LEARNING MANAGEMENT SYSTEM (LMS)

Table of Contents

01. Introduction

Managing a university's academic environment involves coordinating multiple elements like handling lessons, managing course schedules, tracking student assessments, and monitoring submissions. To ensure smooth operation and to provide an efficient experience for both students and lecturers, a comprehensive system is needed to organize all this information. This is where a Learning Management System (LMS) comes into play.

This report aims to describe the design and functionality of a dedicated LMS for university use. Think of it as a digital hub where we can store, manage, and organize everything related to academic activities. The system helps track lessons, student progress, lecturer input, and administrative actions in an organized and accessible manner.

We will guide you through the design process of this LMS step-by-step in simple terms. First, we'll talk about the main actors involved, such as the Admin, Lecturer, and Student. We'll also discuss the key parts of the LMS that need to be managed, including tables for Courses, Lessons, Schedules, Assessments, Submissions, and Course Enrollments.

By the end of this report, you'll gain a clearer understanding of how our LMS efficiently supports university operations, keeps the academic process organized, and allows students, lecturers, and administrators to work seamlessly within the system.

02.Database Design

2.1 Logical Design

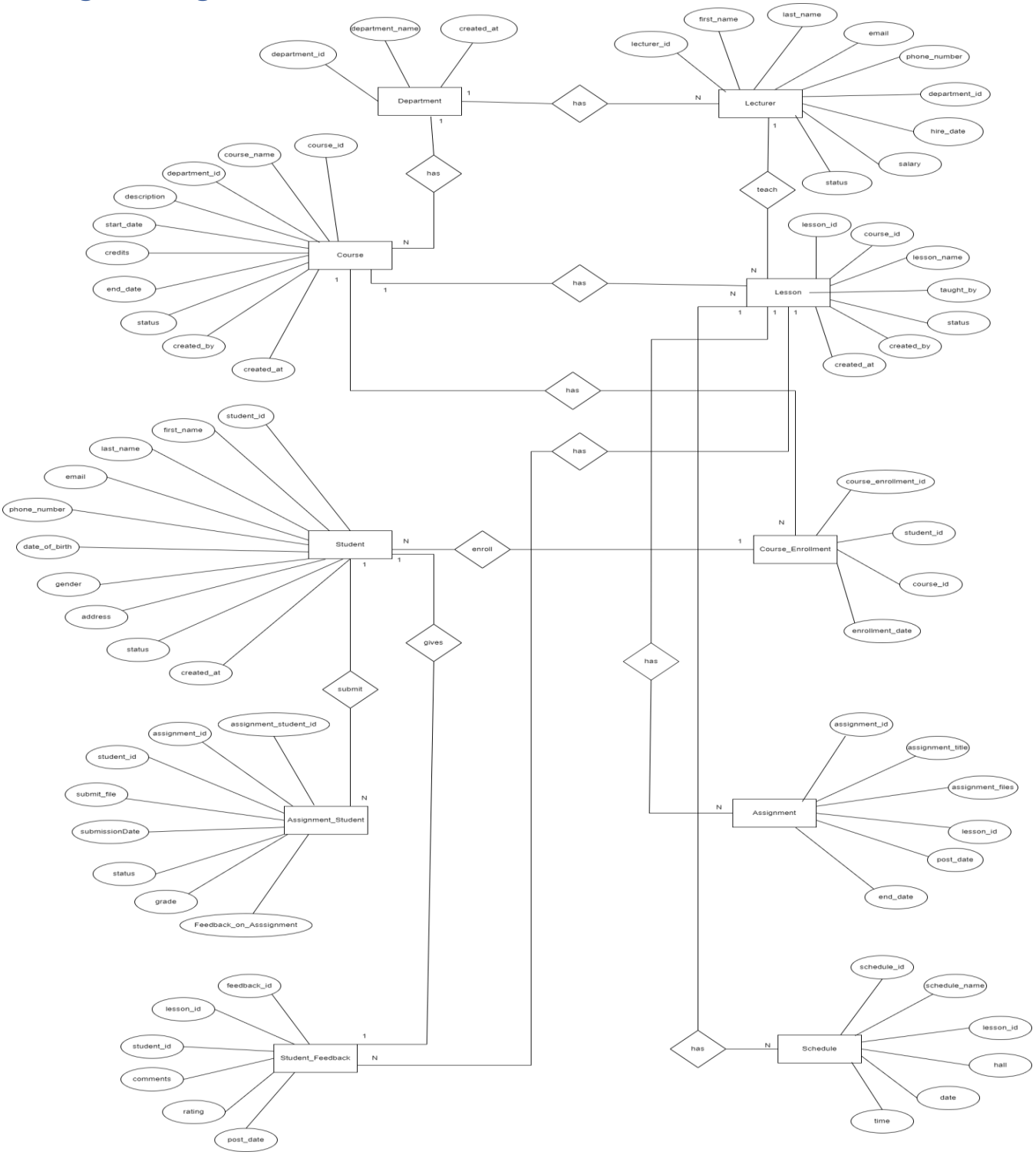


Figure 1

2.2 Physical Design

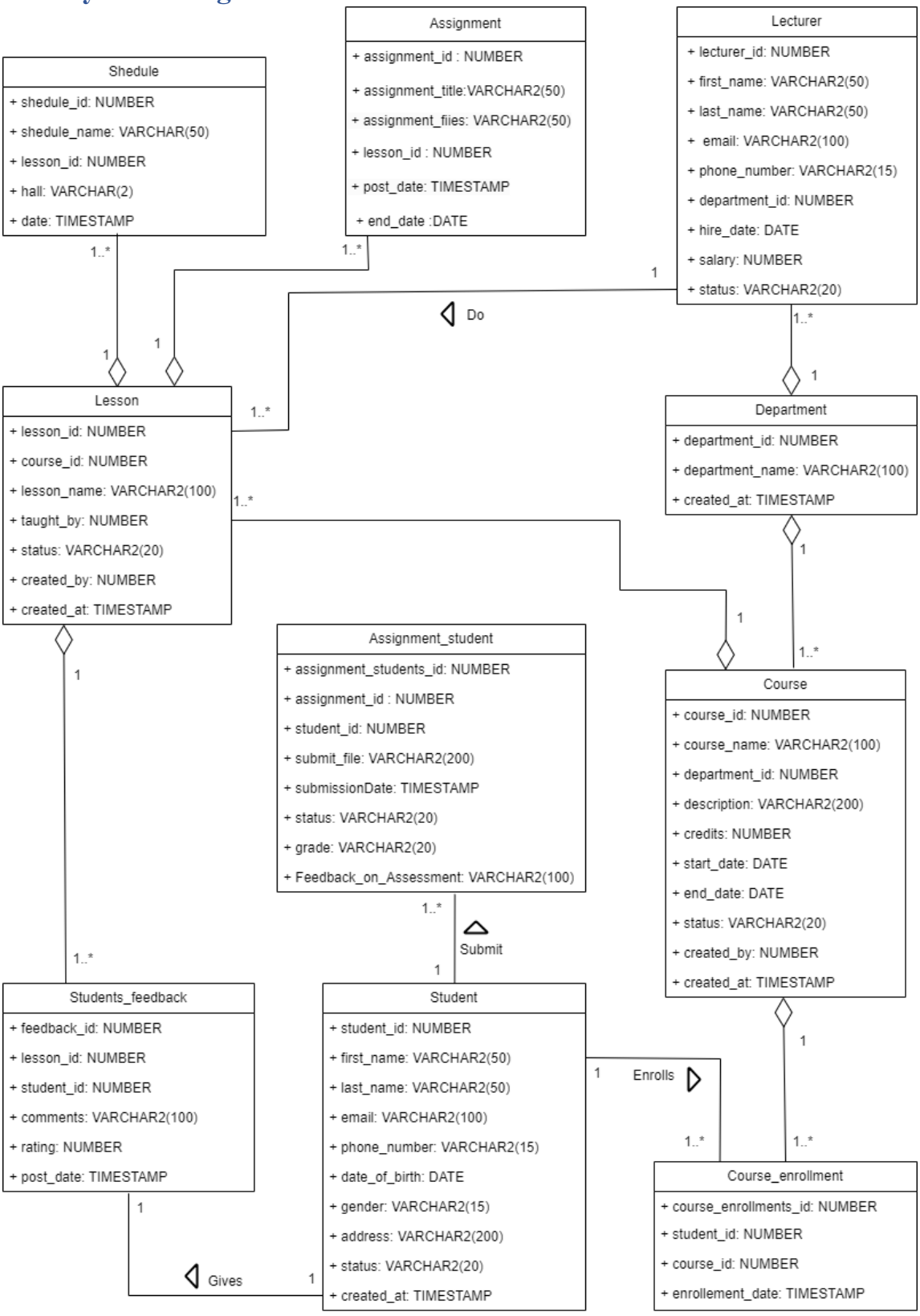


Figure 2

2.3 Normalization of Database Schema

2.3.1 First Normal Form (1NF)

All tables in the schema, including departments, lecturers, courses, lessons, students, course_enrollments, assignment, assignment_students, students_feedback, and schedule, adhere to 1NF. Each table has a primary key, and all columns contain atomic values. There are no repeating groups or arrays.

2.3.2 Second Normal Form (2NF)

All tables in the schema use a single-column primary key. Since no table has a composite primary key, they automatically satisfy 2NF.

Non-key attributes are fully functionally dependent on the primary key in each table. For example:

In the lecturers table, attributes like first_name, last_name, and email are fully dependent on lecturer_id.

In the courses table, attributes such as course_name, credits, and department_id depend entirely on course_id.

2.3.3 Third Normal Form (3NF)

Each table in the schema adheres to 3NF as all non-key columns depend solely on the primary key and not on other non-key columns.

For example:

In the students table, attributes like first_name, email, and phone_number depend directly on student_id.

In the courses table, department_id and created_by (foreign keys) reference departments and lecturers, respectively, but do not cause transitive dependencies.

Other tables, like lessons, course_enrollments, and assignment, similarly maintain only dependencies directly tied to their respective primary keys.

2.3.4 Boyce-Codd Normal Form (BCNF)

All tables in the schema adhere to BCNF as every determinant is a candidate key.

For example:

In the lecturers table, lecturer_id determines all other attributes, and it is the primary key.

In the courses table, course_id determines all other attributes, and it is the primary key.

No non-key attributes determine other non-key attributes, ensuring full adherence to BCNF.

03. LMS Database Setup and Operations

3.1 User and Role Creation for LMS in XE_System

3.1.1 Create Admin User in XE_System

```
CREATE USER ADMIN IDENTIFIED BY ADMIN123
DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp
QUOTA 20M ON users
ACCOUNT UNLOCK;
/
```

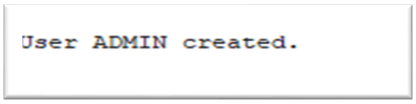


Figure 3.1.1

3.1.2 Grant All Privileges to Admin User

```
GRANT ALL PRIVILEGES TO ADMIN;
```

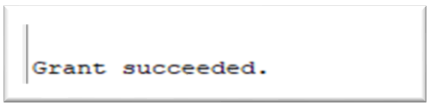


Figure 3.1.2

3.1.3 Create LECTURER User in XE_System

```
CREATE USER LECTURER IDENTIFIED BY LECTURER123
DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp
QUOTA 20M ON users
ACCOUNT UNLOCK;
/
```

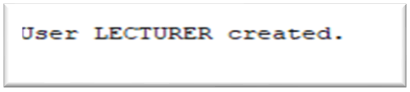


Figure 3.1.3

3.1.4 Create STUDENT User in XE_System

CREATE USER STUDENT IDENTIFIED BY STUDENT123

DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp

QUOTA 20M ON users

ACCOUNT UNLOCK;

/

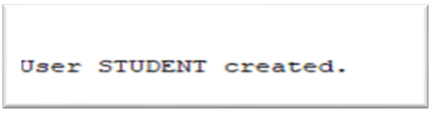


Figure 3.1.4

3.1.5 Create User Role

CREATE ROLE USER_ROLE;

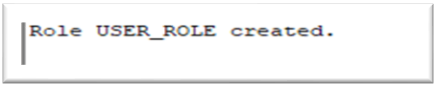


Figure 3.1.5

3.1.6 Grant User Role and Connect Privileges to Student and Lecturer Users

GRANT CONNECT TO USER_ROLE;

GRANT USER_ROLE TO STUDENT;

GRANT USER_ROLE TO LECTURER;

```
Grant succeeded.

Grant succeeded.

Grant succeeded.
```

Figure 3.1.6

3.1.7 Connect ADMIN

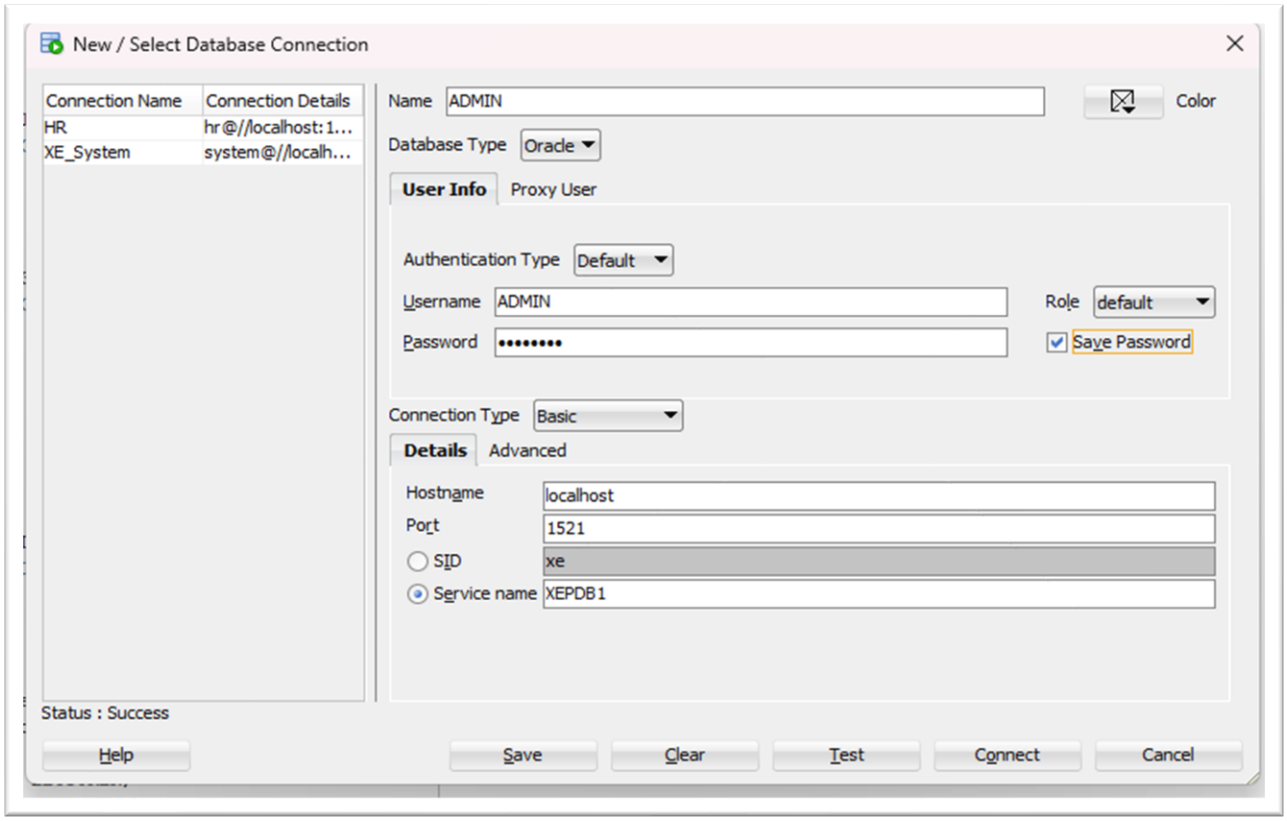


Figure 3.1.7

3.1.8 Connect LECTURER

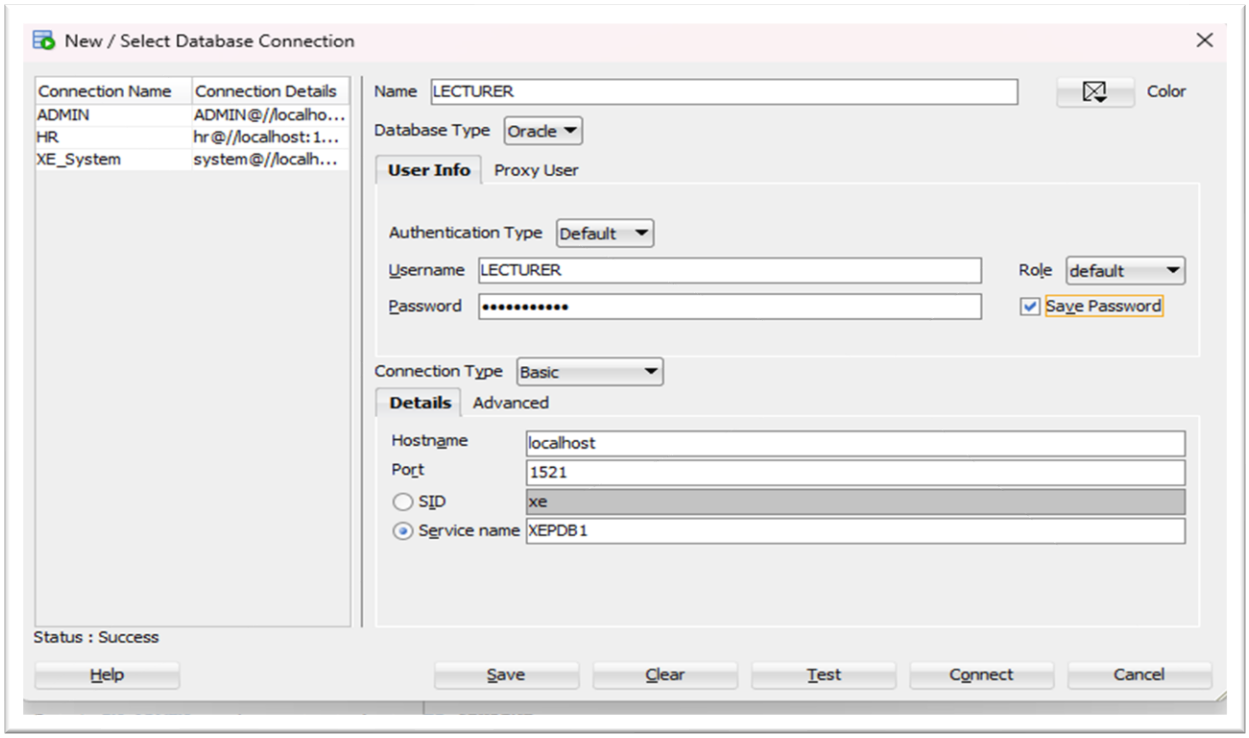


Figure 3.1.8

3.1.9 Connect STUDENT

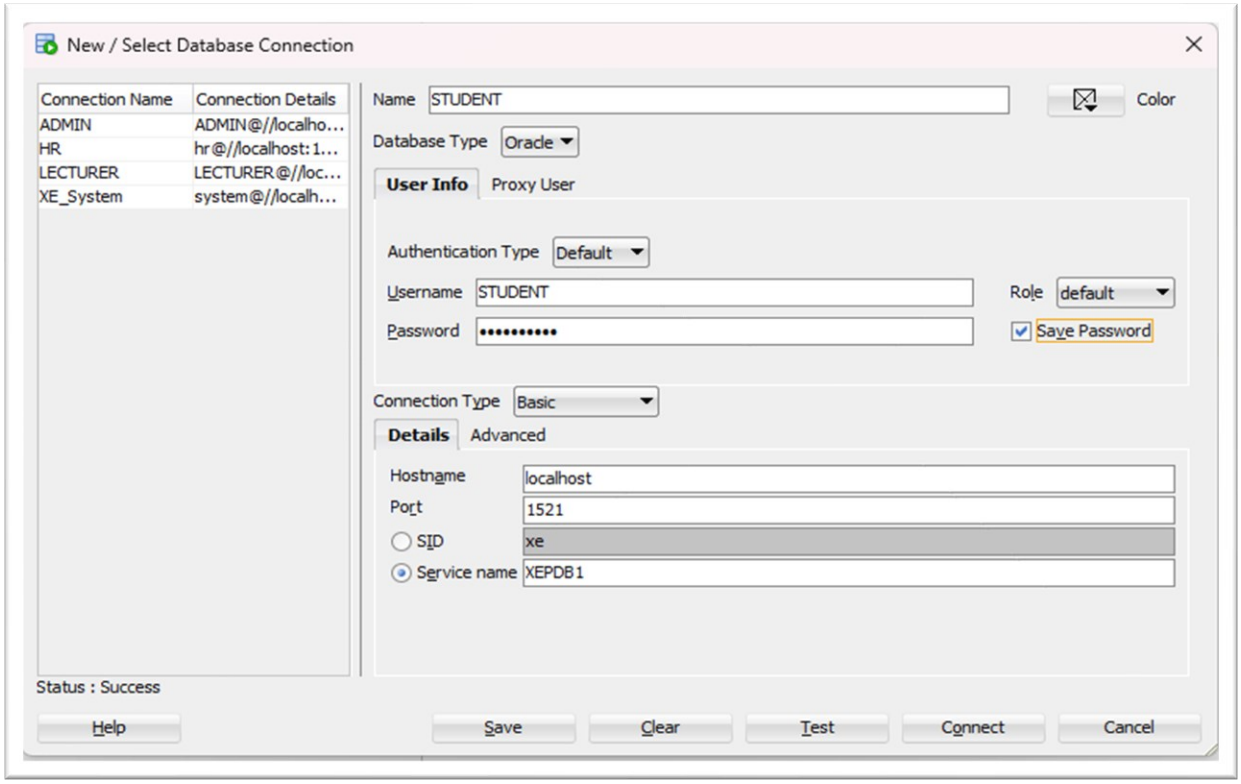


Figure 3.1.9

3.2 Database creation and Table creation

3.2.1 CREATE departments TABLE

```
CREATE TABLE departments (
   department_id NUMBER PRIMARY KEY,
   department_name VARCHAR2(100) NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Figure 3.2.1

3.2.2 Insert Records into departments Table

```
INSERT INTO departments (department_id, department_name) VALUES (1, 'School of Computing');
INSERT INTO departments (department_id, department_name) VALUES (2, 'School of Business');
INSERT INTO departments (department_id, department_name) VALUES (3, 'School of Engineering');
INSERT INTO departments (department_id, department_name) VALUES (4, 'School of Language');
INSERT INTO departments (department_id, department_name) VALUES (5, 'School of Humanities');
```

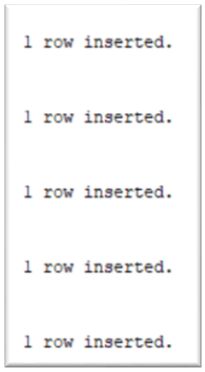


Figure 3.2.2

3.2.3 Retrieve All Records from departments Table

SELECT * FROM departments;

	DEPARTMENT_ID	DEPARTMENT_NAME	CREATED_AT
1	1	School of Computing	18-OCT-24 05.14.26.219000000 PM
2	2	School of Business	18-OCT-24 05.14.26.230000000 PM
3	3	School of Engineering	18-OCT-24 05.14.26.233000000 PM
4	4	School of Language	18-OCT-24 05.14.26.237000000 PM
5	5	School of Humanities	18-OCT-24 05.14.59.726000000 PM

Figure 3.2.3

3.2.4 CREATE lecturers TABLE

```
CREATE TABLE lecturers (
    lecturer_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50),
    email VARCHAR2(100) UNIQUE NOT NULL,
    phone_number VARCHAR2(15) NOT NULL,
    department_id NUMBER,
    hire_date DATE NOT NULL ,
    salary NUMBER CHECK (salary > 0),
    status VARCHAR2(20) DEFAULT 'Active' CHECK (status IN ('Active','Retired','Resigned')),
    FOREIGN KEY (department_id) REFERENCES departments (department_id)
);
```

```
Table LECTURERS created.
```

Figure 3.2.4

3.2.5 Insert Records into lecturers Table

INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id, hire_date, salary, status)

VALUES (1, 'Nimal', 'Perera', 'nimal.perera@schoolofcomputing.edu', '0711234567', 1, TO_DATE('2020-01-15', 'YYYY-MM-DD'), 80000, 'Active');

INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id, hire_date, salary, status)

VALUES (2, 'Saman', 'Silva', 'saman.silva@schoolofbusiness.edu', '0777654321', 2, TO_DATE('2018-03-10', 'YYYY-MM-DD'), 75000, 'Active');

INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id, hire_date, salary, status)

VALUES (3, 'Kamal', 'Fernando', 'kamal.fernando@schoolofengineering.edu', '0723456789', 3, TO_DATE('2019-08-25', 'YYYY-MM-DD'), 90000, 'Retired');

INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id, hire_date, salary, status)

VALUES (4, 'Sunil', 'Jayasinghe', 'sunil.jayasinghe@schooloflanguage.edu', '0709876543', 4, TO_DATE('2021-06-05', 'YYYY-MM-DD'), 60000, 'Active');

INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id, hire_date, salary, status)

VALUES (5, 'Ruwan', 'Dias', 'ruwan.dias@schoolofhumanities.edu', '0751237894', 5, TO_DATE('2017-12-01', 'YYYY-MM-DD'), 85000, 'Resigned');

INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id, hire_date, salary, status)

VALUES (6, 'Anura', 'Kumara', 'anura.kumara@schoolofcomputing.edu', '0765432123', 1, TO_DATE('2022-02-20', 'YYYY-MM-DD'), 72000, 'Active');

INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id, hire_date, salary, status)

VALUES (7, 'Chamara', 'Senanayake', 'chamara.senanayake@schoolofbusiness.edu', '0719876543', 2, TO_DATE('2016-09-18', 'YYYY-MM-DD'), 78000, 'Active');

INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id, hire_date, salary, status)

VALUES (8, 'Lakmal', 'Wijesinghe', 'lakmal.wijesinghe@schoolofengineering.edu', '0742345678', 3, TO_DATE('2020-05-12', 'YYYY-MM-DD'), 82000, 'Active');

INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id, hire_date, salary, status)

VALUES (9, 'Gayan', 'Weerasinghe', 'gayan.weerasinghe@schooloflanguage.edu', '0773219876', 4, TO_DATE('2021-11-23', 'YYYY-MM-DD'), 61000, 'Active');

INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id, hire_date, salary, status)

VALUES (10, 'Priyantha', 'Rajapaksa', 'priyantha.rajapaksa@schoolofhumanities.edu', '0756789123', 5, TO_DATE('2019-07-30', 'YYYY-MM-DD'), 86000, 'Active');

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

Figure 3.2.5

3.2.6 Retrieve All Records from lecturers Table

SELECT * FROM lecturers;

	LECTURER_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	DEPARTMENT_ID	HIRE_DATE	SALARY	STATUS
1	1	Nimal	Perera	nimal.perera@schoolofcomputing.edu	0711234567	1	15-JAN-20	80000	Active
2	2	Saman	Silva	saman.silva@schoolofbusiness.edu	0777654321	2	10-MAR-18	75000	Active
3	3	Kamal	Fernando	kamal.fernando@schoolofengineering.edu	0723456789	3	25-AUG-19	90000	Retired
4	4	Sunil	Jayasinghe	sunil.jayasinghe@schooloflanguage.edu	0709876543	4	05-JUN-21	60000	Active
5	5	Ruwan	Dias	ruwan.dias@schoolofhumanities.edu	0751237894	5	01-DEC-17	85000	Resigned
6	6	Anura	Kumara	anura.kumara@schoolofcomputing.edu	0765432123	1	20-FEB-22	72000	Active
7	7	Chamara	Senanayake	chamara.senanayake@schoolofbusiness.edu	0719876543	2	18-SEP-16	78000	Active
8	8	Lakmal	Wijesinghe	lakmal.wijesinghe@schoolofengineering.edu	0742345678	3	12-MAY-20	82000	Active
9	9	Gayan	Weerasinghe	gayan.weerasinghe@schooloflanguage.edu	0773219876	4	23-NOV-21	61000	Active
10	10	Priyantha	Rajapaksa	priyantha.rajapaksa@schoolofhumanities.edu	0756789123	5	30-JUL-19	86000	Active

Figure 3.2.6

3.2.7 CREATE courses TABLE

```
CREATE TABLE courses (
   course_id NUMBER PRIMARY KEY,
   course_name VARCHAR2(100) UNIQUE NOT NULL,
   department_id NUMBER NOT NULL,
   description VARCHAR2(200),
   credits NUMBER NOT NULL CHECK (credits > 0),
   start_date DATE NOT NULL,
   end_date DATE,
   status VARCHAR2(20) DEFAULT 'Active' CHECK (status IN ('Active', 'Inactive', 'Completed')),
   created_by NUMBER,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   FOREIGN KEY (department_id) REFERENCES departments (department_id),
   FOREIGN KEY (created_by) REFERENCES lecturers (lecturer_id),
   CHECK (end_date > start_date)
);
```

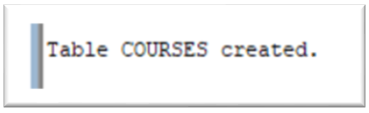


Figure 3.2.7

3.2.8 Insert Records into courses Table

```
INSERT INTO courses (course_id, course_name, department_id, description, credits, start_date,
end_date, created_by) VALUES
(1, 'Diploma in Software Engineering ', 1, 'This program introduces students to the fundamentals of
programming', 40, TO_DATE('2024-01-10', 'YYYY-MM-DD'), TO_DATE('2025-01-15', 'YYYY-MM-DD'), 3);
INSERT INTO courses (course_id, course_name, department_id, description, credits, start_date,
end_date, created_by) VALUES
```

(2, 'Higher National Diploma in Software Engineering ', 1, 'This program introduces students to the advances of programming', 45, TO_DATE('2024-05-10', 'YYYY-MM-DD'), TO_DATE('2025-05-15', 'YYYY-MM-DD'), 3);

INSERT INTO courses (course_id, course_name, department_id, description, credits, start_date, end_date, created_by) VALUES

(3, 'Advanced Diploma in Business Management', 2, 'This program introduces students to the fundamentals of Business', 30, TO_DATE('2024-01-10', 'YYYY-MM-DD'), TO_DATE('2025-01-15', 'YYYY-MM-DD'), 3);

INSERT INTO courses (course_id, course_name, department_id, description, credits, start_date, end_date, created_by) VALUES

(4, 'Higher National Diploma in Business Management', 2, 'This program introduces students to the advances of Business', 35, TO_DATE('2024-05-10', 'YYYY-MM-DD'), TO_DATE('2025-05-15', 'YYYY-MM-DD'), 3);

INSERT INTO courses (course_id, course_name, department_id, description, credits, start_date, end_date, created_by) VALUES

(5, 'Diploma in English', 4, 'This program introduces students to the English', 20, TO_DATE('2024-05-10', 'YYYY-MM-DD'), TO_DATE('2025-11-15', 'YYYY-MM-DD'), 3);

INSERT INTO courses (course_id, course_name, department_id, description, credits, start_date, end_date, created_by) VALUES

(6, 'Higher National Diploma in Network Engineering', 1, 'This program introduces students to the fundamentals of Network', 40, TO_DATE('2024-08-10', 'YYYY-MM-DD'), TO_DATE('2025-08-15', 'YYYY-MM-DD'), 3);

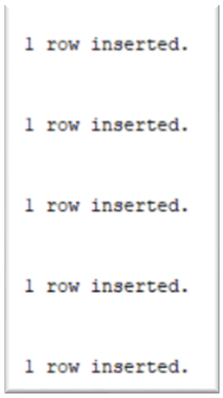


Figure 3.2.8

3.2.9 Retrieve All Records from courses Table

```
SELECT * FROM courses;
```

COURSE...	COURSE_NAME	DEPA...	DESCRIPTION	CREDITS	START_DATE	END_DATE	CREAT...	CREATED_AT	STATUS
1	Diploma in Software Engineering	1	This program introduc...	40	10-JAN-24	15-JAN-25	3	18-OCT-24 05.44.42...	Active
2	Higher National Diploma in Software Engin...	1	This program introduc...	45	10-MAY-24	15-MAY-25	3	18-OCT-24 05.44.42...	Active
3	Advanced Diploma in Business Management	2	This program introduc...	30	10-JAN-24	15-JAN-25	3	18-OCT-24 05.44.42...	Active
4	Higher National Diploma in Business Manag...	2	This program introduc...	35	10-MAY-24	15-MAY-25	3	18-OCT-24 05.44.42...	Active
5	Diploma in English	4	This program introduc...	20	10-MAY-24	15-NOV-25	3	18-OCT-24 05.44.42...	Active
6	Higher National Diploma in Network Engine...	1	This program introduc...	40	10-AUG-24	15-AUG-25	3	18-OCT-24 05.44.42...	Active

Figure 3.2.9

3.2.10 CREATE lessons TABLE

```
CREATE TABLE lessons (
    lesson_id NUMBER PRIMARY KEY,
    course_id NUMBER,
    lesson_name VARCHAR2(100) NOT NULL,
    taught_by NUMBER NOT NULL,
    status VARCHAR2(20) DEFAULT 'Active' CHECK (status IN ('Active','Inactive')),
    created_by NUMBER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (course_id) REFERENCES courses (course_id),
    FOREIGN KEY (taught_by) REFERENCES lecturers (lecturer_id),
    FOREIGN KEY (created_by) REFERENCES lecturers (lecturer_id)
);
```

