Cover Page

TARC
TUNKU ABDUL RAHMAN
UNIVERSITY COLLEGE

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

BACS3183 ADVANCED DATABASE MANAGEMENT

Assignment

Bachelor of Computer Science (Honours) in Software Engineering RSF2 Group 1

Year 2 Semester 2 – 31th Dec 2019

Team members:

No.	Student Name	Registration No.	Signature
01	Tan Kuan Tiong	18 WMR 07386	
02	Tee Yu June	18 WMR 07982	
03	Cheng Chia Shun	18 WMR 00283	

Tutorial Group : RSF2 Group 1

Tutor : Mr. Choong Yun Loong

Table of Contents

Chapter 1: Case Study on the System

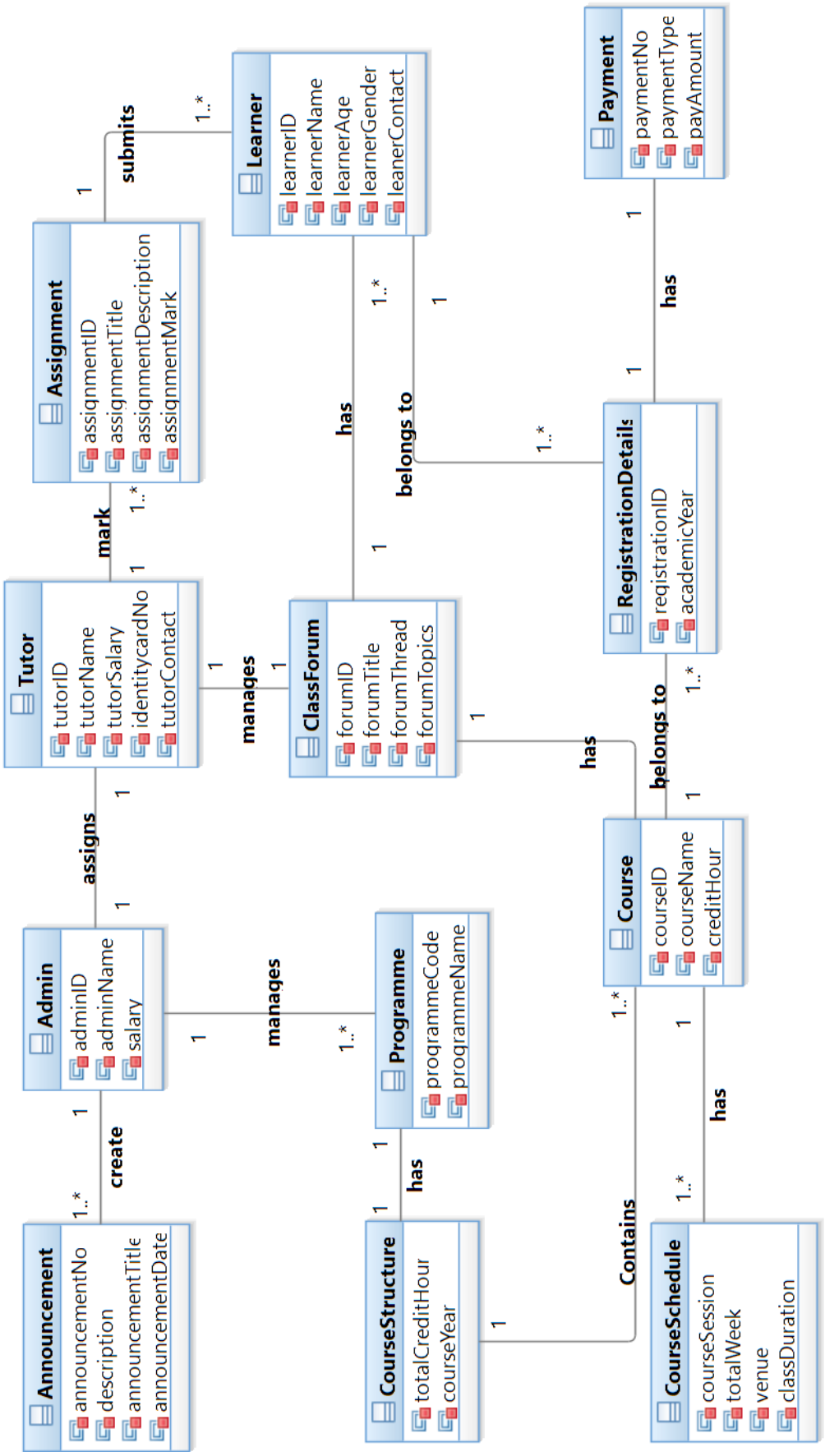
1.1. Background Study of the System

Wow University is a leading Open and Distance Learning (ODL) university in Malaysia and is one of the premier ODL institutions in Asia. The main campus is located in Malaysia and there are a total of 5 degree programme and 3 diploma programme provided. The mission of the university is to remove barriers that can limit access to Higher Education, adopting flexible mode of learning, and providing a conducive and engaging with latest technology-rich innovative learning environment at affordable cost.

The university offers a wide range of academic programmes to meet the needs of Malaysians and international learners. The university provides the highest quality digital learning experience to all the learners regardless of their location anywhere in the world. This is possible as the ODL mode of learning enables courses to be delivered flexibly via many modes such as on-line, self-study and face-to-face interaction with tutors during weekends or weekdays. The university accepts applicants from a wide variety of educational backgrounds but the admission requirements are varied by programmes. Hence, the applicants are required to review the specific Admission Requirements for the programme of their choice.

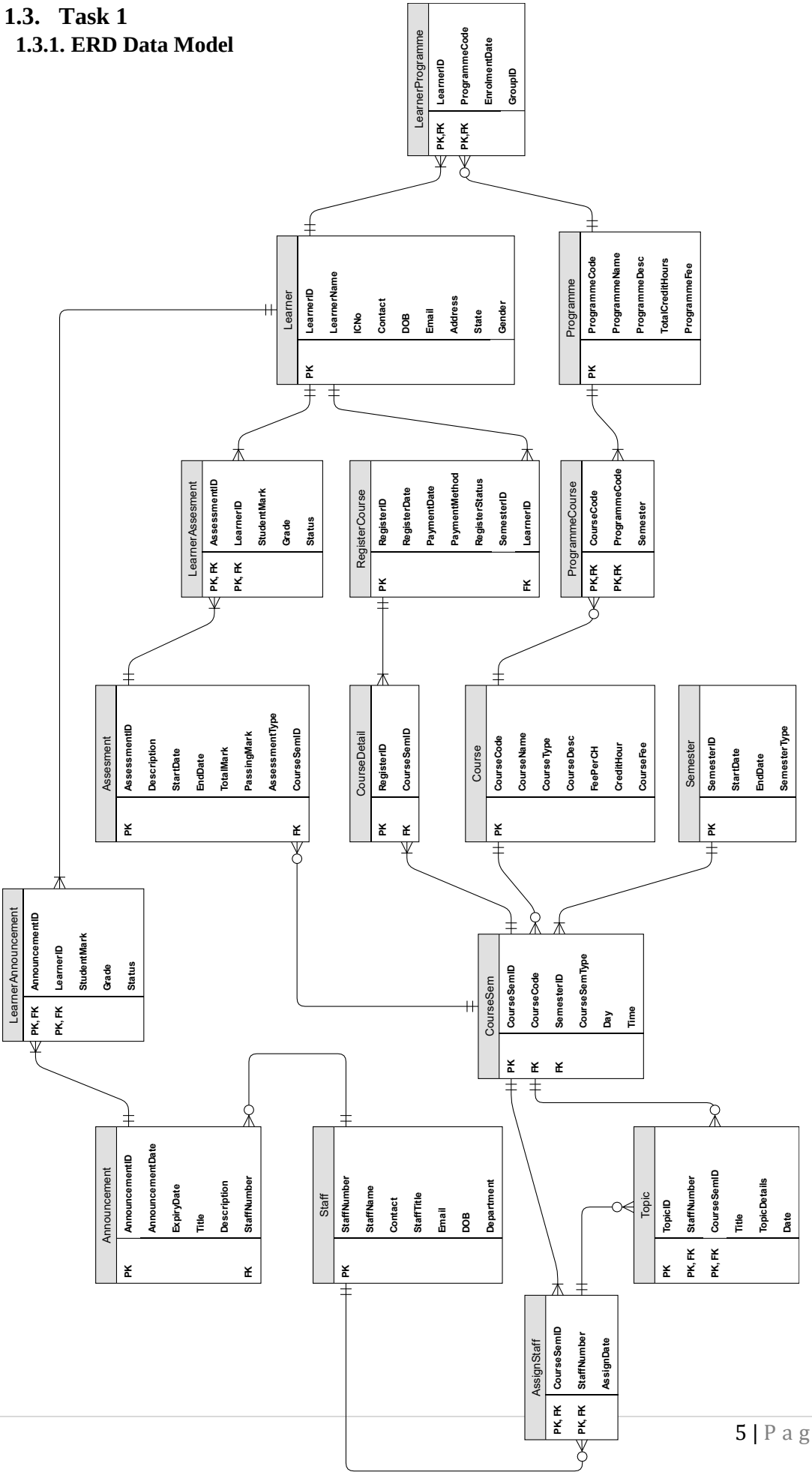
In the E-Learning mode provided by Wow Univeristy had included a total of 5 major modules, which are: Registration Module, responsible for new learner and course registration, Forum Module, allow learners to interact, communicate and learn from tutor or other learners, Announcement Module, allows tutor or admin staff to post announcements to selected group or individual via email, Online Assessment Module, allow the learners to submit their assignment is online and view their grade and marks, and Payment Module, responsible to learner's payment for course registration.

1.2. Old Analysis Class Diagram



1.3. Task 1

1.3.1. ERD Data Model



1.3.2. Assumptions and Enhancements

Assumptions:

1. This University College is currently providing 3 diploma programmes and 5 degree programme.
2. This University College currently only has 1 intake date which is 2nd March in every year which is available for students to enrol in a program.
3. For every programme, learner needs to study for 3 academic years.
4. One academic year includes 2 long semesters and 1 short semester.
5. The maximum number of courses that can be registered for a program in a long semester is 6 while 3 for a short semester.

1.3.3. Business Rules

Business Rules:

1. A learner cannot register more than 7 courses for a long semester and 4 for a short semester.
2. A leaner can only register once for each course in his entire programme study except that the learner had failed the course before.
3. Learners can start to register for their courses as long as the semester start. The registration period for each semester is 14 days after the semester start date. After the registration period, learner can no longer register to study in that semester and had to wait for next semester.
4. Learners are not necessarily needed to pay the course fee during course registration but choose to pay the fee later. However, the learner should pay the registration course fees within 60 days after the semester start date, if not, learner will be expelled from the semester and learner had to retake all courses for that semester again.
5. If learner does not pay the course registration fee within 21 days after the semester start date, an individual announcement will be sent to the learner to remind the learner to pay the course registration fee before the deadline. If learner does not pay the course fee after the deadline, an individual announcement will also be sent to learner to inform learner that the learner has been expelled from the semester.
6. A learner can involve in more than 1 program at different times but should not enrolled in more than 1 program at the same time.
7. 1 learner will receive at least 3 announcements in each semester they study.
8. Only staff can post announcements and assign them to learner who can view them but not learners. Learners can only view the announcement that were assigned to them.
9. Academic staff is not necessarily needed to post at least announcement during his/her teaching semester.
10. Academic staff may not necessarily need to be assigned for teaching during in a semester.
11. The weekly teaching classes of an academic staff should not be more than 9 classes a week and teaching hours should not be more than 54 hours a week.
12. Assigned staff can choose to post or not post a topic for a course that available in a semester.
13. Only assigned staff can delete their own course topic in a semester but not the other staff.
14. The course fee is determined by its credit hours and course type (core, elective, compulsory).

Chapter 2: Data Definition Language (DDL)

2.1. Task 2: Create Tables

Table 1: Learner

No. of Records: 300

```
CREATE TABLE Learner (
    LearnerID       VARCHAR(7)      NOT NULL,
    LearnerName     VARCHAR(20)     NOT NULL,
    ICNo            VARCHAR(14)     NOT NULL,
    Contact         VARCHAR(11)     NOT NULL,
    DOB             DATE,
    Email           VARCHAR(30)     NOT NULL,
    Address         VARCHAR(100),
    State           VARCHAR(20),
    Gender          CHAR(1)         NOT NULL,
    Primary key(LearnerID),
    CONSTRAINT  chk_ic  CHECK(REGEXP_LIKE(ICNo,'[0-9]{6}-[0-9]{2}-[0-
9]{4}')),
    CONSTRAINT   chk_contact   CHECK(REGEXP_LIKE(Contact,'[0-9]{3}-[0-
9]{7}')),
    CONSTRAINT chk_email CHECK(Email LIKE '%@%.%' AND Email NOT LIKE
'@%' AND Email NOT LIKE '%@%@%'),
    CONSTRAINT chk_gender CHECK(Gender IN('M','F'))
);
```

Table 2: Programme

No. of Records: 8

```
CREATE TABLE Programme (
    ProgrammeCode       CHAR(3)         NOT NULL,
    ProgrammeName       VARCHAR(50)     NOT NULL,
    ProgrammeDesc       VARCHAR(130),
    TotalCreditHours    NUMBER(3)       DEFAULT 0,
    Primary key(ProgrammeCode),
    CONSTRAINT chk_tch CHECK(TotalCreditHours >= 0)
);
```

Table 3: Semester

No. of Records: 9

```
CREATE TABLE Semester (
    SemesterID      NUMBER(6)     NOT NULL,
    StartDate       DATE          NOT NULL,
    EndDate         DATE          NOT NULL,
    SemesterType    VARCHAR(5)    NOT NULL,
    Primary key(SemesterID),
    CONSTRAINT chk_SemType CHECK(SemesterType IN('Long','Short'))
);
```

Table 4: Course

No. of Records: 103

```
CREATE TABLE Course (
    CourseCode    VARCHAR(8)      NOT NULL,
    CourseName    VARCHAR(80)     NOT NULL,
    CourseType    VARCHAR(20)     NOT NULL,
    CourseDesc    VARCHAR(130),
    FeePerCH      NUMBER(6,2)     NOT NULL,
    CreditHour    NUMBER(1)       NOT NULL,
    CourseFee     NUMBER(6,2),
    Primary key(CourseCode),
    CONSTRAINT            chk_courseType            CHECK(CourseType
IN('Compulsory','Elective','Core','Core Elective')),
    CONSTRAINT chk_feePerCH CHECK(FeePerCH >= 100.00),
    CONSTRAINT chk_creditHour CHECK(CreditHour BETWEEN 1 AND 3)
);
```

Table 5: Staff

No. of Records: 80

```
CREATE TABLE Staff (
    StaffNumber    NUMBER(5)      NOT NULL,
    StaffName      VARCHAR(40)    NOT NULL,
    Contact        VARCHAR(12),
    StaffTitle     VARCHAR(20)    NOT NULL,
    Email          VARCHAR(30),
    DOB            DATE           NOT NULL,
    Department     VARCHAR(40),
    Primary key(StaffNumber),
    CONSTRAINT  chk_staffContact  CHECK(REGEXP_LIKE(Contact,'[0-9]{3}-
[0-9]{7}|[0-9]{3}-[0-9]{8}')),
    CONSTRAINT chk_staffEmail CHECK(Email LIKE '%@%.%' AND Email NOT
LIKE '@%' AND Email NOT LIKE '%@%@%')
);
```

Table 6: Learner Programme

No. of Records: 300

```
CREATE TABLE LearnerProgramme (
    LearnerID        VARCHAR(7)    NOT NULL,
    ProgrammeCode    CHAR(3)       NOT NULL,
    EnrolmentDate    DATE          NOT NULL,
    GroupID          NUMBER(2),
    Primary key(LearnerID,ProgrammeCode),
    Foreign Key(LearnerID) references Learner(LearnerID),
    Foreign Key(ProgrammeCode) references Programme(ProgrammeCode)
);
```

Table 7: Programme Course

No. of Records: 360

```
CREATE TABLE ProgrammeCourse (
    ProgrammeCode    CHAR(3)       NOT NULL,
    CourseCode       VARCHAR(8)    NOT NULL,
    Semester         CHAR(4)       NOT NULL,
    Primary key(ProgrammeCode,CourseCode),
    Foreign Key(ProgrammeCode) references Programme(ProgrammeCode),
    Foreign Key(CourseCode) references Course(CourseCode)
);
```

Table 8: Course Semester

No. of Records: 904

```
CREATE TABLE CourseSem (
    CourseSemID      NUMBER(6)     NOT NULL,
    CourseCode       VARCHAR(8)    NOT NULL,
    SemesterID       NUMBER(6)     NOT NULL,
    CourseSemType    VARCHAR(9)    NOT NULL,
    Day              VARCHAR(3),
    Time             VARCHAR(9),
    Primary key(CourseSemID),
    Foreign Key(CourseCode) references Course(CourseCode),
    Foreign Key(SemesterID) references Semester(SemesterID),
    CONSTRAINT             chk_csType             CHECK(CourseSemType
IN('Lecture','Tutorial','Practical')),
    CONSTRAINT chk_csTime CHECK(REGEXP_LIKE(Time,'[0-9]{4}-[0-9]{4}'))
);
```

Table 9: Announcement

No. of Records: 33

```
CREATE TABLE Announcement (
    AnnouncementID      NUMBER(4)      NOT NULL,
    AnnouncementDate    DATE           DEFAULT SYSDATE,
    ExpiryDate          DATE,
    Title               VARCHAR(50)    NOT NULL,
    Description         VARCHAR(200),
    StaffNumber         NUMBER(5)      NOT NULL,
    Primary key(AnnouncementID),
    Foreign Key(StaffNumber) references Staff(StaffNumber)
);
```

Table 10: Course Registration

No. of Records: 2700

```
CREATE TABLE RegisterCourse (
    RegisterID        NUMBER(5)     NOT NULL,
    RegisterDate      DATE          NOT NULL,
    PaymentDate       DATE,
    PaymentMethod     VARCHAR(20),
    RegisterStatus    VARCHAR(10)   DEFAULT 'Pending',
    SemesterID        NUMBER(6)     NOT NULL,
    LearnerID         VARCHAR(7)    NOT NULL,
    Primary key(RegisterID),
```

```
    Foreign Key(LearnerID) references Learner(LearnerID),
    Foreign Key(SemesterID) references Semester(SemesterID),
    CONSTRAINT    chk_payMethod    CHECK(PaymentMethod    IN('Online
Banking','Cash','JomPay')),
    CONSTRAINT       chk_regStatus       CHECK       (RegisterStatus
IN('Pending','Success','Reject'))
);
```

