Departmental Database

Part B: Relational Database Implementation

[Name]

[Student Id]

**Document Name**: DatabaseTemplatePartB.docx

Document History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | Version | Revision | Description | Author | Role |
| 31/08/2022 | 1 | 1 | First version document |  | System Analyst |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Table of Contents**

[1. Revised Scope and Boundaries 3](#_Toc110947633)

[2. Revised Business Rules 3](#_Toc110947634)

[3. Final Entity Relationship Diagram 3](#_Toc110947635)

[4. SQL to Create Database and Tables 3](#_Toc110947636)

[5. SQL Queries 0](#_Toc110947637)

[6. Views 0](#_Toc110947638)

[6.1. Creating View 0](#_Toc110947639)

[6.2. Running view 0](#_Toc110947640)

[6.3. Dropping View 0](#_Toc110947641)

[7. Stored Procedures 0](#_Toc110947642)

[7.1. Retrieving Data Using a Stored Procedure 0](#_Toc110947643)

[7.2. Inserting Data Using a Stored Procedure 0](#_Toc110947644)

[7.3. Modifying Data Using a Stored Procedure 0](#_Toc110947645)

[7.4. Stored Procedure Using a Parameter 0](#_Toc110947646)

[7.5. Dropping a Stored Procedure 0](#_Toc110947647)

[8. Triggers 0](#_Toc110947648)

[8.1. Trigger 1 0](#_Toc110947649)

[8.2. Trigger 2 0](#_Toc110947650)

[9. Appendix 1](#_Toc110947651)

# Revised Scope and Boundaries

In Australia, lift time learning is encouraged by the government. Holmesglen is one of the largest providers of vocational and higher education in Victoria.

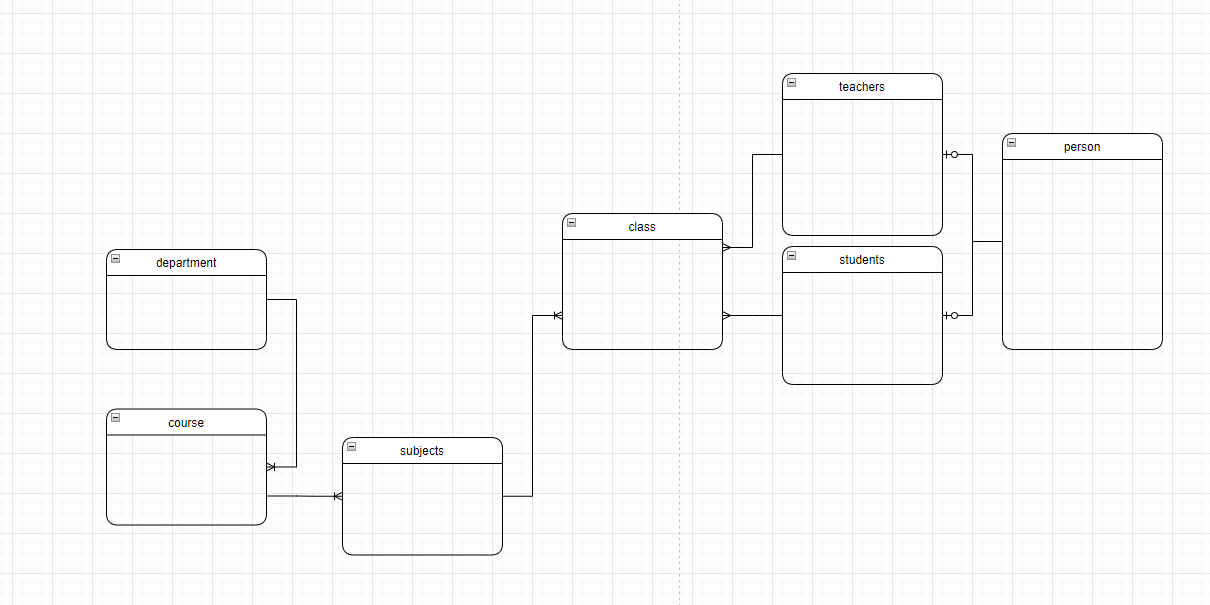
Holmesglen provides mostly formal education (e.g., Cert IV, Diploma), and some informal short courses for all ages. The courses are provided in many ways, but they also have things in common. There are subjects, lessons, teachers, and students. In this article, I will design a database for IT department to keep track of subject, course, students, and teacher.