```
Table LESSONS created.
```

Figure 3.2.10

3.2.11 Insert Records into lessons Table

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (1, 1, 'Introduction to Computer Science', 1, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (2, 1, 'Mathematics for Computing', 2, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (3, 1, 'Database Management Systems', 5, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (4, 2, 'Object-Oriented Programming', 4, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (5, 2, 'Web Development Basics', 6, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (6, 2, 'Programming Fundamentals', 2, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (7, 2, 'Electronics and Computer Architecture', 5, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (8, 1, 'Computer Networks', 2, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (9, 1, 'GUI Application Development', 4, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (10, 1, 'Software Engineering', 5, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (11, 2, 'Enterprise Application Development', 6, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (12, 2, 'Operating Systems', 5, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (13, 4, 'Financial Management', 7, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (14, 4, 'Environmental Management', 8, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (15, 4, 'Marketing Management', 8, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (16, 4, 'Legal Environment', 9, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (17, 4, 'Human Resource Management', 10, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (18, 4, 'Project Management', 10, 3);

INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by) VALUES (19, 4, 'Operations Logistics Management', 9, 3);

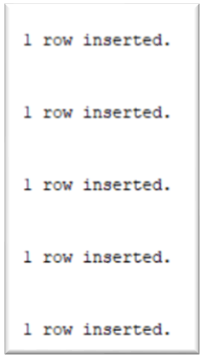
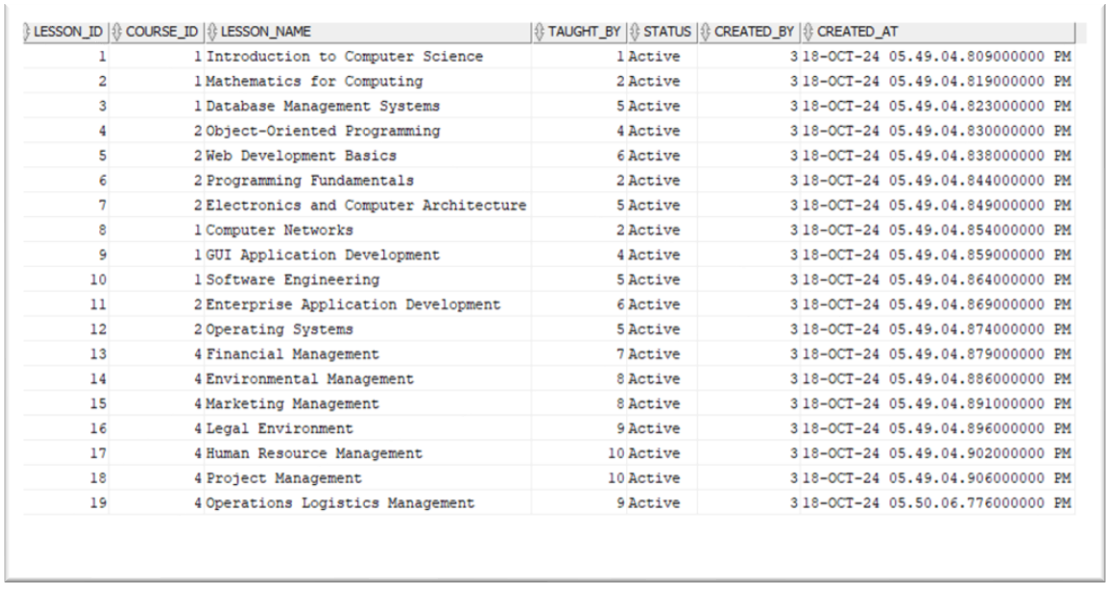


Figure 3.2.11

3.2.12 Retrieve All Records from lessons Table

SELECT * FROM lessons;



LESSON_ID	COURSE_ID	LESSON_NAME	TAUGHT_BY	STATUS	CREATED_BY	CREATED_AT
1	1	Introduction to Computer Science	1	Active	3	18-OCT-24 05.49.04.809000000 PM
2	1	Mathematics for Computing	2	Active	3	18-OCT-24 05.49.04.819000000 PM
3	1	Database Management Systems	5	Active	3	18-OCT-24 05.49.04.823000000 PM
4	2	Object-Oriented Programming	4	Active	3	18-OCT-24 05.49.04.830000000 PM
5	2	Web Development Basics	6	Active	3	18-OCT-24 05.49.04.838000000 PM
6	2	Programming Fundamentals	2	Active	3	18-OCT-24 05.49.04.844000000 PM
7	2	Electronics and Computer Architecture	5	Active	3	18-OCT-24 05.49.04.849000000 PM
8	1	Computer Networks	2	Active	3	18-OCT-24 05.49.04.854000000 PM
9	1	GUI Application Development	4	Active	3	18-OCT-24 05.49.04.859000000 PM
10	1	Software Engineering	5	Active	3	18-OCT-24 05.49.04.864000000 PM
11	2	Enterprise Application Development	6	Active	3	18-OCT-24 05.49.04.869000000 PM
12	2	Operating Systems	5	Active	3	18-OCT-24 05.49.04.874000000 PM
13	4	Financial Management	7	Active	3	18-OCT-24 05.49.04.879000000 PM
14	4	Environmental Management	8	Active	3	18-OCT-24 05.49.04.886000000 PM
15	4	Marketing Management	8	Active	3	18-OCT-24 05.49.04.891000000 PM
16	4	Legal Environment	9	Active	3	18-OCT-24 05.49.04.896000000 PM
17	4	Human Resource Management	10	Active	3	18-OCT-24 05.49.04.902000000 PM
18	4	Project Management	10	Active	3	18-OCT-24 05.49.04.906000000 PM
19	4	Operations Logistics Management	9	Active	3	18-OCT-24 05.50.06.776000000 PM

Figure 3.2.12

3.2.13 CREATE students TABLE

```
CREATE TABLE students (
   student_id NUMBER PRIMARY KEY,
   first_name VARCHAR2(50) NOT NULL,
   last_name VARCHAR2(50) NOT NULL,
   email VARCHAR2(100) UNIQUE NOT NULL,
   phone_number VARCHAR2(15) NOT NULL,
   date_of_birth DATE ,
   gender VARCHAR2(15) ,
   address VARCHAR2(200),
   status VARCHAR2(20) DEFAULT 'Active' CHECK (status IN ('Active','students','Suspended')),
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Figure 3.2.13

3.2.14 Insert Records into students Table

```
INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)
VALUES (1, 'Amila', 'Perera', 'amilaperera@gmail.com', '0711234567', TO_DATE('2001-05-15', 'YYYY-
MM-DD'), 'Male', '123, Colombo Road, Galle');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)
VALUES (2, 'Kasuni', 'Samarasinghe', 'kasunisamarasinghe@gmail.com', '0722345678', TO_DATE('2000-
08-25', 'YYYY-MM-DD'), 'Female', '45, Kandy Street, Kandy');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)
```

VALUES (3, 'Chathura', 'Wijesinghe', 'chathurawijesinghe@gmail.com', '0713456789', TO_DATE('2002-12-10', 'YYYY-MM-DD'), 'Male', '78, Lake View, Kurunegala');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (4, 'Nethmi', 'Jayasinghe', 'nethmijayasinghe@gmail.com', '0704567890', TO_DATE('2003-03-30', 'YYYY-MM-DD'), 'Female', '89, Beach Road, Matara');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (5, 'Sajith', 'Fernando', 'sajithfernando@gmail.com', '0775678901', TO_DATE('2001-07-20', 'YYYY-MM-DD'), 'Male', '65, Temple Lane, Negombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (6, 'Nimesha', 'Dissanayake', 'nimeshadissanayake@gmail.com', '0766789012', TO_DATE('2002-04-15', 'YYYY-MM-DD'), 'Female', '21, School Street, Galle');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (7, 'Ravindu', 'Perera', 'ravinduperera@gmail.com', '0757890123', TO_DATE('2001-10-10', 'YYYY-MM-DD'), 'Male', '45, Main Road, Kandy');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (8, 'Dilani', 'Gunasekara', 'dilanigunasekara@gmail.com', '0748901234', TO_DATE('2000-09-22', 'YYYY-MM-DD'), 'Female', '10, Hilltop, Nuwara Eliya');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (9, 'Tharindu', 'Wijesooriya', 'tharinduwijesooriya@gmail.com', '0739012345', TO_DATE('2002-02-18', 'YYYY-MM-DD'), 'Male', '34, River View, Kegalle');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (10, 'Ayesh', 'Rajapaksha', 'ayeshrajapaksha@gmail.com', '0720123456', TO_DATE('2001-06-30', 'YYYY-MM-DD'), 'Male', '88, Ocean Drive, Colombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (11, 'Chami', 'Chandimal', 'chamichandimal@gmail.com', '0712345678', TO_DATE('2003-05-25', 'YYYY-MM-DD'), 'Female', '44, Park Avenue, Colombo');

```
INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (12, 'Lakmal', 'Senevirathne', 'lakmalsenevirathne@gmail.com', '0703456789', TO_DATE('2001-
01-01', 'YYYY-MM-DD'), 'Male', '56, Green Road, Anuradhapura');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (13, 'Nimasha', 'Karunarathne', 'nimashakarunarathne@gmail.com', '0694567890',
TO_DATE('2002-11-14', 'YYYY-MM-DD'), 'Female', '23, Forest Lane, Ratnapura');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (14, 'Udara', 'Pathirana', 'udarapathirana@gmail.com', '0685678901', TO_DATE('2003-04-20',
'YYYY-MM-DD'), 'Male', '12, Seaside Avenue, Batticaloa');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (15, 'Dhanuka', 'Silva', 'dhanukasilva@gmail.com', '0676789012', TO_DATE('2000-10-30', 'YYYY-
MM-DD'), 'Male', '14, Mountain Road, Matale');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (16, 'Shanika', 'Herath', 'shanikaherath@gmail.com', '0667890123', TO_DATE('2002-03-05',
'YYYY-MM-DD'), 'Female', '19, City Center, Jaffna');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (17, 'Kamal', 'Fernando', 'kamalfernando@gmail.com', '0658901234', TO_DATE('2001-08-12',
'YYYY-MM-DD'), 'Male', '32, New Town, Colombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (18, 'Gayan', 'Perera', 'gayanperera@gmail.com', '0649012345', TO_DATE('2003-06-18', 'YYYY-
MM-DD'), 'Male', '27, Old Market, Galle');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (19, 'Nihal', 'Bandara', 'nihalbandara@gmail.com', '0630123456', TO_DATE('2000-11-26', 'YYYY-
MM-DD'), 'Male', '88, Lake Road, Colombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)
```

VALUES (20, 'Thilini', 'Samarawickrama', 'thilinisanarawickrama@gmail.com', '0621234567', TO_DATE('2002-09-02', 'YYYY-MM-DD'), 'Female', '55, Hill Road, Galle');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (21, 'Anushka', 'Fernando', 'anushkafernando@gmail.com', '0612345678', TO_DATE('2001-12-30', 'YYYY-MM-DD'), 'Female', '75, Church Street, Negombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (22, 'Charitha', 'Abeysekera', 'charithaabeysekera@gmail.com', '0603456789', TO_DATE('2002-07-15', 'YYYY-MM-DD'), 'Male', '24, River Lane, Ratnapura');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (23, 'Ruwangi', 'Senevirathne', 'ruwangisenevirathne@gmail.com', '0594567890', TO_DATE('2001-02-28', 'YYYY-MM-DD'), 'Female', '31, City View, Kandy');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (24, 'Kanishka', 'Jayasinghe', 'kanishkajayasinghe@gmail.com', '0585678901', TO_DATE('2003-01-10', 'YYYY-MM-DD'), 'Male', '99, Coastal Road, Matara');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (25, 'Lahiru', 'Kumarasinghe', 'lahirukumarasinghe@gmail.com', '0576789012', TO_DATE('2002-08-20', 'YYYY-MM-DD'), 'Male', '19, Bay Street, Colombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (26, 'Janani', 'Rathnayake', 'jananirathnayake@gmail.com', '0567890123', TO_DATE('2001-03-16', 'YYYY-MM-DD'), 'Female', '28, Hill Side, Jaffna');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (27, 'Sachin', 'Jayasuriya', 'sachinjayasuriya@gmail.com', '0558901234', TO_DATE('2000-04-22', 'YYYY-MM-DD'), 'Male', '90, Lake Road, Colombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (28, 'Meghna', 'Perera', 'meghnaperera@gmail.com', '0549012345', TO_DATE('2003-07-11', 'YYYY-MM-DD'), 'Female', '45, City Hall, Kandy');

```
INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (29, 'Vihanga', 'Jayasena', 'vihangajayasena@gmail.com', '0530123456', TO_DATE('2002-05-05',
'YYYY-MM-DD'), 'Male', '71, Coastal Road, Colombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (30, 'Saman', 'Tharindu', 'samantaharindu@gmail.com', '0521234567', TO_DATE('2001-09-30',
'YYYY-MM-DD'), 'Male', '30, Market Street, Nuwara Eliya');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (31, 'Sashini', 'Rajapakse', 'sashinirajapakse@gmail.com', '0512345678', TO_DATE('2000-10-15',
'YYYY-MM-DD'), 'Female', '67, Park Road, Colombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (32, 'Harsha', 'Bandarage', 'harshabandara@gmail.com', '0503456789', TO_DATE('2001-11-11',
'YYYY-MM-DD'), 'Male', '90, Green Street, Anuradhapura');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (33, 'Kavindu', 'Gunarathne', 'kavindugunarathne@gmail.com', '0494567890', TO_DATE('2002-
08-28', 'YYYY-MM-DD'), 'Male', '32, Central Road, Colombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (34, 'Manisha', 'Herath', 'manishaherath@gmail.com', '0485678901', TO_DATE('2003-02-14',
'YYYY-MM-DD'), 'Female', '26, Lake Road, Kandy');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (35, 'Dulanjali', 'Fernando', 'dulanjalifernando@gmail.com', '0476789012', TO_DATE('2001-04-
05', 'YYYY-MM-DD'), 'Female', '14, Hilltop, Colombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)

VALUES (36, 'Dilshani', 'Seneviratne', 'dilshaniseneviratne@gmail.com', '0467890123', TO_DATE('2002-
06-21', 'YYYY-MM-DD'), 'Female', '78, Sea View, Matara');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth,
gender, address)
```

VALUES (37, 'Namal', 'Perera', 'namalperera@gmail.com', '0458901234', TO_DATE('2001-09-12', 'YYYY-MM-DD'), 'Male', '33, Beach Road, Galle');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (38, 'Suwanjana', 'Abeysinghe', 'suwanjanaabeysinghe@gmail.com', '0449012345', TO_DATE('2003-03-22', 'YYYY-MM-DD'), 'Female', '56, Temple Road, Colombo');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (39, 'Hasith', 'Samarasekara', 'hasithsamarasekara@gmail.com', '0430123456', TO_DATE('2002-05-01', 'YYYY-MM-DD'), 'Male', '19, New Road, Ratnapura');

INSERT INTO students (student_id, first_name, last_name, email, phone_number, date_of_birth, gender, address)

VALUES (40, 'Chamila', 'Gunarathne', 'chamilagunarathne@gmail.com', '0421234567', TO_DATE('2000-07-29', 'YYYY-MM-DD'), 'Female', '12, Lake View, Kandy');

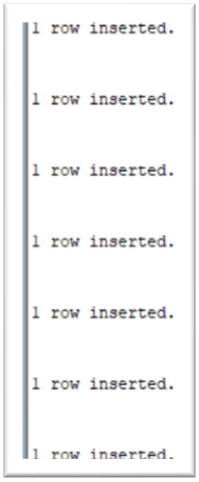


Figure 3.2.14

3.2.15 Retrieve All Records from students Table

SELECT * FROM students;

STUDENT_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	DATE_OF_BIRTH	GENDER	ADDRESS	STATUS	CREATED_AT
1	Amila	Perera	amilaperera@gmail.com	0711234567	15-MAY-01	Male	123, Colombo Road, Galle	Active	18-OCT-24 05.56.08.811000000 PM
2	Kasuni	Samarasinghe	kasunisamarasinghe@gmail.com	0722345678	25-AUG-00	Female	45, Kandy Street, Kandy	Active	18-OCT-24 05.56.08.820000000 PM
3	Chathura	Wijesinghe	chathurawijesinghe@gmail.com	0713456789	10-DEC-02	Male	78, Lake View, Kurunegala	Active	18-OCT-24 05.56.08.826000000 PM
4	Nethmi	Jayasinghe	nethmijayasinghe@gmail.com	0704567890	30-MAR-03	Female	89, Beach Road, Matara	Active	18-OCT-24 05.56.08.829000000 PM
5	Sajith	Fernando	sajithfernando@gmail.com	0775678901	20-JUL-01	Male	65, Temple Lane, Negombo	Active	18-OCT-24 05.56.08.834000000 PM
6	Nimesha	Dissanayake	nimeshadissanayake@gmail.com	0766789012	15-APR-02	Female	21, School Street, Galle	Active	18-OCT-24 05.56.08.840000000 PM
7	Ravindu	Perera	ravinduperera@gmail.com	0757890123	10-OCT-01	Male	45, Main Road, Kandy	Active	18-OCT-24 05.56.08.846000000 PM
8	Dilani	Gunasekara	dilanigunasekara@gmail.com	0748901234	22-SEP-00	Female	10, Hilltop, Nuwara Eliya	Active	18-OCT-24 05.56.08.849000000 PM
9	Tharindu	Wijesooriya	tharinduwijesooriya@gmail.com	0739012345	18-FEB-02	Male	34, River View, Kegalle	Active	18-OCT-24 05.56.08.854000000 PM
10	Ayesh	Rajapaksha	ayeshrajapaksha@gmail.com	0720123456	30-JUN-01	Male	88, Ocean Drive, Colombo	Active	18-OCT-24 05.56.08.858000000 PM
11	Chami	Chandimal	chamichandimal@gmail.com	0712345678	25-MAY-03	Female	44, Park Avenue, Colombo	Active	18-OCT-24 05.56.08.863000000 PM
12	Lakmal	Senevirathne	lakmalsenevirathne@gmail.com	0703456789	01-JAN-01	Male	56, Green Road, Anuradhapura	Active	18-OCT-24 05.56.08.868000000 PM
13	Nimasha	Karunarathne	nimashakarunarathne@gmail.com	0694567890	14-NOV-02	Female	23, Forest Lane, Ratnapura	Active	18-OCT-24 05.56.08.873000000 PM
14	Udara	Pathirana	udarapathirana@gmail.com	0685678901	20-APR-03	Male	12, Seaside Avenue, Batticaloa	Active	18-OCT-24 05.56.08.879000000 PM
15	Dhanuka	Silva	dhanukasilva@gmail.com	0676789012	30-OCT-00	Male	14, Mountain Road, Matale	Active	18-OCT-24 05.56.08.883000000 PM
16	Shanika	Herath	shanikaherath@gmail.com	0667890123	05-MAR-02	Female	19, City Center, Jaffna	Active	18-OCT-24 05.56.08.887000000 PM
17	Kamal	Fernando	kamalfernando@gmail.com	0658901234	12-AUG-01	Male	32, New Town, Colombo	Active	18-OCT-24 05.56.08.891000000 PM
18	Gayan	Perera	gayanperera@gmail.com	0649012345	18-JUN-03	Male	27, Old Market, Galle	Active	18-OCT-24 05.56.08.897000000 PM
19	Nihal	Bandara	nihalbandara@gmail.com	0630123456	26-NOV-00	Male	88, Lake Road, Colombo	Active	18-OCT-24 05.56.08.902000000 PM
20	Thilini	Samarawickrama	thilinisamarawickrama@gmail.com	0621234567	02-SEP-02	Female	55, Hill Road, Galle	Active	18-OCT-24 05.56.08.907000000 PM
21	Anushka	Fernando	anushkafernando@gmail.com	0612345678	30-DEC-01	Female	75, Church Street, Negombo	Active	18-OCT-24 05.56.08.912000000 PM
22	Charitha	Abeysekera	charithaabeysekera@gmail.com	0603456789	15-JUL-02	Male	24, River Lane, Ratnapura	Active	18-OCT-24 05.56.08.918000000 PM
23	Ruwangi	Senevirathne	ruwangisenevirathne@gmail.com	0594567890	28-FEB-01	Female	31, City View, Kandy	Active	18-OCT-24 05.56.08.922000000 PM
24	Kanishka	Jayasinghe	kanishkajayasinghe@gmail.com	0585678901	10-JAN-03	Male	99, Coastal Road, Matara	Active	18-OCT-24 05.56.08.927000000 PM
25	Lahiru	Kumarasinghe	lahirukumarasinghe@gmail.com	0576789012	20-AUG-02	Male	19, Bay Street, Colombo	Active	18-OCT-24 05.56.08.933000000 PM
26	Janani	Rathnayake	jananirathnayake@gmail.com	0567890123	16-MAR-01	Female	28, Hill Side, Jaffna	Active	18-OCT-24 05.56.08.939000000 PM
27	Sachin	Jayasuriya	sachinjayasuriya@gmail.com	0558901234	22-APR-00	Male	90, Lake Road, Colombo	Active	18-OCT-24 05.56.08.944000000 PM
28	Meghna	Perera	meghnaperera@gmail.com	0549012345	11-JUL-03	Female	45, City Hall, Kandy	Active	18-OCT-24 05.56.08.948000000 PM
29	Vihanga	Jayasena	vihangajayasena@gmail.com	0530123456	05-MAY-02	Male	71, Coastal Road, Colombo	Active	18-OCT-24 05.56.08.954000000 PM
30	Saman	Tharindu	samantaharindu@gmail.com	0521234567	30-SEP-01	Male	30, Market Street, Nuwara Eliya	Active	18-OCT-24 05.56.08.958000000 PM
31	Sashini	Rajapakse	sashinirajapakse@gmail.com	0512345678	15-OCT-00	Female	67, Park Road, Colombo	Active	18-OCT-24 05.56.08.964000000 PM
32	Harsha	Bandarage	harshabandara@gmail.com	0503456789	11-NOV-01	Male	90, Green Street, Anuradhapura	Active	18-OCT-24 05.56.08.969000000 PM

Figure 3.2.15

3.2.16 CREATE course_enrollments TABLE

```
CREATE TABLE course_enrollments (
    course_enrollments_id NUMBER PRIMARY KEY,
    student_id NUMBER NOT NULL,
    course_id NUMBER NOT NULL,
    enrollement_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (student_id) REFERENCES students (student_id),
    FOREIGN KEY (course_id) REFERENCES courses (course_id)
);
```

```
Table COURSE_ENROLLMENTS created.
```

Figure 3.2.16

3.2.17 Insert Records into course_enrollments Table

```
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (1, 1, 1);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (2, 2, 1);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (3, 3, 1);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (4, 4, 2);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (5, 5, 2);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (6, 6, 3);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (7, 7, 3);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (8, 8, 4);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (9, 9, 4);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (10, 10, 4);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (11, 11, 1);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (12, 12, 1);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (13, 13, 2);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (14, 14, 2);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (15, 15, 3);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (16, 16, 3);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (17, 17, 4);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (18, 18, 4);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (19, 19, 1);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (20, 20, 1);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (21, 21, 5);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (22, 22, 2);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (23, 23, 3);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (24, 24, 5);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (25, 25, 4);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (26, 26, 4);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (27, 27, 1);
```

```
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (28, 28, 6);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (29, 29, 2);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (30, 30, 2);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (31, 31, 3);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (32, 32, 3);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (33, 33, 4);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (34, 34, 4);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (35, 35, 1);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (36, 36, 1);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (37, 37, 2);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (38, 38, 2);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (39, 39, 3);
INSERT INTO course_enrollments (course_enrollments_id, student_id, course_id) VALUES (40, 40, 3);
```

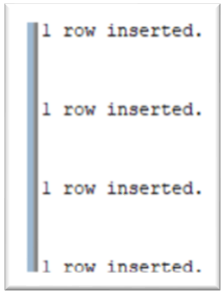


Figure 3.2.17

3.2.18 Retrieve All Records from course_enrollments Table

```
SELECT * FROM course_enrollments;
```

COURSE_ENROLLMENTS_ID	STUDENT_ID	COURSE_ID	ENROLLEMENT_DATE
1	1	1	18-OCT-24 06.07.52.002000000 PM
2	2	1	18-OCT-24 06.07.52.013000000 PM
3	3	1	18-OCT-24 06.07.52.021000000 PM
4	4	2	18-OCT-24 06.07.52.026000000 PM
5	5	2	18-OCT-24 06.07.52.029000000 PM
6	6	3	18-OCT-24 06.07.52.034000000 PM
7	7	3	18-OCT-24 06.07.52.040000000 PM
8	8	4	18-OCT-24 06.07.52.045000000 PM
9	9	4	18-OCT-24 06.07.52.048000000 PM
10	10	4	18-OCT-24 06.07.52.054000000 PM
11	11	1	18-OCT-24 06.07.52.056000000 PM
12	12	1	18-OCT-24 06.07.52.062000000 PM
13	13	2	18-OCT-24 06.07.52.068000000 PM
14	14	2	18-OCT-24 06.07.52.073000000 PM
15	15	3	18-OCT-24 06.07.52.077000000 PM
16	16	3	18-OCT-24 06.07.52.081000000 PM
17	17	4	18-OCT-24 06.07.52.086000000 PM
18	18	4	18-OCT-24 06.07.52.091000000 PM
19	19	1	18-OCT-24 06.07.52.097000000 PM
20	20	1	18-OCT-24 06.07.52.103000000 PM
21	21	5	18-OCT-24 06.07.52.106000000 PM
22	22	2	18-OCT-24 06.07.52.111000000 PM
23	23	3	18-OCT-24 06.07.52.114000000 PM
24	24	5	18-OCT-24 06.07.52.119000000 PM
25	25	4	18-OCT-24 06.07.52.124000000 PM
26	26	4	18-OCT-24 06.07.52.127000000 PM
27	27	1	18-OCT-24 06.07.52.131000000 PM
28	28	6	18-OCT-24 06.07.52.136000000 PM
29	29	2	18-OCT-24 06.07.52.141000000 PM
30	30	2	18-OCT-24 06.07.52.144000000 PM
31	31	3	18-OCT-24 06.07.52.148000000 PM
32	32	3	18-OCT-24 06.07.52.153000000 PM

Figure 3.2.18

3.2.19 CREATE assignment TABLE

```
CREATE TABLE assignment (
   assignment_id  NUMBER PRIMARY KEY,
   assignment_title VARCHAR2(50) NOT NULL,
   assignment_fiies VARCHAR2(50) NOT NULL,
   lesson_id NUMBER NOT NULL,
   post_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   end_date DATE NOT NULL,
   FOREIGN KEY (lesson_id) REFERENCES lessons (lesson_id)
);
```

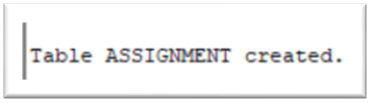


Figure 3.2.19

3.2.20 Insert Records into assignment Table

INSERT INTO assignment (assignment_id, assignment_title, assignment_fiies, lesson_id, end_date)

VALUES (1, 'Assignment 1', 'file1.pdf', 1, TO_DATE('2024-10-10', 'YYYY-MM-DD'));

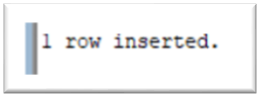
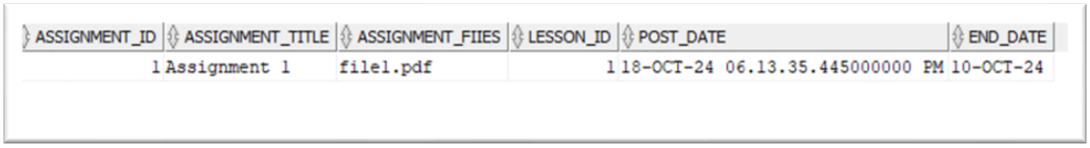


Figure 3.2.20

3.2.21 Retrieve All Records from assignment Table

SELECT * FROM assignment;



ASSIGNMENT_ID	ASSIGNMENT_TITLE	ASSIGNMENT_FIIES	LESSON_ID	POST_DATE	END_DATE
1	Assignment 1	file1.pdf	1	18-OCT-24 06.13.35.445000000 PM	10-OCT-24

Figure 3.2.21

3.2.22 CREATE assignment_students TABLE

```
CREATE TABLE assignment_students (
   assignment_students_id NUMBER PRIMARY KEY,
   assignment_id  NUMBER NOT NULL,
   student_id NUMBER NOT NULL,
   submit_file VARCHAR2(200),
   submissionDate TIMESTAMP,
```

```
    status VARCHAR2(20),
    grade VARCHAR2(20),
    Feedback_on_Assessment VARCHAR2(100) ,
    FOREIGN KEY (assignment_id) REFERENCES assignment (assignment_id),
    FOREIGN KEY (student_id) REFERENCES students (student_id)
);
```

```
Table ASSIGNMENT_STUDENTS created.
```

Figure 3.2.22

3.2.23 Insert Records into assignment_students Table

```
INSERT INTO assignment_students (assignment_students_id, assignment_id, student_id, submit_file, status, submissionDate) VALUES
(1, 1, 1, 'assignment1_student1.docx', 'Submitted',SYSTIMESTAMP);
```

```
1 row inserted.
```

Figure 3.2.23

3.2.24 Retrieve All Records from assignment_students Table

```
SELECT * FROM assignment_students;
```

ASSIGNMENT_STUDENTS_ID	ASSIGNMENT_ID	STUDENT_ID	SUBMIT_FILE	SUBMISSIONDATE	STATUS	GRADE	FEEDBACK_ON_ASSESSMENT
1	1	1	assignment1_student1.docx	18-OCT-24 06.15.23.480000000 PM	Submitted	(null)	(null)

Figure 3.2.24

3.2.25 CREATE assignment_students TABLE

```
CREATE TABLE students_feedback (
```

```
    feedback_id NUMBER PRIMARY KEY,
    lesson_id NUMBER NOT NULL,
    student_id NUMBER NOT NULL,
    comments VARCHAR2(100) NOT NULL,
    rating NUMBER NOT NULL CHECK (rating BETWEEN 1 AND 5),
    post_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (lesson_id) REFERENCES lessons (lesson_id),
    FOREIGN KEY (student_id) REFERENCES students (student_id)
);
```

Figure 3.2.25

3.2.26 Insert Records into students_feedback Table

```
INSERT INTO students_feedback (feedback_id, lesson_id, student_id, comments, rating) VALUES (1, 1, 1, 'The lesson was very informative and engaging.', 5);
```

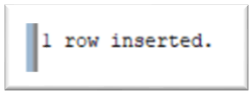
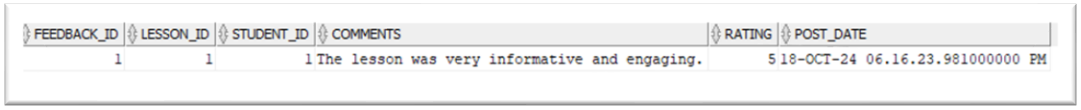


Figure 3.2.26

3.2.27 Retrieve All Records from students_feedback Table

```
SELECT * FROM students_feedback;
```



FEEDBACK_ID	LESSON_ID	STUDENT_ID	COMMENTS	RATING	POST_DATE
1	1	1	The lesson was very informative and engaging.	5	18-OCT-24 06.16.23.981000000 PM

Figure 3.2.27

3.2.28 CREATE schedule TABLE

```
CREATE TABLE schedule (
   schedule_id NUMBER PRIMARY KEY,
   lesson_id NUMBER NOT NULL,
   schedule_date DATE NOT NULL,
   lecture_hall VARCHAR2(50) NOT NULL,
   FOREIGN KEY (lesson_id) REFERENCES lessons (lesson_id)
);
```

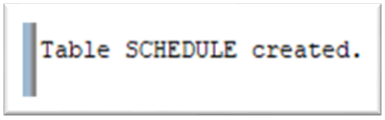


Figure 3.2.28

3.2.29 Insert Records into schedule Table

```
INSERT INTO schedule (schedule_id, lesson_id, schedule_date, lecture_hall) VALUES (1, 1,
TO_DATE('2024-10-22', 'YYYY-MM-DD'), 'Lecture Hall 01');

INSERT INTO schedule (schedule_id, lesson_id, schedule_date, lecture_hall) VALUES (2, 2,
TO_DATE('2024-10-23', 'YYYY-MM-DD'), 'Lecture Hall 02');

INSERT INTO schedule (schedule_id, lesson_id, schedule_date, lecture_hall) VALUES (3, 1,
TO_DATE('2024-10-24', 'YYYY-MM-DD'), 'Lecture Hall 03');

INSERT INTO schedule (schedule_id, lesson_id, schedule_date, lecture_hall) VALUES (4, 4,
TO_DATE('2024-10-25', 'YYYY-MM-DD'), 'Lecture Hall 04');

INSERT INTO schedule (schedule_id, lesson_id, schedule_date, lecture_hall) VALUES (5, 1,
TO_DATE('2024-10-26', 'YYYY-MM-DD'), 'Lecture Hall 05');
```

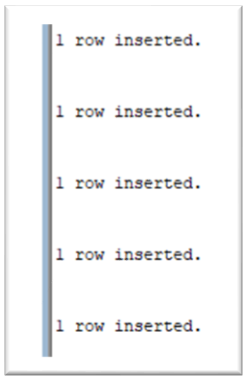


Figure 3.2.29

3.2.30 Retrieve All Records from schedule Table

SELECT * FROM schedule;

SCHEDULE_ID	LESSON_ID	SCHEDULE_DATE	LECTURE_HALL
1	1	22-OCT-24	Lecture Hall 01
2	2	23-OCT-24	Lecture Hall 02
3	1	24-OCT-24	Lecture Hall 03
4	4	25-OCT-24	Lecture Hall 04
5	1	26-OCT-24	Lecture Hall 05

Figure 3.2.30

3.3 Triggers for Automated ID Incrementation and Data Operations

3.3.1 Trigger for Incrementing Student ID Before Insert

```
CREATE OR REPLACE TRIGGER student_id_increment
BEFORE INSERT ON students
FOR EACH ROW
DECLARE
  max_student_id NUMBER;
BEGIN
  SELECT
  CASE
    WHEN MAX(student_id) IS NULL THEN 0 ELSE MAX(student_id)
```

```
    END
    INTO max_student_id FROM students;
    :NEW.student_id := max_student_id + 1;
END student_id_increment;
```

```
Trigger STUDENT_ID_INCREMENT compiled
```