Table 11: Assessment

No. of Records: 952

```
CREATE TABLE Assessment (
    AssessmentID      NUMBER(5)       NOT NULL,
    Description       VARCHAR(100),
    StartDate         DATE            DEFAULT SYSDATE,
    EndDate           DATE,
    TotalMark         NUMBER(5,2),
    PassingMark       NUMBER(5,2),
    AssessmentType    VARCHAR(20)     NOT NULL,
    CourseSemID       NUMBER(6)       NOT NULL,
    Primary key(AssessmentID),
    Foreign Key(CourseSemID) references CourseSem(CourseSemID),
    CONSTRAINT chk_totalMark CHECK(TotalMark >= 0),
    CONSTRAINT chk_assDate   CHECK(EndDate >= StartDate),
    CONSTRAINT chk_passMark  CHECK(PassingMark BETWEEN 30 AND 50),
    CONSTRAINT    chk_assType              CHECK    (AssessmentType
IN('Assignment','Practical Test','Midterm'))
);
```

Table 12: Course Details

No. of Records: 35700

```
CREATE TABLE CourseDetail (
    RegisterID     NUMBER(5)     NOT NULL,
    CourseSemID    NUMBER(6)     NOT NULL,
    Day            VARCHAR(3),
    Time           VARCHAR(9),
    Semester       NUMBER(6)     NOT NULL,
    Primary key(RegisterID,CourseSemID,Semester),
    Foreign Key(RegisterID) references RegisterCourse(RegisterID),
    Foreign Key(CourseSemID) references CourseSem(CourseSemID),
    Foreign Key(Semester) references Semester(SemesterID),
    CONSTRAINT chk_cdTime CHECK(REGEXP_LIKE(Time,'[0-9]{4}-[0-9]{4}'))
);
```

Table 13: Assign Staff

No. of Records: 904

```
CREATE TABLE AssignStaff (
    CourseSemID    NUMBER(6)    NOT NULL,
    StaffNumber    NUMBER(5)    NOT NULL,
    AssignDate     DATE         DEFAULT SYSDATE,
    Primary key(CourseSemID,StaffNumber),
    Foreign key(CourseSemID) references CourseSem(CourseSemID),
    Foreign key(StaffNumber) references Staff(StaffNumber)
);
```

Table 14: Learner Announcement

No. of Records: 5400

```
CREATE TABLE LearnerAnnouncement (
    AnnouncementID    NUMBER(4)     NOT NULL,
    LearnerID         VARCHAR(7)    NOT NULL,
    GroupID           NUMBER(2),
    Primary key(AnnouncementID,LearnerID),
    Foreign               Key(AnnouncementID)              references
Announcement(AnnouncementID),
    Foreign Key(LearnerID) references Learner(LearnerID)
);
```

Table 15: Topic

No. of Records: 50

```
CREATE TABLE Topic (
    TopicID         NUMBER(3)      NOT NULL,
    StaffNumber     NUMBER(5)      NOT NULL,
    CourseSemID     NUMBER(6)      NOT NULL,
    Title           VARCHAR(50)    NOT NULL,
    TopicDetails    VARCHAR(200),
    TopicDate       DATE           DEFAULT  SYSDATE,
    Primary key(TopicID,StaffNumber,CourseSemID),
    Foreign key(StaffNumber) references Staff(StaffNumber),
    Foreign key(CourseSemID) references CourseSem(CourseSemID)
);
```

Table 16: Learner Assessment

No. of Records: 9600

```
CREATE TABLE LearnerAssessment (
    AssessmentID    NUMBER(5)      NOT NULL,
    LearnerID       VARCHAR(7)     NOT NULL,
    StudentMark     NUMBER(4,1),
    Grade           VARCHAR(2),
    Status          VARCHAR(4),
    Primary key(AssessmentID,LearnerID),
    Foreign key(AssessmentID) references Assessment(AssessmentID),
    Foreign key(LearnerID) references Learner(LearnerID),
    CONSTRAINT chk_studMark CHECK(StudentMark BETWEEN 0 AND 100),
    CONSTRAINT    chk_grade    CHECK(Grade    IN('A','A-','B+','B','B-
','C+','C','F')),
    CONSTRAINT chk_status CHECK(Status IN('Pass','Fail'))
);
```

Chapter 3: Queries, Procedures, Triggers and Reports (DQL and DML)

3.1. Tan Kuan Tiong

3.1.1. Query 1: Show Learner's Course Registration That Is Pending

Purpose: This is an operational query where its purpose is to show the admin staff the learner's course registration that is still pending for confirmation. So that, admin staff can validate their registration and approve their registration.

SQL Statement:

```
SELECT RC.PaymentMethod, RC.RegisterDate, RC.PaymentDate, RC.RegisterID, RC.LearnerID,
       LP.ProgrammeCode, LP.GroupID, PC.CourseCode
FROM RegisterCourse RC,
     (SELECT LearnerID, ProgrammeCode, GroupID
      FROM LearnerProgramme) LP,
     (SELECT ProgrammeCode, CourseCode
      FROM ProgrammeCourse
      WHERE Semester = 'Y3S2') PC
WHERE RC.LearnerID = LP.LearnerID AND LP.ProgrammeCode = PC.ProgrammeCode AND RC.SemesterID = 201909 AND
      RC.RegisterStatus = 'Pending' AND RC.PaymentDate IS NOT NULL AND RC.PaymentMethod = 'Online Banking'
GROUP BY RC.PaymentMethod, RC.RegisterDate, RC.PaymentDate, RC.RegisterID, RC.LearnerID,
         sLP.ProgrammeCode, LP.GroupID, PC.CourseCode
ORDER BY 1, 2, 3, 4, 5;
```

Sample Output:

Pending Registration For Students In Semester 201909
With Online Banking

Page : 1

Payment Method	Register Date	Payment Date	Register ID	Learner ID	Prog Code	Group ID	Course Code
Online Banking	14-OCT-19	09-NOV-19	2452	1702002	DSE	1	AACS1074
							AACS1084
							AACS1143
							AAMS1613
							AAMS2613
							BHEL1023
			2632	1901032	REI	2	BAIT1023
							BAIT1083
							BAIT2113
							BAIT2133
							BAIT2173
							BHEL2023
			2638	1901038	REI	1	BAIT1023
							BAIT1083
							BAIT2113
							BAIT2133
							BAIT2173
							BHEL2023

Total Number Of Student: 3

3.1.2. Query 2: Average Workload of Academic Staff by Department

Purpose: This is a tactical query where its purpose is to let the middle management level know which department's academic staff have the heaviest workload based on one academic year and increases between 2 academic years. So that the middle management level will be able to balance out the weekly teaching hour of the academic staff or recruit more academic staff to resolve staff shortage, prevent overworking and provide higher teaching quality service to learner.

SQL Statement:

```
SELECT S.Department, cal_staff_by_dprt(S.Department) noStaff, Y1.TotalClass Y1TC, Y1.TotalHr Y1TH, Y1.AvgHr Y1AH,
       Y2.TotalClass Y2TC, Y2.TotalHr Y2TH, Y2.AvgHr Y2AH,
       to_char((Y2.AvgHr-Y1.AvgHr)/Y1.AvgHr*100,'990.99')||' %' IncreaseRatio
FROM Staff S,
     (SELECT S.Department,
             COUNT(AsS.CourseSemID) TotalClass,
             SUM(cal_weekly_class_hr(CS.CourseSemType)) TotalHr,
             SUM(cal_weekly_class_hr(CS.CourseSemType))/cal_staff_by_dprt(S.Department) AvgHr
      FROM  Staff S, AssignStaff AsS, CourseSem CS
      WHERE S.StaffNumber = AsS.StaffNumber AND
            AsS.CourseSemID = CS.CourseSemID AND CS.SemesterID LIKE &v_year1||'%'
      GROUP BY S.Department) Y1,
     (SELECT S.Department,
             COUNT(AsS.CourseSemID) TotalClass,
             SUM(cal_weekly_class_hr(CS.CourseSemType)) TotalHr,
             SUM(cal_weekly_class_hr(CS.CourseSemType))/cal_staff_by_dprt(S.Department) AvgHr
      FROM  Staff S, AssignStaff AsS, CourseSem CS
      WHERE S.StaffNumber = AsS.StaffNumber AND
            AsS.CourseSemID = CS.CourseSemID AND CS.SemesterID LIKE &v_year2||'%'
      GROUP BY S.Department) Y2
WHERE S.Department = Y1.Department AND S.Department = Y2.Department
GROUP BY S.Department, Y1.TotalClass, Y1.TotalHr, Y1.AvgHr, Y2.TotalClass, Y2.TotalHr, Y2.AvgHr
ORDER BY 9 DESC;
```

```
COLUMN IncHr HEADING "Increase In|Teaching Hour" FORMAT 999.99
TTITLE CENTER 'Greater Increase In Teaching Hours' SKIP 1 CENTER '================================='
SELECT S.Department, cal_staff_by_dprt(S.Department) noStaff, Y1.AvgHr Y1AH, Y2.AvgHr Y2AH,
       Y2.AvgHr-Y1.AvgHr IncHr, to_char((Y2.AvgHr-Y1.AvgHr)/Y1.AvgHr*100,'990.99')||' %' IncreaseRatio
FROM Staff S,
     (SELECT S.Department,
             SUM(cal_weekly_class_hr(CS.CourseSemType))/cal_staff_by_dprt(S.Department) AvgHr
      FROM  Staff S, AssignStaff AsS, CourseSem CS
      WHERE S.StaffNumber = AsS.StaffNumber AND
            AsS.CourseSemID = CS.CourseSemID AND CS.SemesterID LIKE &v_year1||'%'
      GROUP BY S.Department) Y1,
     (SELECT S.Department,
             SUM(cal_weekly_class_hr(CS.CourseSemType))/cal_staff_by_dprt(S.Department) AvgHr
      FROM  Staff S, AssignStaff AsS, CourseSem CS
      WHERE S.StaffNumber = AsS.StaffNumber AND
            AsS.CourseSemID = CS.CourseSemID AND CS.SemesterID LIKE &v_year2||'%'
      GROUP BY S.Department) Y2
WHERE S.Department = Y1.Department AND S.Department = Y2.Department
GROUP BY S.Department, Y1.AvgHr, Y2.AvgHr
ORDER BY 5 DESC
FETCH FIRST ROWS ONLY;

COLUMN DecHr HEADING "Decrease In|Teaching Hour" FORMAT 990.99
TTITLE CENTER 'Greater Decrease In Teaching Hours' SKIP 1 CENTER '================================='

SELECT S.Department, cal_staff_by_dprt(S.Department) noStaff, Y1.AvgHr Y1AH, Y2.AvgHr Y2AH,
       Y2.AvgHr-Y1.AvgHr DecHr, to_char((Y2.AvgHr-Y1.AvgHr)/Y1.AvgHr*100,'990.99')||' %' IncreaseRatio
FROM Staff S,
     (SELECT S.Department,
             SUM(cal_weekly_class_hr(CS.CourseSemType))/cal_staff_by_dprt(S.Department) AvgHr
      FROM  Staff S, AssignStaff AsS, CourseSem CS
```

```
       WHERE S.StaffNumber = AsS.StaffNumber AND
             AsS.CourseSemID = CS.CourseSemID AND CS.SemesterID LIKE &v_year1||'%'
       GROUP BY S.Department) Y1,
      (SELECT S.Department,
              SUM(cal_weekly_class_hr(CS.CourseSemType))/cal_staff_by_dprt(S.Department) AvgHr
       FROM  Staff S, AssignStaff AsS, CourseSem CS
       WHERE S.StaffNumber = AsS.StaffNumber AND
             AsS.CourseSemID = CS.CourseSemID AND CS.SemesterID LIKE &v_year2||'%'
       GROUP BY S.Department) Y2
WHERE S.Department = Y1.Department AND S.Department = Y2.Department
GROUP BY S.Department, Y1.AvgHr, Y2.AvgHr
ORDER BY 5
FETCH FIRST ROWS ONLY;
```

Sample Output:

```
This Query Will Show The Average Staff Workload Details In 2 Academic Years.
Please Enter 2 Academic Years To Proceed.

Enter First Year > 2018
Enter Second Year > 2019
```

Average Academic Staff WorkLoad For Each Department In 2 Academic Years

Department Details	Year 2018				Year 2019			
DEPARTMENT	Total Staff	Total Class	Total Hours	Average Hours	Total Class	Total Hours	Average Hours	Increase Ratio (%)
Department of Information Technology	20	80	480	24.00	169	1,014	50.70	111.25 %
Department of Mathematical Science	16	64	386	24.13	48	288	18.00	-25.39 %
Department of Information Systems	14	56	336	24.00	42	252	18.00	-25.00 %
Department of Computer Science	22	85	508	23.09	66	396	18.00	-22.05 %

Greater Increase In Teaching Hours

DEPARTMENT	Total Staff	Average Hours	Average Hours	Increase In Teaching Hour	Increase Ratio (%)
Department of Information Technology	20	24.00	50.70	26.70	111.25 %

Greater Decrease In Teaching Hours

DEPARTMENT	Total Staff	Average Hours	Average Hours	Decrease In Teaching Hour	Increase Ratio (%)
Department of Mathematical Science	16	24.13	18.00	-6.13	-25.39 %

3.1.3. Query 3: Learner's Commitment on Elective Course

Purpose: This is an strategical query where its purpose is to show the top management level know which elective course is more popular or less favoured among learners. Hence, they can plan to reduce resource put on or abolish the less favoured elective course and promote the more popular elective course to attract more learner to register for the courses.

SQL Statement:

```
 -- Elective Course with Less Learner Commitment Over 2 Academic Year

TTITLE SKIP 1 CENTER 'Elective Course with Less Learner Commitment
Over      2      Academic      Year'      SKIP      1      CENTER
'================================================================='
SKIP 1 ' '
BREAK ON REPORT
COMPUTE AVG LABEL 'Average Of Fees Received(RM):' OF PossibleIncome
ON REPORT

SELECT C.CourseCode, C.CourseName, COUNT(CSD.CourseSemID) as RegNum,
       COUNT(CSD.COURSESEMID)   *   C.FeePerCH   *   C.CreditHour   as
PossibleIncome
FROM Course C LEFT OUTER JOIN
     (SELECT CD.CourseSemID, CS.CourseCode
      FROM CourseDetail CD, CourseSem CS
      WHERE CD.CourseSemID = CS.CourseSemID AND
            CS.SemesterID IN (201803, 201805, 201809, 201903,
                              201905, 201909)
      ) CSD ON C.CourseCode = CSD.CourseCode
WHERE C.CourseType = 'Elective'
GROUP BY C.CourseCode, C.CourseName, C.FeePerCH, C.CreditHour
HAVING COUNT(CSD.CourseSemID) < 100
ORDER BY 3, 1;

-- Elective Course with Great Learner Commitment Over 2 Academic Year

TTITLE SKIP 1 CENTER 'Elective Course with Great Learner Commitment
Over      2      Academic      Year'      SKIP      1      CENTER
'================================================================='
SKIP 1 ' '

SELECT C.CourseCode, C.CourseName, COUNT(CSD.CourseSemID) as RegNum,
       COUNT(CSD.COURSESEMID)   *   C.FeePerCH   *   C.CreditHour   as
PossibleIncome
FROM Course C LEFT OUTER JOIN
     (SELECT CD.CourseSemID, CS.CourseCode
      FROM CourseDetail CD, CourseSem CS
      WHERE CD.CourseSemID = CS.CourseSemID AND
            CS.SemesterID IN (201803, 201805, 201809, 201903,
                              201905, 201909)
     ) CSD ON C.CourseCode = CSD.CourseCode
WHERE C.CourseType = 'Elective'
GROUP BY C.CourseCode, C.CourseName, C.FeePerCH, C.CreditHour
HAVING COUNT(CSD.CourseSemID) > 150
ORDER BY 3 DESC, 1;
```

Sample Output:

```
Elective Course with Less Learner Commitment Over 2 Academic Year
=================================================================
Course                                          No. of    Course Fee
Code     Course Name                          Students Received (RM)
-------- ------------------------------------ -------- -------------
AACS2333 Software Requirements Engineering II        0          0.00
AACS2343 Systems Analysis and Design                 0          0.00
AACS2353 Web Application Programming                 0          0.00
AACS2363 Web Design and Development                  0          0.00
AACS2303 Software Engineering I                     90     54,000.00
AACS2313 Software Engineering II                    90     54,000.00
AACS2323 Software Requirements Engineering I        90     54,000.00
-------------
Average                                                    23,142.86

Elective Course with Great Learner Commitment Over 2 Academic Year
==================================================================

Course                                          No. of    Course Fee
Code     Course Name                          Students Received (RM)
-------- ------------------------------------ -------- -------------
BAIT2183 Software Security                          270    162,000.00
BACS2033 Software Requirements Engineering          180    108,000.00
BACS2103 Software Quality Assurance and Testing     180    108,000.00
BAIT1023 Web Design and Development                 180    108,000.00
BAIT1043 Systems Analysis and Design                180    108,000.00
BAIT1083 Visual Programming                         180    108,000.00
BAIT2113 Web Application Development                180    108,000.00
BAIT2133 Web Engineering                            180    108,000.00
BAIT2173 Web Programming                            180    108,000.00
BAIT3113 Systems Administration                     180    108,000.00
BAIT3153 Software Project Management                180    108,000.00
-------------
Average                                                    112,909.09
```

3.1.4. Procedure 1: Warn or Reject on Learner's Course Registration

Purpose: This procedure will help the admin staff to send an individual announcement to warn the learner to remind them to pay their registration fee within the due day if the learner does not pay the registration fee within 21 days after the semester started or reject the learner's course registration if learner does not resolve the payment within 60 days after the semester started.

