**Phase 2 Python Project**

**Each table design with appropriate data type, data format, identification of primary keys, NULL/NOT NULL definition, explain in your report, how does each table is aligned to your project phase 1 description.**

The six scenarios are administration/login, student registration, course registration, view students/courses, fees and calculation, and view receipt fees. The student table is the initial table to start the table with because it is the most important table in the table design. The primary key for student table is id. The student table has a lot of fields and is required for registering students into courses with unique student identification numbers which is used throughout the whole system. Second table is course where students can see which semester they can register towards to. Course table is used for course registration details from project phase 1. Course\_id is primary key for table course. Administration table for administrators only and primary key is admin\_id. Administration table is aligned with project phase 1 because the administrator is responsible for physically managing the registration system for students. View table is where the primary key is view\_id. The view table is new because it views some details about the school system. Feeinstallment is where the fees of the students are dependant on each subject. Feeinstallment primary key is fee\_id. The fee table is important because it relates to how much does each student pay for their courses and is related to phase 1 with registering students. Registering students is not free and requires students to pay for their courses. Last table is receipt with receipt\_id as primary key. Receipt table is for printing receipts for students for their fees. Receipts are an integral part of viewing the fees from students. These tables are related to project phase 1 description because it is used to explain the school registration system of students and courses where it requires an administrator managing and viewing the related fees and registration process. The student table consists of: student(id INTEGER NOT NULL PRIMARY KEY, name VARCHAR(25), surname VARCHAR(25), email VARCHAR(25), subject VARCHAR(25), gender VARCHAR(25), fee INTEGER, python INTEGER, java INTEGER, c INTEGER. The course table has: course(course\_id INTEGER NOT NULL PRIMARY KEY, course\_name VARCHAR(25), semester VARCHAR(25), year VARCHAR(25) DEFAULT '2021', stu\_id INTEGER. The administration table has: administration(admin\_id INTEGER NOT NULL PRIMARY KEY, initials VARCHAR(25), priority VARCHAR(25) DEFAULT 'value', case\_number VARCHAR(25) DEFAULT 'value', st\_id INTEGER. The view table has: view(view\_id INTEGER NOT NULL PRIMARY KEY, identifier VARCHAR(25), view\_number VARCHAR(25), year VARCHAR(25) DEFAULT '2021', admin\_id INTEGER. The feeinstallment table has: FEEINSTALLMENT(fee\_id INTEGER NOT NULL PRIMARY KEY, TOTAL\_FEE INTEGER, REMAIN\_FEE INTEGER, PAID\_FEE INTEGER, INSTALLMENT INTEGER, ad\_id INTEGER. The receipt table has receipt(receipt\_id INTEGER NOT NULL PRIMARY KEY, name VARCHAR(25), message VARCHAR(25), year VARCHAR(25) DEFAULT '2021', fees\_id INTEGER.

**Design of database tables, when working in group of two members (5 tables, minimum; 5 fields, minimum per table, identification of primary key used, use of secondary key (based on design)**