Figure 3.3.1

3.3.2 Trigger for Incrementing Course Enrollments ID Before Insert

```
CREATE OR REPLACE TRIGGER course_enrollments_id_increment
BEFORE INSERT ON course_enrollments
FOR EACH ROW
DECLARE
    max_course_enrollments_id NUMBER;
BEGIN
    SELECT
    CASE
        WHEN MAX(course_enrollments_id) IS NULL THEN 0 ELSE MAX(course_enrollments_id)
    END
    INTO max_course_enrollments_id FROM course_enrollments;
    :NEW.course_enrollments_id := max_course_enrollments_id + 1;
END;
```

```
Trigger COURSE_ENROLLMENTS_ID_INCREMENT compiled
```

Figure 3.3.2

3.3.3 Trigger for Incrementing Assignment ID Before Insert

```
CREATE OR REPLACE TRIGGER assignment_idincrement
```

```
BEFORE INSERT ON assignment
FOR EACH ROW
DECLARE
  max_assignment_id NUMBER;
BEGIN
  SELECT
  CASE
    WHEN MAX(assignment_id) IS NULL THEN 0 ELSE MAX(assignment_id)
  END
  INTO max_assignment_id FROM assignment;
  :NEW.assignment_id := max_assignment_id + 1;
END;
/
```

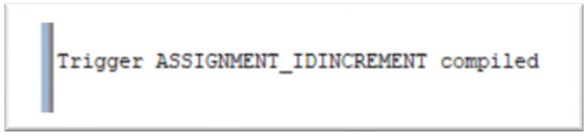


Figure 3.3.3

3.3.4 Trigger for Incrementing Feedback ID Before Insert

```
CREATE OR REPLACE TRIGGER feedback_id_increment
BEFORE INSERT ON students_feedback
FOR EACH ROW
DECLARE
  max_feedback_id NUMBER;
BEGIN
  SELECT
  CASE
    WHEN MAX(feedback_id) IS NULL THEN 0 ELSE MAX(feedback_id)
  END
```

```
    INTO max_feedback_id FROM students_feedback;
    :NEW.feedback_id := max_feedback_id + 1;
END;
/
```

```
Trigger FEEDBACK_ID_INCREMENT compiled
```

Figure 3.3.4

3.3.5 Trigger to Increment Schedule ID Before Insert

```
CREATE OR REPLACE TRIGGER schedule_id_increment
BEFORE INSERT ON schedule
FOR EACH ROW
DECLARE
    max_schedule_id NUMBER;
BEGIN
    SELECT
        CASE
            WHEN MAX(schedule_id) IS NULL THEN 0 ELSE MAX(schedule_id)
        END
    INTO max_schedule_id
    FROM schedule;
    :NEW.schedule_id := max_schedule_id + 1;
END;
/
```

```
Trigger SCHEDULE_ID_INCREMENT compiled
```

Figure 3.3.5

3.3.6 Trigger to Add Students to assignment_students After Insert assignment

```
CREATE SEQUENCE assignment_students_seq
START WITH 1
INCREMENT BY 1;
```

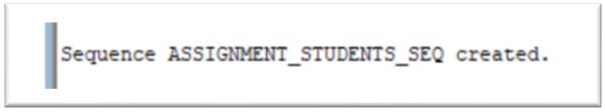


Figure 3.3.6.1

```
CREATE OR REPLACE TRIGGER add_students_to_assignment
AFTER INSERT ON assignment
FOR EACH ROW
DECLARE
  CURSOR student_cursor IS
    SELECT ce.student_id FROM course_enrollments ce JOIN lessons l ON l.course_id = ce.course_id
    WHERE l.lesson_id = :NEW.lesson_id;
BEGIN
  FOR student_record IN student_cursor LOOP
    INSERT INTO assignment_students (assignment_students_id, assignment_id, student_id, status)
    VALUES (assignment_students_seq.NEXTVAL, :NEW.assignment_id, student_record.student_id,
'Not submitted');
  END LOOP;
END;
/
```

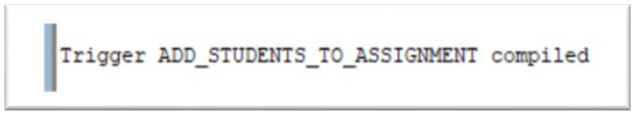


Figure 3.3.6.2

3.4 CRUD Operations Procedure for LMS and Execution

3.4.1 Create Procedure to Insert Department Record

```
CREATE OR REPLACE PROCEDURE insert_department (i_department_id IN NUMBER,
i_department_name IN VARCHAR2) AS
BEGIN
   INSERT INTO departments (department_id, department_name) VALUES (i_department_id,
i_department_name);
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred ');
END ;
```

```
Procedure INSERT_DEPARTMENT compiled
```

Figure 3.4.1

3.4.2 Insert Department Using Variables

```
DECLARE
   iu_department_id NUMBER := &department_id;
   iu_department_name VARCHAR2(100) := '&department_name';
BEGIN
   insert_department(iu_department_id, iu_department_name);
   DBMS_OUTPUT.PUT_LINE('Department "' || iu_department_name || '" with ID ' || iu_department_id
|| ' has been successfully added.');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error inserting department ');
END;
```

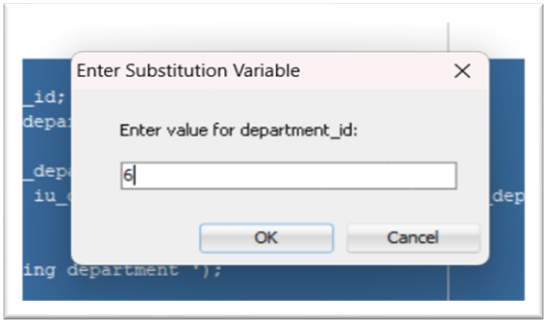


Figure 3.4.2.1

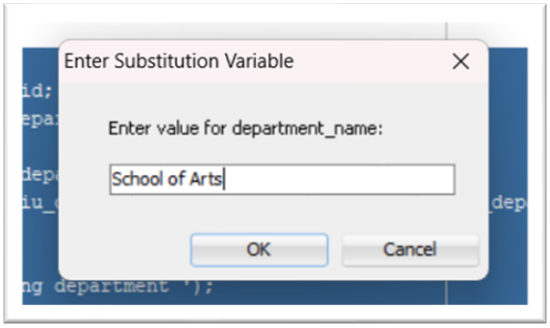


Figure 3.4.2.2

SELECT * FROM departments;

Figure 3.4.2.3

3.4.3 Create Procedure to update Department Record

```
CREATE OR REPLACE PROCEDURE update_department (u_department_id IN NUMBER,
u_department_name IN VARCHAR2) AS u_department_count NUMBER;

BEGIN

  SELECT COUNT(*) INTO u_department_count FROM departments WHERE department_id =
u_department_id;

  IF u_department_count = 0 THEN

    DBMS_OUTPUT.PUT_LINE('No department found ');

  ELSE
```

```
    UPDATE departments SET department_name = u_department_name WHERE department_id =
u_department_id;
    DBMS_OUTPUT.PUT_LINE('Updated');
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred ');
END;
```

```
Procedure UPDATE_DEPARTMENT compiled
```

Figure 3.4.3

3.4.4 Update Department Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
  v_department_id NUMBER := &department_id;
  v_department_name VARCHAR2(100) := '&department_name';
  BEGIN
  update_department(v_department_id, v_department_name);
END;
```

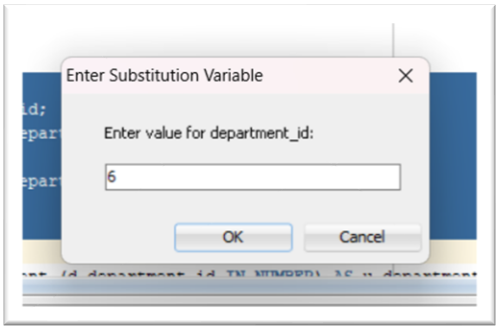


Figure 3.4.4.1

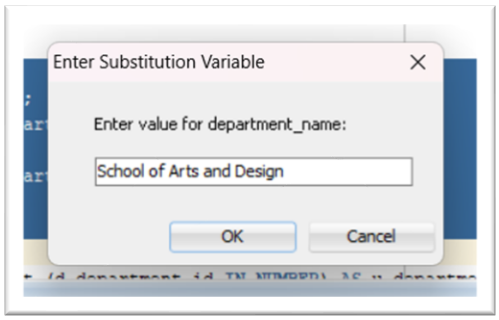


Figure 3.4.4.2

SELECT * FROM departments;

```
         6 School of Arts and Design 18-OCT-24 06.39.34.395000000 PM
```

Figure 3.4.4.3

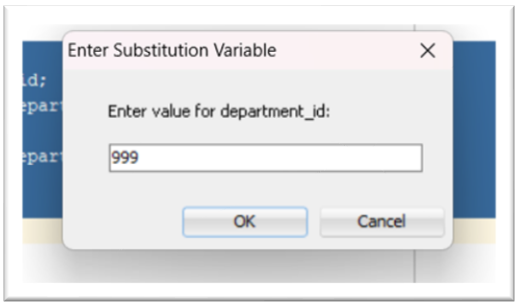


Figure 3.4.4.4

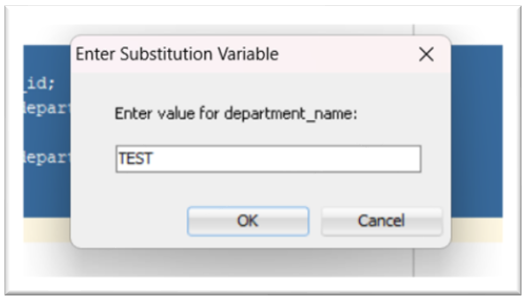


Figure 3.4.4.5

```
END;
No department found
```

Figure 3.4.4.6

3.4.5 Create Procedure to delete Department Record

```
CREATE OR REPLACE PROCEDURE delete_department (d_department_id IN NUMBER) AS
u_department_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO u_department_count FROM departments WHERE department_id =
d_department_id;
   IF u_department_count = 0 THEN
      DBMS_OUTPUT.PUT_LINE('No department found');
   ELSE
      DELETE FROM departments WHERE department_id = d_department_id;
      DBMS_OUTPUT.PUT_LINE('Department deleted');
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred ');
END;
```

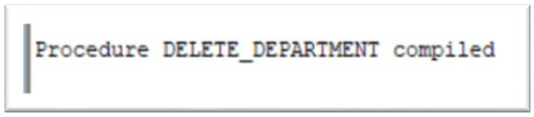


Figure 3.4.5

3.4.6 Delete Department Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
   v_department_id NUMBER := &department_id;
BEGIN
   delete_department(v_department_id);
END;
```

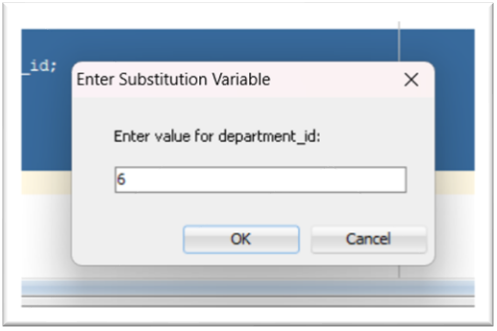


Figure 3.4.6.1

SELECT * FROM departments;

DEPARTMENT_ID	DEPARTMENT_NAME	CREATED_AT
1	School of Computing	18-OCT-24 05.14.26.219000000 PM
2	School of Business	18-OCT-24 05.14.26.230000000 PM
3	School of Engineering	18-OCT-24 05.14.26.233000000 PM
4	School of Language	18-OCT-24 05.14.26.237000000 PM
5	School of Humanities	18-OCT-24 05.14.59.726000000 PM

Figure 3.4.6.2

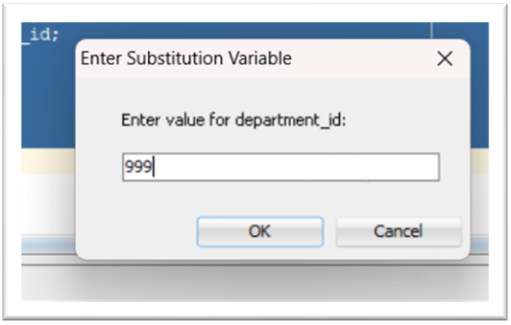


Figure 3.4.6.3

```
END;
No department found
```

Figure 3.4.6.4

3.4.7 Create Procedure to insert lecturer Record

```
CREATE OR REPLACE PROCEDURE insert_lecturer (p_lecturer_id IN NUMBER, p_first_name IN
VARCHAR2, p_last_name IN VARCHAR2, p_email IN VARCHAR2,
   p_phone_number IN VARCHAR2, p_department_id IN NUMBER,p_hire_date IN DATE,p_salary IN
NUMBER) AS
BEGIN
   INSERT INTO lecturers (lecturer_id, first_name, last_name, email, phone_number, department_id,
hire_date, salary)
   VALUES (p_lecturer_id, p_first_name, p_last_name, p_email, p_phone_number, p_department_id,
p_hire_date, p_salary);
   DBMS_OUTPUT.PUT_LINE('Successfully added');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

```
Procedure INSERT_LECTURER compiled
```

Figure 3.4.7

3.4.8 Insert Lecturer Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
   v_lecturer_id NUMBER := &lecturer_id;
   v_first_name VARCHAR2(50) := '&first_name';
   v_last_name VARCHAR2(50) := '&last_name';
   v_email VARCHAR2(100) := '&email';
   v_phone_number VARCHAR2(15) := '&phone_number';
   v_department_id NUMBER := &department_id;
```

```
    v_hire_date DATE := TO_DATE('&hire_date', 'YYYY-MM-DD');
    v_salary NUMBER := &salary;
BEGIN
    insert_lecturer(v_lecturer_id, v_first_name, v_last_name, v_email, v_phone_number,
v_department_id, v_hire_date, v_salary);
END;
/
```

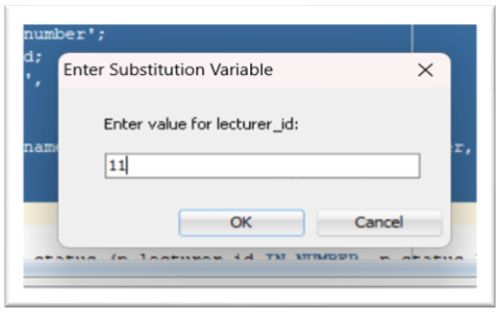


Figure 3.4.8.1

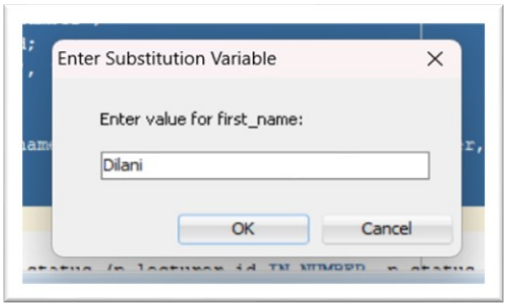


Figure 3.4.8.2

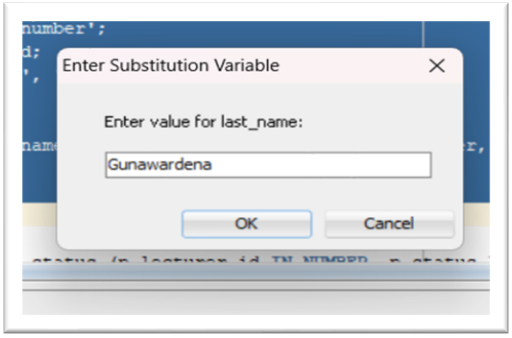


Figure 3.4.8.3

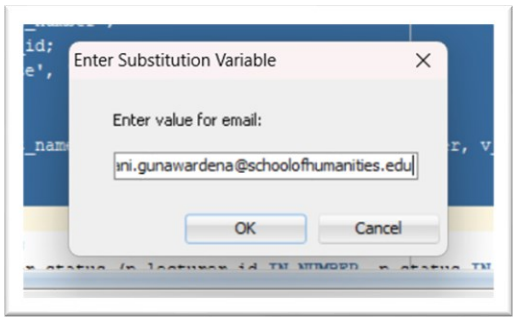


Figure 3.4.8.4

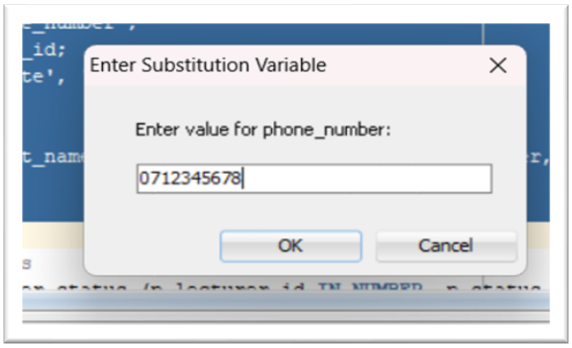


Figure 3.4.8.5

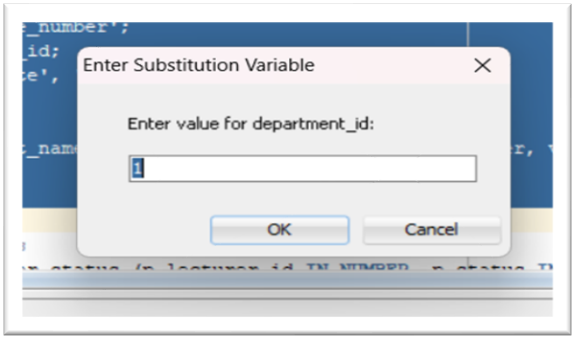


Figure 3.4.8.6

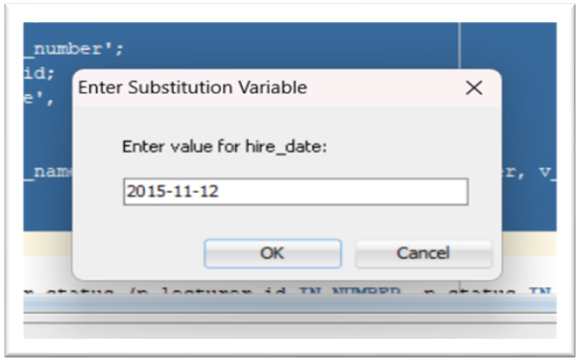


Figure 3.4.8.7

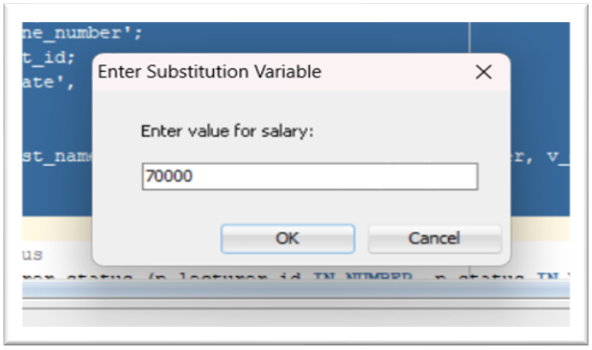


Figure 3.4.8.8

SELECT * FROM lecturers;

Figure 3.4.8.9

3.4.9 Create Procedure to update lecturer Record

CREATE OR REPLACE PROCEDURE update_lecturer_status (p_lecturer_id IN NUMBER, p_status IN VARCHAR2) AS v_lecturer_count NUMBER;

BEGIN

SELECT COUNT(*) INTO v_lecturer_count FROM lecturers WHERE lecturer_id = p_lecturer_id;

IF v_lecturer_count = 0 THEN

DBMS_OUTPUT.PUT_LINE('No lecturer found');

ELSE

```
    UPDATE lecturers SET status = p_status WHERE lecturer_id = p_lecturer_id;
    DBMS_OUTPUT.PUT_LINE('Lecturer status updated ');
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

```
Procedure UPDATE_LECTURER_STATUS compiled
```

Figure 3.4.9

3.4.10 Update Lecturer Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
  v_lecturer_id NUMBER := &lecturer_id;
  v_status VARCHAR2(20) := '&status';
BEGIN
  update_lecturer_status(v_lecturer_id, v_status);
END;
/
```

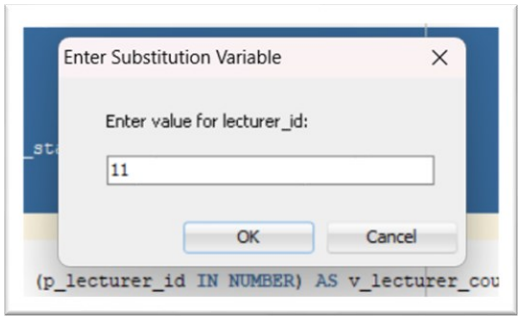
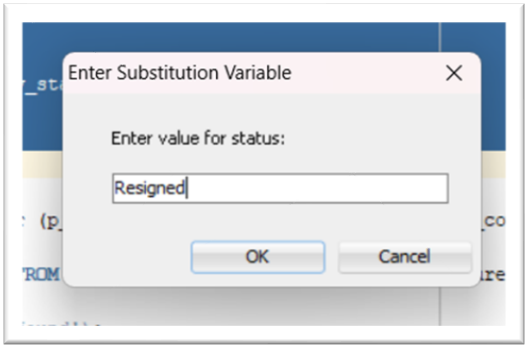


Figure 3.4.10.1



SELECT * FROM lecturers;

11	Dilani	Gunawardena	dilani.gunawardena@schoolofhumanities.edu	0712345678		1	12-NOV-15	70000 Resigned

Figure 3.4.10.2

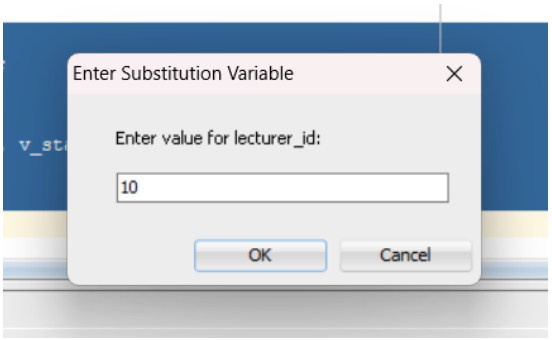


Figure 3.4.10.3

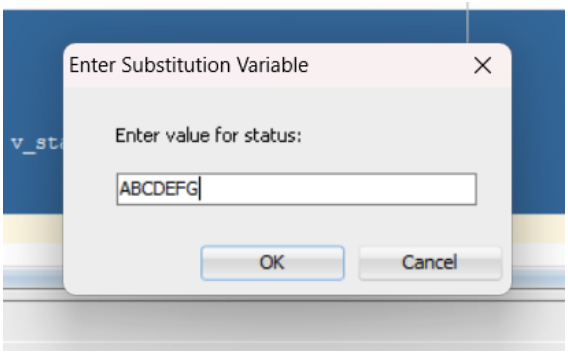


Figure 3.4.10.4

```
An error occurredORA-02290: check constraint (ADMIN.SYS_C008442) violated
```

Figure 3.4.10.5

3.4.11 Create Procedure to delete lecturer Record

```
CREATE OR REPLACE PROCEDURE delete_lecturer (p_lecturer_id IN NUMBER) AS v_lecturer_count
NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_lecturer_count FROM lecturers WHERE lecturer_id = p_lecturer_id;
  IF v_lecturer_count = 0 THEN
    DBMS_OUTPUT.PUT_LINE('No lecturer found');
  ELSE
    DELETE FROM lecturers WHERE lecturer_id = p_lecturer_id;
    DBMS_OUTPUT.PUT_LINE('Lecturer deleted');
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

```
Procedure DELETE_LECTURER compiled
```

Figure 3.4.11

3.4.12 Delete Lecturer Using Variable

```
SET SERVEROUTPUT ON;
DECLARE
   v_lecturer_id NUMBER := &lecturer_id;
BEGIN
   delete_lecturer(v_lecturer_id);
END;
/
```

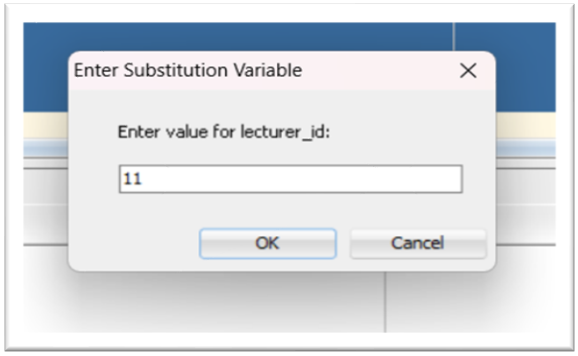


Figure 3.4.12.1

```
SELECT * FROM lecturers;
```

8	Lakmal	Wijesinghe	lakmal.wijesinghe@schoolofengineering.edu	0742345678	3	12-MAY-20	82000	Active
9	Gayan	Weerasinghe	gayan.weerasinghe@schooloflanguage.edu	0773219876	4	23-NOV-21	61000	Active
10	Priyantha	Rajapaksa	priyantha.rajapaksa@schoolofhumanities.edu	0756789123	5	30-JUL-19	86000	Active

Figure 3.4.12.2

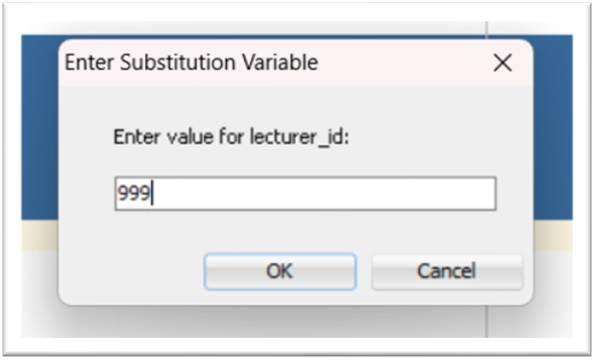


Figure 3.4.12.3

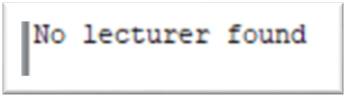


Figure 3.4.12.4

3.4.13 Create Procedure to insert course Record

```
CREATE OR REPLACE PROCEDURE insert_course (p_course_id IN NUMBER, p_course_name IN VARCHAR2, p_department_id IN NUMBER, p_description IN VARCHAR2, p_credits IN NUMBER,
    p_start_date IN DATE, p_end_date IN DATE, p_created_by IN NUMBER) AS
BEGIN
    INSERT INTO courses (course_id, course_name, department_id, description, credits, start_date, end_date, created_by)
    VALUES (p_course_id, p_course_name, p_department_id, p_description, p_credits, p_start_date, p_end_date, p_created_by);
    DBMS_OUTPUT.PUT_LINE('Course successfully added');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

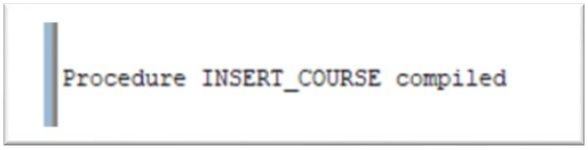


Figure 3.4.13

3.4.14 Insert Course Using Variables

```
SET SERVEROUTPUT ON;

DECLARE

   v_course_id NUMBER := &course_id;

   v_course_name VARCHAR2(100) := '&course_name';

   v_department_id NUMBER := &department_id;

   v_description VARCHAR2(200) := '&description';

   v_credits NUMBER := &credits;

   v_start_date DATE := TO_DATE('&start_date', 'YYYY-MM-DD');

   v_end_date DATE := TO_DATE('&end_date', 'YYYY-MM-DD');

   v_created_by NUMBER := &created_by;

BEGIN

   insert_course(v_course_id, v_course_name, v_department_id, v_description, v_credits, v_start_date,
v_end_date, v_created_by);

END;
```

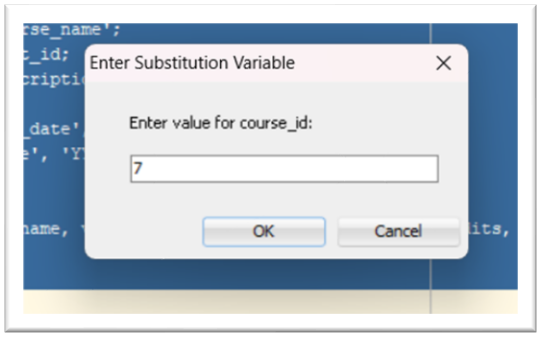


Figure 3.4.14.1

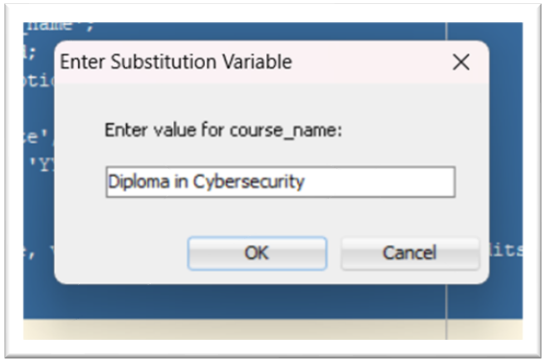


Figure 3.4.14.2

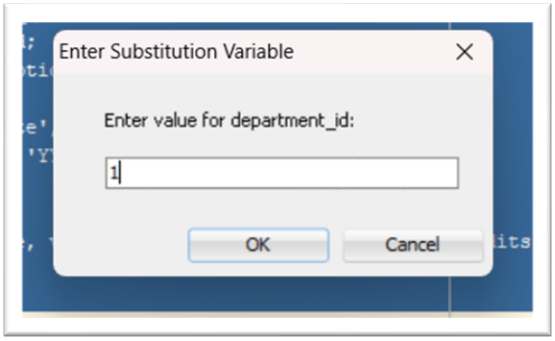


Figure 3.4.14.3

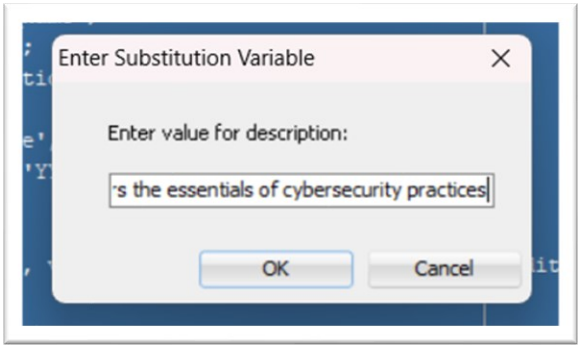


Figure 3.4.14.4

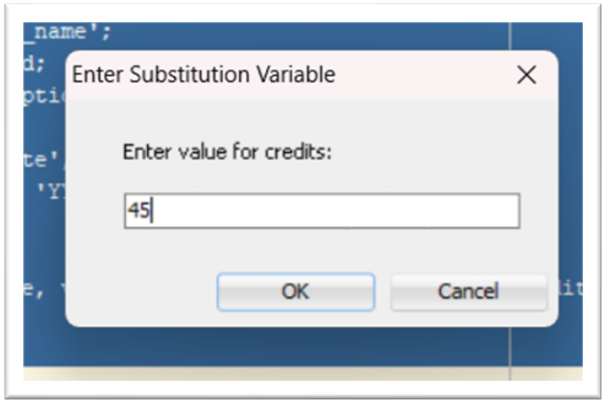


Figure 3.4.14.5

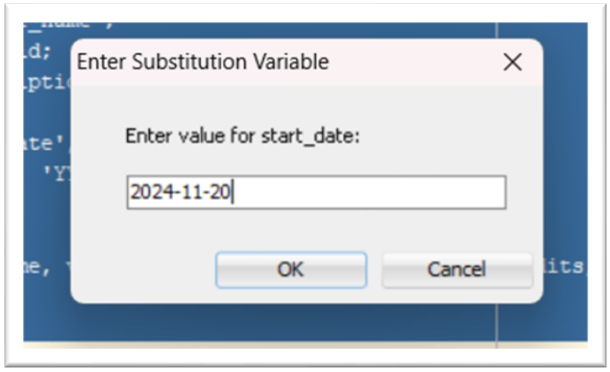


Figure 3.4.14.6

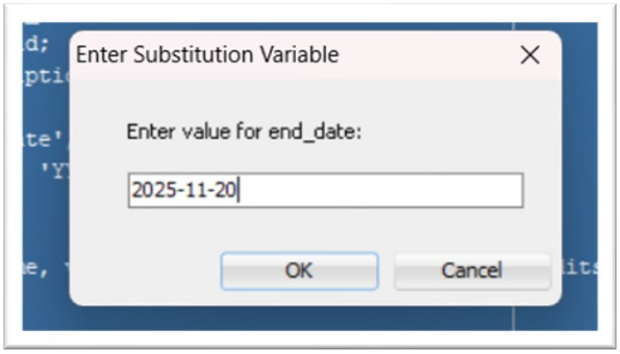


Figure 3.4.14.7

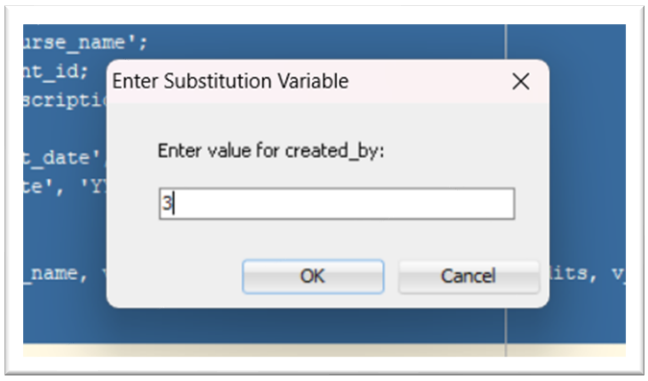


Figure 3.4.14.8

Figure 3.4.14.9

SELECT * FROM courses;

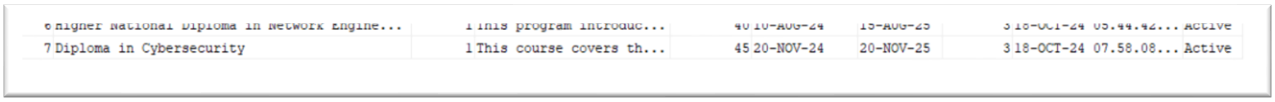


Figure 3.4.14.10

3.4.15 Create Procedure to update course Record

CREATE OR REPLACE PROCEDURE update_course (p_course_id IN NUMBER, p_course_name IN VARCHAR2, p_department_id IN NUMBER, p_description IN VARCHAR2,

p_credits IN NUMBER,p_start_date IN DATE, p_end_date IN DATE, p_status IN VARCHAR2) AS