SQL Statement:

```
CREATE OR REPLACE PROCEDURE prc_warn_payment(v_chkDate IN Date DEFAULT SYSDATE,
                                             v_SemID IN NUMBER DEFAULT 201909) AS
   v_rjctDate    Date;
   v_warnDate    Date;
   v_countRjct   NUMBER(3);
   v_countWarn   NUMBER(3);
   v_announSeq   NUMBER(4);
   v_errMsg      Varchar(100);

   pay_format_invalid EXCEPTION;
   PRAGMA exception_init(pay_format_invalid, -20101);

   CURSOR reg_cursor IS
          SELECT LP.ProgrammeCode, RegisterID, RC.LearnerID, LearnerName, SemesterID, PaymentDate, PaymentMethod
          FROM RegisterCourse RC, Learner L, LearnerProgramme LP
          WHERE RC.LearnerID = L.LearnerID AND L.LearnerID = LP.LearnerID AND RC.SemesterID = v_SemID AND
                (PaymentDate IS NULL OR PaymentMethod IS NULL)
         ORDER BY 1, 2, 3;

   reg_rec  reg_cursor%ROWTYPE;

 BEGIN
    v_rjctDate := get_pay_rjct_date(v_SemID);
    v_warnDate := get_pay_warn_date(v_SemID);
    v_countRjct := 0;
    v_countWarn := 0;
    v_announSeq := 0;
```

```
-- linesize 78
DBMS_OUTPUT.PUT_LINE('Warning Or Rejection Of Student For Course Registration Payment'||chr(10));

DBMS_OUTPUT.PUT_LINE(' No.'||'  '||'Programme'||'  '||'RegisterID'||'  '||'LearnerID'||
                     '  '||RPAD('LearnerName',30,' ')||'  '||'Status');
DBMS_OUTPUT.PUT_LINE(' ---'||'  '||'---------'||'  '||'----------'||'  '||'---------'||
                     '  '||RPAD('-',30,'-')          ||'  '||'-------');

FOR reg_rec IN reg_cursor
LOOP
    IF (reg_rec.PaymentDate IS NULL AND reg_rec.PaymentMethod IS NOT NULL) OR
       (reg_rec.PaymentDate IS NOT NULL AND reg_rec.PaymentMethod IS NULL) THEN
        v_errMsg := 'RegisterID '||reg_rec.RegisterID||' do not have a valid payment record. '||
                    'Please check to prevent human error.';
        raise_application_error(-20101,v_errMsg);

    ELSIF v_chkDate >= v_rjctDate THEN
        DBMS_OUTPUT.PUT_LINE(to_char(reg_cursor%ROWCOUNT,'000')||'     '||
                             RPAD(reg_rec.ProgrammeCode,9,' ') ||'  '||
                             RPAD(reg_rec.RegisterID,8,' ')    ||'  '||
                             RPAD(reg_rec.LearnerID,8,' ')     ||'  '||
                             RPAD(reg_rec.LearnerName,30,' ')  ||'  '||'Reject');

        UPDATE RegisterCourse SET RegisterStatus = 'Reject' WHERE RegisterID = reg_rec.RegisterID;

        v_announSeq := Announcement_Seq.nextval;
        INSERT INTO Announcement
               VALUES (v_announSeq,v_chkDate,v_chkDate + 31,'Rejection Of Learner Registration',
                       'Learner '||reg_rec.LearnerName||'('||reg_rec.LearnerID||
                       ') Was Rejected To Enroll The Courses Due To Overdue Payment',90001);

        INSERT INTO LearnerAnnouncement VALUES (v_announSeq,reg_rec.LearnerID,NULL);
        v_countRjct := v_countRjct + 1;
```

```
        ELSIF v_chkDate >= v_warnDate THEN
            DBMS_OUTPUT.PUT_LINE(to_char(reg_cursor%ROWCOUNT,'000')||'     '||
                                 RPAD(reg_rec.ProgrammeCode,9,' ') ||'  '||
                                 RPAD(reg_rec.RegisterID,8,' ')    ||'  '||
                                 RPAD(reg_rec.LearnerID,8,' ')     ||'  '||
                                 RPAD(reg_rec.LearnerName,30,' ')  ||'  '||'Warning');

            v_announSeq := Announcement_Seq.nextval;
            INSERT INTO Announcement
                   VALUES (v_announSeq,v_chkDate,v_rjctDate,'Warning Of Registration Payment',
                           'Learner '||reg_rec.LearnerName||'('||reg_rec.LearnerID||
                           ') Was Advised To Pay The Registration Fee Before '||v_rjctDate||'.',90001);

            INSERT INTO LearnerAnnouncement VALUES (v_announSeq,reg_rec.LearnerID,NULL);
            v_countWarn := v_countWarn + 1;

        END IF;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE(chr(10)||'No. of Rejection    : '||v_countRjct);
    DBMS_OUTPUT.PUT_LINE('No. of Warning Sent : '||v_countWarn);
 END;
/
```

Sample Output:

```
Warning Or Rejection Of Student For Course Registration Payment

No.  Programme  RegisterID  LearnerID  LearnerName                    Status
---  ---------  ----------  ---------  -----------------------------  -------
001     DST        2701      1902041   Liow Rou Juan                  Reject
002     DST        2702      1902042   Beh Ai Shan                    Reject
003     DST        2703      1902043   Teoh Luo Ning                  Reject
004     DST        2704      1902044   Tan Pei Xie                    Reject
005     DST        2705      1902045   Soh Ke Ke                      Reject
006     DST        2706      1902046   Chuah Pei Eng                  Reject
007     DST        2707      1902047   Teoh Han Fong                  Reject
008     DST        2708      1902048   Teoh Long You                  Reject
009     DST        2709      1902049   Tee Qian Ling                  Reject
010     DST        2710      1902050   Teoh Luo Xin                   Reject

No. of Rejection    : 10
No. of Warning Sent : 0
```

Error Scenario 1: Show the register ID if it does not has a valid payment details

```
Warning Or Rejection Of Student For Course Registration Payment

No.  Programme  RegisterID  LearnerID  LearnerName                    Status
---  ---------  ----------  ---------  -----------------------------  -------
BEGIN prc_warn_payment; END;

*
ERROR at line 1:
ORA-20101: RegisterID 2711 do not have a valid payment record. Please check to prevent human error.
ORA-06512: at "TKT1.PRC_WARN_PAYMENT", line 41
ORA-06512: at "TKT1.PRC_WARN_PAYMENT", line 41
ORA-06512: at line 1
```

3.1.5. Procedure 2: Update Learner's Course Registration Status

Purpose: This procedure will help the adminn staff to update the student course registration status to success if the registration has valid payment details and pay before due date which is within 60 days after the semester started.

SQL Statement:

```
CREATE OR REPLACE PROCEDURE prc_validate_rc_status (v_semID IN NUMBER DEFAULT 201909) AS

  v_startDate  Date;
  v_endDate    Date;
  v_chk_semID  NUMBER(6);
  v_payMethod  VARCHAR(14);
  v_payDate    Date;
  v_count      NUMBER(3) := 0;

  CURSOR rc_cursor IS
     SELECT RC.PaymentMethod pm, RC.PaymentDate pd, RC.RegisterID rid, RC.LearnerID lid, L.LearnerName name
     FROM RegisterCourse RC, Learner L
     WHERE RC.LearnerID = L.LearnerID AND RC.SemesterID = v_semID AND RC.RegisterStatus = 'Pending' AND
           RC.PaymentDate >= v_startDate AND RC.PaymentDate < v_endDate AND RC.PaymentMethod IS NOT NULL AND
           RC.PaymentDate IS NOT NULL
     ORDER BY 1, 2, 3;
  rc_rec  rc_cursor%ROWTYPE;


 BEGIN
    SELECT SemesterID, StartDate, get_pay_rjct_date(v_semID) INTO v_chk_semID, v_startDate, v_endDate
    FROM Semester
    WHERE SemesterID = v_semID;

    DBMS_OUTPUT.PUT_LINE(chr(10)||
                         LPAD('Update Of Status On Valid Learner''s Course Registration',72,' ')||chr(10));
    DBMS_OUTPUT.PUT_LINE('_No.'||'  '||'Payment Method'||'  '||'Payment Date'||'  '||
                         'RegisterID'||'  '||'LearnerID'||'  '||'Learner Name');
```

```
DBMS_OUTPUT.PUT_LINE('====''||'  '||'==============''||'  '||'============''||'  '||
                     '==========''||'  '||'=========''||'  '||'==============================');

FOR rc_rec IN rc_cursor
LOOP
   UPDATE RegisterCourse
   SET RegisterStatus = 'Success'
   WHERE RegisterID = rc_rec.rid;

   IF rc_rec.pm = v_payMethod THEN
      IF rc_rec.pd = v_payDate THEN
         DBMS_OUTPUT.PUT_LINE(chr(13)||TO_CHAR(rc_cursor%ROWCOUNT,'000')||'  '||
                              LPAD(rc_rec.rid,37,' ')||'  ' ||
                              LPAD(rc_rec.lid,11,' ')||'   '||
                              RPAD(rc_rec.name,30,' '));

      ELSE
         DBMS_OUTPUT.PUT_LINE(chr(13)||TO_CHAR(rc_cursor%ROWCOUNT,'000')||'  '||
                              LPAD(to_char(rc_rec.pd,'dd-Mon-yyyy'),27,' ')||'  '||
                              LPAD(rc_rec.rid,8,' ')||'  '  ||
                              LPAD(rc_rec.lid,11,' ')||'   '||
                              RPAD(rc_rec.name,30,' '));
         v_payDate := rc_rec.pd;
      END IF;
   ELSE
         DBMS_OUTPUT.PUT_LINE(chr(10)||TO_CHAR(rc_cursor%ROWCOUNT,'000')||'  '||
                              RPAD(rc_rec.pm,14,' ')||'  '||
                              RPAD(to_char(rc_rec.pd,'dd-Mon-yyyy'),15,' ')||'  '||
                              RPAD(rc_rec.rid,8,' ')||'  '||
                              RPAD(rc_rec.lid,8,' ')||'  '||
                              RPAD(rc_rec.name,30,' '));
         v_payMethod := rc_rec.pm;
         v_payDate := rc_rec.pd;
   END IF;
```

```
       v_count := rc_cursor%ROWCOUNT;
    END LOOP;

    IF v_count = 0 THEN
       DBMS_OUTPUT.PUT_LINE(chr(10)||LPAD('No Register Status Is Updated.',60,' '));
    ELSE
       DBMS_OUTPUT.PUT_LINE(chr(10)||LPAD('Total   of',32,'   ')||to_char(v_count,'00')||'   Register   Statuses   Are
Updated.');
    END IF;

    EXCEPTION
       WHEN NO_DATA_FOUND THEN
         RAISE_APPLICATION_ERROR(-20111,'Semester not Found. Please Enter A Correct Semester.');
 END;
/
```

Sample Output:

Update Of Status On Valid Learner's Course Registration

No.	Payment Method	Payment Date	RegisterID	LearnerID	Learner Name
001	Cash	09-Nov-2019	2451	1702001	Soh Ai Ying
002			2459	1702009	Tee Jun Tiong
003			2481	1702031	Khor Jing Xie
004			2509	1801009	Beh Long Ho
005			2647	1901047	Lim Ren Nuo
006			2668	1902018	Liow Chia Qiang
007			2677	1902027	Ang Kuan Ke
008			2697	1902047	Teoh Han Fong
009	JomPay	09-Nov-2019	2453	1702003	Ang Chia Yi
010			2484	1702034	Tee Zhu Ping
011			2646	1901046	Teoh Jia Ling
012	Online Banking	09-Nov-2019	2452	1702002	Cheng Kuan Bing
013			2632	1901032	Chuah Jun Quan
014			2638	1901038	Khor Kim Wei

Total of 14 Register Statuses Are Updated.

Error Scenario 1: Show invalid semester error message when the semester input is not found in the record.

```
SQL> EXEC prc_validate_rc_status (201908);
BEGIN prc_validate_rc_status (201908); END;

*
ERROR at line 1:
ORA-20111: Semester not Found. Please Enter A Correct Semester.
ORA-06512: at "TKT1.PRC_VALIDATE_RC_STATUS", line 70
ORA-06512: at line 1
```

3.1.6. Trigger 1: Validate Staff-Course Assignation

Purpose: This trigger will validate the staff-course assignation that the assignation will be halted if the staff has reached weekly teaching class limit, 9 classes at most, or weekly teaching hours limit, 54 hours at most, in a semester before or after the assignation.

SQL Statement:

```
CREATE OR REPLACE TRIGGER trg_chk_ass_staff
  BEFORE INSERT ON AssignStaff
  FOR EACH ROW

  DECLARE
     v_noClass  NUMBER(2) := 0;
     v_noHour   NUMBER(3) := 0;
     v_staffNo  NUMBER(5) := 0;
     v_csType   VARCHAR(9) := 'Lecture';

     err_Msg    VARCHAR(200);

     No_Staff_Found EXCEPTION;
     PRAGMA exception_init(No_Staff_Found, -20102);
     Teach_OverLimit EXCEPTION;
     PRAGMA exception_init(Teach_OverLimit, -20103);

  BEGIN
     SELECT StaffNumber INTO v_staffNo FROM Staff WHERE StaffNumber = :NEW.StaffNumber;

     SELECT COUNT(ACS.CourseSemID), SUM(cal_weekly_class_hr(ACS.CourseSemType))
     INTO v_noClass, v_noHour
     FROM Staff S LEFT OUTER JOIN
          (SELECT AsS.StaffNumber, CS.CourseSemID, CS.CourseSemType
           FROM AssignStaff AsS LEFT OUTER JOIN CourseSem CS ON AsS.CourseSemID = CS.CourseSemID
           WHERE CS.SemesterID = 201909
           ) ACS ON S.StaffNumber = ACS.StaffNumber
     WHERE S.StaffNumber = v_staffNo;
```

```
    IF (v_noClass + 1 > 9) OR (v_noHour + 4 > 54) THEN
        err_Msg := chr(10)||'Weekly Teaching Classes Or Hours For Staff '||v_staffNo||
                   ' Is Over The Limit.'||chr(10);
        err_Msg := err_Msg||'Hence, This Staff Will Not Be Assigned To Teach This Course Semester.'||chr(10);
        RAISE_APPLICATION_ERROR(-20103, err_Msg);
    ELSE
        SELECT CourseSemType INTO v_csType
        FROM CourseSem
        WHERE CourseSemID = :NEW.CourseSemID;

        DBMS_OUTPUT.PUT_LINE(' No. of Class: '||(v_noClass + 1));
        DBMS_OUTPUT.PUT_LINE(' No. of Hours: '||(v_noHour + cal_weekly_class_hr(v_csType)));
    END IF;
    EXCEPTION
       WHEN NO_DATA_FOUND THEN
          RAISE_APPLICATION_ERROR(-20102,'StaffNumber Is Not Valid. The Staff Must Be A Current Academic Staff.');
 END;
/
```

Sample Output:

Success Scenario

```
SQL> INSERT INTO AssignStaff VALUES (100901,90050,'07-Oct-19');
Weekly Workload For Staff 90050
No. of Class: 1
No. of Hours: 8

1 row created.
```

Error Scenario 1: When the staff weekly teaching classes or hours have over the limit.

```
SQL> INSERT INTO AssignStaff VALUES (100901,90078,'07-Oct-19');

INSERT INTO AssignStaff VALUES (100901,90078,'07-Oct-19')
            *
ERROR at line 1:
ORA-20103:
Weekly Teaching Classes Or Hours For Staff 90078 Is Over The Limit.
Hence, This Staff Will Not Be Assigned To Teach This Course Semester.
ORA-06512: at "TKT1.TRG_CHK_ASS_STAFF", line 28
ORA-04088: error during execution of trigger 'TKT1.TRG_CHK_ASS_STAFF'
```

Error Scenario 2: When the staff number is not a current working academic staff.

```
SQL> INSERT INTO AssignStaff VALUES (100901,90100,'07-Oct-19');
INSERT INTO AssignStaff VALUES (100901,90100,'07-Oct-19')
            *
ERROR at line 1:
ORA-20102: StaffNumber Is Not Valid. The Staff Must Be A Current Academic Staff.
ORA-06512: at "TKT1.TRG_CHK_ASS_STAFF", line 39
ORA-04088: error during execution of trigger 'TKT1.TRG_CHK_ASS_STAFF'
```

3.1.7. Trigger 2: Validate Learner's Course Registration

Purpose: This trigger will validate the learner's course registration by preventing new registration if the registration date is past more than 14 days after the semester start date and preventing the update of learner's course registration status to 'Success' if it does not has valid payment details or the payment is overdue, which is 60 days after the semester started.

SQL Statement:

```
CREATE OR REPLACE TRIGGER trg_chk_reg_cs
  BEFORE INSERT OR
         UPDATE OR
         DELETE ON RegisterCourse
  FOR EACH ROW

  DECLARE
     v_studID  VARCHAR(7) := '1900050';
     v_regID   NUMBER(5) := 2712;

  BEGIN
     CASE
        WHEN INSERTING THEN
           IF :NEW.RegisterDate >= get_last_reg_date(:NEW.SemesterID) THEN
              RAISE_APPLICATION_ERROR(-20104, chr(10)||'Registration Period For This Semester '||
                                      'Has Ended.'||chr(10)||'Hence, The Registration Is Not Accepted.');
           END IF;

        WHEN UPDATING THEN
           IF :OLD.RegisterStatus = 'Pending' AND :NEW.RegisterStatus = 'Success' THEN
              IF :NEW.PaymentDate IS NULL OR :NEW.PaymentMethod IS NULL THEN
                 RAISE_APPLICATION_ERROR(-20105, chr(10)||'The Payment Detail Is Not Valid.'||chr(10)||
                                         'Please Double Check The Payment Detail To Prevent Human Error.');

              ELSIF :NEW.PaymentDate >= get_pay_rjct_date(:NEW.SemesterID) THEN
                 RAISE_APPLICATION_ERROR(-20106, chr(10)||'This Is An Overdue Payment Which Should Be '||
                                         'Rejected.'||chr(10)||'Hence, Update Of Register Status '||
```

```
                                        'To Success Will Not Be Executed.');
            END IF;
         END IF;

      WHEN DELETING THEN
         IF MONTHS_BETWEEN(SYSDATE,:OLD.RegisterDate) < 7 THEN
            RAISE_APPLICATION_ERROR(-20107, chr(10)||'The Register And Payment Record Of Learner '||
                                    'Should Be Kept'||chr(10)||' For 7 Years Before Removing.');
         END IF;
   END CASE;

   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         RAISE_APPLICATION_ERROR(-20108, chr(10)||'Please Select The Correct Semester Before Register.');
END;
/
```

Sample Output:

Error Scenario 1: When the semester id input is incorrect or not found in the record.

```
SQL> INSERT INTO RegisterCourse VALUES (2711,'14-Oct-19','21-Oct-19','Cash','Pending',NULL,'1902050');
INSERT INTO RegisterCourse VALUES (2711,'14-Oct-19','21-Oct-19','Cash','Pending',NULL,'1902050')
            *
ERROR at line 1:
ORA-20108:
Please Select The Correct Semester Before Register.
ORA-06512: at "TKT1.TRG_CHK_REG_CS", line 30
ORA-04088: error during execution of trigger 'TKT1.TRG_CHK_REG_CS'
```

Error Scenario 2: Reject learner's registration when the registration date has past more than 14 days after the semester started.

```
SQL> INSERT INTO RegisterCourse VALUES (2711,'31-Oct-19','21-Oct-19','Cash','Pending',201909,'1902050');
INSERT INTO RegisterCourse VALUES (2711,'31-Oct-19','21-Oct-19','Cash','Pending',201909,'1902050')
            *
ERROR at line 1:
ORA-20104:
Registration Period For This Semester Has Ended.
Hence, The Registration Is Not Accepted.
ORA-06512: at "TKT1.TRG_CHK_REG_CS", line 9
ORA-04088: error during execution of trigger 'TKT1.TRG_CHK_REG_CS'
```

Error Scenario 3: Reject update of status to 'Success' when the payment details is not valid.

```
SQL> UPDATE RegisterCourse SET RegisterStatus = 'Success' WHERE RegisterID = 2712;
UPDATE RegisterCourse SET RegisterStatus = 'Success' WHERE RegisterID = 2712
       *
ERROR at line 1:
ORA-20105:
The Payment Detail Is Not Valid.
Please Double Check The Payment Detail To Prevent Human Error.
ORA-06512: at "TKT1.TRG_CHK_REG_CS", line 15
ORA-04088: error during execution of trigger 'TKT1.TRG_CHK_REG_CS'
```

Error Scenario 4: Reject update of status to ‘Success’ when the payment has overdue.

```
SQL> UPDATE RegisterCourse SET RegisterStatus = 'Success' WHERE RegisterID = 2712;
UPDATE RegisterCourse SET RegisterStatus = 'Success' WHERE RegisterID = 2712
*
ERROR at line 1:
ORA-20106:
This Is An Overdue Payment Which Should Be Rejected.
Hence, Update Of Register Status To Success Will Not Be Executed.
ORA-06512: at "TKT1.TRG_CHK_REG_CS", line 18
ORA-04088: error during execution of trigger 'TKT1.TRG_CHK_REG_CS'
```

3.1.8. Report 1: On Demand Basis Report on Unresolved Course Registration Payment

Purpose: This is an on demand basis report which show all the unresolved payment for learner's course registration in one semester. This report will include the learner id, learner name, course registered, course fee and total outstanding payment for each learner. This will provide the insight of learner's payment and total owning payment.