The boundary of the project would be the holmesglen’s IT department. All courses and subjects is only limited to holmesglen’s IT department. a different origination would be outside the scope of the project.

# Revised Business Rules

* A subject can only belong to one course, but one course has many subjects.
* A student can enrol in many subjects and each subject can have many students.
* This department has many courses. One course can only belong to one department.
* One subject has many class, one class only teach one subject.
* One teacher can teach many classes, but one class only has one teacher.
* One student can enrol some classes, and one class can has many students.
* One person can be or can not be a teacher. The teacher must be a person.
* One person can be or can not be a student. The student must be a person.

# Final Entity Relationship Diagram



# SQL to Create Database and Tables

-- MySQL Workbench Forward Engineering

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 mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`DEPARTMENT`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`DEPARTMENT` (

`DEPARTMENT\_ID` INT NOT NULL AUTO\_INCREMENT ,

`DEPARTMENT\_NAME` VARCHAR(45) NOT NULL,

PRIMARY KEY (`DEPARTMENT\_ID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`COURSE`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`COURSE` (

`COURSE\_ID` INT NOT NULL AUTO\_INCREMENT ,

`COURSE\_NAME` VARCHAR(45) NOT NULL,

`DEPARTMENT\_ID` INT NOT NULL,

PRIMARY KEY (`COURSE\_ID`),

CONSTRAINT `fk\_COURSE\_1`

FOREIGN KEY (`DEPARTMENT\_ID`)

REFERENCES `mydb`.`DEPARTMENT` (`DEPARTMENT\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_COURSE\_1\_idx` ON `mydb`.`COURSE` (`DEPARTMENT\_ID` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`SUBJECTS`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`SUBJECTS` (

`SUBJECT\_ID` INT NOT NULL AUTO\_INCREMENT,

`NAME` VARCHAR(30) NOT NULL,

`START\_DATE` DATE NOT NULL,

`END\_DATE` DATE NOT NULL,

`COURSE\_ID` VARCHAR(45) NOT NULL,

PRIMARY KEY (`SUBJECT\_ID`),

CONSTRAINT `fk\_SUBJECTS\_1`

FOREIGN KEY ()

REFERENCES `mydb`.`COURSE` ()

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`TEACHERS`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`TEACHERS` (

`TEACHER\_ID` INT NOT NULL AUTO\_INCREMENT ,

`TEACHER\_EMAIL` VARCHAR(45) NOT NULL,

`PAY\_CODE` DECIMAL(8) NOT NULL,

PRIMARY KEY (`TEACHER\_ID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`CLASS`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`CLASS` (

`TEACHER\_ID` INT NOT NULL,

`SUBJECT\_ID` INT NOT NULL,

`CLASS\_NAME` VARCHAR(45) NOT NULL,

`START\_TIME` DATETIME NOT NULL,

`STUDENT\_ID` INT NOT NULL,

`SUBJECTS\_SUBJECT\_ID` INT NOT NULL,

`SUBJECTS\_COURSE\_COURSE\_ID` INT NOT NULL,

`TEACHERS\_TEACHER\_ID` INT NOT NULL,

PRIMARY KEY (`TEACHER\_ID`, `SUBJECTS\_SUBJECT\_ID`, `SUBJECTS\_COURSE\_COURSE\_ID`, `TEACHERS\_TEACHER\_ID`),

CONSTRAINT `fk\_CLASS\_SUBJECTS1`

FOREIGN KEY (`SUBJECTS\_SUBJECT\_ID`)

REFERENCES `mydb`.`SUBJECTS` (`SUBJECT\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_CLASS\_TEACHERS1`