BEGIN

UPDATE courses SET course_name = p_course_name, department_id = p_department_id, description = p_description, credits = p_credits,

start_date = p_start_date, end_date = p_end_date,status = p_status WHERE course_id = p_course_id;

IF SQL%ROWCOUNT = 0 THEN

DBMS_OUTPUT.PUT_LINE('No course found');

```
    ELSE
        DBMS_OUTPUT.PUT_LINE('Course successfully updated');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

Figure 3.4.15

3.4.16 Update Course Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
    v_course_id NUMBER := &course_id;
    v_course_name VARCHAR2(100) := '&course_name';
    v_department_id NUMBER := &department_id;
    v_description VARCHAR2(200) := '&description';
    v_credits NUMBER := &credits;
    v_start_date DATE := TO_DATE('&start_date', 'YYYY-MM-DD');
    v_end_date DATE := TO_DATE('&end_date', 'YYYY-MM-DD');
    v_status VARCHAR2(20) := '&status';
BEGIN
    update_course(v_course_id, v_course_name, v_department_id, v_description, v_credits,
v_start_date, v_end_date, v_status);
END;
```

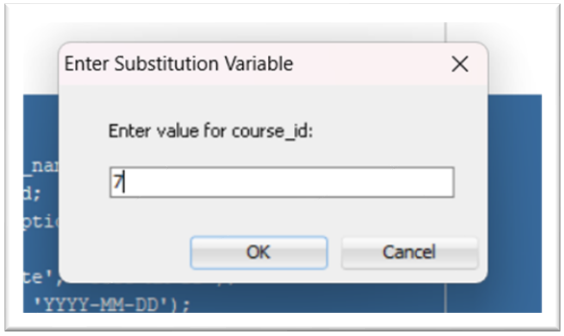


Figure 3.4.16.1

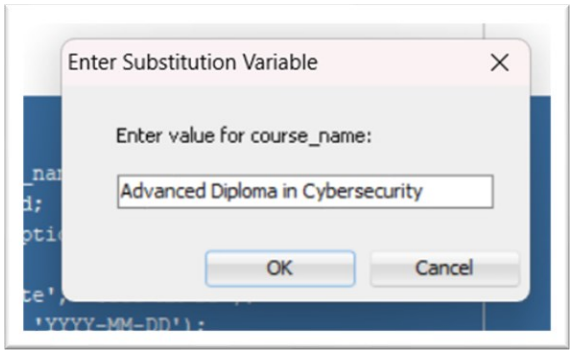


Figure 3.4.16.2

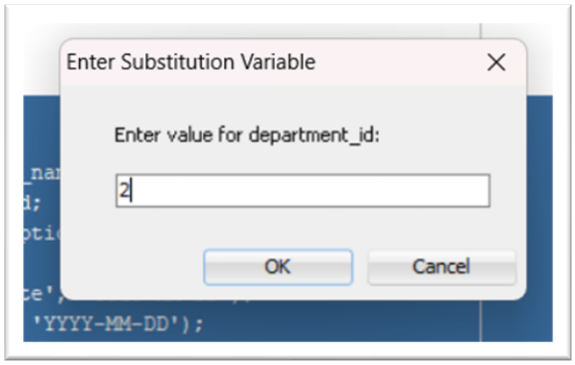


Figure 3.4.16.3

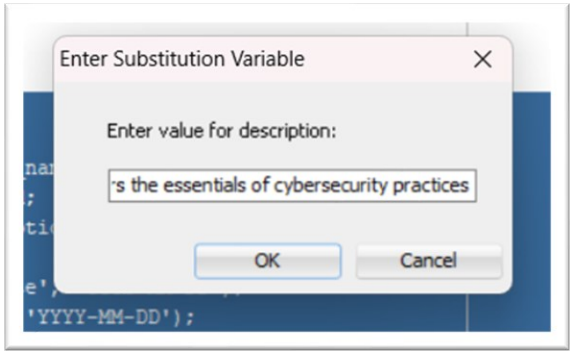


Figure 3.4.16.4

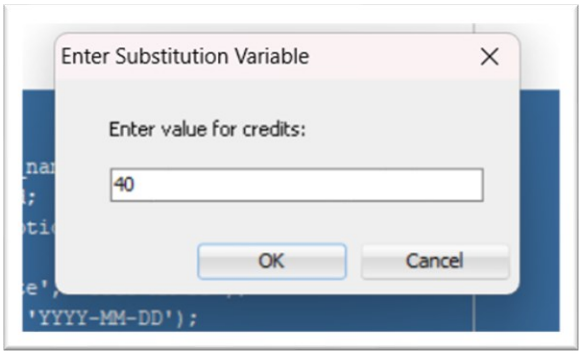


Figure 3.4.16.5

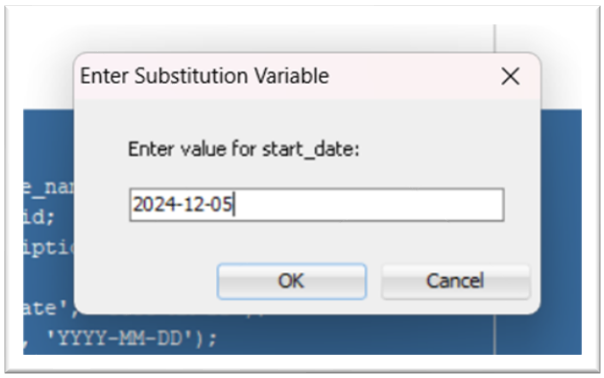


Figure 3.4.16.6

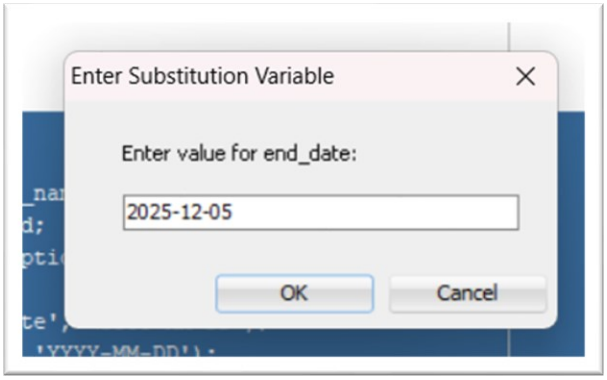


Figure 3.4.16.7

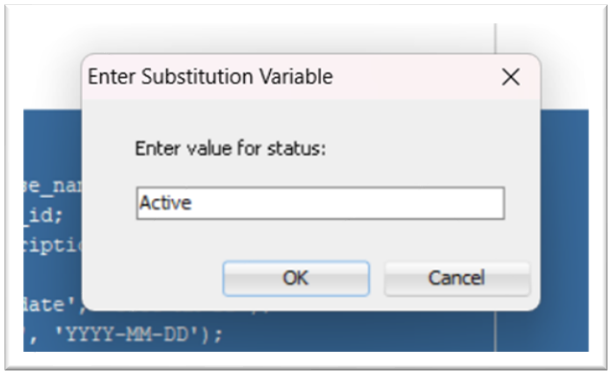


Figure 3.4.16.8

Figure 3.4.16.9

SELECT * FROM courses;

6	Higher National Diploma in Network Engine...	1	This program introduc...	40	10-AUG-24	15-AUG-25	3	18-OCT-24 05.44.42... Active
7	Advanced Diploma in Cybersecurity	2	his course covers the...	40	05-DEC-24	05-DEC-25	3	18-OCT-24 07.58.08... Active

Figure 3.4.16.10

3.4.17 Create Procedure to delete course Record

```
CREATE OR REPLACE PROCEDURE delete_course (p_course_id IN NUMBER) AS
BEGIN
   DELETE FROM courses WHERE course_id = p_course_id;
   IF SQL%ROWCOUNT = 0 THEN
      DBMS_OUTPUT.PUT_LINE('No course found');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Course successfully deleted');
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

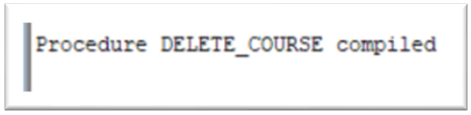


Figure 3.4.17

3.4.18 Delete Course Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
   v_course_id NUMBER := &course_id;
BEGIN
   delete_course(v_course_id);
END;
```

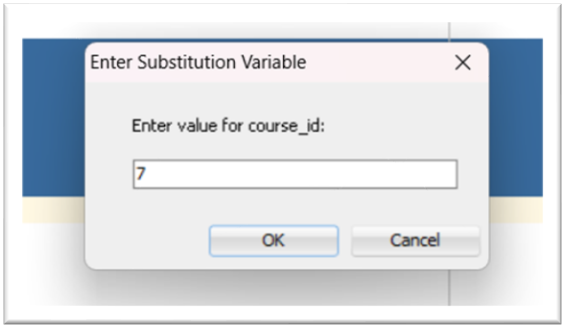


Figure 3.4.18.1

```
END;
Course successfully deleted
```

Figure 3.4.18.2

SELECT * FROM courses;

4	Higher National Diploma in Business Manag...	2	This program introduc...	35	10-MAY-24	15-MAY-25	3	18-OCT-24 05.44.42... Active
5	Diploma in English	4	This program introduc...	20	10-MAY-24	15-NOV-25	3	18-OCT-24 05.44.42... Active
6	Higher National Diploma in Network Engine...	1	This program introduc...	40	10-AUG-24	15-AUG-25	3	18-OCT-24 05.44.42... Active

Figure 3.4.18.3

3.4.19 Create Procedure to insert lesson Record

```
CREATE OR REPLACE PROCEDURE insert_lesson ( p_lesson_id IN NUMBER, p_course_id IN NUMBER,
p_lesson_name IN VARCHAR2,
   p_taught_by IN NUMBER, p_created_by IN NUMBER) AS
BEGIN
   INSERT INTO lessons (lesson_id, course_id, lesson_name, taught_by, created_by)
   VALUES (p_lesson_id, p_course_id, p_lesson_name, p_taught_by, p_created_by);
   DBMS_OUTPUT.PUT_LINE('Lesson successfully added');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred');
```

```
END;
```

```
Procedure INSERT_LESSON compiled
```

Figure 3.4.19

3.4.20 Insert Lesson Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
   v_lesson_id NUMBER := &lesson_id;
   v_course_id NUMBER := &course_id;
   v_lesson_name VARCHAR2(100) := '&lesson_name';
   v_taught_by NUMBER := &taught_by;
   v_created_by NUMBER := &created_by;
BEGIN
   insert_lesson(v_lesson_id, v_course_id, v_lesson_name, v_taught_by, v_created_by);
END;
/
```

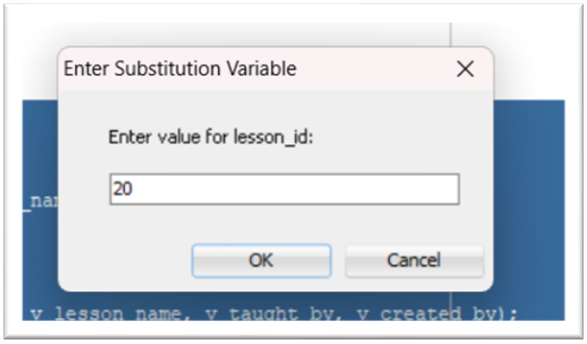


Figure 3.4.20.1

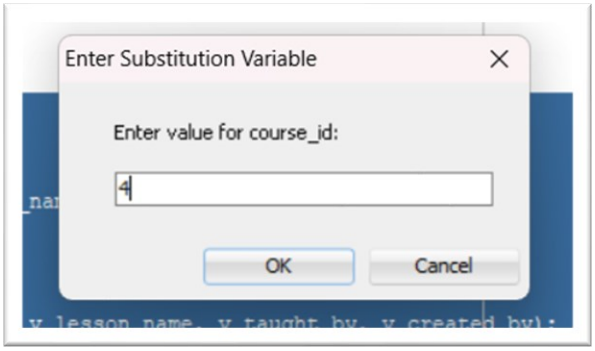


Figure 3.4.20.2

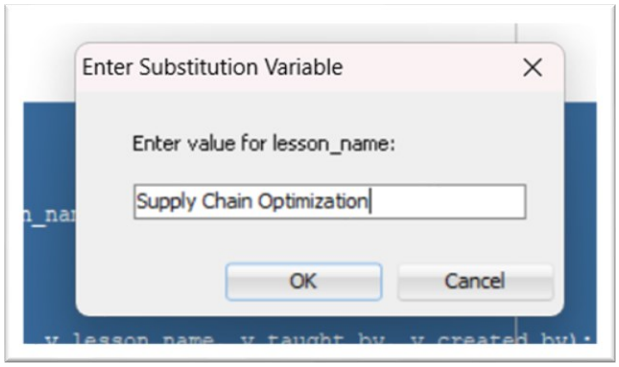


Figure 3.4.20.3

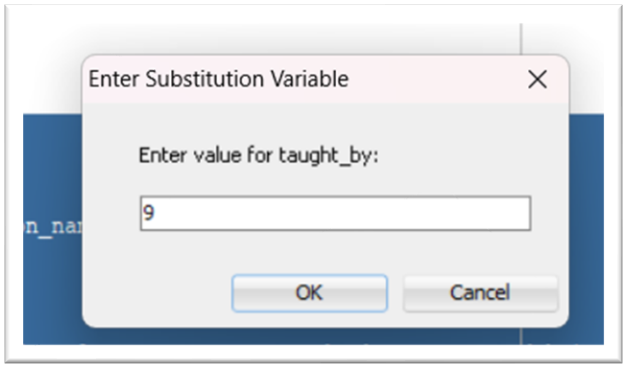


Figure 3.4.20.4

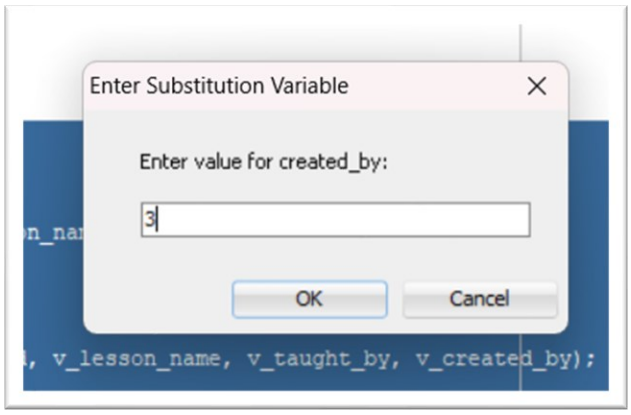


Figure 3.4.20.5

Figure 3.4.20.6

SELECT * FROM lessons;

19	19	4 Operations Logistics Management	9 Active	3 18-OCT-24 05.50.06.776000000 PM	
20	20	4 Supply Chain Optimization	9 Active	3 18-OCT-24 08.25.15.071000000 PM	

Figure 3.4.20.7

3.4.21 Create Procedure to update lesson Record

CREATE OR REPLACE PROCEDURE update_lesson (p_lesson_id IN NUMBER, p_lesson_name IN VARCHAR2,

p_status IN VARCHAR2,p_taught_by IN NUMBER) AS v_lesson_count NUMBER;

BEGIN

SELECT COUNT(*) INTO v_lesson_count FROM lessons WHERE lesson_id = p_lesson_id;

IF v_lesson_count = 0 THEN

DBMS_OUTPUT.PUT_LINE('No lesson found');

ELSE

```
    UPDATE lessons SET lesson_name = p_lesson_name, status = p_status , taught_by = p_taught_by
WHERE lesson_id = p_lesson_id;
    DBMS_OUTPUT.PUT_LINE('Lesson with ID ' || p_lesson_id || ' has been updated.');
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

```
Procedure UPDATE_LESSON compiled
```

Figure 3.4.21

3.4.22 Update Lesson Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
  v_lesson_id NUMBER := &lesson_id;
  v_lesson_name VARCHAR2(100) := '&lesson_name';
  v_status VARCHAR2(20) := '&status';
  v_taught_by NUMBER := &taught_by;
BEGIN
  update_lesson(v_lesson_id, v_lesson_name, v_status, v_taught_by);
END;
/
```

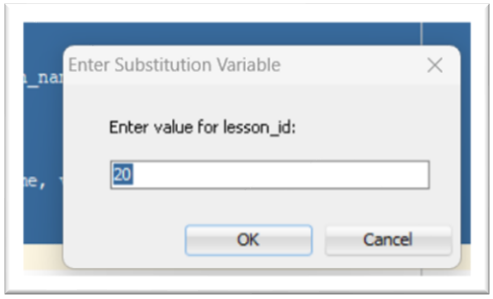


Figure 3.4.22.1

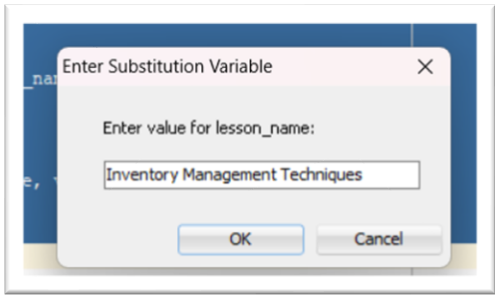


Figure 3.4.22.2

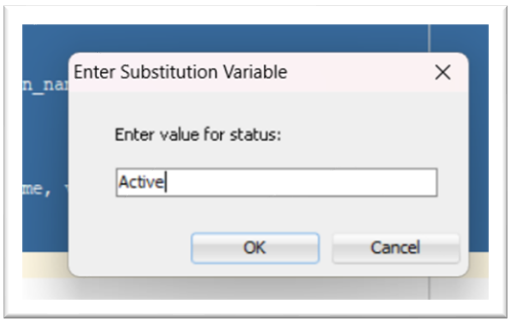


Figure 3.4.22.3

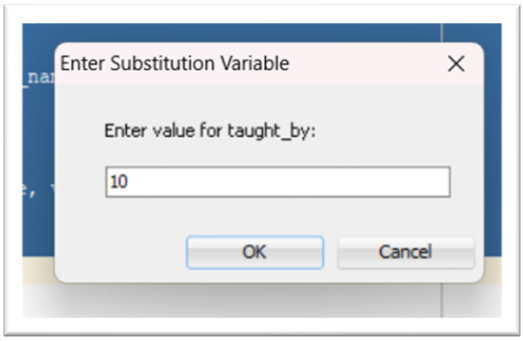


Figure 3.4.22.4

```
END;
Lesson with ID 20 has been updated.
```

Figure 3.4.22.5

SELECT * FROM lessons;

19	4	Operations Logistics Management	9	Active	3	18-OCT-24 05.50.06.776000000 PM
20	4	Inventory Management Techniques	10	Active	3	18-OCT-24 08.25.15.071000000 PM

Figure 3.4.22.6

3.4.23 Create Procedure to delete lesson Record

```
CREATE OR REPLACE PROCEDURE delete_lesson ( p_lesson_id IN NUMBER) AS v_lesson_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_lesson_count FROM lessons WHERE lesson_id = p_lesson_id;
   IF v_lesson_count = 0 THEN
      DBMS_OUTPUT.PUT_LINE('No lesson found');
   ELSE
      DELETE FROM lessons WHERE lesson_id = p_lesson_id;
      DBMS_OUTPUT.PUT_LINE('Lesson deleted.');
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

```
Procedure DELETE_LESSON compiled
```

Figure 3.4.23

3.4.24 Delete Lesson Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
   v_lesson_id NUMBER := &lesson_id;
BEGIN
   delete_lesson(v_lesson_id);
END;
/
```

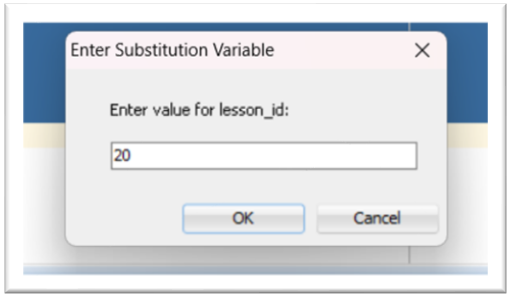


Figure 3.4.24.1

```
END;
Lesson deleted.
```

Figure 3.4.24.2

3.4.25 Create Procedure to insert student Record

```
CREATE OR REPLACE PROCEDURE insert_student (p_first_name IN VARCHAR2, p_last_name IN
VARCHAR2, p_email IN VARCHAR2,
   p_phone_number IN VARCHAR2, p_date_of_birth IN DATE, p_gender IN VARCHAR2, p_address IN
VARCHAR2) AS
BEGIN
   INSERT INTO students (first_name, last_name, email, phone_number, date_of_birth, gender, address)
```

```
    VALUES (p_first_name, p_last_name, p_email, p_phone_number, p_date_of_birth, p_gender,
p_address);
    DBMS_OUTPUT.PUT_LINE('Student successfully added.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

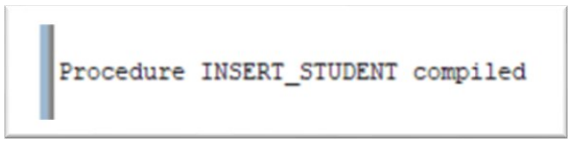


Figure 3.4.25

3.4.26 Insert Student Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
    v_first_name VARCHAR2(50) := '&first_name';
    v_last_name VARCHAR2(50) := '&last_name';
    v_email VARCHAR2(100) := '&email';
    v_phone_number VARCHAR2(15) := '&phone_number';
    v_date_of_birth DATE := TO_DATE('&date_of_birth', 'YYYY-MM-DD');
    v_gender VARCHAR2(15) := '&gender';
    v_address VARCHAR2(200) := '&address';
BEGIN
    insert_student(v_first_name, v_last_name, v_email, v_phone_number, v_date_of_birth, v_gender,
v_address);
END;
```

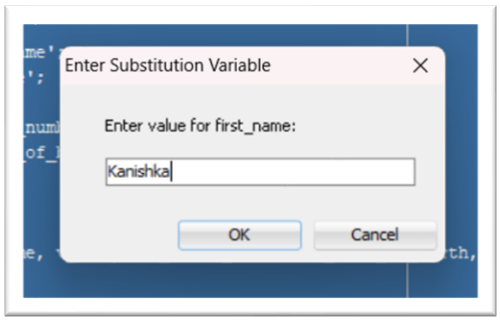


Figure 3.4.26.1

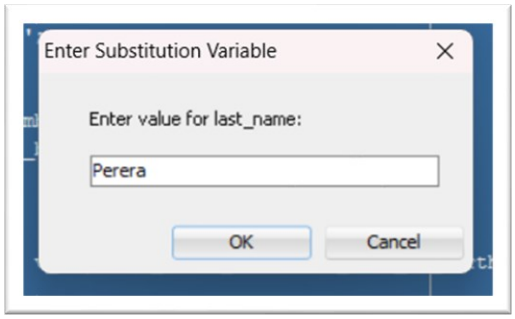


Figure 3.4.26.2

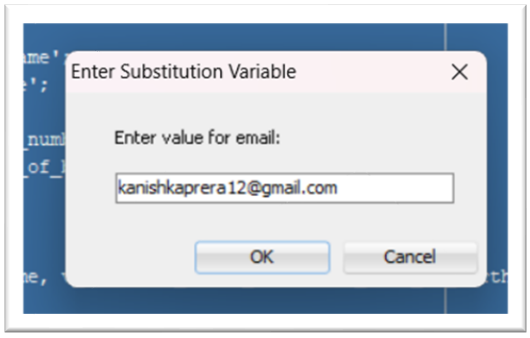


Figure 3.4.26.3

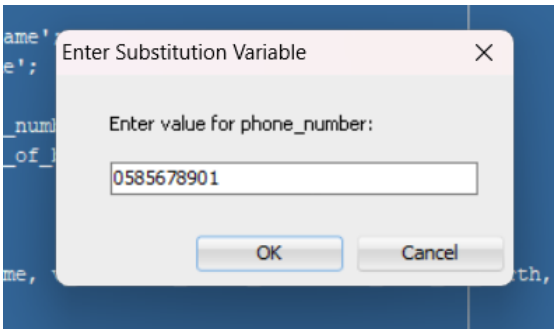


Figure 3.4.26.4

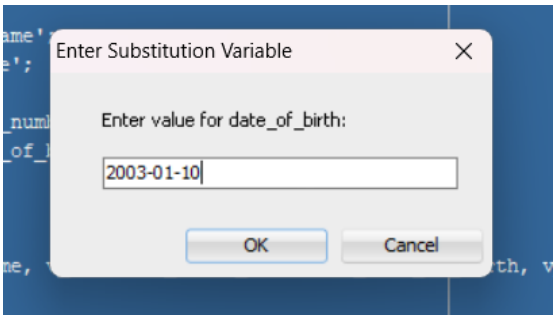


Figure 3.4.26.5

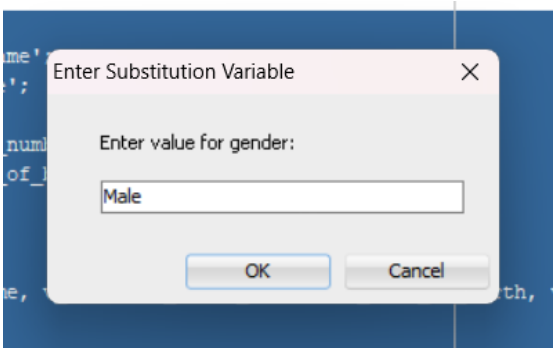


Figure 3.4.26.6

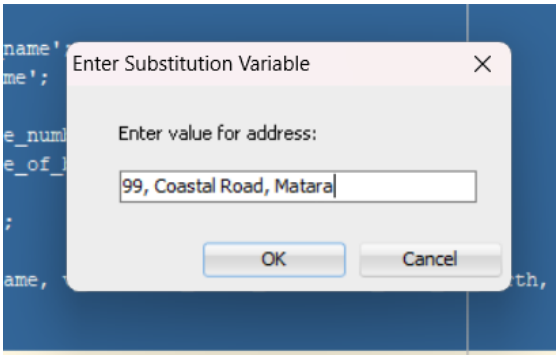


Figure 3.4.26.7

Figure 3.4.26.8

SELECT * FROM students;

J	40	Chamila	Gunarathne	chamilagunarathne@gmail.com	0421234567	29-JUL-00	Female	12, Lake View, Kandy	Active	18-OCT-24 05.56.09.011000000 PM
1	41	Kanishka	Perera	kanishkaperera12@gmail.com	0585678901	10-JAN-03	Male	99, Coastal Road, ...	Active	18-OCT-24 09.51.20.907000000 PM

Figure 3.4.26.9

3.4.27 Create Procedure to update student Record

```
CREATE OR REPLACE PROCEDURE update_student ( p_student_id IN NUMBER, p_first_name IN
VARCHAR2, p_last_name IN VARCHAR2, p_email IN VARCHAR2,
   p_phone_number IN VARCHAR2, p_date_of_birth IN DATE, p_gender IN VARCHAR2, p_address IN
VARCHAR2, p_status IN VARCHAR2) AS
   v_student_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_student_count FROM students WHERE student_id = p_student_id;
   IF v_student_count = 0 THEN
      DBMS_OUTPUT.PUT_LINE('No student found');
   ELSE
      UPDATE students SET first_name = p_first_name, last_name = p_last_name, email = p_email,
phone_number = p_phone_number,
         date_of_birth = p_date_of_birth, gender = p_gender, address = p_address, status = p_status
      WHERE student_id = p_student_id;
      DBMS_OUTPUT.PUT_LINE('Student updated.');
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred');
```

```
END;
```

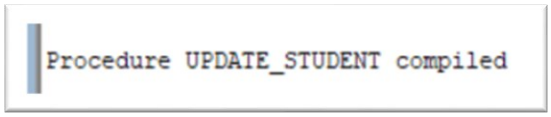


Figure 3.4.27

3.4.28 Update Student Using Variables

```
SET SERVEROUTPUT ON;

DECLARE

   v_student_id NUMBER := &student_id;

   v_first_name VARCHAR2(50) := '&first_name';

   v_last_name VARCHAR2(50) := '&last_name';

   v_email VARCHAR2(100) := '&email';

   v_phone_number VARCHAR2(15) := '&phone_number';

   v_date_of_birth DATE := TO_DATE('&date_of_birth', 'YYYY-MM-DD');

   v_gender VARCHAR2(15) := '&gender';

   v_address VARCHAR2(200) := '&address';

   v_status VARCHAR2(20) := '&status';

BEGIN

   update_student(v_student_id, v_first_name, v_last_name, v_email, v_phone_number,
v_date_of_birth, v_gender, v_address, v_status);

END;
```

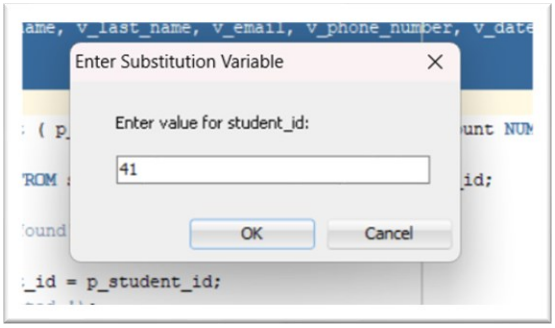


Figure 3.4.28.1

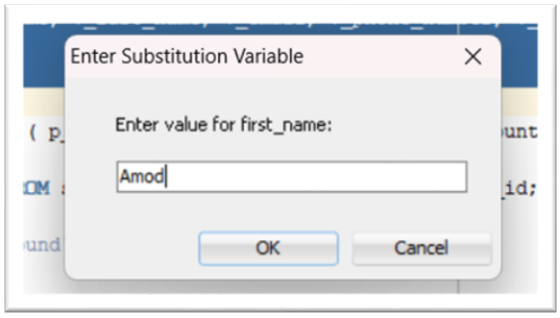


Figure 3.4.28.2

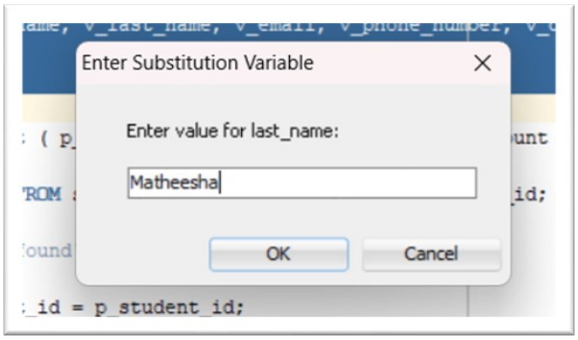


Figure 3.4.28.3

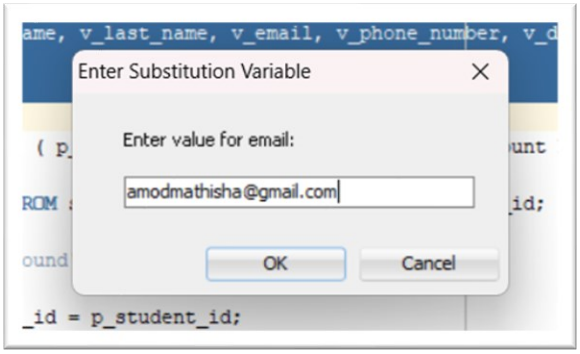


Figure 3.4.28.4

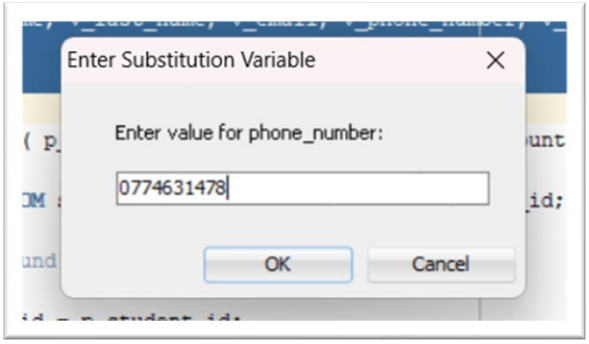


Figure 3.4.28.5

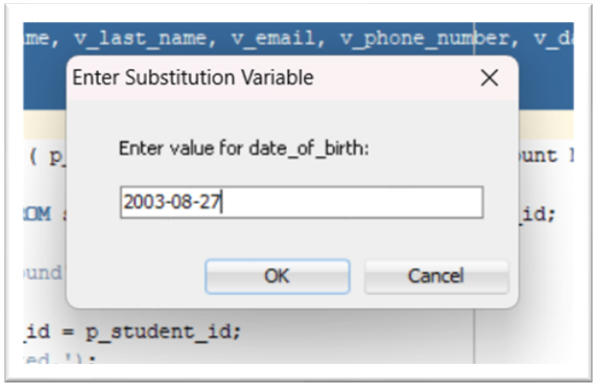


Figure 3.4.28.6

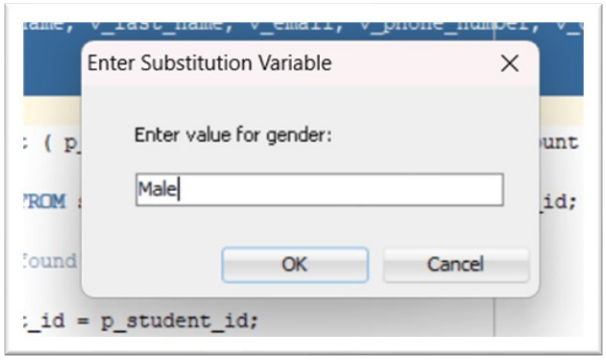


Figure 3.4.28.7

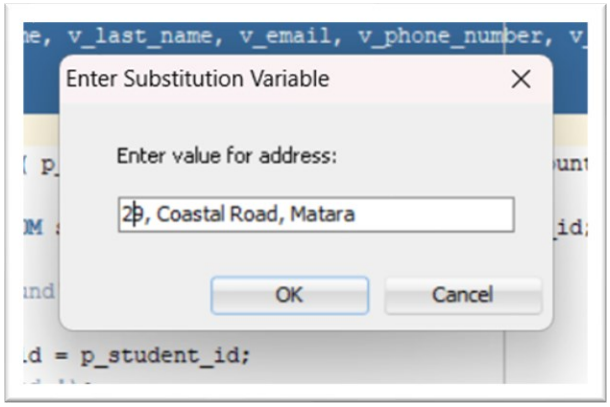


Figure 3.4.28.8

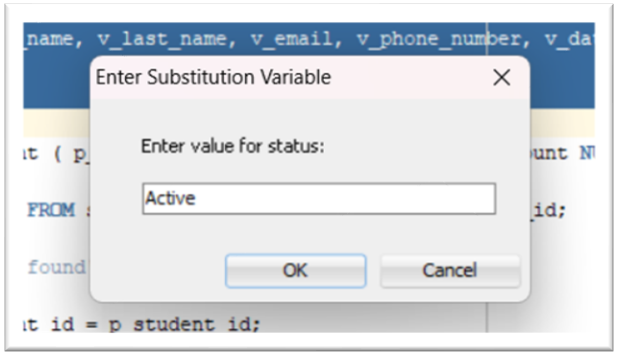


Figure 3.4.28.9

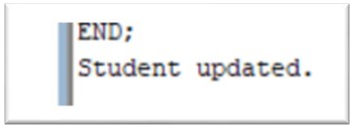


Figure 3.4.28.10

SELECT * FROM students;

41	41	Amod	Matheesha	amodmathisha@gmail.com	0774631478	27-AUG-03	Male	29, Coastal Road, ...	Active	18-OCT-24 09.51.20.907000000 PM

Figure 3.4.28.11

3.4.29 Create Procedure to delete student Record