SQL Statement:

```
CREATE OR REPLACE PROCEDURE prc_print_unpay_rc_report(v_semID IN NUMBER DEFAULT 201909) AS
    v_prog  VARCHAR(3);

    CURSOR  ln_cursor  IS
       SELECT RegisterDate, RegisterID, RC.LearnerID, L.LearnerName, LP.ProgrammeCode
       FROM RegisterCourse RC, Learner L, LearnerProgramme LP
       WHERE RC.LearnerID = L.LearnerID AND L.LearnerID = LP.LearnerID AND
             SemesterID = v_semID AND
             (RC.PaymentDate IS NULL OR RC.PaymentMethod IS NULL)
       ORDER BY 1, 2, 3, 4;

    ln_rec  ln_cursor%ROWTYPE;

    CURSOR  prog_cursor  IS
       SELECT C.CourseCode, CourseName, FeePerCH * CreditHour Fee
       FROM ProgrammeCourse P, Course C
       WHERE P.ProgrammeCode = v_prog AND
             P.CourseCode = C.CourseCode AND
             P.Semester = CASE v_semID
                            WHEN 201909 THEN 'Y3S2'
                            WHEN 201905 THEN 'Y3S1'
                            WHEN 201903 THEN 'Y3S3'
                            WHEN 201809 THEN 'Y2S2'
                            WHEN 201805 THEN 'Y2S1'
                            WHEN 201803 THEN 'Y2S3'
                            WHEN 201709 THEN 'Y1S2'
                            WHEN 201705 THEN 'Y1S1'
```

```
                          WHEN 201703 THEN 'Y1S3'
                          ELSE 'Y3S2'
                       END
      ORDER BY 1;

   prog_rec  prog_cursor%ROWTYPE;

   v_studNo NUMBER(3)   := 0;
   v_fee    NUMBER(6,2) := 0;
   v_total  NUMBER(9,2) := 0;

BEGIN
   DBMS_OUTPUT.PUT_LINE(chr(10)||LPAD('Student''s Registration Details With Unresolved Payment',63,' ')||
                        chr(10)||LPAD('In Semester ',39,' ')||to_char(v_semID,'999999')||
                        chr(10)||LPAD('==================',46,' '));

   FOR ln_rec IN ln_cursor
   LOOP
      v_fee := 0;

      DBMS_OUTPUT.PUT_LINE(chr(10)||LPAD('Student No. :',41,' ')||
                           to_char(ln_cursor%ROWCOUNT,'000')||
                           chr(10)||LPAD('================',45,' '));
      DBMS_OUTPUT.PUT_LINE('Student ID   : '||RPAD(ln_rec.learnerID,20,' ')||
                           'Student Name : '||ln_rec.learnerName);
      DBMS_OUTPUT.PUT_LINE('Programme    : '||ln_rec.ProgrammeCode);
      DBMS_OUTPUT.PUT_LINE('RegisterID   : '||RPAD(ln_rec.RegisterID,20,' ')||
                           'RegisterDate : '||ln_rec.RegisterDate);

      DBMS_OUTPUT.PUT_LINE(chr(10)||'Course Registered');
      DBMS_OUTPUT.PUT_LINE('================');
      DBMS_OUTPUT.PUT_LINE('No.'||'  '||
                           'Course Code'||'  '||
                           RPAD('Course Name',40,' ')||'   '||
```

```
                        'Fees (RM)');
    DBMS_OUTPUT.PUT_LINE('---'||'  '||
                        '-----------'||'  '||
                        RPAD('-',40,'-')||'  '||
                        '-----------');

    v_prog := ln_rec.ProgrammeCode;

    FOR prog_rec IN prog_cursor
    LOOP
       DBMS_OUTPUT.PUT_LINE(to_char(prog_cursor%ROWCOUNT,'000')||'   '||
                            RPAD(prog_rec.CourseCode,10,' ')||'  '||
                            RPAD(prog_rec.CourseName,40,' ')||'  RM'||
                            to_char(prog_rec.Fee, '9,999.99'));
       v_fee := v_fee + prog_rec.Fee;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('---'||'  '||
                        '-----------'||'  '||
                        RPAD('-',40,'-')||'  '||
                        '***********'||chr(10)||
                        LPAD('Individual Student''s Sum : RM',62,' ')||
                        to_char(v_fee, '9,999.99')||chr(10));
    v_studNo := ln_cursor%ROWCOUNT;
    v_total := v_total + v_fee;

  END LOOP;
  DBMS_OUTPUT.PUT_LINE(chr(10)||'No. of Students With Unresolved Payment : '||v_studNo);
  DBMS_OUTPUT.PUT_LINE('Total Of Outstanding Students'' Payment  : RM'||to_char(v_total, '9,999,999.99'));
  DBMS_OUTPUT.PUT_LINE(LPAD('END OF REPORT',42,'-')||RPAD('-',29,'-'));

 END;
/
```

Sample Output:

```
         Student's Registration Details With Unresolved Payment
                           In Semester  201909
                           ==================

                            Student No. : 001
                            =================
Student ID   : 1902041             Student Name : Liow Rou Juan
Programme    : DST
RegisterID   : 2701                RegisterDate : 14-OCT-19

Course Registered
=================
No.  Course Code  Course Name                                  Fees (RM)
---  -----------  ---------------------------------------  -----------
001   AACS1203    Games Technology                          RM   750.00
002   AACS2034    Fundamentals of Computer Networks         RM   750.00
003   AACS3094    GUI and Web Application Programming       RM   750.00
004   AACS3234    Electronic Commerce                       RM   750.00
005   AAMS1634    Fundamental Mathematics                   RM   750.00
006   BHEL1013    English Language                          RM   600.00
---  -----------  ---------------------------------------  ***********
                                 Individual Student's Sum : RM 4,350.00

                            Student No. : 002
                            =================
Student ID   : 1902042             Student Name : Beh Ai Shan
Programme    : DST
RegisterID   : 2702                RegisterDate : 14-OCT-19

Course Registered
=================
No.  Course Code  Course Name                                  Fees (RM)
```

```
---  ----------  ---------------------------------------  ----------
001   AACS1203    Games Technology                        RM   750.00
002   AACS2034    Fundamentals of Computer Networks       RM   750.00
003   AACS3094    GUI and Web Application Programming     RM   750.00
004   AACS3234    Electronic Commerce                     RM   750.00
005   AAMS1634    Fundamental Mathematics                 RM   750.00
006   BHEL1013    English Language                        RM   600.00
---  ----------  ---------------------------------------  **********
                              Individual Student's Sum : RM 4,350.00

                           Student No. : 003
                           =================
Student ID   : 1902043             Student Name : Teoh Luo Ning
Programme    : DST
RegisterID   : 2703                RegisterDate : 14-OCT-19

Course Registered
=================
No.  Course Code  Course Name                              Fees (RM)
---  ----------  ---------------------------------------  ----------
001   AACS1203    Games Technology                        RM   750.00
002   AACS2034    Fundamentals of Computer Networks       RM   750.00
003   AACS3094    GUI and Web Application Programming     RM   750.00
004   AACS3234    Electronic Commerce                     RM   750.00
005   AAMS1634    Fundamental Mathematics                 RM   750.00
006   BHEL1013    English Language                        RM   600.00
---  ----------  ---------------------------------------  **********
                              Individual Student's Sum : RM 4,350.00

No. of Students With Unresolved Payment : 3
Total Of Outstanding Students' Payment  : RM    13,050.00
-----------------------------END OF REPORT-----------------------------

PL/SQL procedure successfully completed.
```

3.1.9. Report 2: Detail Report on Department's Staff Workload

Purpose: This is a detail report that show the average academic staff workload for each department in one semester, which include total class taught, total teaching hours, average teaching hours and workload increase ratio between two academic year. This provide the insight of which department require more work force from year to year and resolve it to prevent sudden workforce shortage.

SQL Statement:

```
CREATE OR REPLACE PROCEDURE prc_print_wl_report(v_y1 IN NUMBER DEFAULT 2018,
                                                v_y2 IN NUMBER DEFAULT 2019) AS
    v_dpmt VARCHAR(40) := 'Department of Information Technology';

    CURSOR  staff_cursor  IS
       SELECT Department, cal_staff_by_dprt(Department) noStaff
       FROM Staff
       GROUP BY Department;
    staff_rec  staff_cursor%ROWTYPE;

    CURSOR  tchr_cursor  IS
       SELECT Y1.TotalClass Y1TC, Y1.TotalHr Y1TH,  Y1.AvgHr Y1AH, Y2.TotalClass Y2TC, Y2.TotalHr Y2TH,
              Y2.AvgHr Y2AH, (Y2.AvgHr-Y1.AvgHr)/Y1.AvgHr*100 IncRatio
       FROM
       (SELECT COUNT(AsS.CourseSemID) TotalClass,
               SUM(cal_weekly_class_hr(CS.CourseSemType)) TotalHr,
               SUM(cal_weekly_class_hr(CS.CourseSemType))/cal_staff_by_dprt(v_dpmt) AvgHr
        FROM  Staff S, AssignStaff AsS, CourseSem CS
        WHERE S.StaffNumber = AsS.StaffNumber AND S.Department = v_dpmt AND
              AsS.CourseSemID = CS.CourseSemID AND CS.SemesterID LIKE v_y1||'%') Y1,
       (SELECT COUNT(AsS.CourseSemID) TotalClass,
               SUM(cal_weekly_class_hr(CS.CourseSemType)) TotalHr,
               SUM(cal_weekly_class_hr(CS.CourseSemType))/cal_staff_by_dprt(v_dpmt) AvgHr
        FROM  Staff S, AssignStaff AsS, CourseSem CS
        WHERE S.StaffNumber = AsS.StaffNumber AND S.Department = v_dpmt AND
              AsS.CourseSemID = CS.CourseSemID AND CS.SemesterID LIKE v_y2||'%') Y2;

    tchr_rec  tchr_cursor%ROWTYPE;
```

```
    v_max_dpmt VARCHAR(40);
    v_max_hr   NUMBER(4,2);
    v_max_rate NUMBER(5,2);
    v_min_dpmt VARCHAR(40);
    v_min_hr   NUMBER(4,2);
    v_min_rate NUMBER(5,2);

  BEGIN
    IF (v_y1 < 2017 OR v_y1 > 2019) OR (v_y2 < 2017 OR v_y2 > 2019) THEN
       RAISE_APPLICATION_ERROR(-20110,'Please Enter Valid Academic Year From 2017 To 2019 Only.');
    END IF;

    OPEN tchr_cursor;
    FETCH tchr_cursor INTO tchr_rec;
       v_max_dpmt := v_dpmt;
       v_max_hr := tchr_rec.Y2AH;
       v_max_rate := ROUND(tchr_rec.IncRatio,2);
       v_min_dpmt := v_dpmt;
       v_min_hr := tchr_rec.Y2AH;
       v_min_rate := ROUND(tchr_rec.IncRatio,2);
    CLOSE tchr_cursor;

    FOR staff_rec IN staff_cursor
    LOOP
       v_dpmt := staff_rec.Department;
       DBMS_OUTPUT.PUT_LINE(chr(10)||chr(10)||LPAD('Department : ',16,' ')||staff_rec.Department||
                            chr(10)||RPAD(' ',53,'-'));

       DBMS_OUTPUT.PUT_LINE(chr(10)||LPAD('Year',11,' ')||to_char(v_y1,'9999')||LPAD('Year',26,' ')||
                            to_char(v_y2,'9999')||chr(10)||LPAD('=========',16,' ')||LPAD('=========',31,' '));

       OPEN tchr_cursor;
       FETCH tchr_cursor INTO tchr_rec;
       WHILE tchr_cursor%FOUND
       LOOP
```

```
        DBMS_OUTPUT.PUT_LINE('Total Class  :'||to_char(tchr_rec.Y1TC,'999999')||LPAD('|',6,' ')||
                             LPAD('Total Class  :',19,' ')||to_char(tchr_rec.Y2TC,'999999'));
        DBMS_OUTPUT.PUT_LINE('Total Hour   :'||to_char(tchr_rec.Y1TH,'999999')||LPAD('|',6,' ')||
                             LPAD('Total Hour   :',19,' ')||to_char(tchr_rec.Y2TH,'999999'));
        DBMS_OUTPUT.PUT_LINE('Average Hour :'||to_char(tchr_rec.Y1AH,'999.99')||LPAD('|',6,' ')||
                             LPAD('Average Hour :',19,' ')||to_char(tchr_rec.Y2AH,'999.99'));
        DBMS_OUTPUT.PUT_LINE(LPAD('-',53,'-')||chr(10)||'Increase Ratio : '||
                             to_char(tchr_rec.IncRatio,'990.99')||'%');

        IF tchr_rec.Y2AH > v_max_hr THEN
           v_max_dpmt := staff_rec.Department;
           v_max_hr := tchr_rec.Y2AH;
           v_max_rate := ROUND(tchr_rec.IncRatio,2);
        ELSIF tchr_rec.Y2AH < v_min_hr THEN
           v_min_dpmt := staff_rec.Department;
           v_min_hr := tchr_rec.Y2AH;
           v_min_rate := ROUND(tchr_rec.IncRatio,2);
        END IF;

        FETCH tchr_cursor INTO tchr_rec;
      END LOOP;
      CLOSE tchr_cursor;
   END LOOP;

   DBMS_OUTPUT.PUT_LINE(chr(10)||LPAD('Workload Summary',49,' ')||chr(10)||LPAD('================',49,' '));
   DBMS_OUTPUT.PUT_LINE(chr(10)||LPAD('Heaviest Workload',29,' ')||LPAD('Lightest Workload',40,' ')||
                        chr(10)||LPAD('================',29,' ')||LPAD('================',40,' '));
   DBMS_OUTPUT.PUT_LINE(chr(10)||LPAD(v_max_dpmt,38,' ')||LPAD('|',3,' ')||LPAD(v_min_dpmt,36,' '));
   DBMS_OUTPUT.PUT_LINE('-'||' Average Hours   :'||to_char(v_max_hr,'999.99')||LPAD('|',15,' ')||
                        LPAD('Average Hours   :',20,' ')||to_char(v_min_hr,'999.99')||
                        chr(10)||'  Increased Ratio :'||to_char(v_max_rate,'990.99')||
                        LPAD('|',15,' ')||LPAD('Increased Ratio :',20,' ')||to_char(v_min_rate,'990.99'));
   DBMS_OUTPUT.PUT_LINE(LPAD('END OF REPORT',46,'-')||RPAD('-',33,'-'));
END;
/
```

Sample Output:

```
This Query Will Show The Average Staff Workload Details For Each Department In 2 Academic Years.
Please Enter 2 Academic Years To Proceed.

Enter the Year 1 > 2018
Enter the Year 2 > 2019


   Department : Department of Information Technology
 ----------------------------------------------------

       Year 2018                     Year 2019
       =========                     =========
Total Class  :     80     |     Total Class  :    169
Total Hour   :    480     |     Total Hour   :   1014
Average Hour :  24.00     |     Average Hour :  50.70
----------------------------------------------------
Increase Ratio :  111.25%


   Department : Department of Information Systems
 ----------------------------------------------------

       Year 2018                     Year 2019
       =========                     =========
Total Class  :     56     |     Total Class  :     43
Total Hour   :    336     |     Total Hour   :    258
Average Hour :  24.00     |     Average Hour :  18.43
----------------------------------------------------
Increase Ratio :  -23.21%


   Department : Department of Computer Science
```

```
 ---------------------------------------------------

       Year 2018                      Year 2019
       =========                      =========
Total Class  :     85     |     Total Class  :     66
Total Hour   :    508     |     Total Hour   :    396
Average Hour :  23.09     |     Average Hour :  18.00
---------------------------------------------------
Increase Ratio :  -22.05%


   Department : Department of Mathematical Science
 ---------------------------------------------------

       Year 2018                      Year 2019
       =========                      =========
Total Class  :     64     |     Total Class  :     48
Total Hour   :    386     |     Total Hour   :    288
Average Hour :  24.13     |     Average Hour :  18.00
---------------------------------------------------
Increase Ratio :  -25.39%

                                Workload Summary
                                ================
           Heaviest Workload                      Lightest Workload
           =================                      =================

  Department of Information Technology  |   Department of Computer Science
- Average Hours   :  50.70              |   Average Hours   :  18.00
  Increased Ratio : 111.25              |   Increased Ratio : -22.05
---------------------------------END OF REPORT---------------------------------

PL/SQL procedure successfully completed.
```

3.1.10. Report 3: Summary Report on Elective Course Popularity

Purpose: This is a summary report that show the elective course with low learner commitment in year 2018 and 2019.