FOREIGN KEY (`TEACHERS\_TEACHER\_ID`)

REFERENCES `mydb`.`TEACHERS` (`TEACHER\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_CLASS\_SUBJECTS1\_idx` ON `mydb`.`CLASS` (`SUBJECTS\_SUBJECT\_ID` ASC, `SUBJECTS\_COURSE\_COURSE\_ID` ASC) VISIBLE;

CREATE INDEX `fk\_CLASS\_TEACHERS1\_idx` ON `mydb`.`CLASS` (`TEACHERS\_TEACHER\_ID` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`STUDENTS`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`STUDENTS` (

`STUDENT\_ID` INT NOT NULL AUTO\_INCREMENT ,

`STUDENT\_EMAIL` VARCHAR(45) NOT NULL,

`STUDENT\_NUMBER` INT NOT NULL,

`CLASS\_TEACHER\_ID` INT NOT NULL,

`CLASS\_SUBJECTS\_SUBJECT\_ID` INT NOT NULL,

`CLASS\_SUBJECTS\_COURSE\_COURSE\_ID` INT NOT NULL,

`CLASS\_TEACHERS\_TEACHER\_ID` INT NOT NULL,

PRIMARY KEY (`STUDENT\_ID`, `CLASS\_TEACHER\_ID`, `CLASS\_SUBJECTS\_SUBJECT\_ID`, `CLASS\_SUBJECTS\_COURSE\_COURSE\_ID`, `CLASS\_TEACHERS\_TEACHER\_ID`),

CONSTRAINT `fk\_STUDENTS\_CLASS1`

FOREIGN KEY (`CLASS\_TEACHER\_ID` , `CLASS\_SUBJECTS\_SUBJECT\_ID` , `CLASS\_SUBJECTS\_COURSE\_COURSE\_ID` , `CLASS\_TEACHERS\_TEACHER\_ID`)

REFERENCES `mydb`.`CLASS` (`TEACHER\_ID` , `SUBJECTS\_SUBJECT\_ID` , `SUBJECTS\_COURSE\_COURSE\_ID` , `TEACHERS\_TEACHER\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_STUDENTS\_CLASS1\_idx` ON `mydb`.`STUDENTS` (`CLASS\_TEACHER\_ID` ASC, `CLASS\_SUBJECTS\_SUBJECT\_ID` ASC, `CLASS\_SUBJECTS\_COURSE\_COURSE\_ID` ASC, `CLASS\_TEACHERS\_TEACHER\_ID` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`PERSON`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`PERSON` (

`PERSON\_ID` VARCHAR(45) NOT NULL AUTO\_INCREMENT,

`LAST\_NAME` VARCHAR(15) NOT NULL,

`FIRST\_NAME` VARCHAR(15) NOT NULL,

`EMAIL` VARCHAR(45) NOT NULL,

PRIMARY KEY (`PERSON\_ID`),

CONSTRAINT `fk\_PERSON\_STUDENTS1`

FOREIGN KEY (`PERSON\_ID`)

REFERENCES `mydb`.`STUDENTS` (`STUDENT\_NUMBER`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_PERSON\_TEACHERS1`

FOREIGN KEY (`PERSON\_ID`)

REFERENCES `mydb`.`TEACHERS` (`PAY\_CODE`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- MySQL Workbench Forward Engineering

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 mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`DEPARTMENT`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`DEPARTMENT` (

`DEPARTMENT\_ID` INT NOT NULL AUTO\_INCREMENT ,

`DEPARTMENT\_NAME` VARCHAR(45) NOT NULL,

PRIMARY KEY (`DEPARTMENT\_ID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`COURSE`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`COURSE` (

`COURSE\_ID` INT NOT NULL AUTO\_INCREMENT ,

`COURSE\_NAME` VARCHAR(45) NOT NULL,

`DEPARTMENT\_ID` INT NOT NULL,

PRIMARY KEY (`COURSE\_ID`),

CONSTRAINT `fk\_COURSE\_1`

FOREIGN KEY (`DEPARTMENT\_ID`)