```
CREATE OR REPLACE PROCEDURE delete_student ( p_student_id IN NUMBER) AS v_student_count
NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_student_count FROM students WHERE student_id = p_student_id;
   IF v_student_count = 0 THEN
      DBMS_OUTPUT.PUT_LINE('No student found');
   ELSE
      DELETE FROM students WHERE student_id = p_student_id;
      DBMS_OUTPUT.PUT_LINE('Student deleted.');
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred ');
END;
```

```
Procedure DELETE_STUDENT compiled
```

Figure 3.4.29

3.4.30 Delete Student Using Variable

```
SET SERVEROUTPUT ON;
DECLARE
   v_student_id NUMBER := &student_id;
```

```
BEGIN
   delete_student(v_student_id);
END;
```

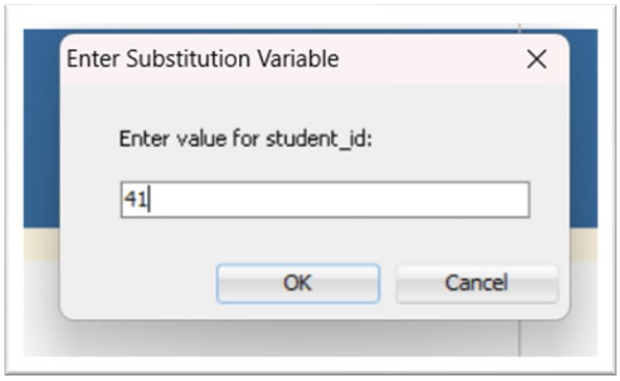


Figure 3.4.30.1

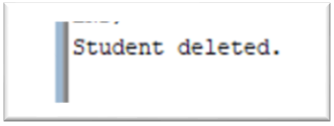


Figure 3.4.30.2

3.4.31 Create Procedure to insert course_enrollment Record

```
CREATE OR REPLACE PROCEDURE insert_course_enrollment ( p_student_id IN NUMBER, p_course_id IN
NUMBER) AS
BEGIN
   INSERT INTO course_enrollments (student_id, course_id) VALUES (p_student_id, p_course_id);
   DBMS_OUTPUT.PUT_LINE('Course enrollment added successfully.');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred');
END ;
```

```
Procedure INSERT_COURSE_ENROLLMENT compiled
```

Figure 3.4.31

3.4.32 Enroll Student in Course Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
   v_student_id NUMBER := &student_id;
   v_course_id NUMBER := &course_id;
BEGIN
   insert_course_enrollment(v_student_id, v_course_id);
END;
```

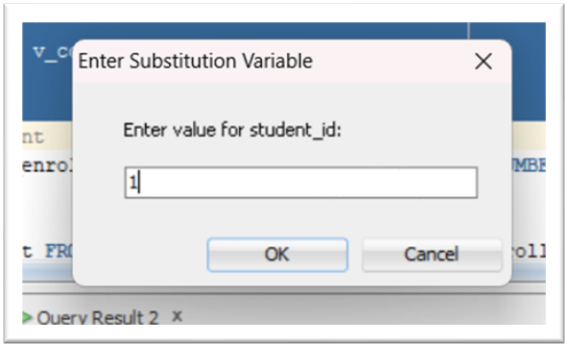


Figure 3.4.32.1

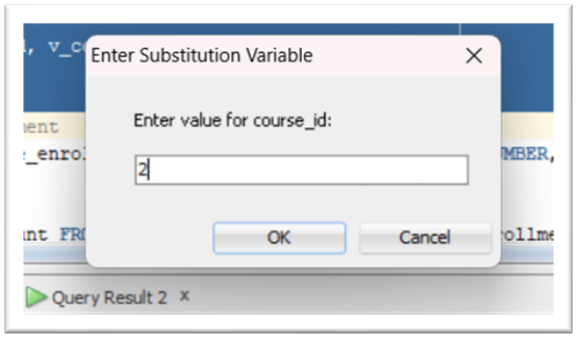


Figure 3.4.32.2

Figure 3.4.32.3

SELECT * FROM course_enrollments;

Figure 3.4.32.4

3.4.33 Create Procedure to update course_enrollment Record

```
CREATE OR REPLACE PROCEDURE update_course_enrollment ( p_course_enrollments_id IN NUMBER, p_student_id IN NUMBER, p_course_id IN NUMBER) AS
   v_enrollment_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_enrollment_count FROM course_enrollments WHERE course_enrollments_id = p_course_enrollments_id;
   IF v_enrollment_count = 0 THEN
      DBMS_OUTPUT.PUT_LINE('No course enrollment found');
   ELSE
      UPDATE course_enrollments SET student_id = p_student_id, course_id = p_course_id WHERE course_enrollments_id = p_course_enrollments_id;
      DBMS_OUTPUT.PUT_LINE('Course enrollment updated');
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

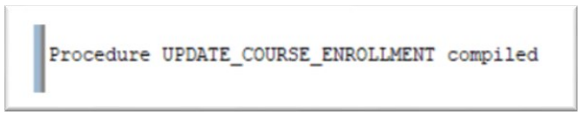


Figure 3.4.33

3.4.34 Update Course Enrollment Using Variables

```
SET SERVEROUTPUT ON;
DECLARE
   v_course_enrollments_id NUMBER := &course_enrollments_id;
   v_student_id NUMBER := &student_id;
   v_course_id NUMBER := &course_id;
BEGIN
   update_course_enrollment(v_course_enrollments_id, v_student_id, v_course_id);
END;
```

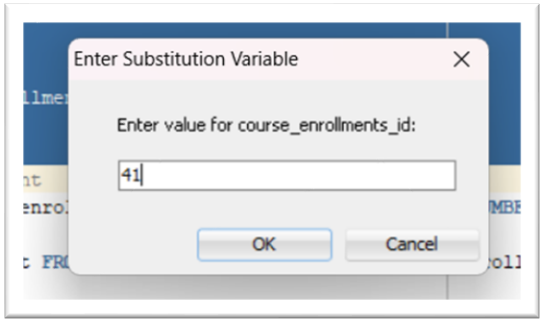


Figure 3.4.34.1

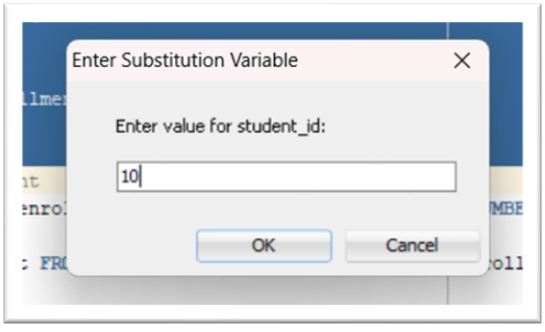


Figure 3.4.34.2

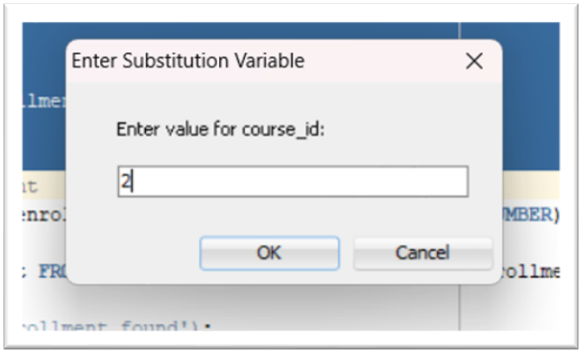


Figure 3.4.34.3

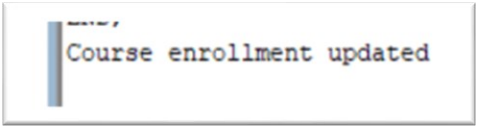


Figure 3.4.34.4

SELECT * FROM course_enrollments;

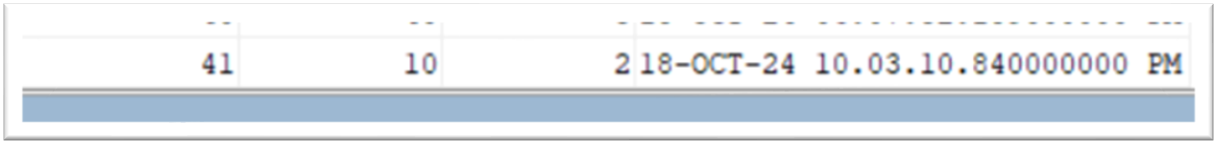


Figure 3.4.34.5

3.4.35 Create Procedure to delete course_enrollment Record

CREATE OR REPLACE PROCEDURE delete_course_enrollment (p_course_enrollments_id IN NUMBER) AS v_enrollment_count NUMBER;

BEGIN

SELECT COUNT(*) INTO v_enrollment_count FROM course_enrollments WHERE course_enrollments_id = p_course_enrollments_id;

IF v_enrollment_count = 0 THEN

DBMS_OUTPUT.PUT_LINE('No course enrollment found');

ELSE

DELETE FROM course_enrollments WHERE course_enrollments_id = p_course_enrollments_id;

DBMS_OUTPUT.PUT_LINE('Course enrollment deleted.');

```
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
```

```
Procedure DELETE_COURSE_ENROLLMENT compiled
```

Figure 3.4.35

3.4.36 Delete Course Enrollment Using Variable

```
SET SERVEROUTPUT ON;
DECLARE
    v_course_enrollments_id NUMBER := &course_enrollments_id;
BEGIN
    delete_course_enrollment(v_course_enrollments_id);
END;
```

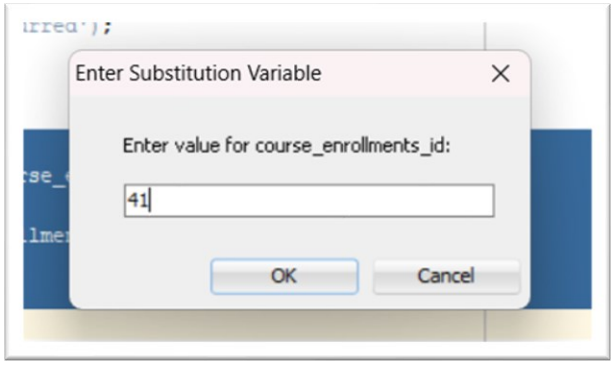


Figure 3.4.36.1

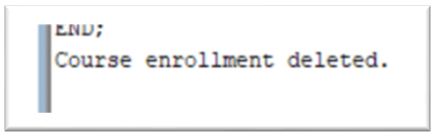


Figure 3.4.36.2

3.4.37 Create Procedure to Insert Student Feedback

```
CREATE OR REPLACE PROCEDURE insert_student_feedback (p_course_id IN NUMBER,p_student_id IN NUMBER,p_comments IN VARCHAR2,p_rating IN NUMBER) AS
   v_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO v_count FROM course_enrollments WHERE student_id = p_student_id AND course_id = p_course_id;
   IF v_count > 0 THEN
      INSERT INTO students_feedback (lesson_id, student_id, comments, rating, post_date) VALUES (p_course_id, p_student_id, p_comments, p_rating, SYSTIMESTAMP);
      DBMS_OUTPUT.PUT_LINE('Feedback inserted successfully.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('The student is not related to the specified lesson');
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error occurred');
END;
/
```

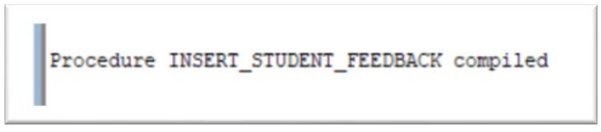


Figure 3.4.37

3.4.38 Insert Student Lesson Feedback

```
SET SERVEROUTPUT ON;
DECLARE
   v_course_id NUMBER;
   v_student_id NUMBER;
   v_comments VARCHAR2(255);
   v_rating NUMBER;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Enter Course ID:');
   v_course_id := &course_id;
   DBMS_OUTPUT.PUT_LINE('Enter Student ID:');
   v_student_id := &student_id;
   DBMS_OUTPUT.PUT_LINE('Enter Comments:');
   v_comments := '&comments';
   DBMS_OUTPUT.PUT_LINE('Enter Rating (1-5):');
   v_rating := &rating;
   ADMIN.insert_student_feedback(v_course_id, v_student_id, v_comments, v_rating);
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

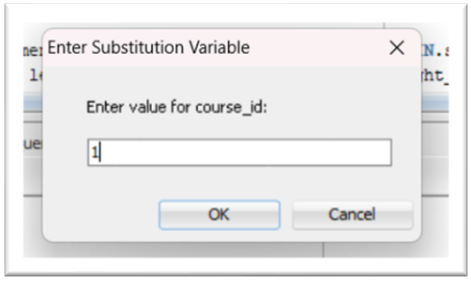


Figure 3.4.38.1

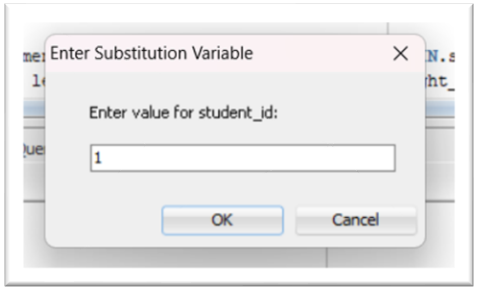


Figure 3.4.38.2

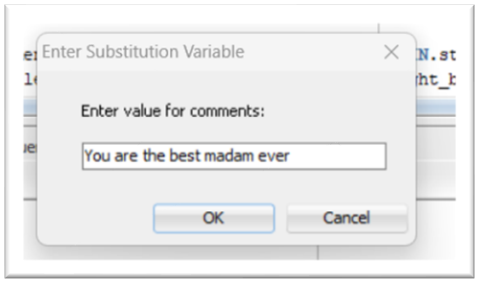


Figure 3.4.38.3

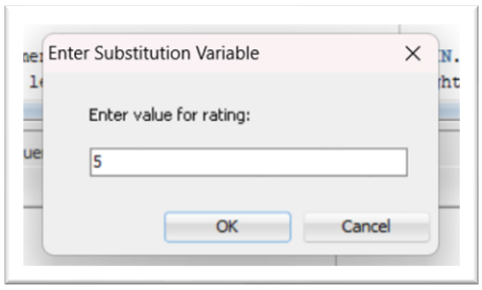


Figure 3.4.38.4

```
END;
Enter Course ID:
Enter Student ID:
Enter Comments:
Enter Rating (1-5):
Feedback inserted successfully.
```

Figure 3.4.38.5

SELECT * FROM students_feedback;

FEEDBACK_ID	LESSON_ID	STUDE...	COMMENTS	RATING	POST_DATE
1	1	1	You are the best madam ever	5	18-OCT-24 11.42.21.475000000 PM

Figure 3.4.38.6

3.4.39 Insert Student Lesson Feedback with Invalid Student ID

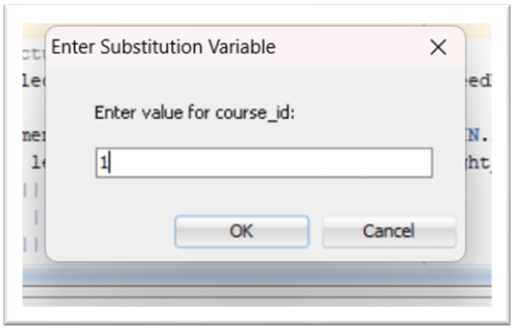


Figure 3.4.39.1

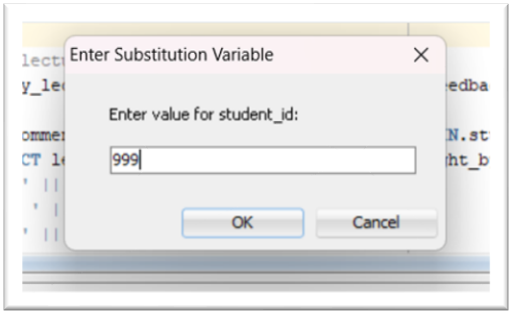


Figure 3.4.39.2

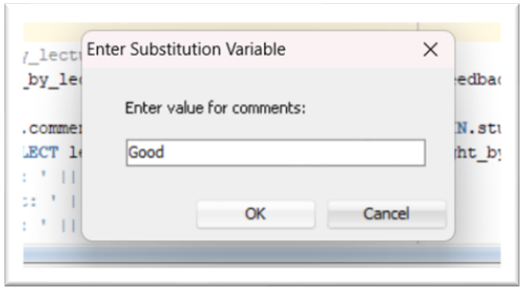


Figure 3.4.39.3

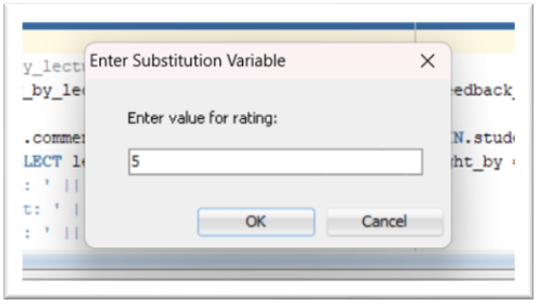


Figure 3.4.39.4

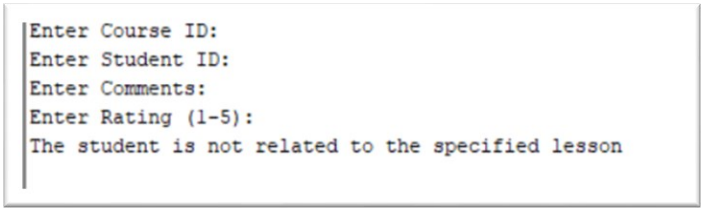


Figure 3.4.39.5

INSERT INTO students_feedback (lesson_id, student_id, comments, rating)

VALUES (1, 2, 'I enjoyed the interactive activities and practical examples.', 5);

INSERT INTO students_feedback (lesson_id, student_id, comments, rating)

VALUES (1, 11, 'Very detailed and helpful. I learned a lot from this lesson.', 4);

INSERT INTO students_feedback (lesson_id, student_id, comments, rating)

VALUES (1, 19, 'The lesson was engaging, but the lecture duration was too long.', 3);

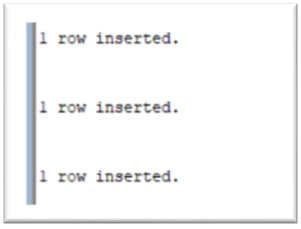


Figure 3.4.39.6

3.4.40 Create Procedure to Retrieve Feedback by Lecturer

```
CREATE OR REPLACE PROCEDURE get_feedback_by_lecturer (p_lecturer_id IN NUMBER) AS
feedback_found BOOLEAN := FALSE;

BEGIN

   FOR rec IN (SELECT l.lesson_name, sf.comments, sf.rating, sf.post_date FROM
ADMIN.students_feedback sf JOIN lessons l ON sf.lesson_id = l.lesson_id

            WHERE l.lesson_id IN (SELECT lesson_id FROM ADMIN.lessons WHERE taught_by =
p_lecturer_id)) LOOP

         DBMS_OUTPUT.PUT_LINE('Lesson: ' || rec.lesson_name );

         DBMS_OUTPUT.PUT_LINE('Comment: ' || rec.comments );

         DBMS_OUTPUT.PUT_LINE('Rating: ' || rec.rating );

         DBMS_OUTPUT.PUT_LINE('Date: ' || rec.post_date );

         DBMS_OUTPUT.PUT_LINE(' ');

         feedback_found := TRUE;

   END LOOP;

   IF NOT feedback_found THEN

      DBMS_OUTPUT.PUT_LINE('No feedback found');

   END IF;

EXCEPTION

   WHEN OTHERS THEN

      DBMS_OUTPUT.PUT_LINE('An error occurred');

END get_feedback_by_lecturer;

/
```

```
Procedure GET_FEEDBACK_BY_LECTURER compiled
```

Figure 3.4.40

3.4.41 Fetch Feedback by Lecturer ID

```
SET SERVEROUTPUT ON;
DECLARE
   v_lecturer_id NUMBER;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Enter Lecturer ID:');
   v_lecturer_id := &lecturer_id;
   ADMIN.get_feedback_by_lecturer(v_lecturer_id);
END;
/
```

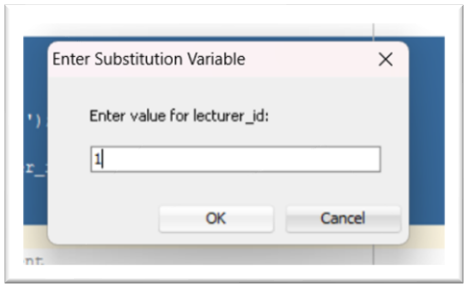


Figure 3.4.41.1

```
Enter Lecturer ID:
Lesson: Introduction to Computer Science
Comment: You are the best madam ever
Rating: 5
Date: 18-OCT-24 11.42.21.475000 PM

Lesson: Introduction to Computer Science
Comment: I enjoyed the interactive activities and practical examples.
Rating: 5
Date: 18-OCT-24 11.50.09.811000 PM

Lesson: Introduction to Computer Science
Comment: Very detailed and helpful. I learned a lot from this lesson.
Rating: 4
Date: 18-OCT-24 11.50.09.816000 PM

Lesson: Introduction to Computer Science
Comment: The lesson was engaging, but the lecture duration was too long.
Rating: 3
Date: 18-OCT-24 11.50.09.822000 PM
```

Figure 3.4.41.2

3.4.42 Fetch Feedback by Handling Invalid Lecturer ID

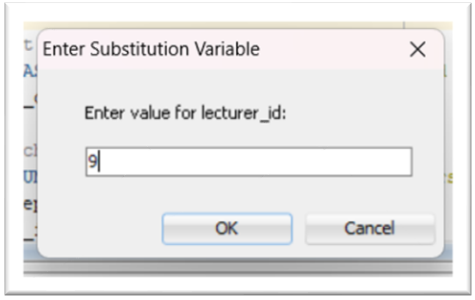


Figure 3.4.42.1

```
Enter Lecturer ID:
No feedback found
```

Figure 3.4.42.2

3.4.43 Procedure to Add a Schedule

```
CREATE OR REPLACE PROCEDURE add_schedule_proc(
   p_lesson_id lessons.lesson_id%TYPE,
   p_schedule_date schedule.schedule_date%TYPE,
   p_lecture_hall schedule.lecture_hall%TYPE) IS
BEGIN
   INSERT INTO schedule (lesson_id, schedule_date, lecture_hall)
   VALUES (p_lesson_id, p_schedule_date, p_lecture_hall);
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error');
END;
/
```

```
Procedure ADD_SCHEDULE_PROC compiled
```

Figure 3.4.43

3.4.44 Execute Schedule Addition Procedure

```
BEGIN
   add_schedule_proc(1, TO_DATE('2024-10-26', 'YYYY-MM-DD'), 'Lecture Hall 05');
END;
/
```

```
PL/SQL procedure successfully completed.
```

Figure 3.4.44.1

select * from schedule;

SCHEDULE_ID	LESSON_ID	SCHEDULE_DATE	LECTURE_HALL
1	1	22-OCT-24	Lecture Hall 01
2	2	23-OCT-24	Lecture Hall 02
3	1	24-OCT-24	Lecture Hall 03
4	4	25-OCT-24	Lecture Hall 04
5	1	26-OCT-24	Lecture Hall 05
6	1	26-OCT-24	Lecture Hall 05

Figure 3.4.44.2

3.4.45 Create Procedure to insert assignment Record

```
CREATE OR REPLACE PROCEDURE insert_assignment (p_assignment_title IN VARCHAR2,
p_assignment_fiies IN VARCHAR2, p_lesson_id IN NUMBER, p_end_date IN DATE) AS

BEGIN

   INSERT INTO ADMIN.assignment (assignment_title,assignment_fiies,lesson_id,end_date) VALUES
(p_assignment_title, p_assignment_fiies, p_lesson_id, p_end_date);

EXCEPTION

   WHEN OTHERS THEN

      DBMS_OUTPUT.PUT_LINE('Error occurred try agin');

END;

/
```

```
Procedure INSERT_ASSIGNMENT compiled
```

Figure 3.4.45.1

```
DELETE FROM assignment_students  WHERE assignment_students_id = 1;
```

```
1 row deleted.
```

Figure 3.4.45.2

SELECT * FROM assignment_students;

Figure 3.4.45.3

DELETE FROM assignment WHERE assignment_id = 1;

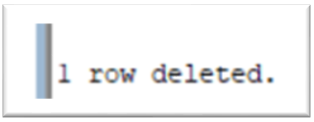


Figure 3.4.45.4

SELECT * FROM assignment;

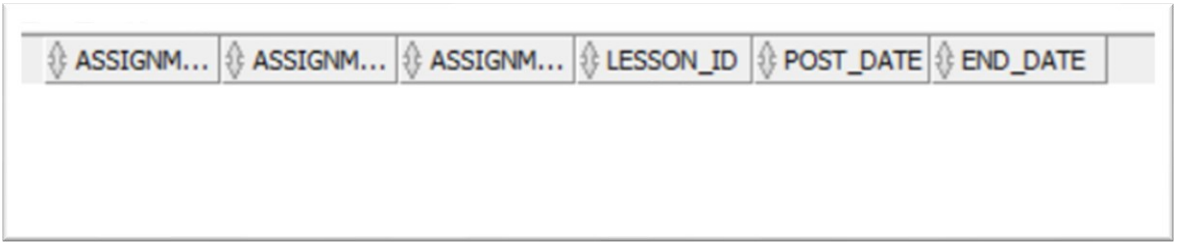


Figure 3.4.45.5

3.4.46 Insert Assignment Using User Input and Variables

```
SET SERVEROUTPUT ON;
DECLARE
   v_assignment_title VARCHAR2(255);
   v_assignment_fiies VARCHAR2(255);
   v_lesson_id NUMBER;
   v_end_date DATE;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Enter Assignment Title:');
   v_assignment_title := '&assignment_title';
```

```
    DBMS_OUTPUT.PUT_LINE('Enter Assignment File Name:');
    v_assignment_fiies := '&assignment_fiies';
    DBMS_OUTPUT.PUT_LINE('Enter Lesson ID:');
    v_lesson_id := &lesson_id;
    DBMS_OUTPUT.PUT_LINE('Enter End Date(YYYY-MM-DD):');
    v_end_date := TO_DATE('&end_date','YYYY-MM-DD');
    ADMIN.insert_assignment(v_assignment_title, v_assignment_fiies, v_lesson_id, v_end_date);
    DBMS_OUTPUT.PUT_LINE('Assignment inserted successfully');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred');
END;
/
```

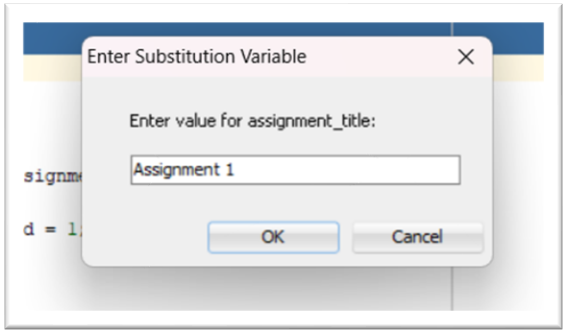


Figure 3.4.46.1

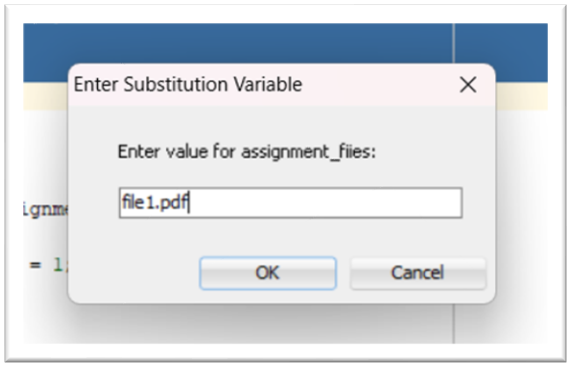


Figure 3.4.46.2

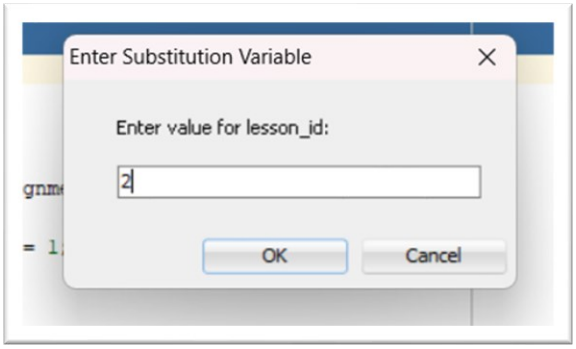


Figure 3.4.46.3

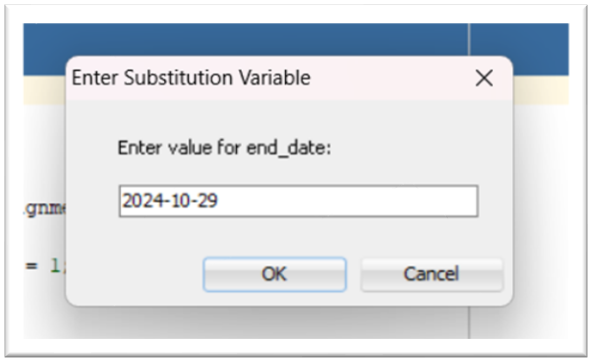


Figure 3.4.46.4

```
Enter Assignment Title:
Enter Assignment File Name:
Enter Lesson ID:
Enter End Date(YYYY-MM-DD):
Assignment inserted successfully


PL/SQL procedure successfully completed.
```

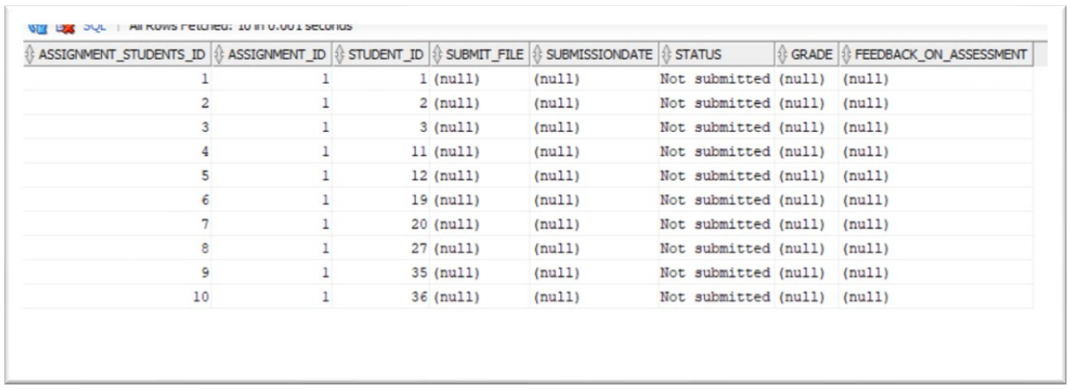
Figure 3.4.46.5

SELECT * FROM assignment;

ASSIGNMENT_ID	ASSIGNMENT_TITLE	ASSIGNMENT_FIIES	LESSON_ID	POST_DATE	END_DATE
1	Assignment 1	file1.pdf	2	18-OCT-24 11.02.14.529000000 PM	29-OCT-24

Figure 3.4.46.6

SELECT * FROM assignment_students;



ASSIGNMENT_STUDENTS_ID	ASSIGNMENT_ID	STUDENT_ID	SUBMIT_FILE	SUBMISSIONDATE	STATUS	GRADE	FEEDBACK_ON_ASSESSMENT
1	1	1	(null)	(null)	Not submitted	(null)	(null)
2	1	2	(null)	(null)	Not submitted	(null)	(null)
3	1	3	(null)	(null)	Not submitted	(null)	(null)
4	1	11	(null)	(null)	Not submitted	(null)	(null)
5	1	12	(null)	(null)	Not submitted	(null)	(null)
6	1	19	(null)	(null)	Not submitted	(null)	(null)
7	1	20	(null)	(null)	Not submitted	(null)	(null)
8	1	27	(null)	(null)	Not submitted	(null)	(null)
9	1	35	(null)	(null)	Not submitted	(null)	(null)
10	1	36	(null)	(null)	Not submitted	(null)	(null)

Figure 3.4.46.7

3.4.47 Create Procedure to Submit Assignment for Student

```
CREATE OR REPLACE PROCEDURE submit_assignment_student (p_assignment_id IN
NUMBER,p_students_id IN NUMBER,p_submit_file IN VARCHAR2) AS

BEGIN

   UPDATE ADMIN.assignment_students SET submit_file = p_submit_file, status = 'Submitted',
submissionDate = SYSTIMESTAMP

   WHERE assignment_id = p_assignment_id AND student_id = p_students_id;

   DBMS_OUTPUT.PUT_LINE('Record updated successfully');

EXCEPTION

   WHEN NO_DATA_FOUND THEN

      DBMS_OUTPUT.PUT_LINE('No record found');

   WHEN OTHERS THEN

      DBMS_OUTPUT.PUT_LINE('Error');

END;

/
```