SQL Statement:

```
CREATE OR REPLACE PROCEDURE prc_print_ec_report(v_y1 IN NUMBER DEFAULT 2018,
                                                v_y2 IN NUMBER DEFAULT 2019) AS
    v_regnum   NUMBER(4) := 0;
    v_ccode    VARCHAR(8);
    v_count    NUMBER(2) := 0;
    v_csem_low NUMBER(2) := 0;
    v_low_reg  NUMBER(6,2) := 0;
    v_low_fee  NUMBER(9,2) := 0;
    v_csem_all NUMBER(2) := 0;
    v_all_reg  NUMBER(6,2) := 0;
    v_all_fee  NUMBER(9,2) := 0;

    CURSOR  csem_cursor  IS
       SELECT C.CourseCode, C.CourseName, COUNT(CSD.CourseSemID) as RegNum,
              COUNT(CSD.COURSESEMID) * C.FeePerCH * C.CreditHour as FeeReceive
       FROM Course C LEFT OUTER JOIN
            ( SELECT CD.CourseSemID, CS.CourseCode
              FROM CourseDetail CD, CourseSem CS
              WHERE CD.CourseSemID = CS.CourseSemID AND
                    CS.SemesterID IN (201803, 201805, 201809, 201903, 201905, 201909)
            ) CSD ON C.CourseCode = CSD.CourseCode
       WHERE C.CourseType = 'Elective'
       GROUP BY C.CourseCode, C.CourseName, C.FeePerCH, C.CreditHour
       ORDER BY 3, 1;

    csem_rec  csem_cursor%ROWTYPE;

 BEGIN
    DBMS_OUTPUT.PUT_LINE(chr(10)||'Summary Report On Elective Course With Low Student Commitment
```

```
In Academic Year 2018 And 2019');
    DBMS_OUTPUT.PUT_LINE('=================================================================================================
======');

    DBMS_OUTPUT.PUT_LINE('No.'||'  '||'Course Code'||'  '||RPAD('Course Name',40,' ')||
                         'No. Of Register'||'  '||'Fees Received(RM)');
    DBMS_OUTPUT.PUT_LINE('---'||'  '||'-----------'||'  '||RPAD('-',38,'-')||'  '||
                         '---------------'||'  '||'-----------------');

    FOR csem_rec IN csem_cursor
    LOOP
       IF csem_rec.RegNum < 100 THEN
          v_count := v_count + 1;
          DBMS_OUTPUT.PUT_LINE(to_char(v_count,'00')||'  '||
                               LPAD(csem_rec.CourseCode,10,' ')||'    '||
                               RPAD(csem_rec.CourseName,40,' ')||
                               LPAD(csem_rec.RegNum,15,' ')||'  '||
                               LPAD(to_char(csem_rec.FeeReceive,'999,990.00'),17,' '));

          v_csem_low := v_csem_low + 1;
          v_low_reg := v_low_reg + csem_rec.RegNum;
          v_low_fee := v_low_fee + csem_rec.FeeReceive;
       END IF;
       v_csem_all := v_csem_all + 1;
       v_all_reg := v_all_reg + csem_rec.RegNum;
       v_all_fee := v_all_fee + csem_rec.FeeReceive;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('Expected Outcome For Elective Course');
    DBMS_OUTPUT.PUT_LINE('====================================');
    DBMS_OUTPUT.PUT_LINE('Number Of Student''s Registrations : '||ROUND(v_all_reg/v_csem_all,0));
    DBMS_OUTPUT.PUT_LINE('Total Of Fees Received For Course : RM'||
                         to_char(ROUND(v_all_fee/v_csem_all,0),'999,999.00'));
```

```
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('Average Outcome Of Elective Course With Low Student Commitment');
    DBMS_OUTPUT.PUT_LINE('============================================================');
    DBMS_OUTPUT.PUT_LINE('Number Of Student''s Registrations : '||ROUND(v_low_reg/v_csem_low,0));
    DBMS_OUTPUT.PUT_LINE('Total Of Fees Received For Course : RM'||
                         to_char(ROUND(v_low_fee/v_csem_low,0),'999,999.00'));
    DBMS_OUTPUT.PUT_LINE(LPAD('END OF REPORT',46,'-')||RPAD('-',33,'-'));
    DBMS_OUTPUT.PUT_LINE('Possible Revenue Lost Per Course  : RM'||
                         to_char(ROUND((v_all_fee/v_csem_all)-(v_low_fee/v_csem_low),0),'999,999.00'));

 END;
/
```

Sample Output:

```
Summary Report On Elective Course With Low Student Commitment In Academic Year 2018 And 2019
================================================================================================
No.  Course Code  Course Name                            No. Of Register  Fees Received(RM)
---  -----------  -------------------------------------  ---------------  -----------------
01    AACS2333    Software Requirements Engineering II                 0               0.00
02    AACS2343    Systems Analysis and Design                          0               0.00
03    AACS2353    Web Application Programming                          0               0.00
04    AACS2363    Web Design and Development                           0               0.00
05    AACS2303    Software Engineering I                              90          54,000.00
06    AACS2313    Software Engineering II                             90          54,000.00
07    AACS2323    Software Requirements Engineering I                 90          54,000.00


Expected Outcome For Elective Course
====================================
Number Of Student's Registrations : 133
Total Of Fees Received For Course : RM  79,527.00


Average Outcome Of Elective Course With Low Student Commitment
==============================================================
Number Of Student's Registrations : 39
Total Of Fees Received For Course : RM  23,143.00
Possible Revenue Lost Per Course  : RM  56,384.00
---------------------------------END OF REPORT---------------------------------

PL/SQL procedure successfully completed.
```

3.2. Tee Yu June

3.2.1. Query 1: Yearly Comparison of Number of Intake Students in each program

Purpose: The purpose of this query is to display all the number of intake students in each program between 2 years inputted by the user. This query is used to address the strategic information in which the percentage changes in student's enrolment in the respective inputted years can be clearly shown and compared to make decisions in achieving long term vision and target. For example, staff can assign more tutor to the most popular program or eliminate the least popular program to reduce capital cost.

SQL Statement:

```
SELECT P.ProgrammeCode, P.ProgrammeName, VL.StudNUm AS FirstYear, VP.StudNUm AS SecondYear,
       ROUND(((VP.StudNUm-VL.StudNUm)/VP.StudNUm)*100,2) AS Differences
FROM Programme P, VIEW_LearnerProgramme VL, VIEW_LearnerProgramme VP
WHERE P.ProgrammeCode = VL.ProgrammeCode AND
      VP.ProgrammeCode = P.ProgrammeCode AND
      EXTRACT(YEAR FROM VL.ENROLMENTDATE) = '&firstYear' AND
      EXTRACT(YEAR FROM VP.ENROLMENTDATE) = '&secondYear'
ORDER BY Differences DESC;
```

Sample Output:

Total Intake Number of Students for each Program in 2018,2019

Program Code	Program Name	2018	2019	Differences(%)
RIS	Bachelor in Information Security	10	20	50.00
RSF	Bachelor in Software Engineering	10	15	33.33
REI	Bachelor in Interactive Software Technology	10	13	23.08
RIT	Bachelor in Internet Technology	10	12	16.67
DSE	Diploma in Software Engineering	10	11	9.09
RSD	Bachelor in Software System Development	10	11	9.09
DST	Diploma in Interactive Software Technology	16	17	5.88
DIT	Diploma in Internet Technology	24	25	4.00

3.2.2. Query 2: Relationships between Program Details and Number of Students enrolled in each program

Purpose: The purpose of this query is to compare the relationships between program details such as total courses, credit hours, fees and the total students' enrolment in each program. This query can address the tactical information and it can determine which program detail (total course, credit hours, fees) has a strong relationship with the total number of student enrolment. Tactical plan such as inserting more courses, decreasing program fees can be made by observing the tactical information generated by this query to increase student enrolments.

SQL Statement:

```
SELECT P.ProgrammeCode, P.ProgrammeName, COUNT(C.CourseCode) AS TotalCourses,
       SUM(C.CreditHour) AS TotalCredits,SUM(C.FeePerCH * C.CreditHour) AS FEE,
       (SELECT COUNT(LP.LearnerID)
       FROM LearnerProgramme LP
       WHERE LP.ProgrammeCode=P.ProgrammeCode) AS Students
FROM Course C, ProgrammeCourse PC, Programme P
WHERE P.ProgrammeCode = PC.ProgrammeCode AND
      PC.CourseCode = C.CourseCode
GROUP BY P.ProgrammeCode, P.ProgrammeName
ORDER BY 4 DESC;
```

Sample Output:

Relationship Between Programs Details And Total Students Enrolled

Code	Program Name	Total Courses	Total Credits	Program Fees	Total Students
DIT	Diploma in Internet Technology	50	142	$35,100.00	73
DST	Diploma in Interactive Software Technology	48	136	$32,775.00	49
RIS	Bachelor in Information Security	47	133	$32,625.00	40
REI	Bachelor in Interactive Software Technology	45	127	$30,525.00	33
RSD	Bachelor in Software System Development	45	127	$32,175.00	31
DSE	Diploma in Software Engineering	45	127	$31,725.00	31
RSF	Bachelor in Software Engineering	45	127	$31,575.00	35
RIT	Bachelor in Internet Technology	45	127	$30,525.00	32

3.2.3. Query 3: Program Details on every Semester

Purpose: The purpose of this query is to display the program details such as its program course and subtotal program fees in every semester. The operational information generated by this query enables staff to check whether a semester has a lack, of course, the course fee is too expensive or too much of course in a semester.

SQL Statements:

```
SELECT PC.Semester, P.ProgrammeCode, P.ProgrammeName, PC.CourseCode,C.CourseType,(C.FeePerCH*C.Credithour) AS Fee
FROM Programme P, ProgrammeCourse PC, Course C
WHERE P.ProgrammeCode = PC.ProgrammeCode AND
      PC.COurseCode = C.CourseCode AND
      P.ProgrammeCode = '&v_Program_Code'
ORDER BY PC.semester;
```

Sample Output:

```
Enter the Program Code > RSF

                          RSF Program Course Details From Y1S1 to Y3S3


SEMESTER Programme Code Programe Name                                  COURSECO Type                   FEE
-------- ------------- ------------------------------------------- -------- ------------- -----------
Y1S1     RSF           Bachelor in Software Engineering            BHEL1013 Compulsory        $600.00
                                                                   BAIT3095 Core              $750.00
                                                                   BACS1033 Core              $750.00
                                                                   BACS2224 Core              $750.00
                                                                   BAIT3094 Core              $750.00
                                                                   BACS2003 Core              $750.00
                       ******************************************                           -----------
                       Average Course Fees :                                                  $725.00
                       Current Semester Fees :                                              $4,350.00
```

3.2.4. Procedure 1: Insert Course to a Program

Purpose: The purpose of this procedure is to insert a current available course to a specific program and users need to input the course code, program code, program structure(semester), course type, day, start time and end time when calling this procedure. This procedure will also include some validation for users' input in order to ensure input is correct before added into the database.

Sample Output:

```
CREATE OR REPLACE PROCEDURE PRC_INSERT_PROGRAM_COURSE( v_courseCode IN VARCHAR,
                                                       v_programCode IN CHAR,
                                                       v_semesterID IN VARCHAR,
                                                       v_courseType IN VARCHAR,
                                                       v_day IN VARCHAR,
                                                       v_start_time IN VARCHAR,
                                                       v_end_time IN VARCHAR) AS
    v_Sem                     ProgrammeCourse.Semester%TYPE;
    v_time                    CourseSem.Time%TYPE;
    SEMESTER_NOT_AVAILABLE    exception;
    NUMERIC_TIME_ERROR        exception;
    TIME_LOGIC_ERROR          exception;
    MIN_TIME_ERROR            exception;
BEGIN
    IF v_semesterID = 201705 THEN
        v_Sem :='Y1S1';
```

```
ELSIF v_semesterID = 201709 THEN
    v_Sem :='Y1S2';
ELSIF v_semesterID = 201703 THEN
    v_Sem :='Y1S3';
ELSIF v_semesterID = 201805 THEN
    v_Sem :='Y2S1';
ELSIF v_semesterID = 201809 THEN
    v_Sem :='Y2S2';
ELSIF v_semesterID = 201803 THEN
    v_Sem :='Y2S3';
ELSIF v_semesterID = 201905 THEN
    v_Sem :='Y3S1';
ELSIF v_semesterID = 201909 THEN
    v_Sem :='Y3S2';
ELSIF v_semesterID = 201903 THEN
    v_Sem :='Y3S3';
ELSE
    RAISE SEMESTER_NOT_AVAILABLE;
END IF;
IF(is_number(v_start_time)=1) THEN
```

```
        IF(is_number(v_end_time)=1) THEN
            IF(v_end_time > v_start_time) THEN
                IF((v_end_time - v_start_time) >= 100) THEN
                    v_time := v_start_time||'-'||v_end_time;
                ELSE
                    RAISE MIN_TIME_ERROR;
                END IF;
            ELSE RAISE TIME_LOGIC_ERROR;
            END IF;
        ELSE
            RAISE NUMERIC_TIME_ERROR;
        END IF;
    ELSE
        RAISE NUMERIC_TIME_ERROR;
END IF;

INSERT INTO CourseSem Values
(CourseSem_Seq.nextval,v_courseCode,v_semesterID,v_courseType,v_day,v_time);
INSERT INTO ProgrammeCourse Values(v_programCode,v_courseCode,v_Sem);
```

```
DBMS_OUTPUT.PUT_LINE(chr(10));
DBMS_OUTPUT.PUT_LINE('New course: ' ||v_courseCode ||' successfully inserted into '||v_programCode||
                     ' program');
DBMS_OUTPUT.PUT_LINE(chr(10));
DBMS_OUTPUT.PUT_LINE('========== Inserted Record ==========');
DBMS_OUTPUT.PUT_LINE('Semester ID : ' || v_semesterID);
DBMS_OUTPUT.PUT_LINE('Program     : ' || v_programCode);
DBMS_OUTPUT.PUT_LINE('Course      : ' || v_courseCode);
DBMS_OUTPUT.PUT_LINE('Day         : ' || v_day);
DBMS_OUTPUT.PUT_LINE('Time        : ' || v_time);
DBMS_OUTPUT.PUT_LINE('====================================');
EXCEPTION
    WHEN SEMESTER_NOT_AVAILABLE THEN
        RAISE_APPLICATION_ERROR(-20000,v_semesterID||' Semester ID is not available ! ');
    WHEN NUMERIC_TIME_ERROR THEN
        RAISE_APPLICATION_ERROR(-20000,'Non-Numeric Time FOUND ! ');
    WHEN TIME_LOGIC_ERROR THEN
        RAISE_APPLICATION_ERROR(-20000,'End time must be later than Start time.');
    WHEN MIN_TIME_ERROR THEN
        RAISE_APPLICATION_ERROR(-20000,'Class must be at least 1 hour.');
```

```
END;
/
```

Sample Output:

Error Scenario 1: When the semester input is not available or invalid semester.

```
BEGIN PRC_INSERT_PROGRAM_COURSE('AACS1473','RSD','201708','Lecture','Mon',1800,1930); END;
*
ERROR at line 1:
ORA-20000: 201708 Semester ID is not available !
ORA-06512: at "JUNE.PRC_INSERT_PROGRAM_COURSE", line 92
ORA-06512: at line 1
```

Error Scenario 2 : Reject non-numeric time input such as alphebetical value (six thirthy).

```
BEGIN PRC_INSERT_PROGRAM_COURSE('AACS1473','RSD','201709','Lecture','Mon','six thirty',1930); END;
*
ERROR at line 1:
ORA-20000: Non-Numeric Time FOUND !
ORA-06512: at "JUNE.PRC_INSERT_PROGRAM_COURSE", line 95
ORA-06512: at line 1
```

Error Scenario 3 : Reject the time input where end time is earlier than start time of a class.

```
BEGIN PRC_INSERT_PROGRAM_COURSE('AACS1473','RSD','201705','Lecture','Mon',1900,1730); END;
*
ERROR at line 1:
ORA-20000: End time must be later than Start time.
ORA-06512: at "JUNE.PRC_INSERT_PROGRAM_COURSE", line 98
ORA-06512: at line 1
```

Error Scenario 4 : Reject the class duration which is less than one hour.

```
BEGIN PRC_INSERT_PROGRAM_COURSE('AACS1473','RSD','201705','Lecture','Mon',1800,1830); END;
*
ERROR at line 1:
ORA-20000: Class must be at least 1 hour.
ORA-06512: at "JUNE.PRC_INSERT_PROGRAM_COURSE", line 101
ORA-06512: at line 1
```

Success Scenario:

```
New course: AACS1473 successfully inserted into RSD program
========== Inserted Record ==========
Semester ID : 201705
Program     : RSD
Course      : AACS1473
Day         : Mon
Time        : 1800-1930
=====================================
```

3.2.5. Procedure 2: Update all course fees by percentage based on course type.

Purpose: The purpose of this procedure is to update all the course fee based on the course type given. If the input given is invalid such as 'ten' instead of numerical 10 or the course type is not available, then it will prompt application errors for each of these invalid input. If all the input is correct, then successful message and updated course fee in every program details will be shown.