We have group of 4 members so 6 tables are: feeinstallment, student, administration, course, view, and receipt. The 6 fields for feeinstallment are: fee\_id INTEGER NOT NULL PRIMARY KEY, TOTAL\_FEE INTEGER, REMAIN\_FEE INTEGER, PAID\_FEE INTEGER, INSTALLMENT INTEGER, ad\_id INTEGER foreign key from administration table. Student table has 10 fields are: id INTEGER NOT NULL PRIMARY KEY, name VARCHAR(25), surname VARCHAR(25), email VARCHAR(25), subject VARCHAR(25), gender VARCHAR(25), fee INTEGER, python INTEGER, java INTEGER, c INTEGER. Administration table fields are: admin\_id INTEGER NOT NULL PRIMARY KEY, initials VARCHAR(25), priority VARCHAR(25) DEFAULT 'value', case\_number VARCHAR(25) DEFAULT 'value', st\_id INTEGER (foreign key from student). Course table has 5 fields: course\_id INTEGER NOT NULL PRIMARY KEY, course\_name VARCHAR(25), semester VARCHAR(25), year VARCHAR(25) DEFAULT '2021', stu\_id INTEGER (foreign key from student). View has 5 fields: view\_id INTEGER NOT NULL PRIMARY KEY, identifier VARCHAR(25), view\_number VARCHAR(25), year VARCHAR(25) DEFAULT '2021', admin\_id INTEGER (foreign key from administration. Receipt table has 5 fields: receipt\_id INTEGER NOT NULL PRIMARY KEY, name VARCHAR(25), message VARCHAR(25), year VARCHAR(25) DEFAULT '2021', fees\_id INTEGER (foreign key from feeinstallment). When designing the tables, we had the six scenarios changes as well so we had to change the fields from phase 1. The database design is based on the scenarios. The administration login table is base on the administration/login scenario. The student table is based on student registration scenario. The course table is based on course registration scenario. The view table is based on students and courses scenario. The feeinstallment table is based on the fees calculation scenario. The view and receipt tables are based on the

**Take screenshot of each table description and crop and paste. Copy and paste the script to create tables 10 Marks**

Script to Create Tables:

**from tkinter import \***

**from tkinter import ttk**

**import mysql.connector**

**import tkinter.messagebox**

**from datetime import datetime**

**from tkinter import filedialog**

**import shutil**

**import os**

**from tkinter import Text,Tk**

**import tkinter as tk**

**mydb = mysql.connector.connect(host='127.0.0.1', user='root', password='1234', database='school')**

**my\_cursor = mydb.cursor()**

**my\_cursor.execute("use school")**

**my\_cursor.execute("CREATE TABLE IF NOT EXISTS student(id INTEGER NOT NULL PRIMARY KEY, name VARCHAR(25), surname VARCHAR(25), email VARCHAR(25), subject VARCHAR(25), gender VARCHAR(25), fee INTEGER, python INTEGER, java INTEGER, c INTEGER)")**

**my\_cursor.execute("CREATE TABLE IF NOT EXISTS course(course\_id INTEGER NOT NULL PRIMARY KEY, course\_name VARCHAR(25), semester VARCHAR(25), year VARCHAR(25) DEFAULT '2021', stu\_id INTEGER)")**

**my\_cursor.execute("ALTER TABLE course add foreign key(stu\_id) references student(id)")**

**my\_cursor.execute("CREATE TABLE IF NOT EXISTS administration(admin\_id INTEGER NOT NULL PRIMARY KEY, initials VARCHAR(25), priority VARCHAR(25) DEFAULT 'value', case\_number VARCHAR(25) DEFAULT 'value', st\_id INTEGER)")**

**my\_cursor.execute("ALTER TABLE administration add foreign key (st\_id) references student(id)")**

**my\_cursor.execute("CREATE TABLE IF NOT EXISTS view(view\_id INTEGER NOT NULL PRIMARY KEY, identifier VARCHAR(25), view\_number VARCHAR(25), year VARCHAR(25) DEFAULT '2021', admin\_id INTEGER)")**

**my\_cursor.execute("ALTER TABLE view add foreign key (admin\_id) references administration(admin\_id)")**

**my\_cursor.execute("CREATE TABLE IF NOT EXISTS FEEINSTALLMENT(fee\_id INTEGER NOT NULL PRIMARY KEY, TOTAL\_FEE INTEGER, REMAIN\_FEE INTEGER, PAID\_FEE INTEGER, INSTALLMENT INTEGER, ad\_id INTEGER)")**

**my\_cursor.execute("ALTER TABLE FEEINSTALLMENT add foreign key (ad\_id) references administration(admin\_id)")**

**my\_cursor.execute("CREATE TABLE IF NOT EXISTS receipt(receipt\_id INTEGER NOT NULL PRIMARY KEY, name VARCHAR(25), message VARCHAR(25), year VARCHAR(25) DEFAULT '2021', fees\_id INTEGER)")**