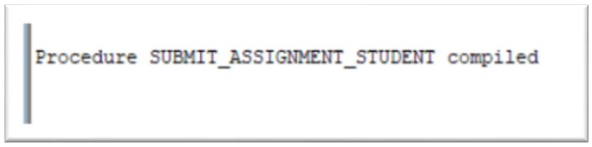


Figure 3.4.47

3.4.48 Submit Assignment for Student Using User Input and Variables

```
SET SERVEROUTPUT ON;
DECLARE
   v_assignment_id NUMBER;
   v_students_id NUMBER;
   v_submit_file VARCHAR2(255);
BEGIN
   DBMS_OUTPUT.PUT_LINE('Enter Assignment ID:');
   v_assignment_id := &assignment_id;
   DBMS_OUTPUT.PUT_LINE('Enter Assignment Student ID:');
   v_students_id := &students_id;
   DBMS_OUTPUT.PUT_LINE('Enter Submit File Name:');
   v_submit_file := '&submit_file';
   ADMIN.submit_assignment_student(v_assignment_id, v_students_id, v_submit_file);
   DBMS_OUTPUT.PUT_LINE('Assignment student record updated successfully.');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error occurred');
END;
/
```

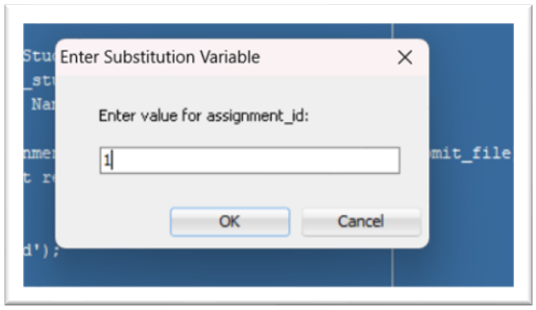


Figure 3.4.48.1

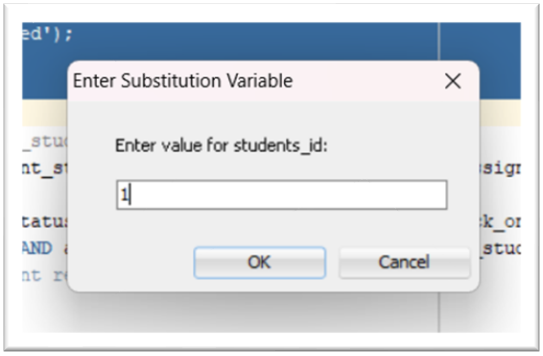


Figure 3.4.48.2

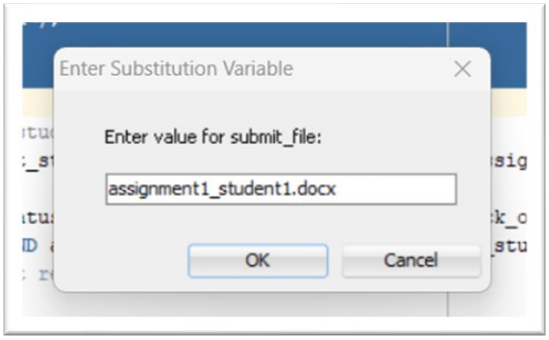


Figure 3.4.48.3

```
Enter Assignment ID:
Enter Assignment Student ID:
Enter Submit File Name:
Record updated successfully
Assignment student record updated successfully.


PL/SQL procedure successfully completed.
```

Figure 3.4.48.4

SELECT * FROM assignment_students;

ASSIGNMENT_STUDENTS_ID	ASSIGNMENT_ID	STUDENT_ID	SUBMIT_FILE	SUBMISSIONDATE	STATUS	GRADE	FEEDBACK_ON_ASSESSMENT
1	1	1	assignment1_student1.docx	18-OCT-24 11.13.40.554000000 PM	Submitted	(null)	(null)
2	1	2	(null)	(null)	Not submitted	(null)	(null)
3	1	3	(null)	(null)	Not submitted	(null)	(null)
4	1	11	(null)	(null)	Not submitted	(null)	(null)

Figure 3.4.48.5

3.4.49 Create Procedure to Grade Student Assignment

```
CREATE OR REPLACE PROCEDURE grade_assignment_student (p_assignment_id IN
NUMBER,p_student_id IN NUMBER,p_grade IN VARCHAR2,p_feedback IN VARCHAR2) AS
BEGIN
   UPDATE ADMIN.assignment_students SET status = 'Graded', grade = p_grade,
Feedback_on_Assessment = p_feedback
   WHERE assignment_id = p_assignment_id AND student_id = p_student_id;
   DBMS_OUTPUT.PUT_LINE('Assignment student record graded successfully.');
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No record found');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error occurred');
END;
/
```

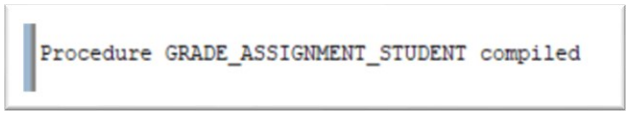


Figure 3.4.49

3.4.50 Grade Student Assignment Using User Input and Variables

```
SET SERVEROUTPUT ON;
DECLARE
   v_assignment_id NUMBER;
   v_student_id NUMBER;
   v_grade VARCHAR2(20);
   v_feedback VARCHAR2(100);
BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE('Enter Assignment ID:');
    v_assignment_id := &assignment_id;
    DBMS_OUTPUT.PUT_LINE('Enter Assignment Student ID:');
    v_student_id := &student_id;
    DBMS_OUTPUT.PUT_LINE('Enter Grade:');
    v_grade := '&grade';
    DBMS_OUTPUT.PUT_LINE('Enter Feedback:');
    v_feedback := '&feedback';
    ADMIN.grade_assignment_student(v_assignment_id, v_student_id, v_grade, v_feedback);
    DBMS_OUTPUT.PUT_LINE('Assignment student record updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred.');
END;
/
```

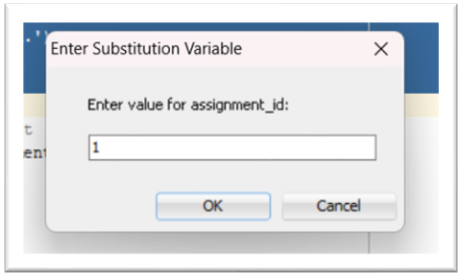


Figure 3.4.50.1

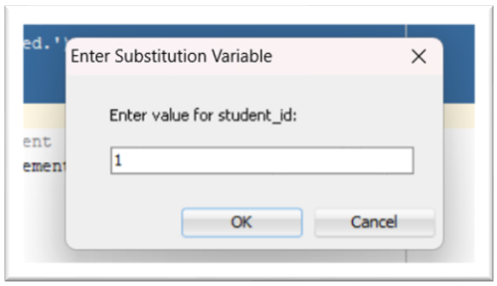


Figure 3.4.50.2

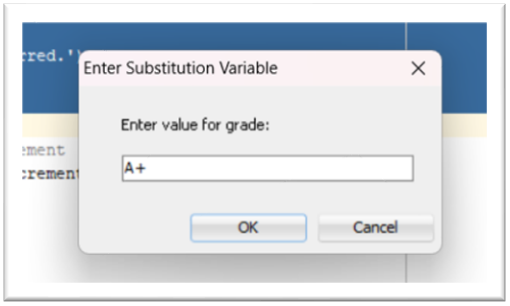


Figure 3.4.50.3

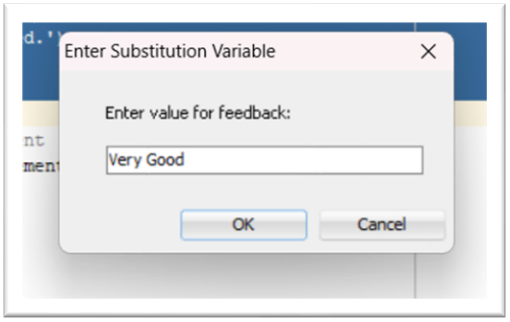


Figure 3.4.50.4

```
Enter Assignment ID:
Enter Assignment Student ID:
Enter Grade:
Enter Feedback:
Assignment student record graded successfully.
Assignment student record updated successfully.


PL/SQL procedure successfully completed.
```

Figure 3.4.50.5

SELECT * FROM assignment_students;

ASSIGNMENT_STUDENTS_ID	ASSIGNMENT_ID	STUDENT_ID	SUBMIT_FILE	SUBMISSIONDATE	STATUS	GRADE	FEEDBACK_ON_ASSESSMENT
1	1	1	assignment1_student1.docx	18-OCT-24 11.13.40.554000000 PM	Graded	A+	Very Good
2	1	2	(null)	(null)	Not submitted	(null)	(null)

Figure 3.4.50.6

3.5 Use of Cursor

3.5.1 Display Lesson and Schedule Details Using Cursor

```
SET SERVEROUTPUT ON;
DECLARE
   CURSOR schedule_cursor IS
      SELECT schedule_id, lesson_id, schedule_date, lecture_hall FROM schedule;
   schedule_rec schedule_cursor%ROWTYPE;
   lesson_name lessons.lesson_name%TYPE;
BEGIN
   FOR schedule_rec IN schedule_cursor LOOP
      SELECT lesson_name INTO lesson_name FROM lessons WHERE lesson_id = schedule_rec.lesson_id;
      DBMS_OUTPUT.PUT_LINE('Lesson Name: ' || lesson_name);
      DBMS_OUTPUT.PUT_LINE('Schedule Date: ' || TO_CHAR(schedule_rec.schedule_date, 'DD-MON-
YYYY'));
      DBMS_OUTPUT.PUT_LINE('Lecture Hall: ' || schedule_rec.lecture_hall);
      DBMS_OUTPUT.PUT_LINE(' ');
   END LOOP;
END;
/
```

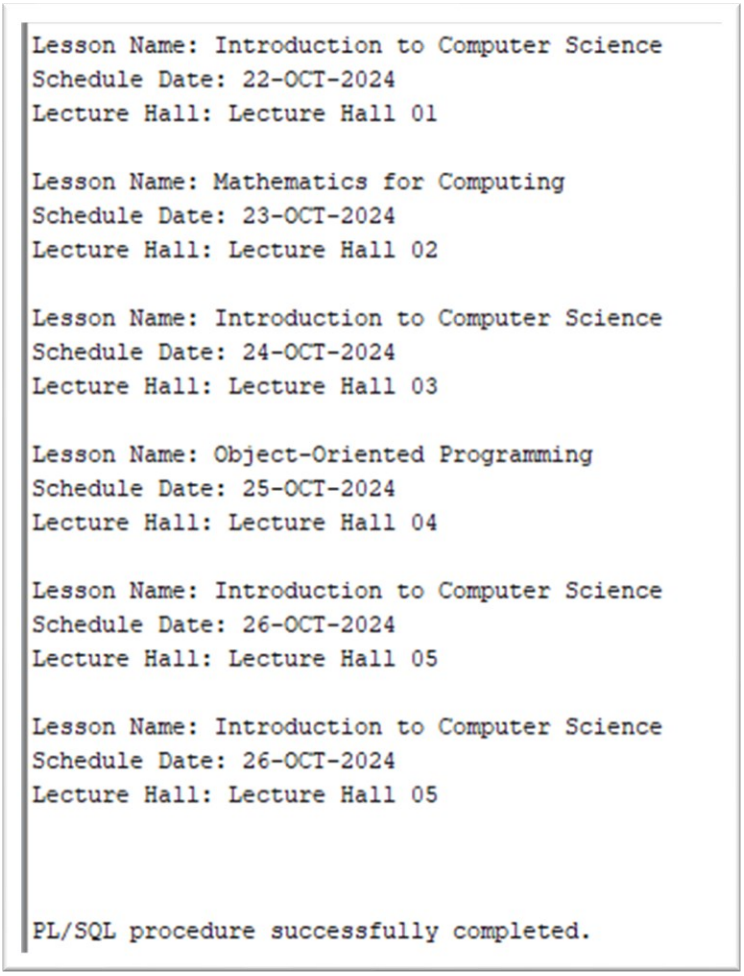


Figure 3.5.1

3.6 Granting Permissions and Roles to Students and Lecturers

3.6.1 Assign User Role Permissions to Students and Lecturers

```
GRANT SELECT ON ADMIN.assignment TO USER_ROLE;
GRANT SELECT ON ADMIN.assignment_students TO USER_ROLE;
GRANT SELECT ON ADMIN.courses TO USER_ROLE;
GRANT SELECT ON ADMIN.departments TO USER_ROLE;
GRANT SELECT ON ADMIN.lecturers TO USER_ROLE;
GRANT SELECT ON ADMIN.lessons TO USER_ROLE;
GRANT SELECT ON ADMIN.schedule TO USER_ROLE;
```

GRANT SELECT ON ADMIN.students TO USER_ROLE;

GRANT USER_ROLE TO STUDENT;

GRANT USER_ROLE TO LECTURER;

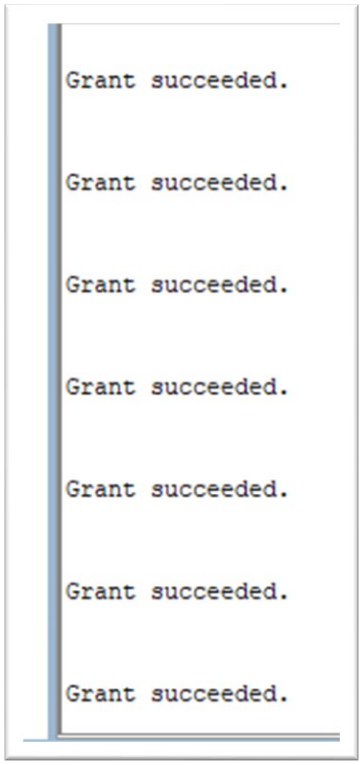


Figure 3.6.1

3.6.2 Grant DML and Execute Permissions to Lecturers

GRANT INSERT,DELETE,UPDATE ON ADMIN.assignment TO LECTURER;

GRANT UPDATE,DELETE ON ADMIN.assignment_students TO LECTURER;

GRANT UPDATE ON ADMIN.courses TO LECTURER;

GRANT UPDATE ON ADMIN.lessons TO LECTURER;

GRANT INSERT,UPDATE,DELETE ON ADMIN.schedule TO LECTURER;

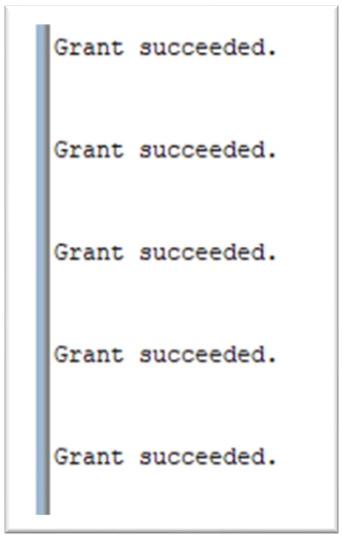


Figure 3.6.2

3.6.3 Grant Update and Execute Permissions to Students

GRANT UPDATE (submit_file, status, submissionDate) ON ADMIN.assignment_students TO STUDENT;

GRANT EXECUTE ON ADMIN.submit_assignment_student TO STUDENT;

GRANT EXECUTE ON ADMIN.insert_student_feedback TO STUDENT;

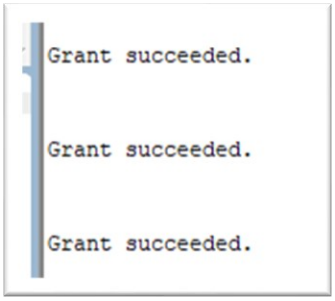


Figure 3.6.3

3.7 View

3.7.1 View for Department Course and Lesson

```
CREATE OR REPLACE VIEW department_course_lesson_stats AS SELECT
    d.department_id,
    d.department_name,
    COUNT(DISTINCT c.course_id) AS total_courses,
    COUNT(DISTINCT CASE WHEN c.status = 'Active' THEN c.course_id END) AS active_courses,
    COUNT(DISTINCT CASE WHEN c.status = 'Inactive' THEN c.course_id END) AS inactive_courses,
    COUNT(DISTINCT l.lesson_id) AS total_lessons,
    COUNT(DISTINCT CASE WHEN l.status = 'Active' THEN l.lesson_id END) AS active_lessons,
    COUNT(DISTINCT CASE WHEN l.status = 'Inactive' THEN l.lesson_id END) AS inactive_lessons
FROM departments d LEFT JOIN courses c ON d.department_id = c.department_id
LEFT JOIN lessons l ON c.course_id = l.course_id
GROUP BY d.department_id, d.department_name;
```

```
View DEPARTMENT_COURSE_LESSON_STATS created.
```

Figure 3.7.1.1

```
SELECT * FROM department_course_lesson_stats ;
```

DEPARTMENT_ID	DEPARTMENT_NAME	TOTAL_COURSES	ACTIVE_COURSES	INACTIVE_COURSES	TOTAL_LESSONS	ACTIVE_LESSONS	INACTIVE_LESSONS
1	School of Computing	3	3	0	12	12	0
2	School of Business	2	2	0	7	7	0
3	School of Engineering	0	0	0	0	0	0
5	School of Humanities	0	0	0	0	0	0
4	School of Language	1	1	0	0	0	0

Figure 3.7.1.2

3.7.2 View for Upcoming Assignments

```
CREATE OR REPLACE VIEW upcoming_assignments AS

SELECT a.assignment_id, a.assignment_title, a.end_date, l.lesson_name, l.taught_by

FROM assignment a JOIN lessons l ON a.lesson_id = l.lesson_id

WHERE a.end_date > CURRENT_TIMESTAMP
```

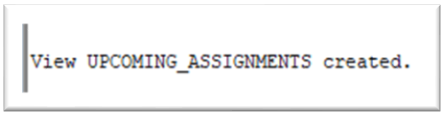
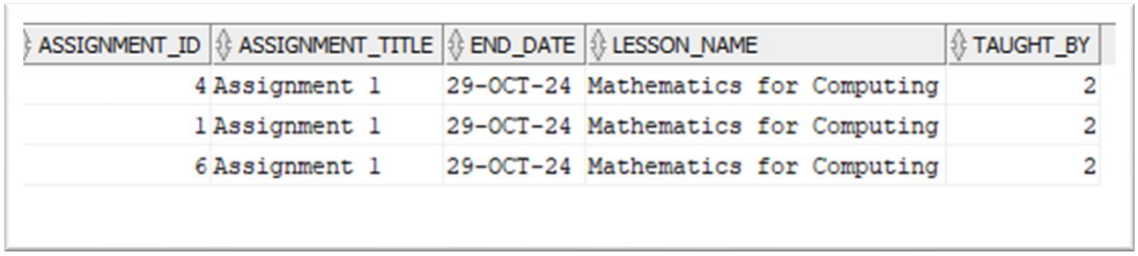


Figure 3.7.2.1

```
SELECT * FROM upcoming_assignments ;
```



ASSIGNMENT_ID	ASSIGNMENT_TITLE	END_DATE	LESSON_NAME	TAUGHT_BY
4	Assignment 1	29-OCT-24	Mathematics for Computing	2
1	Assignment 1	29-OCT-24	Mathematics for Computing	2
6	Assignment 1	29-OCT-24	Mathematics for Computing	2

Figure 3.7.2.1

3.7.3 View for student feedback for lessons

```
CREATE OR REPLACE VIEW student_feedback_for_lessons AS

SELECT

   sf.lesson_id AS feedback_lesson_id,

   sf.student_id AS feedback_student_id,

   sf.comments,

   sf.rating,

   sf.post_date

FROM

   students_feedback sf

JOIN

   lessons les ON sf.lesson_id = les.lesson_id;
```

```
View STUDENT_FEEDBACK_FOR_LESSONS created.
```

Figure 3.7.3.1

SELECT * FROM student_feedback_for_lessons;

FEEDBACK_LESSON_ID	FEEDBACK_STUDENT_ID	COMMENTS	RATING	POST_DATE
1	1	You are the best madam ever	5	18-OCT-24 11.42.21.475000000 PM
1	2	I enjoyed the interactive activities and practical examples.	5	18-OCT-24 11.50.09.811000000 PM
1	11	Very detailed and helpful. I learned a lot from this lesson.	4	18-OCT-24 11.50.09.816000000 PM
1	19	The lesson was engaging, but the lecture duration was too long.	3	18-OCT-24 11.50.09.822000000 PM

Figure 3.7.3.2

3.8 DQL Commands

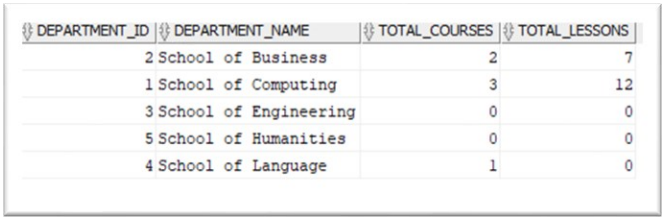
3.8.1 Number of lessons and courses Offered by Each Department

```
SELECT d.department_id, d.department_name, COUNT(DISTINCT c.course_id) AS total_courses,
COUNT(l.lesson_id) AS total_lessons

FROM departments d LEFT JOIN courses c ON d.department_id = c.department_id

LEFT JOIN lessons l ON c.course_id = l.course_id

GROUP BY d.department_id, d.department_name ORDER BY d.department_name;
```



DEPARTMENT_ID	DEPARTMENT_NAME	TOTAL_COURSES	TOTAL_LESSONS
2	School of Business	2	7
1	School of Computing	3	12
3	School of Engineering	0	0
5	School of Humanities	0	0
4	School of Language	1	0

Figure 3.8.1

3.8.2 Lecturers Who Have Taught More Than two Lessons

```
SELECT l.lecturer_id, l.first_name, l.last_name, COUNT(ls.lesson_id) AS lessons_taught

FROM lecturers l INNER JOIN lessons ls ON l.lecturer_id = ls.taught_by

GROUP BY l.lecturer_id, l.first_name, l.last_name

HAVING COUNT(ls.lesson_id) > 2

ORDER BY lessons_taught DESC;
```

LECTURER_ID	FIRST_NAME	LAST_NAME	LESSONS_TAUGHT
5	Ruwan	Dias	4
2	Saman	Silva	3

Figure 3.8.2

3.8.3 All Students Enrolled in Diploma in Software Engineering

SELECT s.student_id, s.first_name, s.last_name, s.email

FROM students s INNER JOIN course_enrollments ce ON s.student_id = ce.student_id

INNER JOIN courses c ON ce.course_id = c.course_id

WHERE c.course_name = 'Diploma in Software Engineering ';

STUDENT_ID	FIRST_NAME	LAST_NAME	EMAIL
1	Amila	Perera	amilaperera@gmail.com
2	Kasuni	Samarasinghe	kasunisamarasinghe@gmail.com
3	Chathura	Wijesinghe	chathurawijesinghe@gmail.com
11	Chami	Chandimal	chamichandimal@gmail.com
12	Lakmal	Senevirathne	lakmalsenevirathne@gmail.com
19	Nihal	Bandara	nihalbandara@gmail.com
20	Thilini	Samarawickrama	thilinisanarawickrama@gmail.com
27	Sachin	Jayasuriya	sachinjayasuriya@gmail.com
35	Dulanjali	Fernando	dulanjalifernando@gmail.com
36	Dilshani	Seneviratne	dilshanisenevirathne@gmail.com

Figure 3.8.3

3.8.4 All Assignments Due in Next 7 Days

INSERT INTO assignment (assignment_title, assignment_fiies, lesson_id, end_date)

VALUES (' Database Management System Assignment', 'dbms_assignment.docx', 3, TO_DATE('2024-10-21', 'YYYY-MM-DD'));

SELECT * from assignment;

ASSIGNMENT_ID	ASSIGNMENT_TITLE	ASSIGNMENT_FIIES	LESSON_ID	POST_DATE	END_DATE
2	Database Management Sys...	dbms_assignment.docx	3	19-OCT-24 11.00.49.473000000 PM	21-OCT-24
1	Assignment 1	file1.pdf	2	18-OCT-24 11.02.14.529000000 PM	29-OCT-24

Figure 3.8.4.1

SELECT * from assignment_students;

ASSIGNMENT_STUDENTS_ID	ASSIGNMENT_ID	STUDENT_ID	SUBMIT_FILE	SUBMISSIONDATE	STATUS	GRADE	FEEDBACK_ON_ASSESSMENT
11	2	1	(null)	(null)	Not submitted	(null)	(null)
12	2	2	(null)	(null)	Not submitted	(null)	(null)
13	2	3	(null)	(null)	Not submitted	(null)	(null)
14	2	11	(null)	(null)	Not submitted	(null)	(null)
15	2	12	(null)	(null)	Not submitted	(null)	(null)
16	2	19	(null)	(null)	Not submitted	(null)	(null)
17	2	20	(null)	(null)	Not submitted	(null)	(null)
18	2	27	(null)	(null)	Not submitted	(null)	(null)
19	2	35	(null)	(null)	Not submitted	(null)	(null)
20	2	36	(null)	(null)	Not submitted	(null)	(null)
1	1	1	assignment1_student1.docx	18-OCT-24 11.13.40.554000000 PM	Graded	A+	Very Good
2	1	2	(null)	(null)	Not submitted	(null)	(null)
3	1	3	(null)	(null)	Not submitted	(null)	(null)
4	1	11	(null)	(null)	Not submitted	(null)	(null)
5	1	12	(null)	(null)	Not submitted	(null)	(null)
6	1	19	(null)	(null)	Not submitted	(null)	(null)
7	1	20	(null)	(null)	Not submitted	(null)	(null)
8	1	27	(null)	(null)	Not submitted	(null)	(null)
9	1	35	(null)	(null)	Not submitted	(null)	(null)
10	1	36	(null)	(null)	Not submitted	(null)	(null)

Figure 3.8.4.2

```
SELECT a.assignment_title, a.end_date, c.course_name, l.lesson_name
FROM assignment a INNER JOIN lessons l ON a.lesson_id = l.lesson_id
INNER JOIN courses c ON l.course_id = c.course_id
WHERE a.end_date BETWEEN SYSDATE AND SYSDATE + 7
ORDER BY a.end_date;
```

ASSIGNMENT_TITLE	END_DATE	COURSE_NAME	LESSON_NAME
Database Management System Assignment	21-OCT-24	Diploma in Software Engineering	Database Management Systems

Figure 3.8.4.3

3.8.5 Average, Maximum, and Minimum Salaries by Department

```
SELECT
   d.department_name,
   AVG(l.salary) AS average_salary,
   MAX(l.salary) AS max_salary,
   MIN(l.salary) AS min_salary
FROM departments d INNER JOIN lecturers l
ON d.department_id = l.department_id GROUP BY d.department_name;
```

DEPARTMENT_NAME	AVERAGE_SALARY	MAX_SALARY	MIN_SALARY
School of Computing	76000	80000	72000
School of Business	76500	78000	75000
School of Engineering	86000	90000	82000
School of Language	60500	61000	60000
School of Humanities	85500	86000	85000

Figure 3.8.5

3.8.6 Number of student enrollments

SELECT c.course_name, COUNT(ce.student_id) AS number_of_enrollments

FROM courses c INNER JOIN course_enrollments ce ON c.course_id = ce.course_id

GROUP BY c.course_name ORDER BY number_of_enrollments DESC;

COURSE_NAME	NUMBER_OF_ENROLLMENTS
Diploma in Software Engineering	10
Higher National Diploma in Software Engineering	9
Advanced Diploma in Business Management	9
Higher National Diploma in Business Management	9
Diploma in English	2
Higher National Diploma in Network Engineering	1

Figure 3.8.6

3.8.7 Courses Without any enrollments

UPDATE course_enrollments SET course_id = 5 WHERE

student_id IN (SELECT student_id FROM course_enrollments WHERE course_id = 6);

(Because all courses have enrollments)

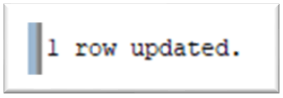


Figure 3.8.7.1

```
SELECT c.* FROM courses c
LEFT JOIN course_enrollments ce ON c.course_id = ce.course_id
WHERE ce.course_id IS NULL;
```

COURSE_ID	COURSE_NAME	DEPA...	DESCRIPTION	CREDITS	START_DATE	END_DATE	STATUS	CREATED_BY	CREATED_AT
6	Higher National Diploma in Network Engineering	1	This program introduc...	40	10-AUG-24	15-AUG-25	Active	3	18-OCT-24 05.44.42.697000000 PM

Figure 3.8.7.2

3.8.8 Total number of lessons per course

```
SELECT c.course_name, COUNT(l.lesson_id) AS total_lessons
FROM courses c LEFT JOIN lessons l ON c.course_id = l.course_id
GROUP BY c.course_name ORDER BY total_lessons DESC;
```

3.8.9 Fetch Unsubmitted Assignments for Specific Student ID

```
SELECT a.student_id,a.status,ast.assignment_title,ast.assignment_fiies,ast.post_date,ast.end_date
FROM ADMIN.assignment_students a JOIN ADMIN.assignment ast on a.assignment_id =
ast.assignment_id
WHERE status = 'Not submitted' AND student_id = 1 ;
```

3.8.10 Fetch Students Based on Name, Email, and Phone

```
SELECT * FROM students WHERE
   (first_name LIKE 'A%' OR last_name LIKE '%ndo')
   AND email LIKE '%@gmail.com'
   AND phone_number LIKE '07_%'
```

ORDER BY last_name, first_name;

STUDENT_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	DATE_OF_BIRTH	GENDER	ADDRESS	STATUS	CREATED_AT
5	Sajith	Fernando	sajithfernando@gmail.com	0775678901	20-JUL-01	Male	65, Temple Lane, Negombo	Active	18-OCT-24 05.56.08.834000000 PM
1	Amila	Perera	amilaperera@gmail.com	0711234567	15-MAY-01	Male	123, Colombo Road, Galle	Active	18-OCT-24 05.56.08.811000000 PM
10	Ayesh	Rajapaksha	ayeshrajapaksha@gmail.com	0720123456	30-JUN-01	Male	88, Ocean Drive, Colombo	Active	18-OCT-24 05.56.08.858000000 PM

Figure 3.8.10

3.8.11 Retrieve Upcoming Assignments from View

SELECT * FROM upcoming_assignments;

ASSIGNMENT_ID	ASSIGNMENT_TITLE	END_DATE	LESSON_NAME	TAUGHT_BY
2	Database Management System Assignment	21-OCT-24	Database Management Systems	5
1	Assignment 1	29-OCT-24	Mathematics for Computing	2

Figure 3.8.11

3.8.12 Retrieve department course lesson from View

SELECT * FROM department_course_lesson_stats;

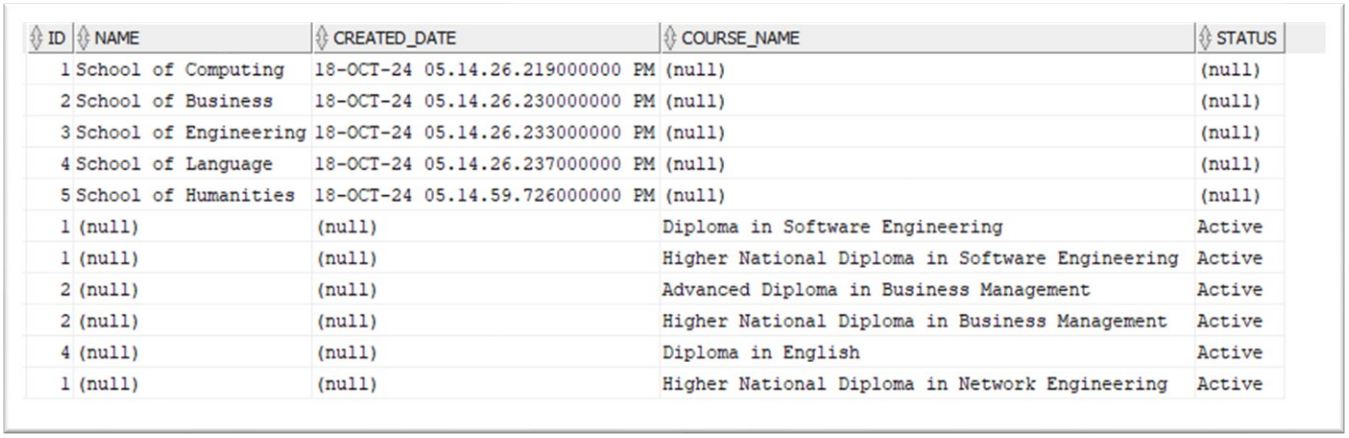
DEPARTMENT_ID	DEPARTMENT_NAME	TOTAL_COURSES	ACTIVE_COURSES	INACTIVE_COURSES	TOTAL_LESSONS	ACTIVE_LESSONS	INACTIVE_LESSONS
1	School of Computing	3	3	0	12	12	0
2	School of Business	2	2	0	7	7	0
3	School of Engineering	0	0	0	0	0	0
5	School of Humanities	0	0	0	0	0	0
4	School of Language	1	1	0	0	0	0

Figure 3.8.12

3.8.13 Combine Department and Course Information Using UNION

```
SELECT department_id AS id,department_name AS name,created_at AS created_date,
    NULL AS course_name,NULL AS status
FROM departments
UNION ALL
```

```
SELECT department_id AS id,NULL AS name,NULL AS created_date,
   course_name AS course_name,status
FROM courses;
```