Sample Statement:

```
CREATE OR REPLACE PROCEDURE PRC_UPDATE_FEE(In_Updt_Percent IN VARCHAR, In_Course_Type IN VARCHAR) IS


v_num_records  number;
v_old_FeePerCH Course.FeePerCH%TYPE;
v_new_FeePerCH Course.FeePerCH%TYPE;
NON_NUMERIC_PERCENT_ERROR exception;
NO_THIS_COURSE_TYPE  exception;
v_empty_cursor boolean;

CURSOR CourseFee_Cursor IS
    SELECT PC.ProgrammeCode AS ProgrammeCode,C.CourseCode, C.CourseName, C.CourseType, C.FeePerCH
    FROM Course C, ProgrammeCourse PC
WHERE C.CourseCode = PC.CourseCode AND
      UPPER(C.CourseType) = UPPER(In_Course_Type);

CourseFee_rec CourseFee_Cursor%ROWTYPE;
```

```
BEGIN
    IF(is_number(In_Updt_Percent)=0) THEN
        RAISE NON_NUMERIC_PERCENT_ERROR;
    END IF;

    v_empty_cursor := true;

    FOR CourseFee_rec IN CourseFee_Cursor LOOP
        v_empty_cursor := false;
    END LOOP;

    IF(v_empty_cursor = true) THEN
        RAISE NO_THIS_COURSE_TYPE;
    END IF;
    v_num_records := 0;

    DBMS_OUTPUT.PUT_LINE(chr(10));
DBMS_OUTPUT.PUT_LINE(' Fee Per Credits of '||In_Course_Type ||
' Successfully Increased by '||In_Updt_Percent||'%');
```

```
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('============================ REPORT ==============================');
DBMS_OUTPUT.PUT_LINE('NO  '||'Program Code '||' Course Code '||' Course Type    '||' Old Fee '||
                     ' Updated Fee ');
DBMS_OUTPUT.PUT_LINE('=== '||'============ '||' ========== '||' ============== '||' ======= '||
                     ' ========== ');
    DBMS_OUTPUT.PUT_LINE(chr(10));
    FOR CourseFee_rec IN CourseFee_Cursor LOOP
        v_old_FeePerCH := CourseFee_rec.FeePerCH;
        v_new_FeePerCH := v_old_FeePerCH * ((100 + In_Updt_Percent)/100);

        UPDATE COURSE
        SET FeePerCH = v_new_FeePerCH
        WHERE CourseCode = CourseFee_rec.CourseCode
        AND CourseType = In_Course_Type;

DBMS_OUTPUT.PUT_LINE(to_char(CourseFee_Cursor%rowcount,'000')||'  '||
                        RPAD(CourseFee_rec.ProgrammeCode,13,' ')||
                        RPAD(CourseFee_rec.CourseCode,13,' ')||
                     RPAD(CourseFee_rec.CourseType,14,' ')||
```

```
                    RPAD(to_char(v_old_FeePerCH,'$999.99'),9,' ')||
                        RPAD(to_char(v_new_FeePerCH,'$999.99'),9,' '));

    v_num_records := v_num_records+1;

    END LOOP;

    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('========================== END OF REPORT ==========================');
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('Number of Records Updated : '||v_num_records);

    EXCEPTION
        WHEN NON_NUMERIC_PERCENT_ERROR THEN
            RAISE_APPLICATION_ERROR(-20000,'Update Pecent Must Be a Number!');

        WHEN NO_THIS_COURSE_TYPE THEN
            RAISE_APPLICATION_ERROR(-20000,'Inserted Course Type "'||In_Course_Type||'" Cannot Be Found!');
END;
/
```

Sample Output:

Error Scenario 1: Reject alphabetical input such as ‘Ten’ instead of number (10) for the update’s percent.

```
BEGIN PRC_UPDATE_FEE('Ten', 'Core'); END;
*
ERROR at line 1:
ORA-20000: Update Pecent Must Be a Number!
ORA-06512: at "JUNE.PRC_UPDATE_FEE", line 67
ORA-06512: at line 1
```

Error Scenario 2: Reject the inputted course type which is not available or invalid course.

```
BEGIN PRC_UPDATE_FEE(10, 'Main Core'); END;
*
ERROR at line 1:
ORA-20000: Inserted Course Type "Main Core" Cannot Be Found!
ORA-06512: at "JUNE.PRC_UPDATE_FEE", line 70
ORA-06512: at line 1
```

Successful Scenario:

```
Fee Per Credits of Core Elective Successfully Increased by 10%


============================ REPORT ================================
NO  Program Code  Course Code  Course Type     Old Fee  Updated Fee
=== ============  ===========  ==============  =======  ===========
001  RSF           BACS2042     Core Elective  $225.00  $247.50
002  RSF           BACS3033     Core Elective  $225.00  $247.50
003  RSF           BACS2073     Core Elective  $225.00  $247.50
004  RSF           BACS2163     Core Elective  $225.00  $247.50
005  RSF           BACS3003     Core Elective  $225.00  $247.50
006  RIT           BACS2042     Core Elective  $225.00  $247.50
007  RIT           BACS3033     Core Elective  $225.00  $247.50
008  RIT           BACS2073     Core Elective  $225.00  $247.50
009  RIT           BACS2163     Core Elective  $225.00  $247.50
010  RIT           BACS3003     Core Elective  $225.00  $247.50
011  REI           BACS2042     Core Elective  $225.00  $247.50
012  REI           BACS3033     Core Elective  $225.00  $247.50
013  REI           BACS2073     Core Elective  $225.00  $247.50
014  REI           BACS2163     Core Elective  $225.00  $247.50
```

```
015  REI          BACS3003     Core Elective  $225.00  $247.50
016  RSD          BACS2042     Core Elective  $225.00  $247.50
017  RSD          BACS3033     Core Elective  $225.00  $247.50
018  RSD          BACS2073     Core Elective  $225.00  $247.50
019  RSD          BACS2163     Core Elective  $225.00  $247.50
020  RSD          BACS3003     Core Elective  $225.00  $247.50
021  DSE          AAMS1613     Core Elective  $225.00  $247.50
022  DSE          AACS1143     Core Elective  $225.00  $247.50
023  DSE          AAMS2613     Core Elective  $225.00  $247.50
024  DSE          AACS1074     Core Elective  $225.00  $247.50
025  DSE          AACS1084     Core Elective  $225.00  $247.50
026  DST          AAMS1613     Core Elective  $225.00  $247.50
027  DST          AACS1143     Core Elective  $225.00  $247.50
028  DST          AAMS2613     Core Elective  $225.00  $247.50
029  DST          AACS1074     Core Elective  $225.00  $247.50
030  DST          AACS1084     Core Elective  $225.00  $247.50
031  RIS          AACS1143     Core Elective  $225.00  $247.50

========================== END OF REPORT ==========================

Number of Records Updated : 31
```

3.2.6. Trigger 1: Update Total Program Fees

Purpose: The purpose of this trigger is to update which is increasing the program fees by summing up the new course fee and total credit hours after that particular course was inserted into that particular program or decreasing a particular program fee and credit hours by subtracting the deleted program course fees and credit hours in that particular program when a program course was removed from it.

Trigger Code:

```
LINE TEXT
---- --------------------------------------------------------------------------------------------
 001 TRIGGER TRG_UPT_PROGRAM_FEE
 002 AFTER INSERT OR
 003       DELETE ON ProgrammeCourse
 004
 005 FOR EACH ROW
 006 DECLARE
 007
 008     v_courseFee         Course.CourseFee%TYPE;
 009     v_creditHours       Programme.TotalCreditHours%TYPE;
 010
 011 BEGIN
 012
 013     CASE
 014
 015         WHEN INSERTING THEN
 016
 017             SELECT (FeePerCh * CreditHour), CreditHour into v_courseFee, v_creditHours
 018             FROM Course
 019             WHERE CourseCode = :new.CourseCode;
 020
 021             IF SQL%FOUND THEN
 022
 023                 UPDATE Programme
 024                 SET ProgrammeFee = ProgrammeFee + v_courseFee,
 025                     TotalCreditHours = TotalCreditHours + v_creditHours
 026                 WHERE ProgrammeCode = :new.ProgrammeCode;
```

```
027
028             END IF;
029
030         WHEN DELETING THEN
031
032             SELECT (FeePerCh * CreditHour), CreditHour into v_courseFee, v_creditHours
033             FROM Course
034             WHERE CourseCode = :old.CourseCode;
035
036             IF SQL%FOUND THEN
037
038                 UPDATE Programme
039                 SET ProgrammeFee = ProgrammeFee - v_courseFee,
040                     TotalCreditHours = TotalCreditHours - v_creditHours
041                 WHERE ProgrammeCode = :old.ProgrammeCode;
042
043             END IF;
044
045     END CASE;
046
047     EXCEPTION
048         WHEN NO_DATA_FOUND THEN
049             RAISE_APPLICATION_ERROR(-20000,'No Data Found! Try Again!');
050
051         WHEN OTHERS THEN
052             RAISE_APPLICATION_ERROR(-20000,'ERROR 404');
053
054 END;
```

Sample output Before Triggers:

Program Code	Program Name	TOTALCREDITHOURS	Program Fees
RSF	Bachelor in Software Engineering	0	$.00
RIS	Bachelor in Information Security	0	$.00
RIT	Bachelor in Internet Technology	0	$.00
REI	Bachelor in Interactive Software Technology	0	$.00
RSD	Bachelor in Software System Development	0	$.00
DSE	Diploma in Software Engineering	0	$.00
DIT	Diploma in Internet Technology	0	$.00
DST	Diploma in Interactive Software Technology	0	$.00

Sample Output After Triggers:

Program Code	Program Name	TOTALCREDITHOURS	Program Fees
RSF	Bachelor in Software Engineering	127	$31,575.00
RIS	Bachelor in Information Security	127	$31,350.00
RIT	Bachelor in Internet Technology	127	$30,525.00
REI	Bachelor in Interactive Software Technology	127	$30,525.00
RSD	Bachelor in Software System Development	124	$31,425.00
DSE	Diploma in Software Engineering	127	$31,725.00
DIT	Diploma in Internet Technology	127	$31,350.00

3.2.7. Trigger 2: Validate inserted program course

Purpose: The purpose of this trigger is to validate the course being inserted into a program is correct and follows the system's business rules. For example, if the inserted course is not available in the system, number of course in short semester is more than 4, the number of course in a long semester is more than 7, insert course into an invalid program, the course already existed in a program will prompt an error before the information was inserted into the database. If the inserted course information is all correct, then it will store it in the database.

Trigger Code:

```
LINE TEXT
---- ----------------------------------------------------------------------------------------------------
 001 TRIGGER TRG_VALIDATE_INSERT_CP
 002
 003 BEFORE INSERT ON ProgrammeCourse
 004
 005 FOR EACH ROW
 006 DECLARE
 007
 008     OVER_PERSEMCOURSE_LIMIT EXCEPTION;
 009     ALREADY_EXIST_ERROR     EXCEPTION;
 010     INVALID_SEM_ERROR       EXCEPTION;
 011     OVER_PERSHORTSEMCOURSE_LIMIT EXCEPTION;
 012     v_CourseCode         Course.CourseCode%TYPE;
```

```
013     v_ProgrammeCode      Programme.ProgrammeCode%TYPE;
014     v_PerSemCourse       NUMBER;
015
016 BEGIN
017
018     SELECT ProgrammeCode INTO v_ProgrammeCode
019     FROM Programme
020     WHERE ProgrammeCode = :new.ProgrammeCode;
021
022     IF SQL%FOUND THEN
023
024         BEGIN
025             SELECT CourseCode INTO v_CourseCode
026             FROM Course
027             WHERE CourseCode = :new.CourseCode;
028
029             IF SQL%FOUND THEN
030
031                 SELECT COUNT(CourseCode) INTO v_PerSemCourse
032                 FROM ProgrammeCourse
033                 WHERE ProgrammeCode = :new.ProgrammeCode
```

```
034                 AND Semester=:new.Semester;
035
036                 IF :new.Semester IN ('Y1S3','Y2S3','Y3S3') THEN
037                     IF v_PerSemCourse>3 THEN
038                         RAISE OVER_PERSHORTSEMCOURSE_LIMIT;
039                     END IF;
040
041                 ELSIF :new.Semester IN ('Y1S1','Y1S2','Y2S1','Y2S2','Y3S1','Y3S2') THEN
042                     IF v_PerSemCourse>6 THEN
043                         RAISE OVER_PERSEMCOURSE_LIMIT;
044                     END IF;
045
046                 ELSE
047                     RAISE INVALID_SEM_ERROR;
048                 END IF;
049
050                     BEGIN
051
052                     SELECT ProgrammeCode, CourseCode INTO v_ProgrammeCode,v_CourseCode
053                     FROM ProgrammeCourse
054                     WHERE ProgrammeCode = :new.ProgrammeCode
```

```
055                     AND CourseCode = :new.CourseCode;
056
057                     IF SQL%FOUND THEN
058                        RAISE ALREADY_EXIST_ERROR;
059                     ELSE
060                        RAISE NO_DATA_FOUND;
061                     END IF;
062                         EXCEPTION
063                             WHEN NO_DATA_FOUND THEN
064                                NULL;
065                             WHEN ALREADY_EXIST_ERROR THEN
066                                RAISE_APPLICATION_ERROR(-20000,v_CourseCode||
                                                        ' Course already exist   in '||
                                                        v_ProgrammeCode||' Program');
067                     END;
068
069             ELSE
070                 RAISE NO_DATA_FOUND;
071             END IF;
072
073             EXCEPTION
```

```
074                 WHEN NO_DATA_FOUND THEN
075                     RAISE_APPLICATION_ERROR(-20000,:new.CourseCode||' Course does not available!');
076                 WHEN OVER_PERSEMCOURSE_LIMIT THEN
077                     RAISE_APPLICATION_ERROR(-20000,'Long Semester only allowed MAX 7 courses!');
078                 WHEN INVALID_SEM_ERROR THEN
079                     RAISE_APPLICATION_ERROR(-20000,:new.Semester||' semester is invalid! ');
080                 WHEN OVER_PERSHORTSEMCOURSE_LIMIT THEN
081                     RAISE_APPLICATION_ERROR(-20000,'Short Semester only allowed MAX 4 courses!');
082
083         END;
084
085     END IF;
086
087     EXCEPTION
088
089        WHEN NO_DATA_FOUND THEN
090            RAISE_APPLICATION_ERROR(-20000, :new.ProgrammeCode ||' Program does not available!');
091
092 END;
```

Sample output:

Error Scenario 1: Reject Course which already exists in program.

```
insert into programmeCourse Values('RSF','BACS2003','Y1S1')
            *
ERROR at line 1:
ORA-20000: BACS2003 Course already exist in RSF Program
ORA-06512: at "JUNE.TRG_VALIDATE_INSERT_CP", line 61
ORA-04088: error during execution of trigger 'JUNE.TRG_VALIDATE_INSERT_CP'
```

Error Scenario 2: Reject invalid program which is not exits in the system.

```
insert into programmeCourse Values('RSS','BACS2003','Y1S1')
            *
ERROR at line 1:
ORA-20000: RSS Program does not available!
ORA-06512: at "JUNE.TRG_VALIDATE_INSERT_CP", line 85
ORA-04088: error during execution of trigger 'JUNE.TRG_VALIDATE_INSERT_CP'
```

Error Scenario 3: Reject Course which is not exits in the system.

```
insert into programmeCourse Values('RSF','BACS2000','Y1S1')
            *
ERROR at line 1:
ORA-20000: BACS2000 Course does not available!
ORA-06512: at "JUNE.TRG_VALIDATE_INSERT_CP", line 70
ORA-04088: error during execution of trigger 'JUNE.TRG_VALIDATE_INSERT_CP'
```

Error Scenario 4: Reject invalid semester or a semester which is not available in the system.

```
insert into programmeCourse Values('RSF','AACS1084','Y1S4')
            *
ERROR at line 1:
ORA-20000: Y1S4 semester is invalid!
ORA-06512: at "JUNE.TRG_VALIDATE_INSERT_CP", line 74
ORA-04088: error during execution of trigger 'JUNE.TRG_VALIDATE_INSERT_CP'
```

Error Scenario 5: Reject course to insert when a long semester already have max 7 courses.

```
insert into programmeCourse Values('RSF','AACS2123','Y1S1')
            *
ERROR at line 1:
ORA-20000: Long Semester only allowed MAX 7 courses!
ORA-06512: at "JUNE.TRG_VALIDATE_INSERT_CP", line 72
ORA-04088: error during execution of trigger 'JUNE.TRG_VALIDATE_INSERT_CP'
```

Error Scenario 6 : Reject course to insert when a short semester already have max 4 courses.

```
insert into programmeCourse Values('RSF','AAMS3184','Y1S3')
            *
ERROR at line 1:
ORA-20000: Short Semester only allowed MAX 4 courses!
ORA-06512: at "JUNE.TRG_VALIDATE_INSERT_CP", line 76
ORA-04088: error during execution of trigger 'JUNE.TRG_VALIDATE_INSERT_CP'
```

3.2.8. Report 1: On-Demand Report of Yearly Number of Intake Students in each Program

Purpose: The purpose of this on-demand report display all the number of intake students and percentage changes of students enrolment in each program between 2 years inputted by the user. This report will also calculate and display the overall total and average number of student enrolment in each of the inputted years and the overall percentage changes of students enrolment between 2 inputted years. By using this report, users can know which program is the populest or least populest and take some action on it such as adding more tutor to a high students' enrolment program.

PL/SQL Code:

```
CREATE OR REPLACE PROCEDURE proc_onDemandReport(In_Year1 IN NUMBER, In_Year2 IN NUMBER)AS


var_LearnerNumber1   NUMBER(3);
var_LearnerNumber2   NUMBER(3);
var_TotalLearner1    NUMBER(3):=0;
var_TotalLearner2    NUMBER(3):=0;
var_dif_in_Percentage NUMBER(5,2);
var_difTotal_in_Percentage NUMBER(5,2);
var_avg_firstyear    NUMBER(2);
var_avg_secondyear   NUMBER(2);
var_count            NUMBER(2):=0;
var_total_differences NUMBER(5,2):=0;
var_avg_differences NUMBER(5,2);
```

```
CURSOR pro_cursor IS

     SELECT P.ProgrammeCode AS ProgrammeCode, P.ProgrammeName AS ProgrammeName,
            VL.StudNUm AS var_LearnerNumber1, VP.StudNUm AS var_LearnerNumber2,
            ROUND(((VP.StudNUm-VL.StudNUm)/VP.StudNUm)*100,2) AS var_dif_in_Percentage
     FROM Programme P, VIEW_LearnerProgramme VL, VIEW_LearnerProgramme VP
     WHERE P.ProgrammeCode = VL.ProgrammeCode AND
           VP.ProgrammeCode = P.ProgrammeCode AND
           EXTRACT(YEAR FROM VL.ENROLMENTDATE) = In_Year1 AND
           EXTRACT(YEAR FROM VP.ENROLMENTDATE) = In_Year2
     ORDER BY var_dif_in_Percentage DESC;

pro_rec pro_cursor%ROWTYPE;

BEGIN
     DBMS_OUTPUT.PUT_LINE(chr(10));
     DBMS_OUTPUT.PUT_LINE(RPAD('--',30)||'Yearly Total Program Students On Demand Report');
     DBMS_OUTPUT.PUT_LINE(chr(10));
     DBMS_OUTPUT.PUT_LINE(RPAD('Printed Date: '||To_Char(sysdate,'dd-mm-yyyy'),95,' ')||'Page: 1');
     DBMS_OUTPUT.PUT_LINE(chr(10));
```

```
DBMS_OUTPUT.PUT_LINE('================================================================================
                      ========================');
 DBMS_OUTPUT.PUT_LINE('NO '||'Program Code'||RPAD(' Program Name',47,' ')||
 RPAD(In_Year1,14,' ')||RPAD(In_Year2,12,' ')||RPAD('Differences(%)',15,' '));
DBMS_OUTPUT.PUT_LINE('================================================================================
                      ========================');

FOR pro_rec IN pro_cursor LOOP

    var_TotalLearner1 := var_TotalLearner1 + pro_rec.var_LearnerNumber1;
    var_TotalLearner2 := var_TotalLearner2 + pro_rec.var_LearnerNumber2;
    var_total_differences := var_total_differences + var_dif_in_Percentage;

    DBMS_OUTPUT.PUT_LINE(to_char(pro_cursor%rowcount,'00')||'  '||
                         RPAD(pro_rec.ProgrammeCode,13,' ')||
                         RPAD(pro_rec.ProgrammeName,44,' ')||
                         RPAD(to_char(pro_rec.var_LearnerNumber1,'999'),14,' ')||
                         RPAD(to_char(pro_rec.var_LearnerNumber2,'999'),14,' ')||
                         to_char(pro_rec.var_dif_in_Percentage,'999.99')||'%');
    var_count:=var_count+1;
END LOOP;
```

```
    var_difTotal_in_Percentage := ((var_TotalLearner2-var_TotalLearner1)/var_TotalLearner2)*100;
    var_avg_firstyear := var_TotalLearner1/var_count;
    var_avg_secondyear := var_TotalLearner2/var_count;
    var_avg_differences := ((var_avg_secondyear-var_avg_firstyear)/var_avg_secondyear)*100;

    DBMS_OUTPUT.PUT_LINE(RPAD('--',55)||'     '||'---------- '||' --------------'||'  --------------');
     DBMS_OUTPUT.PUT_LINE(RPAD('--',50)||'TOTAL  :    '||RPAD(var_TotalLearner1,14,' ') ||
                          RPAD(var_TotalLearner2,15,' ')||var_difTotal_in_Percentage||'%');
     DBMS_OUTPUT.PUT_LINE(RPAD('--',50)||'AVG    :     '||RPAD(var_avg_firstyear,14,' ') ||
                          RPAD(var_avg_secondyear,14,' ')||var_avg_differences||'%');
    DBMS_OUTPUT.PUT_LINE(RPAD('--',55)||'     '||'---------- '||' --------------'||'  --------------');
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE(RPAD('--',45)||'End of Report');
END;
/
```

Sample Output:

```
--                          Yearly Total Program Students On Demand Report

Printed Date: 02-01-2020                                                    Page: 1

=============================================================================================
NO Program Code Program Name                                 2018          2019        Differences(%)
=============================================================================================
01  RIS          Bachelor in Information Security             10            20            50.00%
02  RSF          Bachelor in Software Engineering             10            15            33.33%
03  REI          Bachelor in Interactive Software Technology  10            13            23.08%
04  RIT          Bachelor in Internet Technology              10            12            16.67%
05  DSE          Diploma in Software Engineering              10            11             9.09%
06  RSD          Bachelor in Software System Development      10            11             9.09%
07  DST          Diploma in Interactive Software Technology   16            17             5.88%
08  DIT          Diploma in Internet Technology               24            25             4.00%
--                                                        ----------  -------------  -------------
--                                            TOTAL  :     100           124            19.35%
--                                            AVG    :      13            16            18.75%
--                                                        ----------  -------------  -------------

--                                      End of Report
```

3.2.9. Report 2: Summary Report of Program Details and Numbers of Students enrolled in each program

Purpose: The purpose of this summary report is to display total course, total students, total fees in each program and it will also calculate and display the total and average course number, students and fees for every program which sum up. Besides, the highest, lowest of course number and students enrolment in a program will be shown in this report. The user then can decide whether to manage the course number in a program by observing the total students' enrolment in that particular program.