REFERENCES `mydb`.`DEPARTMENT` (`DEPARTMENT\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_COURSE\_1\_idx` ON `mydb`.`COURSE` (`DEPARTMENT\_ID` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`SUBJECTS`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`SUBJECTS` (

`SUBJECT\_ID` INT NOT NULL AUTO\_INCREMENT,

`NAME` VARCHAR(30) NOT NULL,

`START\_DATE` DATE NOT NULL,

`END\_DATE` DATE NOT NULL,

`COURSE\_ID` VARCHAR(45) NOT NULL,

PRIMARY KEY (`SUBJECT\_ID`),

CONSTRAINT `fk\_SUBJECTS\_1`

FOREIGN KEY ()

REFERENCES `mydb`.`COURSE` ()

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`TEACHERS`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`TEACHERS` (

`TEACHER\_ID` INT NOT NULL AUTO\_INCREMENT ,

`TEACHER\_EMAIL` VARCHAR(45) NOT NULL,

`PAY\_CODE` DECIMAL(8) NOT NULL,

PRIMARY KEY (`TEACHER\_ID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`CLASS`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`CLASS` (

`TEACHER\_ID` INT NOT NULL,

`SUBJECT\_ID` INT NOT NULL,

`CLASS\_NAME` VARCHAR(45) NOT NULL,

`START\_TIME` DATETIME NOT NULL,

`STUDENT\_ID` INT NOT NULL,

`SUBJECTS\_SUBJECT\_ID` INT NOT NULL,

`SUBJECTS\_COURSE\_COURSE\_ID` INT NOT NULL,

`TEACHERS\_TEACHER\_ID` INT NOT NULL,

PRIMARY KEY (`TEACHER\_ID`, `SUBJECTS\_SUBJECT\_ID`, `SUBJECTS\_COURSE\_COURSE\_ID`, `TEACHERS\_TEACHER\_ID`),

CONSTRAINT `fk\_CLASS\_SUBJECTS1`

FOREIGN KEY (`SUBJECTS\_SUBJECT\_ID`)

REFERENCES `mydb`.`SUBJECTS` (`SUBJECT\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_CLASS\_TEACHERS1`

FOREIGN KEY (`TEACHERS\_TEACHER\_ID`)

REFERENCES `mydb`.`TEACHERS` (`TEACHER\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_CLASS\_SUBJECTS1\_idx` ON `mydb`.`CLASS` (`SUBJECTS\_SUBJECT\_ID` ASC, `SUBJECTS\_COURSE\_COURSE\_ID` ASC) VISIBLE;

CREATE INDEX `fk\_CLASS\_TEACHERS1\_idx` ON `mydb`.`CLASS` (`TEACHERS\_TEACHER\_ID` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`STUDENTS`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`STUDENTS` (

`STUDENT\_ID` INT NOT NULL,

`STUDENT\_EMAIL` VARCHAR(45) NOT NULL,

`STUDENT\_NUMBER` INT NOT NULL,

`CLASS\_TEACHER\_ID` INT NOT NULL,

`CLASS\_SUBJECTS\_SUBJECT\_ID` INT NOT NULL,

`CLASS\_SUBJECTS\_COURSE\_COURSE\_ID` INT NOT NULL,

`CLASS\_TEACHERS\_TEACHER\_ID` INT NOT NULL,

PRIMARY KEY (`STUDENT\_ID`, `CLASS\_TEACHER\_ID`, `CLASS\_SUBJECTS\_SUBJECT\_ID`, `CLASS\_SUBJECTS\_COURSE\_COURSE\_ID`, `CLASS\_TEACHERS\_TEACHER\_ID`),

CONSTRAINT `fk\_STUDENTS\_CLASS1`

FOREIGN KEY (`CLASS\_TEACHER\_ID` , `CLASS\_SUBJECTS\_SUBJECT\_ID` , `CLASS\_SUBJECTS\_COURSE\_COURSE\_ID` , `CLASS\_TEACHERS\_TEACHER\_ID`)

