fk\_fees\_1`  FOREIGN KEY (`student`)  REFERENCES `students`.`students` (`id`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  -- -----------------------------------------------------  -- Table `students`.`student\_hostel`  -- -----------------------------------------------------  CREATE TABLE IF NOT EXISTS `students`.`student\_hostel` (  `student` INT NOT NULL,  `hostel` INT NULL,  PRIMARY KEY (`student`),  INDEX `fk\_student\_hostel\_1\_idx` (`hostel` ASC) VISIBLE,  CONSTRAINT `fk\_student\_hostel\_1`  FOREIGN KEY (`hostel`)  REFERENCES `students`.`hostels` (`id`)  ON DELETE NO ACTION  ON UPDATE NO ACTION,  CONSTRAINT `fk\_student\_hostel\_2`  FOREIGN KEY (`student`)  REFERENCES `students`.`students` (`id`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  SET SQL\_MODE=@OLD\_SQL\_MODE;  SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;  SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS; |