PL/SQL Code:

```
CREATE OR REPLACE PROCEDURE proc_summaryReport AS


var_CourseNumber        NUMBER(5,2);
var_CreditHour          NUMBER(3);
var_ProgramFee          NUMBER(8,2);
var_Total_Courses       NUMBER(6,2):=0;
var_Total_Credits       NUMBER(4):=0;
var_Total_Fees          NUMBER(10,2):=0;
var_Students            NUMBER(5,2);
var_Total_Students      NUMBER(6,2):=0;
var_Highest_Course      NUMBER(2):=0;
var_Lowest_Course       NUMBER(2):=99;
var_Highest_students    NUMBER(3):=0;
var_Lowest_students     NUMBER(3):=999;
var_Highest_Course_Pro  Programme.ProgrammeCode%TYPE;
```

```
var_Lowest_Course_Pro   Programme.ProgrammeCode%TYPE;
var_Highest_Student_Pro Programme.ProgrammeCode%TYPE;
var_Lowest_Student_Pro  Programme.ProgrammeCode%TYPE;
var_avg_Course          NUMBER(5,2);
var_avg_Student         NUMBER(5,2);
var_avg_ProgrammeFee    NUMBER(10,2):=0;
var_loop_num            NUMBER(2):=0;

CURSOR program_cursor IS
    SELECT ProgrammeCode, ProgrammeName
    FROM Programme;
program_rec program_cursor%ROWTYPE;

BEGIN
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE(RPAD('--',25,' ')||'Total Programme Course And Students Summary Report');
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE(RPAD('Printed Date: '||To_Char(sysdate,'dd-mm-yyyy'),95,' ')||'Page: 1');
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('================================================================================
                          ==========================');
```

```
DBMS_OUTPUT.PUT_LINE('NO '||'Program Code'||RPAD(' Program Name',46,' ')||'Total Courses  '||
                     'Total Students  '||'Programme Fee');
DBMS_OUTPUT.PUT_LINE('================================================================================
                      ============================');

FOR program_rec IN program_cursor LOOP
    var_loop_num := var_loop_num+1;

    SELECT COUNT(PC.CourseCode),  SUM(CreditHour), SUM(C.FeePerCH * CreditHour)
           INTO var_CourseNumber,var_CreditHour, var_ProgramFee
    FROM ProgrammeCourse PC, Course C
    WHERE PC.CourseCode = C.CourseCode AND
          PC.ProgrammeCode = program_rec.ProgrammeCode;

    SELECT COUNT(LearnerID) INTO var_Students
    FROM learnerProgramme
    WHERE ProgrammeCode = program_rec.ProgrammeCode;

    IF (var_CourseNumber > var_Highest_Course) THEN
        var_Highest_Course := var_CourseNumber;
        var_Highest_Course_Pro := program_rec.ProgrammeCode;
```

```
ELSIF (var_CourseNumber < var_Lowest_Course) THEN
    var_Lowest_Course := var_CourseNumber;
    var_Lowest_Course_Pro := program_rec.ProgrammeCode;
END IF;

IF (var_Students > var_Highest_students) THEN
    var_Highest_students := var_Students;
    var_Highest_Student_Pro := program_rec.ProgrammeCode;

ELSIF (var_Students < var_Lowest_Course) THEN
    var_Lowest_students := var_Students;
    var_Lowest_Student_Pro := program_rec.ProgrammeCode;
END IF;

DBMS_OUTPUT.PUT_LINE(to_char(program_cursor%rowcount,'00')||
                     '  '||RPAD(program_rec.ProgrammeCode,13,' ')||
                     RPAD(program_rec.ProgrammeName,46,' ')||
                     RPAD(to_char(var_CourseNumber,'999'),14,' ')||
                     RPAD(to_char(var_Students,'999'),14,' ')||
                     RPAD(to_char(var_ProgramFee,'$999,999.99'),14,' '));
```

```
    var_Total_Courses := var_Total_Courses + var_CourseNumber;
    var_Total_Students := var_Total_Students + var_Students;
    var_Total_Fees    := var_Total_Fees    + var_ProgramFee;

    END LOOP;
    var_avg_Course := var_Total_Courses/var_loop_num;
    var_avg_Student:= var_Total_Students/var_loop_num;
    var_avg_ProgrammeFee := var_Total_Fees/var_loop_num;

    DBMS_OUTPUT.PUT_LINE(RPAD('--',58)||'     '||'---------- '||' -----------'||'    --------------');
    DBMS_OUTPUT.PUT_LINE(RPAD('--',54,' ')||'TOTAL   :'||RPAD(to_char(var_Total_Courses,'999.99'),13,' ')||'
'||RPAD(to_char(var_Total_Students,'999.99'),14,' ')|| RPAD(to_char(var_Total_Fees,'$999,999.99'),14,' '));
    DBMS_OUTPUT.PUT_LINE(RPAD('--',54,' ')||'AVERAGE :  '||
                         RPAD(var_avg_Course,13,' ') ||' '||RPAD(var_avg_Student,12,' ')||
                         RPAD(to_char(var_avg_ProgrammeFee,'$999,999.99'),14,' '));
    DBMS_OUTPUT.PUT_LINE(RPAD('--',58)||'     '||'---------- '||' -----------'||'    --------------');
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('Highest Number of Courses  : '||RPAD(var_Highest_Course,15,' ')||
                         'Programme Code: '||var_Highest_Course_Pro);
    DBMS_OUTPUT.PUT_LINE('Lowest Number of Courses   : '||RPAD(var_Lowest_Course,15,' ')||
                         'Programme Code: '||var_Lowest_Course_Pro);
    DBMS_OUTPUT.PUT_LINE('Highest Number of Students : '||RPAD(var_Highest_students,15,' ')||
```

```
                        'Programme Code: '||var_Highest_Student_Pro);
    DBMS_OUTPUT.PUT_LINE('Lowest Number of Students  : '||RPAD(var_Lowest_students,15,' ')||
                        'Programme Code: '||var_Lowest_Student_Pro);
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE(RPAD('--',45)||'End of Report');
END;
/
```

Sample Output:

```
--                        Total Programme Course And Students Summary Report

Printed Date: 02-01-2020                                                    Page: 1


=============================================================================================================
NO Program Code Program Name                                  Total Courses  Total Students  Programme Fee
=============================================================================================================
01  RSF          Bachelor in Software Engineering                    45            35           $31,575.00
02  RIS          Bachelor in Information Security                    45            40           $31,350.00
03  RIT          Bachelor in Internet Technology                     45            32           $30,525.00
04  REI          Bachelor in Interactive Software Technology         45            33           $30,525.00
05  RSD          Bachelor in Software System Development             44            31           $31,425.00
06  DSE          Diploma in Software Engineering                     45            31           $31,725.00
07  DIT          Diploma in Internet Technology                      45            73           $31,350.00
08  DST          Diploma in Interactive Software Technology          45            49           $30,525.00
--                                                                 ----------  -----------    --------------
--                                                  TOTAL    : 359.00        324.00         $249,000.00
--                                                  AVERAGE :  44.88         40.5           $31,125.00
--                                                             ----------  -----------    --------------
```

```
Highest Number of Courses  : 45            Programme Code: RSF
Lowest Number of Courses   : 44            Programme Code: RSD
Highest Number of Students : 73            Programme Code: DIT
Lowest Number of Students  : 31            Programme Code: DSE

--                                          End of Report
```

3.2.10. Report 3: Details Report of Program Details in every Semester

Purpose: The purpose of this report is to display the number of courses, total credits hours, semester fees in every semester for a particular inputted program. The user then can use this detail report to know which semester has the least number of courses or which semester fee is too expensive and thence, to some appropriate actions such as managing course numbers for each semester.

PL/SQL Code:

```
CREATE OR REPLACE PROCEDURE proc_details_report(In_Programme IN VARCHAR)AS

var_course_num        NUMBER(2);
var_course_fee        NUMBER(7,2);
var_credits           NUMBER(3);
var_total_course      NUMBER(3):=0;
var_total_credits     NUMBER(3):=0;
var_total_fees        NUMBER(8,2):=0;

CURSOR programme_cursor IS
    SELECT UNIQUE SEMESTER
    FROM ProgrammeCourse
    ORDER BY 1;

programme_rec programme_cursor%ROWTYPE;
```

```
BEGIN
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE(RPAD('--',11,' ')||'Programme Semester Detail Report');
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE(RPAD('Printed Date: '||To_Char(sysdate,'dd-mm-yyyy'),51,' ')||'Page: 1');
    DBMS_OUTPUT.PUT_LINE('Programme Code: '||In_Programme);
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE('============================================================');
     DBMS_OUTPUT.PUT_LINE('NO '||' Semester  '||RPAD(' Course Number',17,' ')||
                          RPAD('Total Credits',20,' ')||RPAD('Fee',13,' '));
    DBMS_OUTPUT.PUT_LINE('============================================================');

    FOR programme_rec IN programme_cursor LOOP
        SELECT COUNT(PC.CourseCode), SUM(C.FeePerCH*C.CreditHour),SUM(C.CreditHour)
               INTO var_course_num, var_course_fee,var_credits
        FROM ProgrammeCourse PC, Course C
        WHERE PC.ProgrammeCode = In_Programme
        AND PC.Semester = programme_rec.semester
        AND PC.CourseCode = C.CourseCode;

        DBMS_OUTPUT.PUT_LINE(to_char(programme_cursor%rowcount,'00')||
```

```
                              '   '||RPAD(programme_rec.Semester,15,' ')||
                              RPAD(var_course_num,15,' ')||RPAD(var_credits,12,' ')||
                              RPAD(to_char(var_course_fee,'$99,999.00'),14,' '));

        var_total_course := var_total_course + var_course_num;
        var_total_credits:= var_total_credits + var_credits;
        var_total_fees   := var_total_fees + var_course_fee;

    END LOOP;
    DBMS_OUTPUT.PUT_LINE(RPAD('--',18,' ')||RPAD('------',15,' ')||RPAD('------',13,' ')||
                           '------------');
    DBMS_OUTPUT.PUT_LINE(RPAD('--',11,' ')||'Total : '||
                         RPAD(var_total_course,15,' ')||RPAD(var_total_credits,12,' ')||
                         to_char(var_total_fees,'$999,999.00'));
    DBMS_OUTPUT.PUT_LINE(RPAD('--',18,' ')||RPAD('------',15,' ')||
                         RPAD('------',13,' ')||'------------');
    DBMS_OUTPUT.PUT_LINE(chr(10));
    DBMS_OUTPUT.PUT_LINE(RPAD('--',20,' ')||'End of Report');
END;
/
```

Sample Output:

```
--         Programme Semester Detail Report


Printed Date: 02-01-2020                          Page: 1
Programme Code: DST


=============================================================
NO  Semester   Course Number   Total Credits       Fee
=============================================================
01   Y1S1           6              17           $3,975.00
02   Y1S2           6              17           $3,600.00
03   Y1S3           3              8            $1,800.00
04   Y2S1           6              17           $3,750.00
05   Y2S2           6              17           $4,350.00
06   Y2S3           3              8            $2,100.00
07   Y3S1           6              17           $4,350.00
08   Y3S2           6              17           $4,350.00
09   Y3S3           3              9            $2,250.00
--                ------         ------       ------------
--         Total : 45             127          $30,525.00
--                ------         ------       ------------


--                 End of Report
```

3.3. Cheng Chia Shun

3.3.1. Query 1: Display top 3 average total highest mark of student within two semester.

Purpose: The purpose of this query is to display the top 3 highest mark to the college know these three student are good in their exam result then the college may give some award for these student or use these data for the enrollment new student.

SQL Statement:

```
select * from(select
l.learnerid,l.learnername,sum(a.studentmark/32)as highestmark

from learner l,learnerassessment a

where l.learnerid=a.learnerid

    group by l.learnerid,l.learnername

    order by 3 desc

    )

  where rownum<=3;
```

Sample output:

```
LEARNER LEARNERNAME          HIGHESTMARK

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

1901016 Cheng Chia Fen            92.625

1701012 Ang Ye Shan               92.625

1701045 Tan Jun Li                92.125
```

3.3.2. Query 2: Display topic that related with software.

Purpose: The purpose of this query is to display the topic which is hot discuss during 2017-2019, then the college can know for these data to improve such as adding more courses that relate to the topic which is hot discuss at their computer faculty.

SQL statement:

```
select s.staffname,s.stafftitle,t.topicdate,t.title
   from staff s,assignstaff a,topic t
    where s.staffnumber=a.staffnumber and
    a.coursesemid=t.coursesemid
    and t.title like '%Software%';
```

Sample output:

```
STAFFNAME                                 STAFFTITLE           TOPICDATE TITLE
----------------------------------------- -------------------- --------- ----------------------------------------
------------
Ts. Dr Yu Yong Poh                        Programme Leader     10-JAN-18 Software Design and Architecture
Dr Goh Ching Pang                         Senior Lecturer      14-OCT-18 Software Engineering
Dr Poh Tze Ven                            Senior Lecturer      11-JAN-18 Software Evolution and Maintenance
Ts. Dr Tew Yiqi                           Senior Lecturer      14-DEC-18 Software Requirements Engineering
Puan Anurehka A/p Magheswaran             Lecturer             01-JAN-19 Software Security
Puan Anurehka A/p Magheswaran             Lecturer             14-NOV-18 Software Project Management
Puan Lee Shu Gyan                         Programme Leader     12-JAN-18 Software Quality Assurance and Testing
```

3.3.3. Query 3: Display relate details of specify staff

Purpose: The purpose of this query is to display the detaisl such as course and course type that staff is teaching.

SQL statement:

```
select c.coursename,a.coursesemtype,s.staffname
    from course c,coursesem a,assignstaff b,staff s
    where c.coursecode=a.coursecode
    and a.coursesemid=b.coursesemid and
    b.staffnumber=s.staffnumber and
    s.staffname='Encik Ong Jia Hui';
```

Sample output:

```
COURSENAME                          COURSESEM            STAFFNAME
----------------------------------- --------------       ----------------

Introduction to Computer Security     Tutorial        Encik Ong Jia Hui

Introduction to Computer Systems          Tutorial         Encik Ong Jia Hui

Multimedia Development for the Web        Lecture         Encik Ong Jia Hui

Network Security                           Lecture         Encik Ong Jia Hui

Web Design and Development                Practical        Encik Ong Jia Hui
```

Web Engineering	Practical	Encik Ong Jia Hui
ObjectOriented Programming Techniques	Practical	Encik Ong Jia Hui
ObjectOriented Programming Techniques	Tutorial	Encik Ong Jia Hui
ObjectOriented Programming Techniques	Lecture	Encik Ong Jia Hui
Web Design and Development	Tutorial	Encik Ong Jia Hui
WebBased Multimedia Applications	Tutorial	Encik Ong Jia Hui

3.3.4. Procedure 1: Validate of the data type of learner data execute

Purpose: The purpose of this procedure to validate whether the data insert to the table learner is correct or wrong.

SQL statement:

```
create or replace procedure programme_validate(p_id varchar,p_name
varchar,p_icno varchar,p_contact varchar,

    p_dob date,p_email varchar,p_address varchar,p_state
varchar,p_gender varchar)

    is is_valid int ;

    begin

    select count(8) into is_valid from learner

    where LearnerID=p_ID and

          learnername=p_name and

          icno=p_icno and

          contact=p_contact and

         dob=p_dob and

         email=p_email and

         address=P_address and

         state=p_state and

         gender=p_gender;

   if is_valid =0 then

   raise_application_error(-20000,'Invalid data type enter');

   else

   DBMS_OUTPUT.PUT_LINE('Correct data type insert');

   end if;

   end;

   /
```

Sample output:

```
SQL> exec programme_validate(1701001,'Tan Bai Ying','951127-09-
1751','018-6241311','27-Dec-95','tanby@student.tarc.edu.my','No.96,
Lorong Kembang Lama, Taman Kembang Lama','Penang','F')

Correct data type insert
PL/SQL procedure successfully completed.

SQL> exec programme_validate(1701001,'Tan Bai Ying','951127-09-
1751','018-6241311','27-Dec-95','tanby@student.tarc.edu.my','No.96,
Lorong Kembang Lama, Taman Kembang Lama','Penang',1)

BEGIN programme_validate(1701001,'Tan Bai Ying','951127-09-
1751','018-6241311','27-Dec-95','tanby@student.tarc.edu.my','No.96,
Lorong Kembang Lama, Taman Kembang Lama','Penang',1); END;

*
ERROR at line 1:
ORA-20000: Invalid data type enter
ORA-06512: at "SHUN0922.PROGRAMME_VALIDATE", line 16
ORA-06512: at line 1
```

3.3.5. Procedure 2: Update staff title to retired

Purpose: The purpose of this procedure is update the staff title when the staff is retired.