ID	NAME	CREATED_DATE	COURSE_NAME	STATUS
1	School of Computing	18-OCT-24 05.14.26.219000000 PM	(null)	(null)
2	School of Business	18-OCT-24 05.14.26.230000000 PM	(null)	(null)
3	School of Engineering	18-OCT-24 05.14.26.233000000 PM	(null)	(null)
4	School of Language	18-OCT-24 05.14.26.237000000 PM	(null)	(null)
5	School of Humanities	18-OCT-24 05.14.59.726000000 PM	(null)	(null)
1	(null)	(null)	Diploma in Software Engineering	Active
1	(null)	(null)	Higher National Diploma in Software Engineering	Active
2	(null)	(null)	Advanced Diploma in Business Management	Active
2	(null)	(null)	Higher National Diploma in Business Management	Active
4	(null)	(null)	Diploma in English	Active
1	(null)	(null)	Higher National Diploma in Network Engineering	Active

Figure 3.8.13

3.9 Function and Execution

3.9.1 Create Function to Count Scheduled Lectures by lesson_id

```
DECLARE
   v_lecture_count NUMBER; -- Variable to hold the result from the function
BEGIN
   -- Call the function and store the result in v_lecture_count
   v_lecture_count := get_scheduled_lectures(2); -- Replace 1 with the desired lesson ID

   -- Display the count of scheduled lectures
   DBMS_OUTPUT.PUT_LINE('Number of scheduled lectures for lesson ID 1: ' || v_lecture_count);
END;
/
```

```
Function GET_SCHEDULED_LECTURES compiled
```

Figure 3.9.1

3.9.2 Display Count of Scheduled Lectures for a Lesson ID

```
SET SERVEROUTPUT ON;
DECLARE
   v_lecture_count NUMBER;
   v_lesson_id NUMBER;
BEGIN
   v_lesson_id := &lesson_id;
   v_lecture_count := get_scheduled_lectures(v_lesson_id);
   DBMS_OUTPUT.PUT_LINE('Number of scheduled lectures for lesson ID ' || v_lesson_id || ': ' ||
v_lecture_count);
END;
/
```

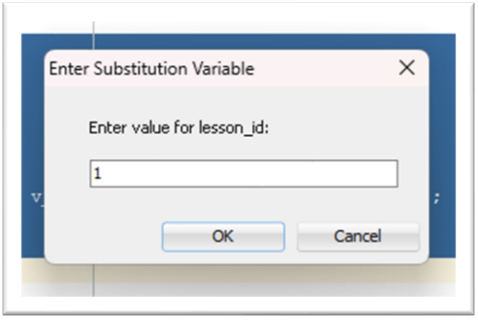


Figure 3.9.2.1

```
Number of scheduled lectures for lesson ID 1: 4
```

Figure 3.9.2.2

3.9.3 Get Assignment Deadline for a Student

```
CREATE OR REPLACE FUNCTION get_assignment_deadline (p_student_id NUMBER, p_assignment_id
NUMBER)
RETURN DATE IS v_end_date DATE;
BEGIN
   SELECT a.end_date INTO v_end_date FROM assignment a
   JOIN assignment_students ast ON a.assignment_id = ast.assignment_id
   WHERE ast.student_id = p_student_id AND a.assignment_id = p_assignment_id;
   RETURN v_end_date;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN NULL;
END;
/
```

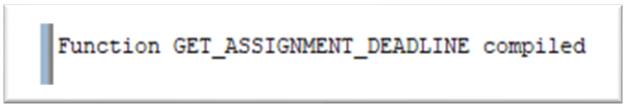


Figure 3.9.3

3.9.4 Display Assignment Deadline for a Student

```
SET SERVEROUTPUT ON;
DECLARE
   v_deadline DATE;
BEGIN
   v_deadline := get_assignment_deadline(1, 1); --student_id and assignment_id
   DBMS_OUTPUT.PUT_LINE('Assignment Deadline: ' || v_deadline);
END;
```

/

```
Assignment Deadline: 29-OCT-24
```

Figure 3.9.4

3.9.5 Average Rating for a Lesson

```
CREATE OR REPLACE FUNCTION get_average_rating ( p_lesson_id NUMBER)
RETURN NUMBER IS v_avg_rating NUMBER;
BEGIN
   SELECT AVG(f.rating) INTO v_avg_rating FROM students_feedback f
   WHERE f.lesson_id = p_lesson_id;
   RETURN v_avg_rating;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN 0;
END;
/
```

```
Function GET_AVERAGE_RATING compiled
```

3.9.6 Retrieve average rating for a lesson

```
SET SERVEROUTPUT ON;
DECLARE
   v_avg_rating NUMBER;
BEGIN
   v_avg_rating := get_average_rating(1);  --lesson ID
```

```
    DBMS_OUTPUT.PUT_LINE('Average Rating: ' || v_avg_rating);
END;
/
```

```
Average Rating: 4.25
```

Figure 3.9.6

3.9.7 Calculate GPA for a Student

```
CREATE OR REPLACE FUNCTION calculate_gpa (p_student_id NUMBER)
RETURN NUMBER IS
    v_total_points NUMBER := 0;
    v_total_assignments NUMBER := 0;
    v_grade_point NUMBER;
BEGIN
    FOR rec IN (SELECT grade FROM assignment_students WHERE student_id = p_student_id
    AND grade IS NOT NULL) LOOP
        CASE rec.grade
            WHEN 'A+' THEN v_grade_point := 4.0;
            WHEN 'A'  THEN v_grade_point := 4.0;
            WHEN 'A-' THEN v_grade_point := 3.7;
            WHEN 'B+' THEN v_grade_point := 3.3;
            WHEN 'B'  THEN v_grade_point := 3.0;
            WHEN 'B-' THEN v_grade_point := 2.7;
            WHEN 'C+' THEN v_grade_point := 2.3;
            WHEN 'C'  THEN v_grade_point := 2.0;
            WHEN 'C-' THEN v_grade_point := 1.7;
            WHEN 'D+' THEN v_grade_point := 1.3;
```

```
        WHEN 'D'  THEN v_grade_point := 1.0;
        ELSE v_grade_point := 0;
    END CASE;
    v_total_points := v_total_points + v_grade_point;
    v_total_assignments := v_total_assignments + 1;
        -- This is not a correct way to find GPA
  END LOOP;
  IF v_total_assignments = 0 THEN
    RETURN 0;
  ELSE
    RETURN v_total_points / v_total_assignments;
  END IF;
END;
/
```

```
Function CALCULATE_GPA compiled
```

Figure 3.9.7

3.9.8 Retrieve GPA for a student

```
SET SERVEROUTPUT ON;
DECLARE
  v_gpa NUMBER;
BEGIN
  v_gpa := calculate_gpa(1);  --student ID
  DBMS_OUTPUT.PUT_LINE('Student GPA: ' || v_gpa);
END;
/
```

```
Student GPA: 4
```

Figure 3.9.8

04. BACKUP

4.1 Logical Backup [Export / Import utility]

4.1.1 Export utility

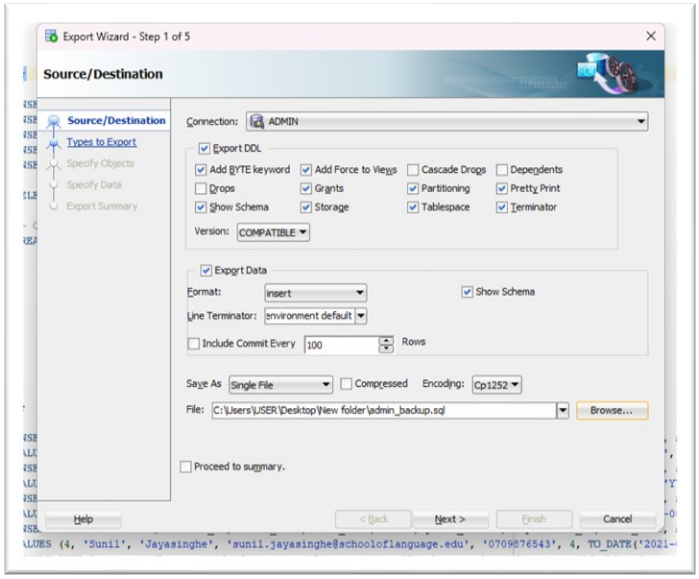


Figure 4.1.1.1

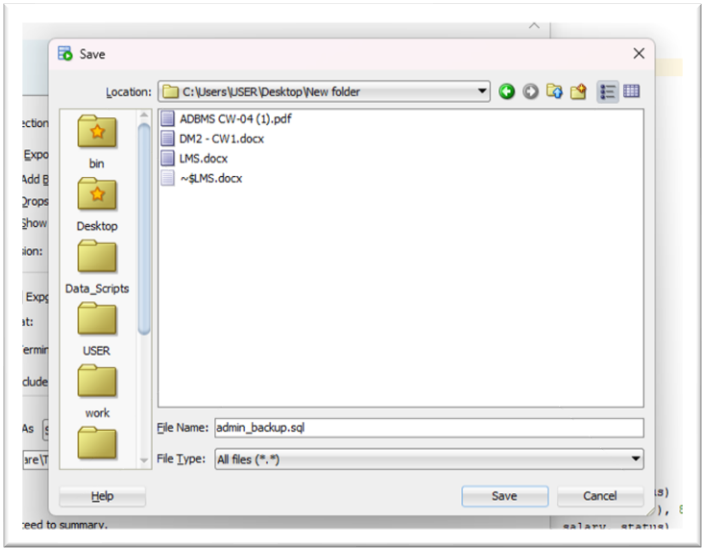


Figure 5.1.1.2

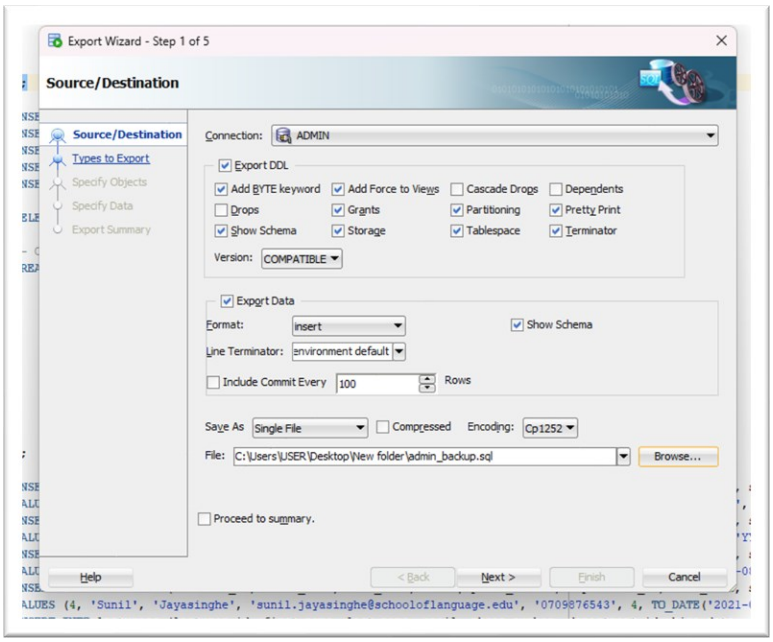


Figure 6.1.1.3

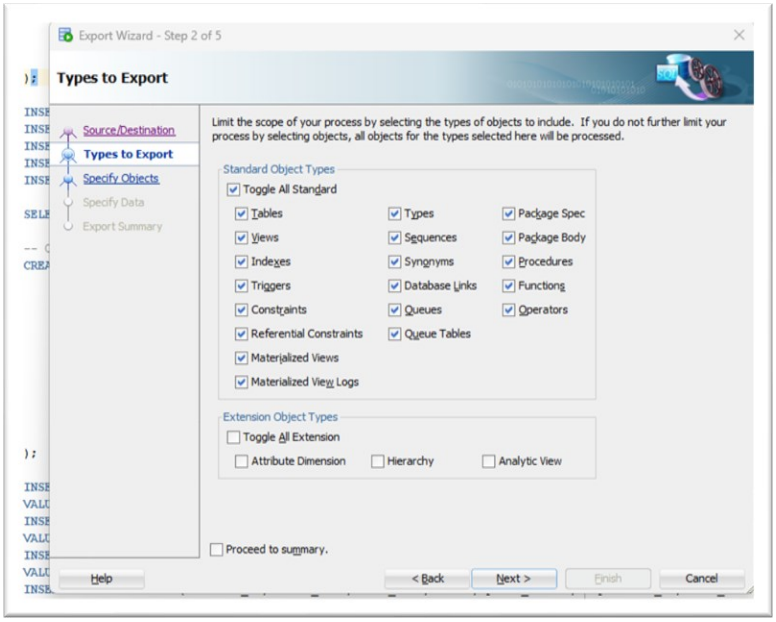


Figure 7.1.1.4

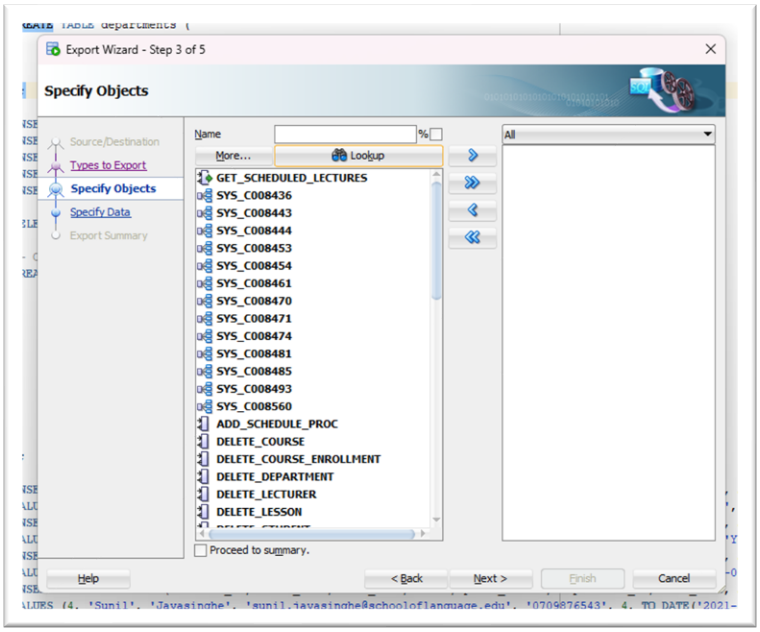


Figure 8.1.1.5

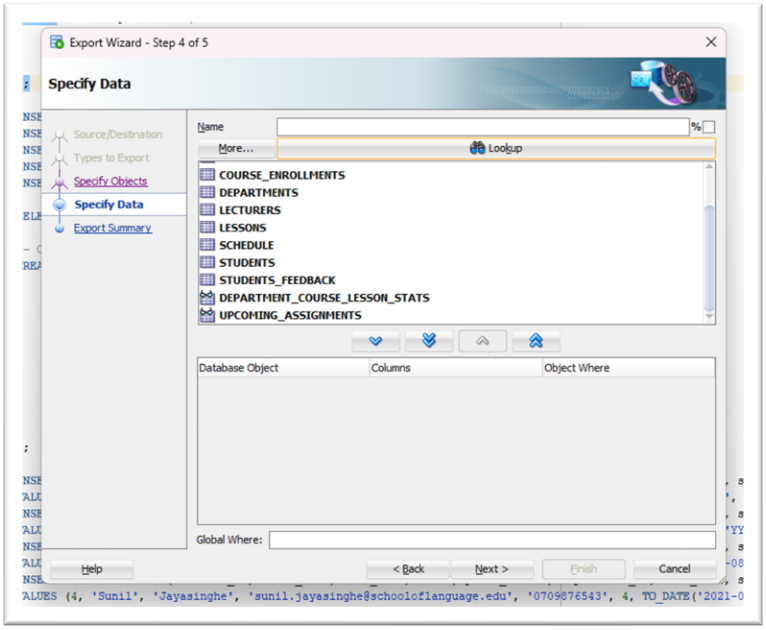


Figure 9.1.1.6

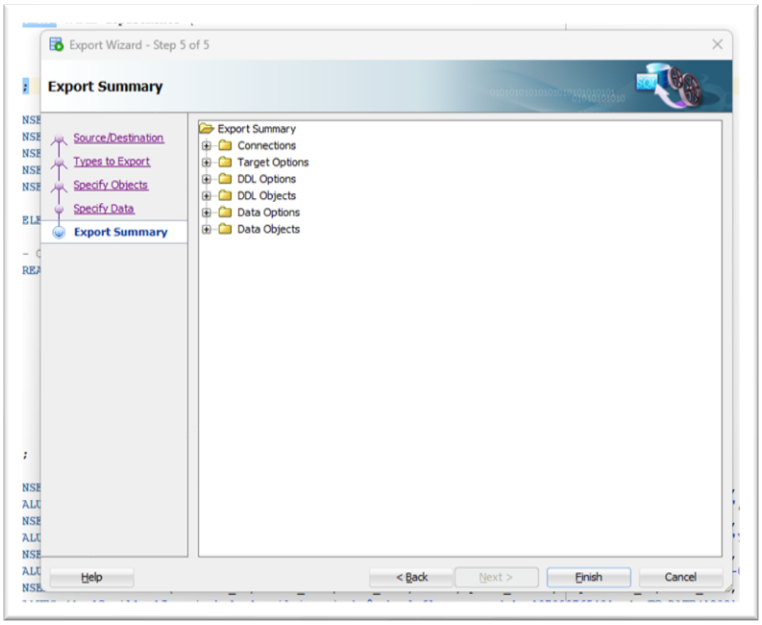


Figure 10.1.1.7

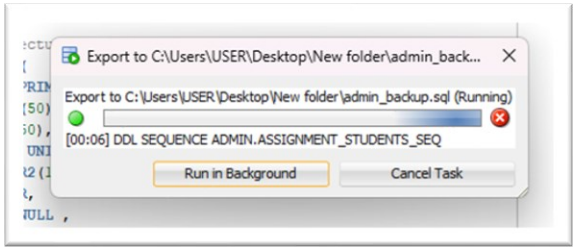


Figure 11.1.1.8

4.1.2 Import utility

CREATE USER ADMIN2 IDENTIFIED BY ADMIN123;

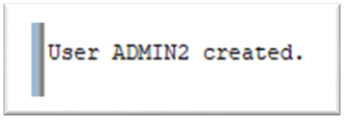


Figure 4.1.2 1

GRANT ALL PRIVILEGES TO ADMIN2;

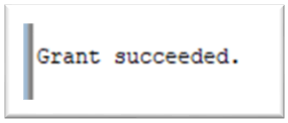


Figure 4.1.2 2

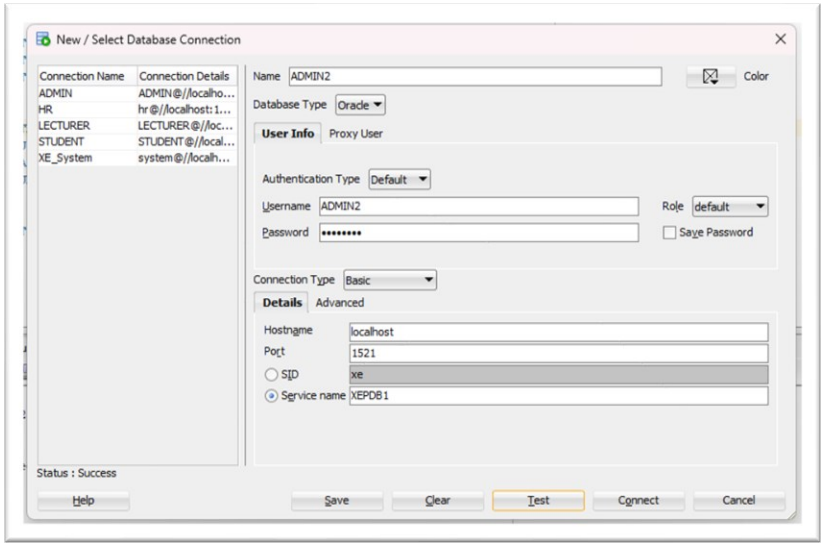


Figure 4.1.2 3

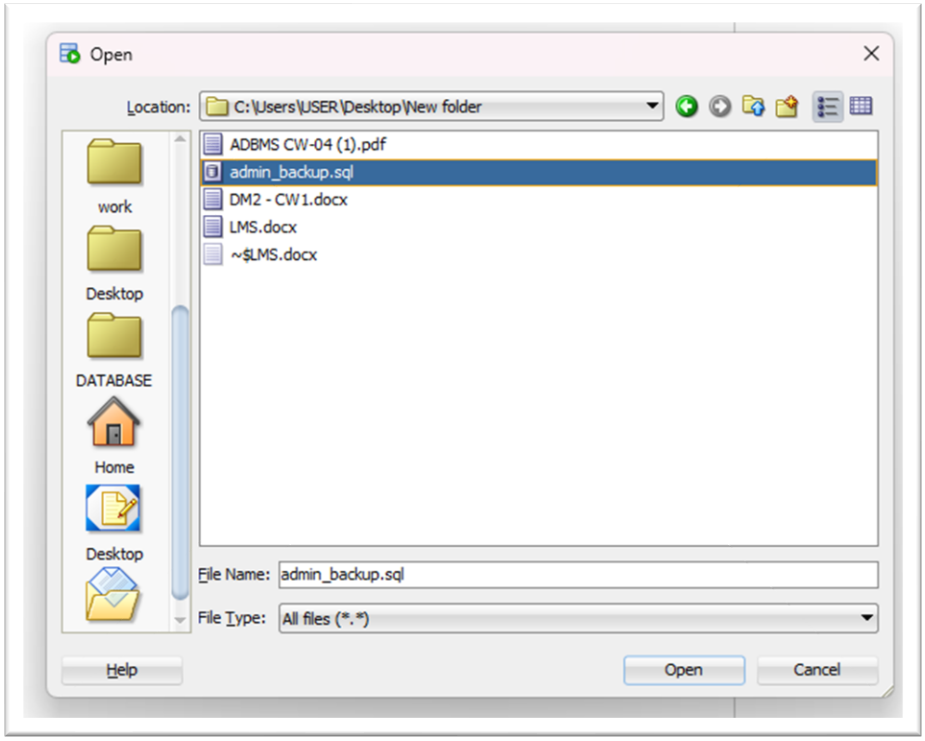


Figure 4.1.2 4

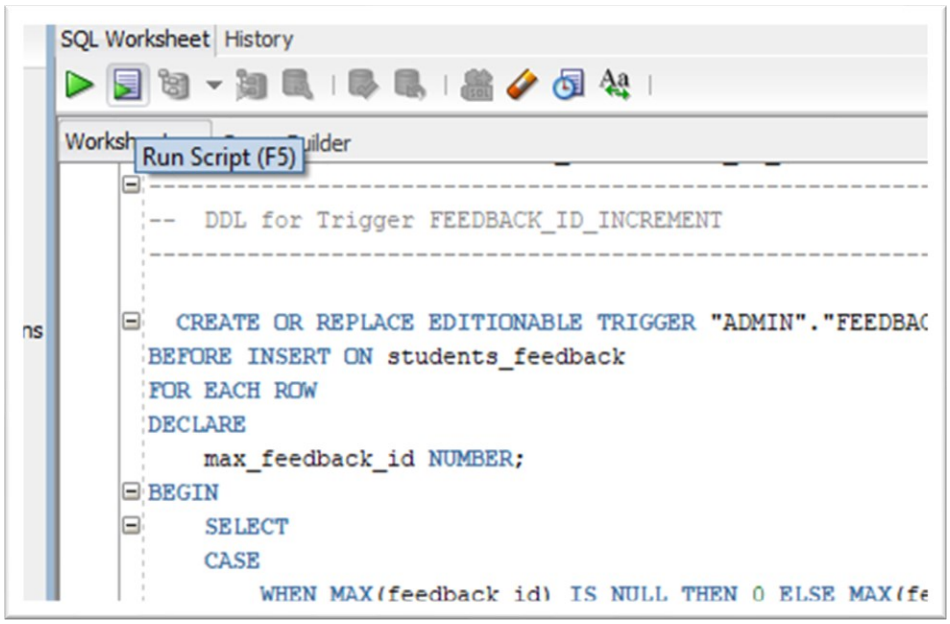
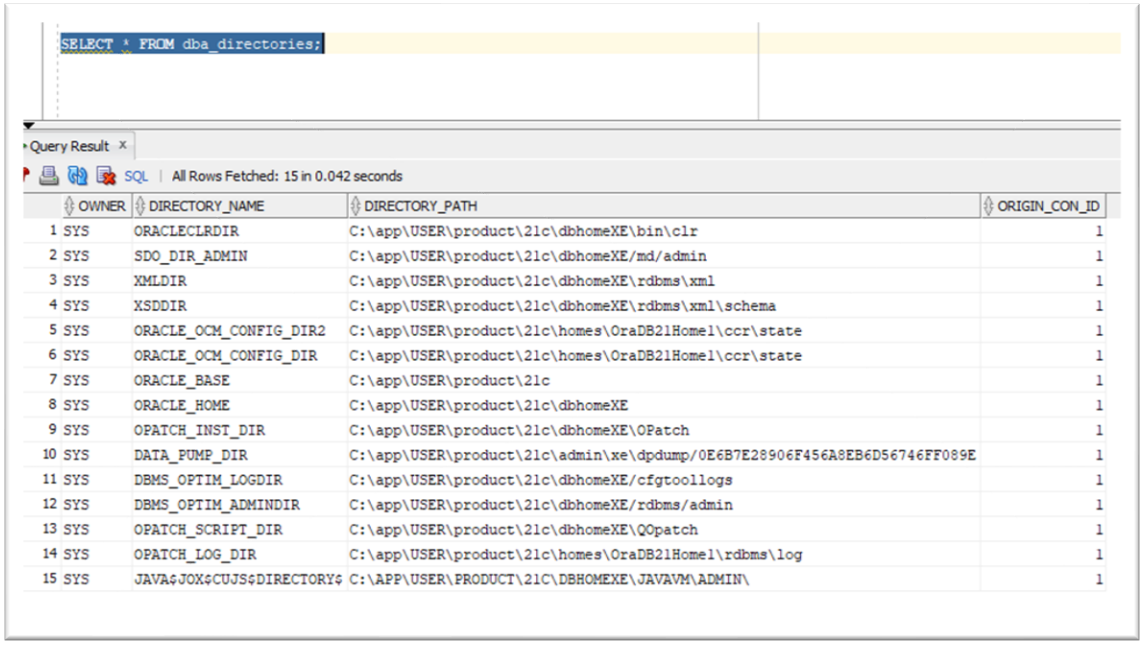

SQL Worksheet
History
Run Script (F5)
-- DDL for Trigger FEEDBACK_ID_INCREMENT
CREATE OR REPLACE EDITIONABLE TRIGGER "ADMIN"."FEEDBAC
BEFORE INSERT ON students_feedback
FOR EACH ROW
DECLARE
max_feedback_id NUMBER;
BEGIN
SELECT
CASE
WHEN MAX(feedback_id) IS NULL THEN 0 ELSE MAX(fe

Figure 4.1.2 5

4.2 Logical Backup

4.2.1 Data Pump Export

SELECT * FROM dba_directories;



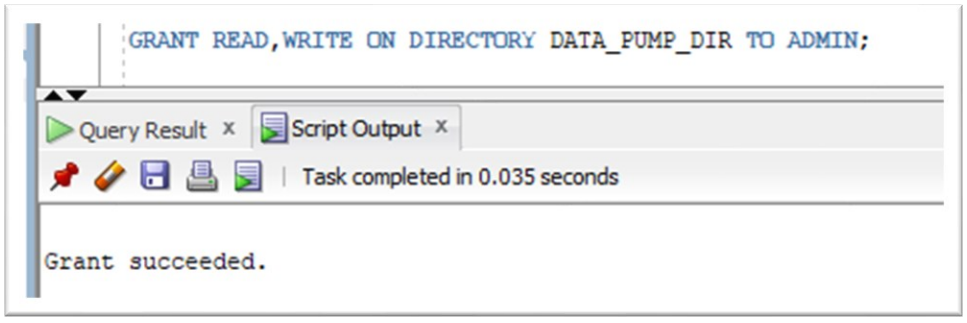
SELECT * FROM dba_directories;

Query Result | SQL | All Rows Fetched: 15 in 0.042 seconds

	OWNER	DIRECTORY_NAME	DIRECTORY_PATH	ORIGIN_CON_ID
1	SYS	ORACLECLRDIR	C:\app\USER\product\21c\dbhomeXE\bin\clr	1
2	SYS	SDO_DIR_ADMIN	C:\app\USER\product\21c\dbhomeXE/md/admin	1
3	SYS	XMLDIR	C:\app\USER\product\21c\dbhomeXE\rdbms\xml	1
4	SYS	XSDDIR	C:\app\USER\product\21c\dbhomeXE\rdbms\xml\schema	1
5	SYS	ORACLE_OCM_CONFIG_DIR2	C:\app\USER\product\21c\homes\OraDB21Home1\ccr\state	1
6	SYS	ORACLE_OCM_CONFIG_DIR	C:\app\USER\product\21c\homes\OraDB21Home1\ccr\state	1
7	SYS	ORACLE_BASE	C:\app\USER\product\21c	1
8	SYS	ORACLE_HOME	C:\app\USER\product\21c\dbhomeXE	1
9	SYS	OPATCH_INST_DIR	C:\app\USER\product\21c\dbhomeXE\OPatch	1
10	SYS	DATA_PUMP_DIR	C:\app\USER\product\21c\admin\xe\dpdump/0E6B7E28906F456A8EB6D56746FF089E	1
11	SYS	DBMS_OPTIM_LOGDIR	C:\app\USER\product\21c\dbhomeXE/cfgtoollogs	1
12	SYS	DBMS_OPTIM_ADMINDIR	C:\app\USER\product\21c\dbhomeXE/rdbms/admin	1
13	SYS	OPATCH_SCRIPT_DIR	C:\app\USER\product\21c\dbhomeXE\QOpatch	1
14	SYS	OPATCH_LOG_DIR	C:\app\USER\product\21c\homes\OraDB21Home1\rdbms\log	1
15	SYS	JAVAJOXCUJS$DIRECTORY$	C:\APP\USER\PRODUCT\21C\DBHOMEXE\JAVAVM\ADMIN\	1

Figure 4.1.2 6

GRANT READ,WRITE ON DIRECTORY DATA_PUMP_DIR TO ADMIN;



GRANT READ,WRITE ON DIRECTORY DATA_PUMP_DIR TO ADMIN;

Query Result | Script Output

Task completed in 0.035 seconds

Grant succeeded.