REFERENCES `mydb`.`CLASS` (`TEACHER\_ID` , `SUBJECTS\_SUBJECT\_ID` , `SUBJECTS\_COURSE\_COURSE\_ID` , `TEACHERS\_TEACHER\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `fk\_STUDENTS\_CLASS1\_idx` ON `mydb`.`STUDENTS` (`CLASS\_TEACHER\_ID` ASC, `CLASS\_SUBJECTS\_SUBJECT\_ID` ASC, `CLASS\_SUBJECTS\_COURSE\_COURSE\_ID` ASC, `CLASS\_TEACHERS\_TEACHER\_ID` ASC) VISIBLE;

-- -----------------------------------------------------

-- Table `mydb`.`PERSON`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`PERSON` (

`PERSON\_ID` VARCHAR(45) NOT NULL AUTO\_INCREMENT ,

`LAST\_NAME` VARCHAR(15) NOT NULL,

`FIRST\_NAME` VARCHAR(15) NOT NULL,

`EMAIL` VARCHAR(45) NOT NULL,

PRIMARY KEY (`PERSON\_ID`),

CONSTRAINT `fk\_PERSON\_STUDENTS1`

FOREIGN KEY (`PERSON\_ID`)

REFERENCES `mydb`.`STUDENTS` (`STUDENT\_NUMBER`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_PERSON\_TEACHERS1`

FOREIGN KEY (`PERSON\_ID`)

REFERENCES `mydb`.`TEACHERS` (`PAY\_CODE`)

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;

**Below is my insert sql statement**

**TABLE\_DEPARTMENT:**

INSERT INTO mydb.DEPARTMENT (DEPARTMENT\_NAME, DEPARTMENT\_ID)

VALUES ('BUSINESS DEPARTMENT', 1);

INSERT INTO mydb.DEPARTMENT (DEPARTMENT\_NAME, DEPARTMENT\_ID)

VALUES ('COMPUTER SCIENCE DEPARTMENT', 2);

INSERT INTO mydb.DEPARTMENT (DEPARTMENT\_NAME, DEPARTMENT\_ID)

VALUES ('COMMERCE DEPARTMENT',3);

SELECT \* FROM mydb.DEPARTMENT;

**TABLE\_PERSON:**

SELECT \* FROM mydb.PERSON;

INSERT INTO mydb.PERSON(FIRST\_NAME, LAST\_NAME, EMAIL, MOBILE, AGE, DOB)

VALUES('ALICE', 'ROBERSTON', 'ROBERTSON.ALICE@HOTMAIL.COM' , 470894685, 34, ‘1988-09-02’);

INSERT INTO mydb.PERSON(FIRST\_NAME, LAST\_NAME, EMAIL, MOBILE, AGE, DOB)

VALUES('BOB', 'JOHNSON', 'JOHNSON.BOB@HOTMAIL.COM', 403598764, 26, ‘1996-03-15’);

INSERT INTO mydb.PERSON(FIRST\_NAME, LAST\_NAME, EMAIL, MOBILE, AGE, DOB)

VALUES('CHRIS', 'WONG', 'WONG.CHRIS@HOTMAIL.COM', 428976874, 18, ‘2004-08-12’ );

INSERT INTO mydb.PERSON(FIRST\_NAME, LAST\_NAME, EMAIL, MOBILE,AGE, DOB)

VALUES('DAVE', 'CHARMCHY', 'CHARMCHY.DAVE@HOTMAIL.COM', 412937952, 45, ‘1977-03-17’ );

INSERT INTO mydb.PERSON(FIRST\_NAME, LAST\_NAME, EMAIL, MOBILE, AGE, DOB)

VALUES('EVE', 'MORRISON', 'EVE.MORRISON@HOTMAIL.COM', 412937952,23, ‘1999-05-16’ );

INSERT INTO mydb.PERSON(FIRST\_NAME, LAST\_NAME, EMAIL, MOBILE, AGE, DOB)