SQL statement:

```
create or replace procedure prc_staff_retire(v_staffname in varchar)

is

     v_stafftitle varchar2(20);

   begin

       Select stafftitle into v_stafftitle

       from staff

       where staffname=v_staffname;

       v_stafftitle :='retired';

      update staff

      set stafftitle=v_stafftitle

      where upper('staffname')=upper('v_staffname');

   DBMS_OUTPUT.PUT_LINE(RPAD('Staff Name',25)|| 'Stafftitle');

   DBMS_OUTPUT.PUT_LINE('====================== '||
'===============');

   DBMS_OUTPUT.PUT_LINE(RPAD(v_staffname,25)||v_stafftitle);

   end;

   /
```

Sample output:

```
exec prc_staff_retire('Ts. Lim Mei Shyan');
Staff Name               Stafftitle
======================== ===============
Ts. Lim Mei Shyan        retired
```

3.3.6. Trigger 1: Check student whether exist before insert

Purpose: The purpose of this trigger is check whether the student is exist, if the student is exist then the trigger will allow the user insert value.

SQL statement:

```
create or replace trigger check_student before insert on
learnerassessment

    for each row

    declare

           v_learnerid varchar(7);

    begin

    select learnerid into v_learnerid

    from learner

    where learnerid = :new.learnerid;

   DBMS_OUTPUT.PUT_LINE('The learner is exist');


   exception

     when no_data_found

     then RAISE_APPLICATION_ERROR(-20200,'The learner is not exist');


   end;

   /
```

Sample output:

```
 INSERT INTO LearnerAssessment VALUES
(10866,'1902049',66.0,'B','Pass');

The learner is exist
```

3.3.7. Trigger 2: Update assessment status

Purpose: The purpose of this trigger is update assessment status during the user update the learner assessment mark.

SQL statement:

```
create or replace trigger upd_ass

   before update of studentmark on learnerassessment

   for each row

   declare

   begin

   if (:new.studentmark>49) then

        :new.status :='Pass';

  else

        :new.status:='Fail';

        :new.grade:='F';

  end if;

  end;

  /
```

Sample output:

```
update learnerassessment

   set studentmark=48

   where assessmentid=10007 and learnerid='1701001';
```

ASSESSMENTID	LEARNER	STUDENTMARK	GR	STAT
10007	1701001	48	F	Fail
10008	1701001	30	C+	Fail

3.3.8. Report 1: Detail Report of topic detail

Purpose: The purpose of this report is display all detail of the topic that related with the staff.

SQL STATEMENT:

```
create or replace procedure summary_topic(v_topic in number)

   as cursor prod_cursor is

         select * from topic

         where topicid=v_topic;

      prod_rec prod_cursor%rowtype;

      v_staff topic.staffnumber%type;

   begin

      Select staffnumber into v_staff

      from topic

      where topicid =v_topic;

      DBMS_OUTPUT.PUT_LINE('-------------------Detail Report of
Topic--------------------');



   DBMS_OUTPUT.PUT_LINE('Topic ID : '||v_topic);

   DBMS_OUTPUT.PUT_LINE('Staff ID:'||v_staff);

   DBMS_OUTPUT.PUT_LINE(RPAD('No',5)||RPAD('Course semester
ID',25)||RPAD('Title',15)||

    RPAD('Topic Detail',40)||'Date');

         DBMS_OUTPUT.PUT_LINE('==== '||'=================='||'
=================='||

  '  ==================================='||'  ==========');

          FOR Prod_rec IN Prod_cursor LOOP
DBMS_OUTPUT.PUT_LINE(RPAD(TO_CHAR(prod_cursor%rowcount),5)||RPAD(Prod
_rec.coursesemid,20)
||RPAD(Prod_rec.title,25)||RPAD(Prod_rec.topicdetails,30)
 ||to_date(Prod_rec.topicdate,'dd/mm/yyyy'));
```

```
   end loop;

   end;  /
```

Sample output:

```
SQL> exec summary_topic('100')
-------------------Detail Report of Topic--------------------
Topic ID : 100
Staff ID:90001
No   Course semester ID     Title        Topic Detail       Date
==== ==================  ============    ============      ==========
1   100001         Computer Network    Discuss of Network 05-JAN-17
```

3.3.9. Report 2: Summary Report about assessment

Purpose: The purpose of this report is to display what requirement of the assessment have if the student wants to pass and what type of assessment the college have.

SQL Statement:

```
 create or replace procedure summary_assessment(v_assessmentid in
number)

   as cursor prod_cursor is

           select * from assessment

           where assessmentid=v_assessmentid;

           prod_rec prod_cursor%rowtype;



    begin

         DBMS_OUTPUT.PUT_LINE('-------------------Summary Report of
Assessment--------------------');

         DBMS_OUTPUT.PUT_LINE('Assessment ID:'||v_assessmentid);

   FOR Prod_rec IN Prod_cursor LOOP

         DBMS_OUTPUT.PUT_LINE('The total mark of all assessment is
'||(prod_rec.totalmark));

         DBMS_OUTPUT.PUT_LINE('The all student must above
'||prod_rec.passingmark||' then only can pass');

         DBMS_OUTPUT.PUT_LINE('There have three assessment type it is
assigment,midterm and partical');

   end loop;

   end;

   /
```

Sample output:

```
SQL> exec summary_assessment(10001);
-------------------Summary Report of Assessment--------------------
Assessment ID:10001
The total mark of all assessment is 100
The all student must above 50 then only can pass
There have three assessment type it is assigment,midterm and partical
```

3.3.10. Report 3: On demand report about user update passing mark

Purpose: The purpose of this report is to let the user can update the passing mark due to some institute change of mark.The original mark of passing is 50.

SQL Statement:

```
create or replace procedure dmd_mark(v_assessmentid in number,v_mark
in number)

   as  cursor prod_cursor is

          select * from assessment

          where assessmentid=v_assessmentid;

       prod_rec prod_cursor%rowtype;

    begin

      DBMS_OUTPUT.PUT_LINE('--------------------On Demand Report of
Passing Mark--------------------');

      DBMS_OUTPUT.PUT_LINE(RPAD('No',5)||'Assessment ID'||' Original
Passing Mark');

      DBMS_OUTPUT.PUT_LINE('=== '||'============='||'
====================');



   for prod_rec in prod_cursor loop


DBMS_OUTPUT.PUT_LINE(RPAD(TO_CHAR(prod_cursor%rowcount),5)||RPAD(v_as
sessmentid,15)||

            (prod_rec.passingmark));

       prod_rec.passingmark:=v_mark;

      DBMS_OUTPUT.PUT_LINE(RPAD('No',5)||'Assessment ID'||' Updated
Passing Mark');

      DBMS_OUTPUT.PUT_LINE('=== '||'============='||'
====================');

        DBMS_OUTPUT.PUT_LINE(RPAD(TO_CHAR(prod_cursor%rowcount),5)||

   RPAD(v_assessmentid,15)||prod_rec.passingmark);

   end loop;

   end;/
```

Sample output:

```
exec dmd_mark(10001,10);
------------------On Demand Report of Passing Mark------------------
No   Assessment ID Original Passing Mark
=== ============= ====================
1    10001          50
No   Assessment ID Updated Passing Mark
=== ============= ====================
1    10001          10
```

Chapter 4: Personal Reflection Report

4.1. Tan Kuan Tiong

First of all, I would like to appreciate my tutor and my teammates as they always willing to give me constructive support and advices that sufficient enough to help me complete this assignment, without them, this assignment might not reach such completeness and insightful as should be achieved.

During this assignment, I have met a few obstacles, such as: ERD Data Model, DDL, DQL DML and report design, due to my lack of experience and underwhelming knowledge about the database design. Hence, my team mates and I have had a few group meetings for sharing of thoughts and understandings among each other. The outcome was astonishing and appealing. We were able to establish strong rapport with each other and resolve the conflict peacefully and unite together to complete the ERD data model, table creation and data insertion and yet improve our productivity in this assignment.

However, due to the overwhelming workload from other assignment, I was not able to prepare myself completely and explore myself deeper into the knowledge sea of database. It is a pity that I have not fully utilize all the knowledge that I learnt from this course into the assignment. For the next time, I will try to improve my time managing skill and strive to utilize all of my knowledge where appropriate and produce a better assignment.

4.2. Tee Yu June

Firstly, I would like to express my sincerely appreciation by giving a big respect and shout out to our practical tutor. He's not only giving us a very interesting advanced database lesson in practical and lecture class, but he also 'discipline' us to be more self-dependent on learning. Even though he is strict in teaching, but we learnt and grow a lot throughout this semester. Therefore, I have applied all the lessons that I've learnt during the class and it turn out a fruitful outcome especially for my assignment.

When I was doing my assignment, there are definitely lots of frustrating problem that keep stopping me especially drawing ERD diagram which has to be in 3rd normal because of lacking skills and knowledge in this subject. Luckily, I was able to seek the help from my friends especially my group assignment members who always guiding me and correcting me when I made mistakes. After hours of hard-work and night-to-night discussion, we are able to produce a well ERD diagram and proceed to the next phase of our assignments.

I was a little bit disappointed because I was not able to perform 100% in this course because it required determination and hard-work in learning. Even though this subject is a challenge for me, but I will not give up easily due to my passion to dive deep in this interesting subject.

4.3. Cheng Chia Shun

Firstly,I would like to thank my tutor that teaching me how to create database coding.He is not only teaching us the lesson of database,he teaching us also the responsible a student must have.I am remember deeply as I don't know what coding I am doing,he will fail me.So that I am hardworking to do my assignment but not really well.I also have learning from this database assignment are query,trigger and procedures.During I doing this assignment I have face problem about how to create triggers,but luckily my teammate teach me how to solve the problem.

When we create the data,I found out this assignment must work together with my teammate,because each data is related with each other.If my task does not done,then my teammates may waiting for me until I done.

In my future,I hope I will explore more database coding and knowledge to explain my future life.Because database is important to any company,if have not database to store business data,a company cannot operate enduring and cannot earn more money due does not have customer data,product data and so on.

Chapter 5: Extra Effort

5.1. Group Effort

5.1.1. Sequences

Name: RegisterID_Seq
Purpose: Sequence for primary key of RegisterCourse table.

```
CREATE SEQUENCE RegisterID_Seq
MINVALUE 2713
MAXVALUE 99999
START WITH 2713
INCREMENT BY 1
NOCACHE;
```

Name: StaffNo_Seq
Purpose: Sequence for primary key of Staff table.

```
CREATE SEQUENCE StaffNo_Seq
MINVALUE 90081
MAXVALUE 99999
START WITH 90081
INCREMENT BY 1
NOCACHE;
```

Name: CourseSem_Seq
Purpose: Sequence for primary key of coursesem table.

```
CREATE SEQUENCE CourseSem_Seq
MINVALUE 100000
MAXVALUE 999999
START WITH 100001
INCREMENT BY 1;
```

Name: Announcement_Seq
Purpose: Sequence for primary key of Announcement table.

```
CREATE SEQUENCE Announcement_Seq
MINVALUE 1034
MAXVALUE 9999
START WITH 1034
INCREMENT BY 1
NOCACHE;
```

Name: TopicID_Seq
Purpose: Sequence for primary key of Topic table.

```
CREATE SEQUENCE TopicID_Seq
MINVALUE 151
MAXVALUE 999
START WITH 151
INCREMENT BY 1
NOCACHE;
```

Name: AssessmentID_Seq
Purpose: Sequence for primary key of AssessmentID table.

```
CREATE SEQUENCE AssessmentID_Seq
MINVALUE 10953
MAXVALUE 19999
START WITH 10953
INCREMENT BY 1
NOCACHE;
```

5.2. Individual Effort

5.2.1. Tan Kuan Tiong

5.2.1.1. Views

View : **View on Learner's Course Registration That Pending for Validation**

Purpose : This view is to display the learner course registration details that is paid by online banking method and is still pending for validation for current ongoing semester. So that the admin staff can manually validate the learner course registration detail.

SQL Statement :

```
CREATE OR REPLACE VIEW pend_reg_OB(PaymentMethod, RegisterDate, PaymentDate, RegisterID, LearnerID, ProgrammeCode,
GroupID, CourseCode) AS

   SELECT RC.PaymentMethod, RC.RegisterDate, RC.PaymentDate, RC.RegisterID, RC.LearnerID,
          LP.ProgrammeCode, LP.GroupID, PC.CourseCode

   FROM RegisterCourse RC,

        (SELECT LearnerID, ProgrammeCode, GroupID

         FROM LearnerProgramme) LP,

        (SELECT ProgrammeCode, CourseCode

         FROM ProgrammeCourse

         WHERE Semester = 'Y3S2') PC

   WHERE RC.LearnerID = LP.LearnerID AND LP.ProgrammeCode = PC.ProgrammeCode AND RC.SemesterID = 201909 AND
         RC.RegisterStatus = 'Pending' AND RC.PaymentDate IS NOT NULL AND RC.PaymentMethod = 'Online Banking'

   GROUP BY RC.PaymentMethod, RC.RegisterDate, RC.PaymentDate, RC.RegisterID, RC.LearnerID,
            LP.ProgrammeCode, LP.GroupID, PC.CourseCode

   ORDER BY 1, 2, 3, 4, 5;
```

Sample Output :

```
            Pending Registration For Students In Semester 201909
                           With Online Banking
                                                                    Page :  1
 Payment        Register  Payment   Register Learner Prog  Group Course
 Method         Date      Date            ID ID      Code     ID Code
 -------------- --------- --------- -------- ------- ----- ----- --------
 Online Banking 14-OCT-19 09-NOV-19     2452 1702002 DSE       1 AACS1074
                                                                  AACS1084
                                                                  AACS1143
                                                                  AAMS1613
                                                                  AAMS2613
                                                                  BHEL1023

                                        2632 1901032 REI       2 BAIT1023
                                                                  BAIT1083
                                                                  BAIT2113
                                                                  BAIT2133
                                                                  BAIT2173
                                                                  BHEL2023

                                        2638 1901038 REI       1 BAIT1023
                                                                  BAIT1083
                                                                  BAIT2113
                                                                  BAIT2133
                                                                  BAIT2173
                                                                  BHEL2023
 -------------- --------- --------- -------- ------- ----- ----- --------
Total Number Of Student:                          3
```

5.2.1.2. User Defined Functions

Function 1

Name : cal_weekly_class_hr

Purpose: Return the teaching hours teached by academic staff for each class based on its class type, where lecture = 6 hours, tutorial = 4 hours, practical = 8 hours and others = 2 hours.

SQL Statement:

```
CREATE OR REPLACE FUNCTION cal_weekly_class_hr(v_csType in varchar)
  RETURN NUMBER
  IS weekly_hr NUMBER;

  BEGIN
    weekly_hr := 0;

    CASE v_csType
    WHEN 'Lecture' THEN
        weekly_hr := 6;
    WHEN 'Tutorial' THEN
        weekly_hr := 4;
    WHEN 'Practical' THEN
        weekly_hr := 8;
    ELSE
        weekly_hr := 2;
    END CASE;
    RETURN weekly_hr;
  END;
/
```

Function 2

Name : cal_staff_by_dprt

Purpose: Return the number of academic staff in one department

SQL Statement:

```
CREATE OR REPLACE FUNCTION cal_staff_by_dprt(v_dprt in varchar)
  RETURN NUMBER
  IS staff_no NUMBER;

  BEGIN
    SELECT COUNT(StaffNumber) INTO staff_no
    FROM Staff
    WHERE Department = v_dprt AND StaffTitle NOT IN ('Dean','Deputy
Dean','Associate Dean');
    RETURN (staff_no);
  END;
/
```

Function 3

Name : get_pay_warn_date

Purpose: Return the payment warning date based on the semester parameter which is 3 weeks after the semester date

SQL Statement:

```
CREATE OR REPLACE FUNCTION get_pay_warn_date(v_SemID IN NUMBER)
   RETURN Date
   IS WarnDate Date;
 BEGIN
   SELECT StartDate + 21 INTO WarnDate
   FROM Semester
   WHERE SemesterID = v_SemID;
   RETURN(WarnDate);
 END;
/
```

Function 4

Name : get_pay_rjct_date

Purpose: Return the payment rejection date based on the semester parameter which is 60 days after the semester date

SQL Statement:

```
CREATE OR REPLACE FUNCTION get_pay_rjct_date(v_SemID IN NUMBER)
   RETURN Date
   IS RjctDate Date;
 BEGIN
   SELECT StartDate + 60 INTO RjctDate
   FROM Semester
   WHERE SemesterID = v_SemID;
   RETURN(RjctDate);
 END;
/
```

Function 5

Name : get_last_reg_date

Purpose: Return the last register date based on the semester parameter which is 14 days after the semester start date

SQL Statement:

```
CREATE OR REPLACE FUNCTION get_last_reg_date(v_SemID IN NUMBER)
   RETURN Date
   IS RegDate Date;
 BEGIN
   SELECT StartDate + 14 INTO RegDate
   FROM Semester
   WHERE SemesterID = v_SemID;
   RETURN(RegDate);
 END;
/
```

5.2.2. Tee Yu June

5.2.2.1. Views

Views on Total Number of student's enrolment in each program

Purpose: Use to generate data values which can then be used to comapre the value inside a single table. For example, compare the number of students enrolments in each years which all the students enrolments infdormation such as time intake is inside a single table.

Staff then can take appropriate actions on those program with low students enrolment.

Sample Code:

```
CREATE OR REPLACE VIEW VIEW_LearnerProgramme AS
    SELECT ProgrammeCode,EnrolmentDate,COUNT(LearnerID) AS StudNUm
    FROM LearnerProgramme
    GROUP BY ProgrammeCode,EnrolmentDate;
```

5.2.2.2. User Defined Functions

Name: Is_Number

Purpose: Used for checking whether an string input is a numeric number or alphabetic value. If it is numeric, convert it to number and return 0, else return 0 for error.

SQL Statement:

```
CREATE OR REPLACE FUNCTION is_number (p_string IN VARCHAR2)
RETURN INT
IS
  v_num NUMBER;
BEGIN
  v_num := TO_NUMBER(p_string);
  RETURN 1;
EXCEPTION
WHEN VALUE_ERROR THEN
  RETURN 0;
END is_number;
/
```

5.2.3. Cheng Chia Shun

5.2.3.1. View

Purpose: Use to view how many students and who are fail in which assessment and give the refer to university management thinking for decrease the number of students who fail.

SQL Statement:

```
create or replace view learner_assessment as

select *

from learnerassessment

where grade='F';


select status,studentmark,assessmentid

from learner_assessment;
```

5.2.3.2. User Defined Functions

Name: goodbye_msg

Purpose: Used for display goodbye message that include a user name.

SQL Statement:

```
create or replace function goodbye_msg(p_name in varchar)

return varchar

is

begin

return('goodbye '||p_name||' see you next time');

end;

/
```