**my\_cursor.execute("ALTER TABLE receipt add foreign key (fees\_id) references FEEINSTALLMENT(fee\_id)")**

**mydb.commit()**

**mydb.close()**

Describing Tables:

Administration Table:

A picture containing text

Description automatically generated

Course Table:

Text

Description automatically generated with medium confidence

Feeinstallment Table:

A screenshot of a computer

Description automatically generated with medium confidence

Receipt Table:

Text

Description automatically generated with medium confidence

Student Table:

A picture containing calendar

Description automatically generated

View Table:

Text

Description automatically generated with low confidence

**E-R diagram of the complete system 10 Marks**

A picture containing diagram

Description automatically generated

**Based on Scenarios, enter data and test different scenarios (Test at least 5 scenarios from your project description)**

**Take screenshots, crop and paste each scenario based on your project description**

Student registration query:

my\_cursor.execute("INSERT INTO student(id, name, surname, email, subject, gender, fee, python, java, c) VALUES ('13245', 'bob', 'tob', 'bob@gmail.com', 'MECHANICS', '1', '3000', '0', '0', '0')")



Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated

Search Registration with Student ID 13245:

My\_cursor.execute(‘SELECT name FROM student WHERE id = %s’,((SEARCH,))

Text

Description automatically generated

Graphical user interface

Description automatically generated

Course Registration:

my\_cursor.execute("INSERT INTO course(course\_id, course\_name, semester, year, stu\_id) VALUES ('1231', 'MECHANICS', 'WINTER', '2021', '13245')")



Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated

Course View details:

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Administration input:

my\_cursor.execute("INSERT INTO administration(admin\_id, initials, priority, case\_number, st\_id) VALUES ('132456', 'SQ', 'HIGH', '1231', '13245')")



Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

View administration details:

View feeinstallment fees for student id 13245 query:

View administration details:

Text

Description automatically generated

Graphical user interface

Description automatically generated

View total fees for student id 13245:

my\_cursor.execute(‘SELECT fee FROM student WHERE id = %s’, (FE,))

Text

Description automatically generated

Graphical user interface

Description automatically generated

Input receipt information of student:

Input receipt data query:

my\_cursor.execute("INSERT INTO receipt(receipt\_id, name, message, year, fees\_id) VALUES ('13245', 'bob', 'Thanks', '2021', '123')")



Graphical user interface

Description automatically generated

View receipt details query:

Text

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated with medium confidence

Insert into feeinstallment: my\_cursor.execute("INSERT INTO FEEINSTALLMENT(fee\_id, TOTAL\_FEE, REMAIN\_FEE, PAID\_FEE, INSTALLMENT, ad\_id) VALUES ('123', '3000', '1212', '500', '500', '132456')") with fees id 123



Graphical user interface, text, application

Description automatically generated

**Summarize challenges that you had to overcome or change your initial design and reason for those changes. Also explain, contribution of each group member in the project**

Assing foreign keys is hard because of the syntax errors and invalid columns. The challenge was to unable to make the foreign keys because the column was not first created in table initially. Then we realized the mistake and created the column first in the “CREATE TABLE” statement, and then add the foreign key constraint. Victor was responsible for simplifying the scenarios because in phase 1 the scenarios it was too complex and did not make sense. Hamid was responsible for helping code the functions within the scenarios. Amin and Salman was responsible in helping the functions of code and integration of Python with and MySQL. Everyone contributed equally in terms of the code for all the scenarios. Some other challenges included: linking the primary keys to foreign keys in database system, inputting data to test cases, and trying to connect python to MySQL. Linking the primary keys to foreign keys was difficult because we had to think of the relationship between each of the tables and then assign the corresponding keys. Testing cases was hard too because sometimes the data type did not match our input. Trying to connect to MySQL was difficult because it involved installing lots of different packages and files. Our initial tables were too complex and hard in phase 1 and we had to simplify them by trying to focus on the main topic of school management system. So we had to delete the parking permit table, teacher table, scholarships table, and honour roll graduation table.