VALUES('COCO', 'WILLAM', 'COCO.WILLAM@HOTMAIL.COM', 412937952, 34, ‘1988-06-07’);

**TABLE\_STUDENTS:**

SELECT \* FROM mydb.STUDENTS;

INSERT INTO mydb.STUDENTS(FIRST\_NAME,LAST\_NAME,STUDENT\_EMAIL, STUDENT\_NUMBER, PERSON\_ID)

VALUES ('Alice','ROBERTSON','b7838945@tafe.edu.au',7838945, 2);

INSERT INTO mydb.STUDENTS(FIRST\_NAME,LAST\_NAME,STUDNET\_EMAIL, STUDENT\_NUMBER, PERSON\_ID)

VALUES ('Bob', 'JOHNSON', 'd6574743@tafe.edu.au', 6574743, 3);

INSERT INTO mydb.STUDENTS(FIRST\_NAME,LAST\_NAME, STUDENT\_EMAIL, STUDENT\_NUMBER, PERSON\_ID)

VALUES ('Chris','WONG','e8945837@tafe.edu.au', 8945837, 1);

**TABLE\_COURSE**

INSERT INTO COURSES (COURSE\_ID, DEPARTMENT\_ID, COURSE\_NAME)

VALUES (2, ‘ICT40220’) ;

INSERT INTO COURSES (DEPARTMENT\_ID, COURSE\_NAME)

VALUES(2, ‘ICT30220’, 4);

INSERT INTO COURSES (DEPARTMENT\_ID, COURSE\_NAME

VALUES (2, ‘ICT50220’);

SELECT \* FROM mydb.COURSE

**TABLE\_SUBJECT**

INSERT INTO SUBJECTS (SUBJECT\_ID, SUBJECT\_NAME, COURSE\_ID, AVAGE\_SCORE)

VALUES ('DIGITAL IMAGES', 78)

INSERT INTO SUBJECTS (SUBJECT\_NAME, COURSE\_ID, AVERAGE\_SCORE)

VALUES ('WEB DEV', 80)

INSERT INTO SUBJECTS (SUBJECT\_NAME, COURSE\_ID, AVERAGE\_SCORE)

VALUES ('DATABASE', 82)

SELECT \* FROM mydb.SUBJECT

**TABLE\_CLASS**

SELECT \* FROM mydb.CLASS;

INSERT INTO mydb.CLASS(CLASS\_ID, TEACHER\_ID, SUBJECT\_ID, START\_TIME, STUDENT\_ID, NUMBER\_OF\_STUDENTS, CLASS\_NAME)

VALUES (1, 1, 2,'Wed 9am', 3, 2);

INSERT INTO mydb.CLASS(CLASS\_ID, TEACHER\_ID, SUBJECT\_ID, START\_TIME, STUDENT\_ID, NUMBER\_OF\_STUDENTS)

VALUES (2, 3, 2,'Tues 11am', 2, 3);

INSERT INTO mydb.CLASS(CLASS\_ID, TEACHER\_ID, SUBJECT\_ID, START\_TIME, STUDENT\_ID, NUMBER\_OF\_STUDENTS)

VALUES (3, 2, 1,'Fri 9am', 1, 3);

|  |  |  |
| --- | --- | --- |
| **Purpose** | **Tables Involved** | **Command(s) Involved and**  **Result** |
| *A:,List all students in alphabetical order* | *StudentS* | *SELECT \* FROM mydb\_UPDATE.STUDENTS;*  *SELECT STUDENT\_ID, FIRST\_NAME, LAST\_NAME*  *FROM STUDENTS*  *ORDER BY FIRST\_NAME ASC;* |
| B:Retrieve student age records | Students, person |  |
| C: the number of persons | person | SELECT COUNT(PERSON\_ID)  FROM mydb\_UPDATE.PERSON; |
| D:student\_id, who select which couse | Class, subject | This answer is same as question e, same screenshot and same code.  SELECT C.CLASS\_ID, S.AVERAGE\_SCORE  FROM mydb\_UPDATE.CLASS C, mydb\_UPDATE.SUBJECTS S  WHERE C.SUBJECT\_ID = S.SUBJECT\_ID  AND S.AVERAGE\_SCORE > 79; |
| E:the class whose subject average score is higher than 20 | Class and subject |  |
| F:square root (average\_score )\*10 |  | SELECT (SQRT(AVERAGE\_SCORE)\* 10) FROM mydb\_UPDATE.SUBJECTS; |
| G:DOB is greater than 1990-0101 |  | SELECT \* FROM mydb\_UPDATE.PERSON  WHERE DOB > '1990-01-01'  ORDER BY DOB DESC |

# SQL Queries

# Views

## Creating View, Running view, Dropping View

CREATE VIEW mydb.SCORE\_GT\_82 AS

SELECT C.CLASS\_ID, S.AVERAGE\_SCORE

FROM mydb.CLASS C, mydb.SUBJECTS S

WHERE C.SUBJECT\_ID = S.SUBJECT\_ID

AND S.AVERAGE\_SCORE >= 82;

SELECT \* FROM mydb.SCORE\_GT\_82;

DROP VIEW mydb.SCORE\_GT\_82;

# Stored Procedures

## Retrieving Data Using a Stored Procedure

DELIMITER //

CREATE PROCEDURE SP\_GETPERSON()

BEGIN

SELECT CONCAT(FIRST\_NAME,' ',LAST\_NAME) FROM mydb\_UPDATE.PERSON;

END //

DELIMITER ;

CALL SP\_GETPERSON()

## Inserting Data Using a Stored Procedure

DELIMITER //

CREATE PROCEDURE sp\_Addstudents(in courseid int, in coursename varchar(45), in departmentid int)

BEGIN

insert into COURSE(course\_id, course\_name, department\_id)

values (courseid, coursename, departmentid);

END //

DELIMITER ;

call sp\_Addstudents(5, 'test\_course',1 );

## Modifying Data Using a Stored Procedure

DELIMITER //

CREATE PROCEDURE sp\_update\_course(in courseid int, in coursename varchar(45))

BEGIN

UPDATE COURSE SET COURSE\_NAME = coursename

WHERE COURSE\_ID = courseid ;

END //

DELIMITER ;

call sp\_update\_course(5, 'advanced computer scicence');

## Stored Procedure Using a Parameter

DELIMITER //

CREATE PROCEDURE sp\_update\_student\_email(in studentid int, in studentemail varchar(45))

BEGIN

UPDATE STUDENTS SET STUDENT\_EMAIL= studentemail

WHERE STUDENT\_ID = studentid ;

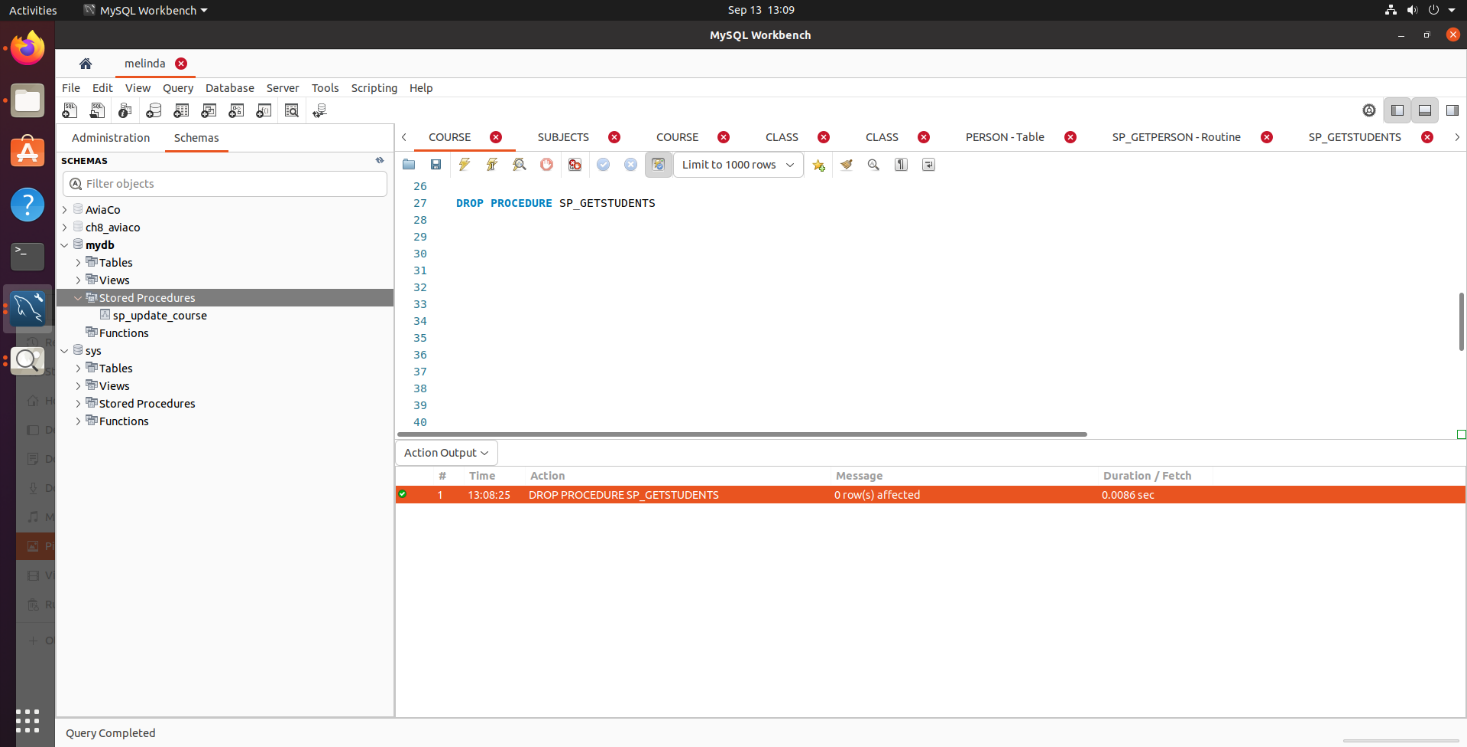
END //

DELIMITER ;

call sp\_update\_student\_email(5, 'tomhanks@hotmail.com');

## Dropping a Stored Procedure

DROP PROCEDURE SP\_GETSTUDENTS



# Triggers

## Trigger 1

## Trigger 2

CREATE TABLE PERSON\_LOG(

LOG\_ID INT NOT NULL AUTO\_INCREMENT,

COMMENT nvarchar(500) NOT NULL,

LOG\_DATE datetime NOT NULL ,

primary key(LOG\_ID)

);

delimiter |

CREATE TRIGGER TRIG\_PERSON\_LOG

AFTER INSERT ON PERSON

FOR EACH ROW

BEGIN

INSERT INTO PERSON\_LOG

(COMMENT,LOG\_DATE)

VALUES

(concat(NEW.FIRST\_NAME, '', NEW.LAST\_NAME, ' new person'), curdate());

END;

|

delimiter ;

select \* from PERSON\_LOG;

INSERT INTO mydb\_UPDATE.PERSON(PERSON\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, MOBILE, AGE, DOB)

VALUES(10, 'COCO-888', 'WILLAM', 'COCO888.WILLAM@HOTMAIL.COM', 412937852, 24, '1998-06-07');

select \* from PERSON;

select \* from PERSON\_LOG;

my explanation about my trigger:

1. specify the name of the trigger is TRIG\_PERSON\_LOG
2. specify the trigger action time is AFTER, which indicates that the trigger is invoked after each row is modified.
3. The operation of activating the trigger is INSERT.
4. The name of table to which the trigger belongs PERSON
5. When the trigger invokes, the statement will be executed: insert a record including ‘COMMENT’ and ‘LOG\_DATE’ into table PERSON\_LOG

# Appendix