Figure 4.1.2 7

GRANT DBA TO ADMIN;

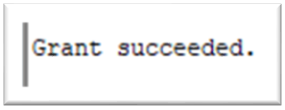


Figure 4.1.2 8

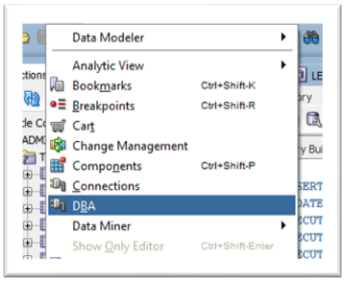


Figure 4.1.2 9

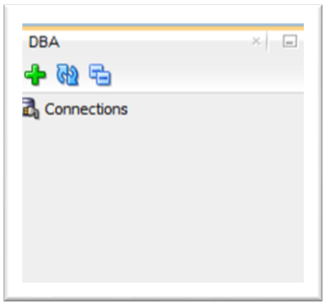


Figure 4.1.2 10

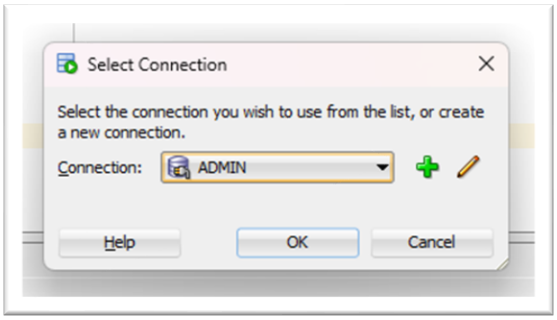


Figure 4.1.2 11

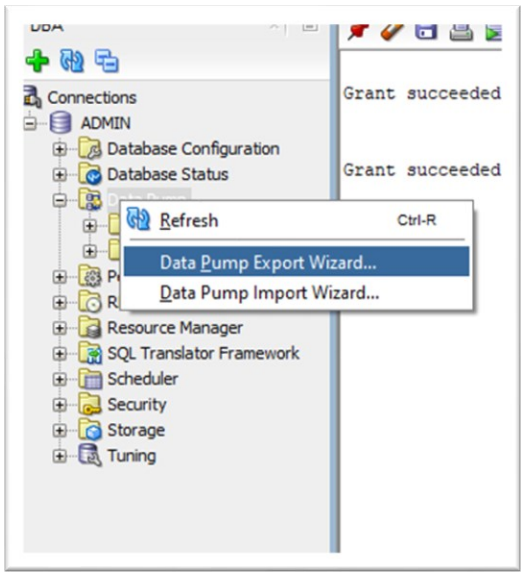


Figure 4.1.2 12

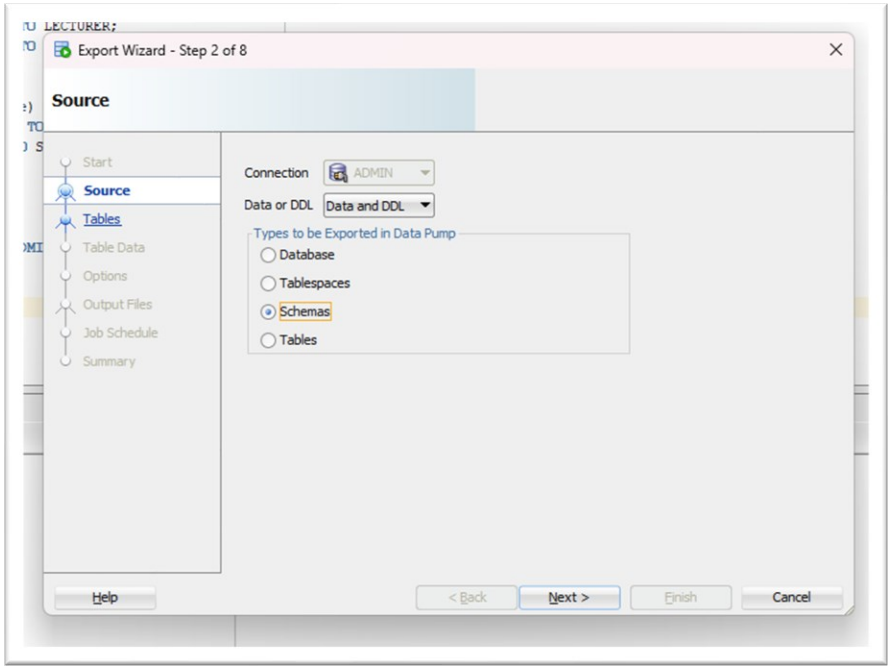


Figure 4.1.2 13

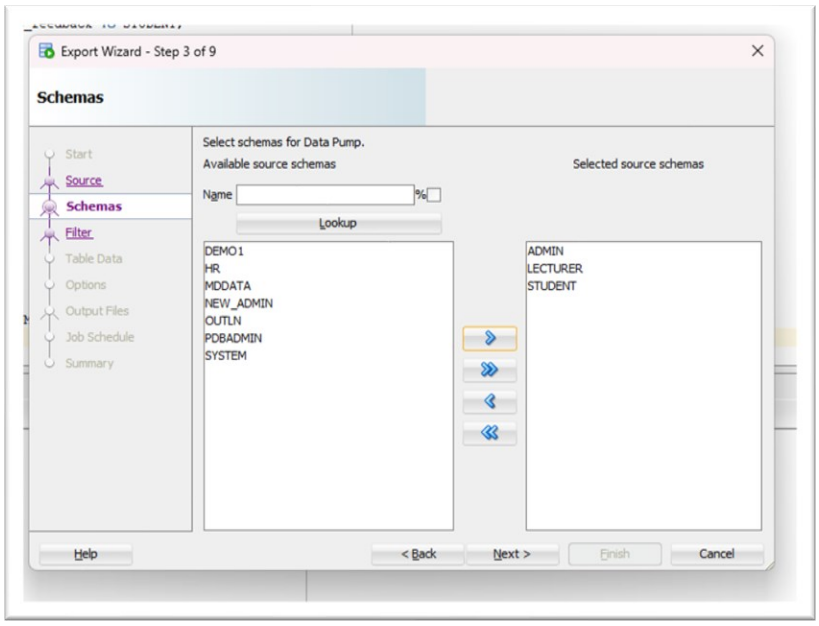


Figure 4.1.2 14

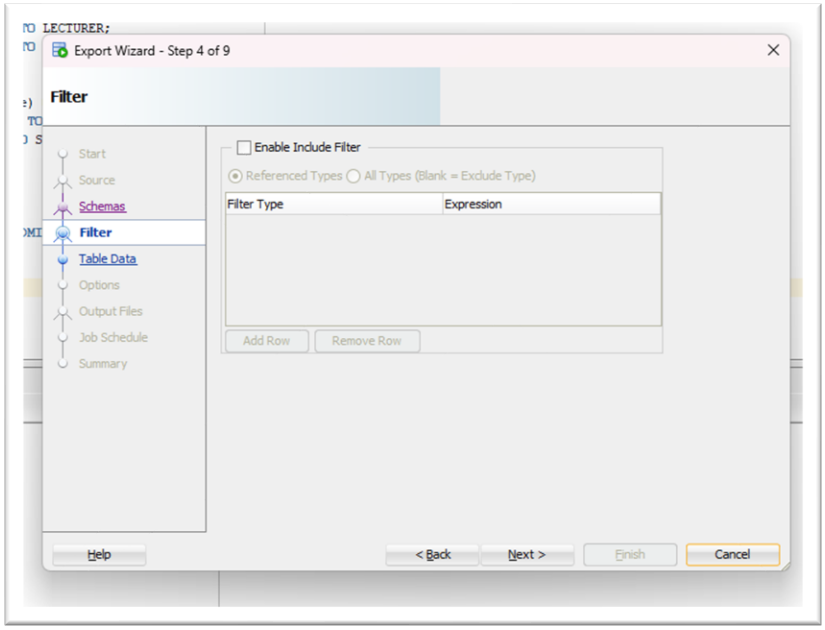


Figure 4.1.2 15

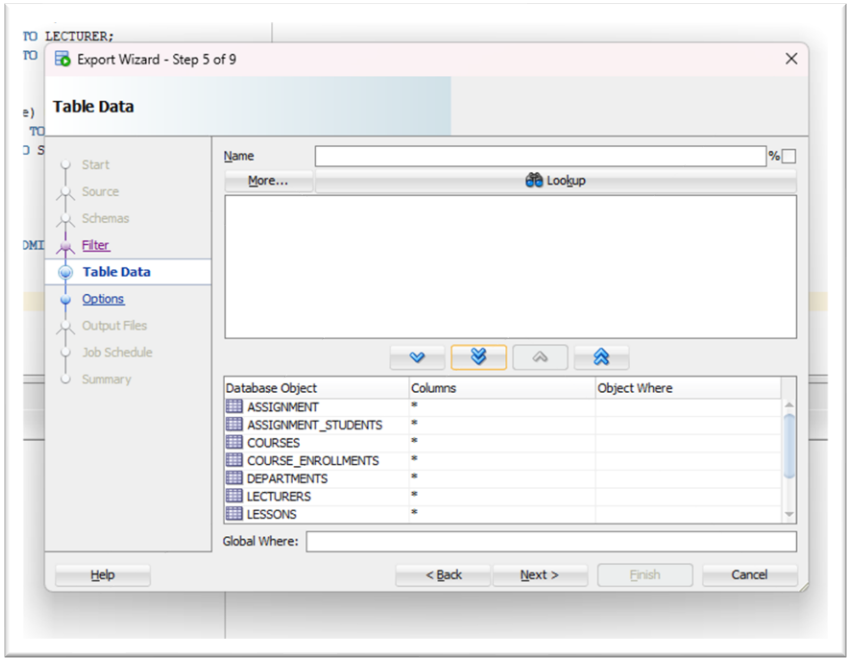


Figure 4.1.2 16

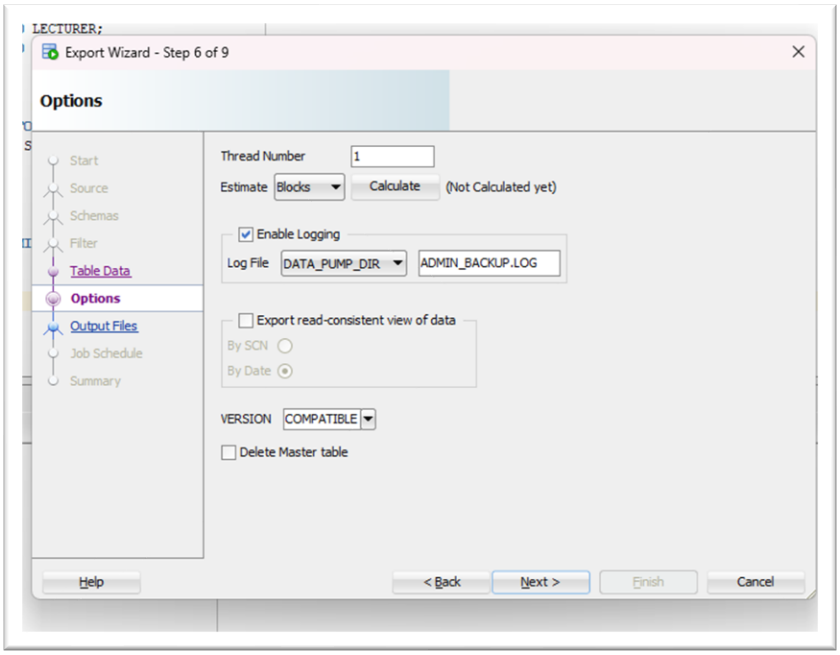


Figure 4.1.2 17

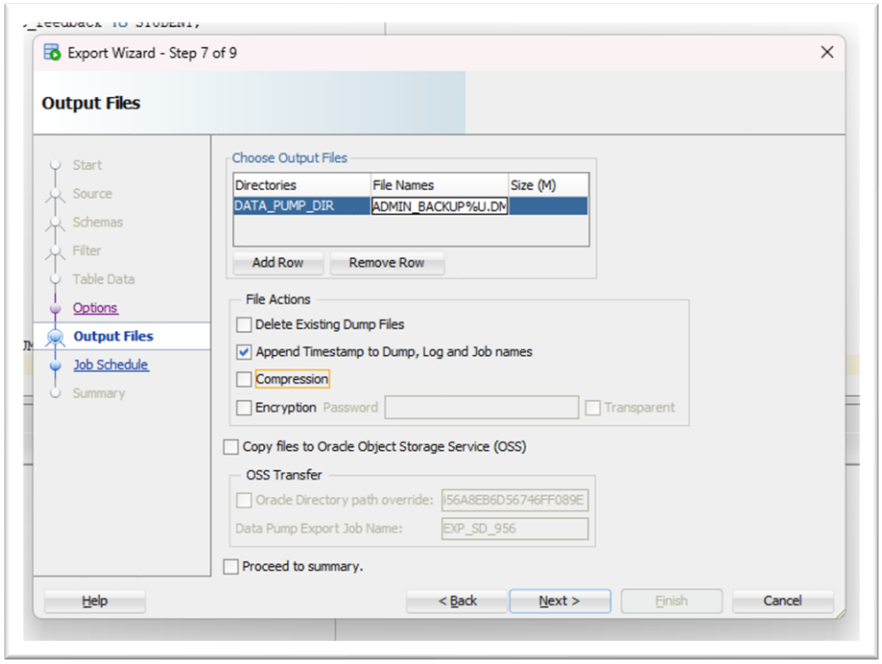


Figure 4.1.2 18

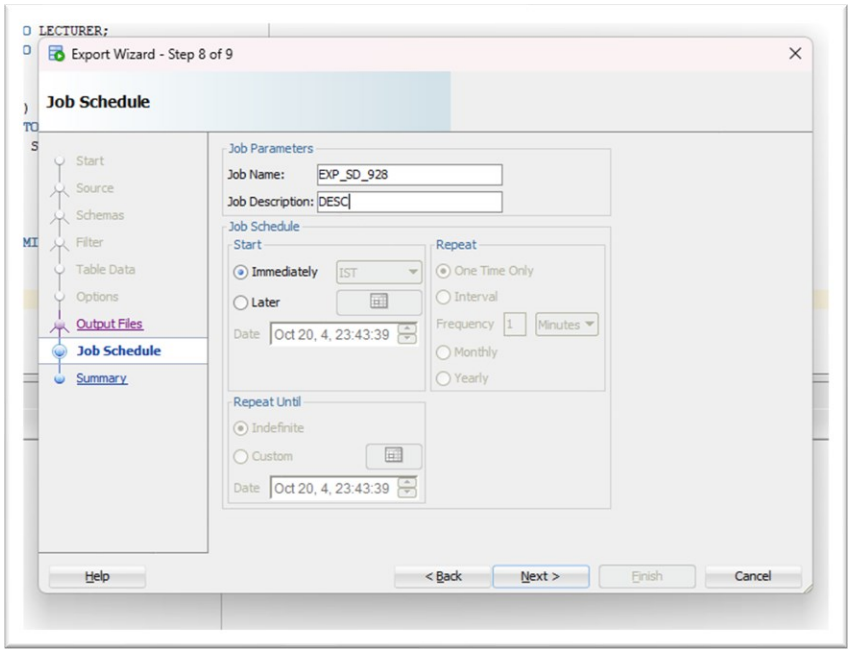


Figure 4.1.2 19

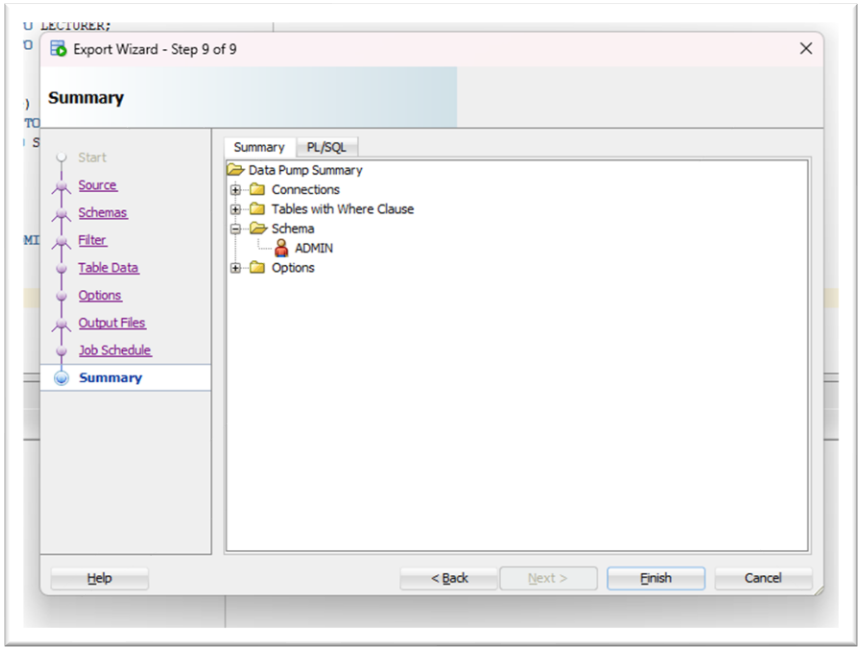


Figure 4.1.2 20

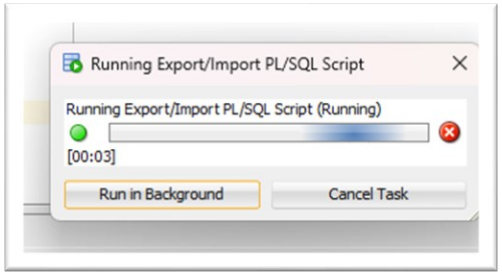


Figure 4.1.2 21

```
Dump file set for ADMIN.EXP_SD_956-11_41_41 is:

C:\APP\USER\PRODUCT\21C\ADMIN\XE\DPDUMP\0E6B7E28906F456A8EB6D56746FF089E\LMS_AD
MIN_BACKUP01-11_52_39.DMP
Job "ADMIN"."EXP_SD_956-11_41_41" successfully completed at Tue Oct 22 11:53:12
2024 elapsed 0 00:00:33
```

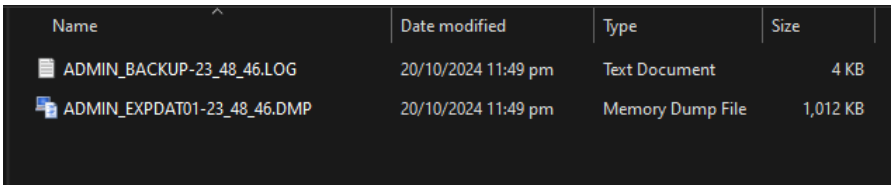


Figure 4.1.2 22

4.2.2 Data Pump import

CREATE USER ADMIN_BACKUP IDENTIFIED BY new123;

```
User ADMIN_BACKUP created.
```

GRANT ALL PRIVILEGES TO ADMIN_BACKUP;

```
Grant succeeded.
```

Figure 4.2.2 1

GRANT READ,WRITE ON DIRECTORY DATA_PUMP_DIR TO ADMIN_BACKUP;

```
Grant succeeded.
```

Figure 4.2.2 2

GRANT DBA TO ADMIN_BACKUP;

```
Grant succeeded.
```

Figure 4.2.2 3

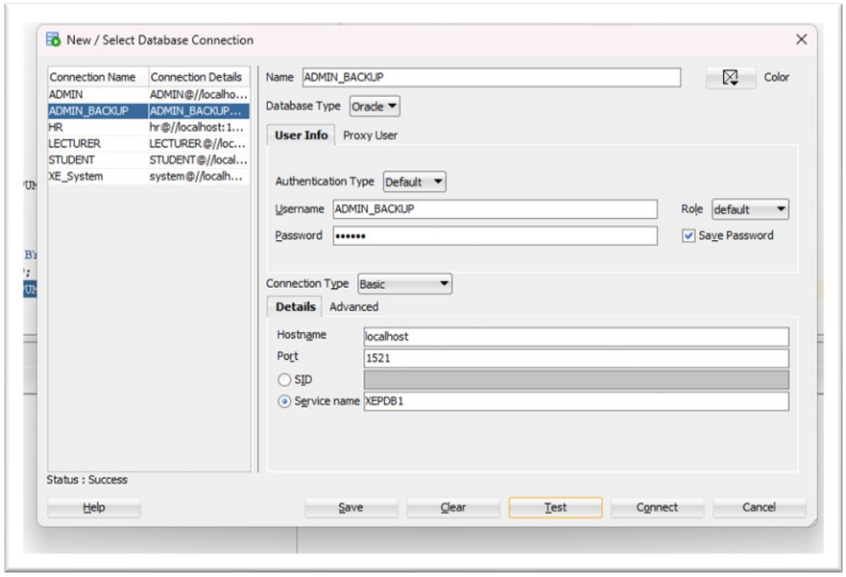


Figure 4.2.2 4

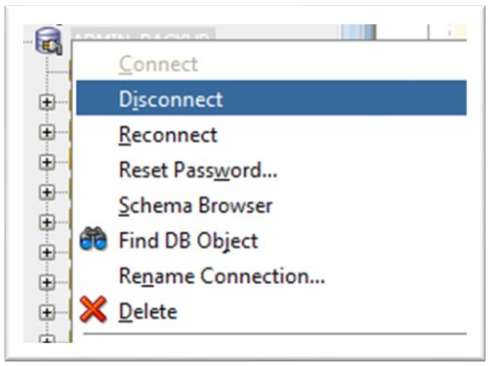


Figure 4.2.2 5

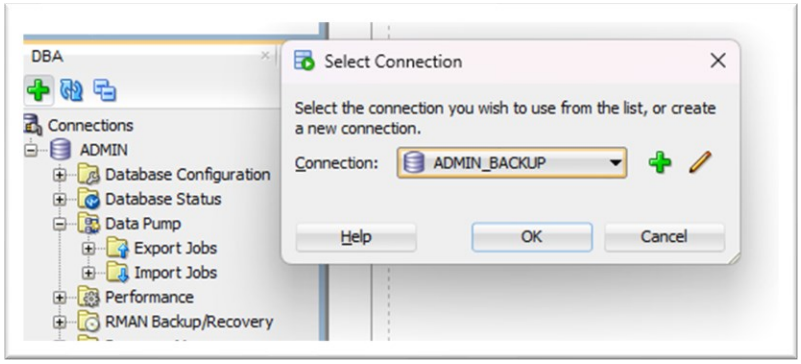


Figure 4.2.2 6

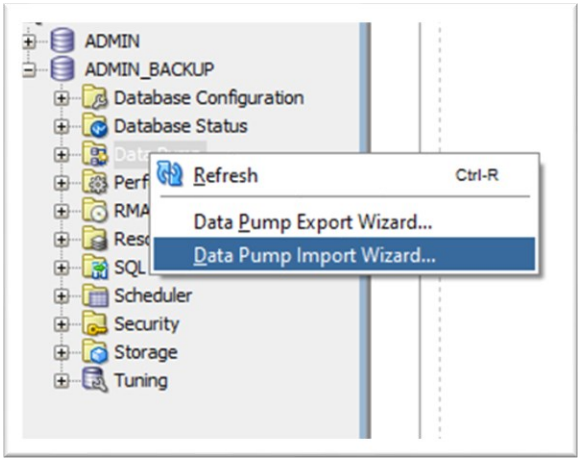


Figure 4.2.2 7

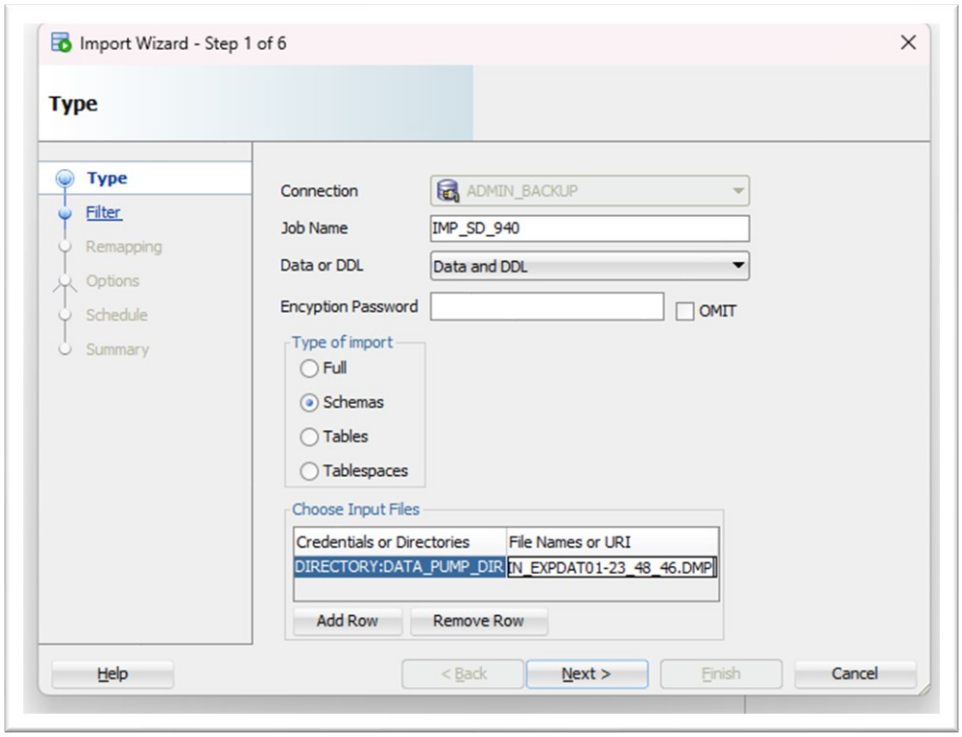


Figure 4.2.2 8

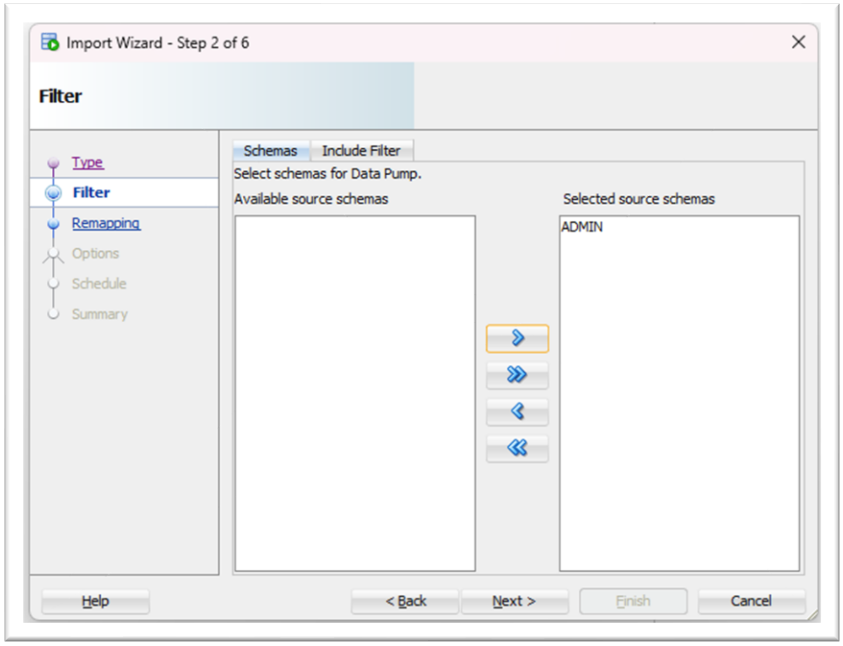


Figure 4.2.2 9

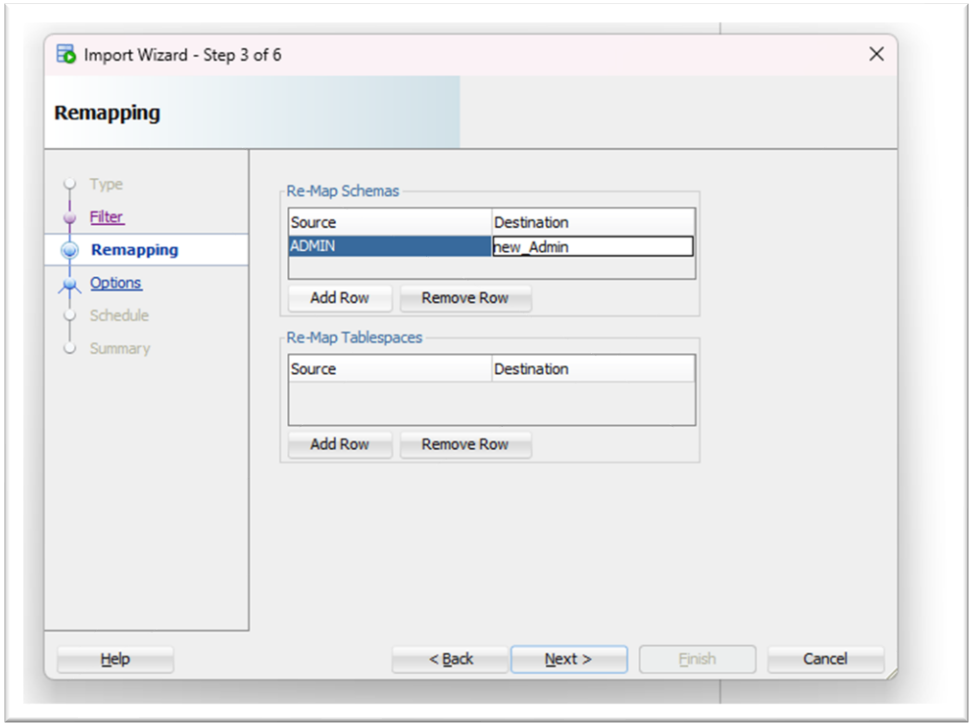


Figure 4.2.2 10

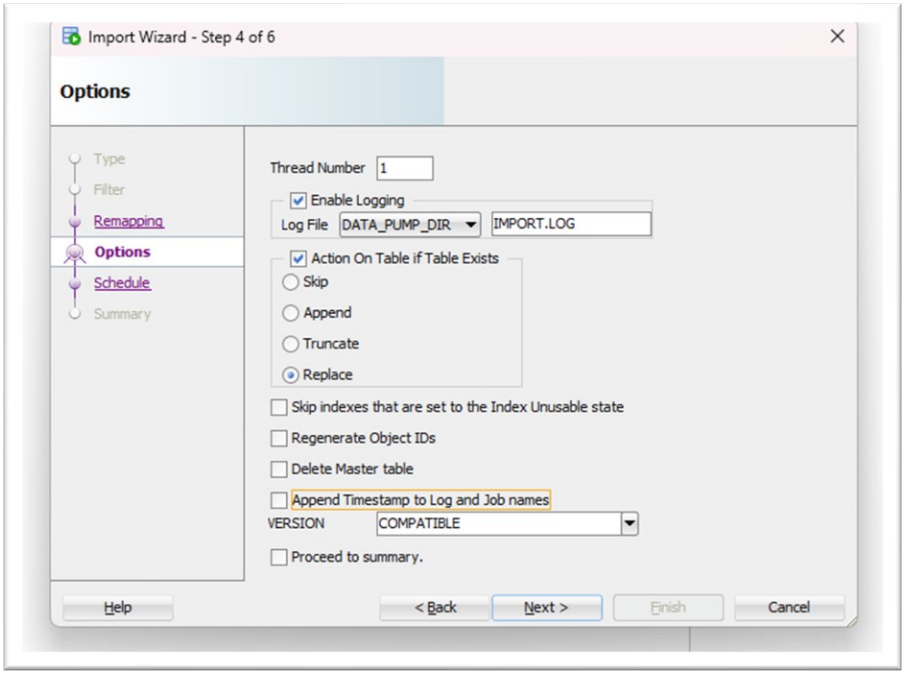


Figure 4.2.2 11

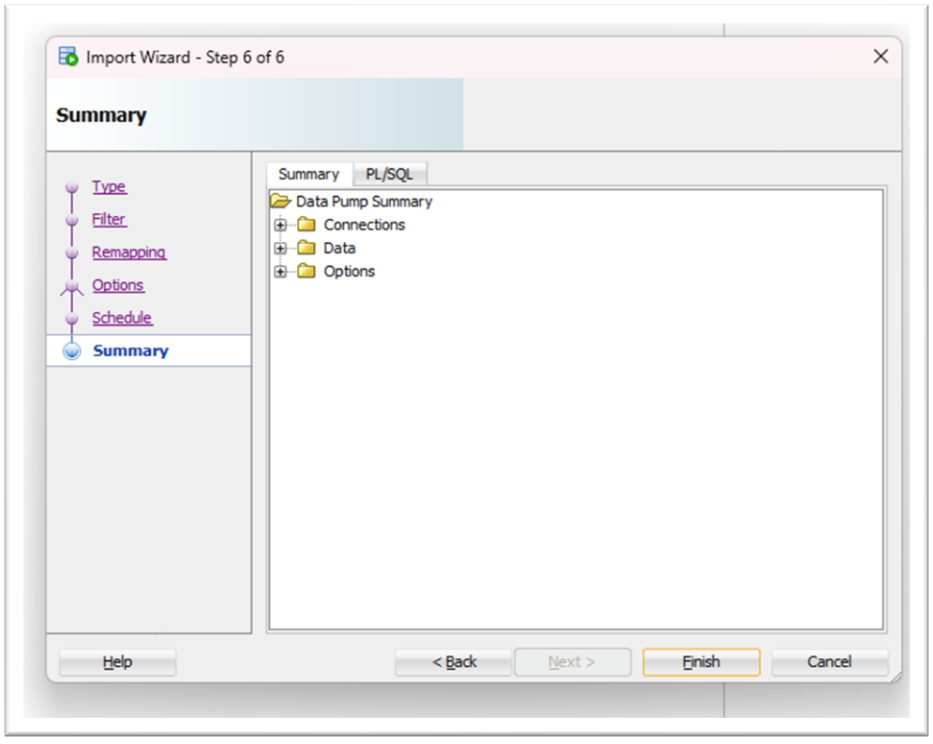


Figure 4.2.2 12

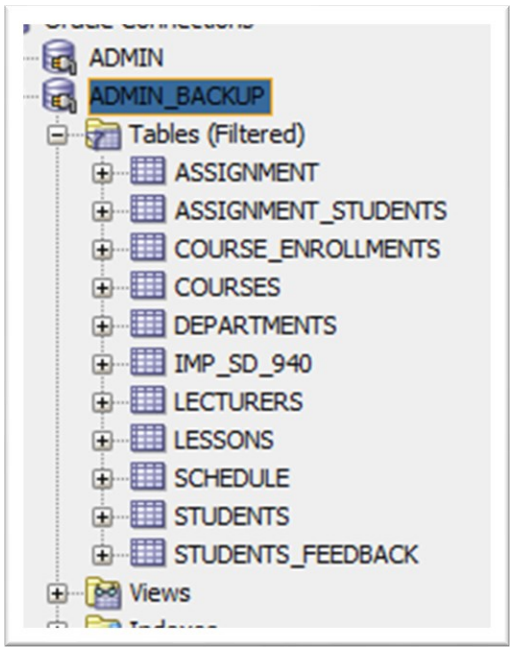


Figure 4.2.2 13

4.3 Backup Plan

4.3.1 Daily Incremental Backups

A daily incremental backup will capture any changes made to the database during the day. This reduces storage costs and ensures that recent data can be restored quickly.

4.3.2 Weekly Full Backups

A full backup will be performed weekly to provide a complete copy of the database. This ensures that if a major failure occurs, the entire system can be restored from the last full backup.

4.3.3 Monthly Full Backups

Every month, a full backup will be stored offsite for disaster recovery purposes. This backup ensures data can be restored even in the case of significant system failures, natural disasters, or data center issues.

4.3.4 Backup Testing

Regular tests will be conducted on the backup files to verify their integrity and ensure they can be restored successfully when needed.

05. Database Administration

5.1 User Roles and Access Control

In the Learning Management System (LMS), different user roles are defined to manage access to the database, ensuring that users have the permissions necessary for their functions while adhering to the principle of least privilege. The following roles are established:

5.1.1 Admin User

Full access to all tables and the ability to perform all CRUD (Create, Read, Update, Delete) operations.

The admin user is responsible for managing the entire database system, including creating and managing users, performing backups, monitoring performance, and ensuring data integrity and security.

5.1.2 Lecturer User

Restricted access to manage their lessons, assignments, and view students' feedback.

Lecturers can create, read, update, and delete (CRUD) their lessons and assignments. They can also view feedback from students related to their courses. However, they do not have access to modify other lecturers' data or sensitive system configurations.

5.1.3 Student User

Limited access to their own course enrollments, assignments, and feedback submissions.

Students can view their course enrollments, submit assignments, and access their feedback.

They cannot modify or view other students' information or system-level configurations.

5.2 Enforcing Least Privilege Principles

The system is designed to enforce the principle of least privilege, minimizing access risks by ensuring that users have only the permissions necessary to perform their tasks.

5.2.1 Role-Based Access Control (RBAC)

Each user role is associated with specific permissions. For example, while the admin has full access, lecturers and students are granted only the necessary permissions relevant to their roles. This approach reduces the risk of unauthorized access to sensitive data.

5.2.2 Database Permissions

SQL commands are utilized to set permissions for each role. For instance, the admin role is granted privileges on all tables, while the lecturer role is granted permissions only on specific tables related to their lessons and assignments. The student role is similarly restricted.

5.2.3 Auditing and Monitoring

Regular audits are conducted to monitor access and modifications within the database. This ensures that any unauthorized access attempts can be detected and addressed promptly.

5.2.4 Use of Views

Database views can be created to provide a simplified and secure way for lecturers and students to access the data they need without exposing the underlying tables. For instance, a view can be created for lecturers to see only their lessons and related feedback, preventing access to other lecturers' data.

06. Cloud Platform Deployment

6.1 Proposed Cloud-Based Database Platform

The business can migrate the Learning Management System (LMS) to Amazon Web Services (AWS) Relational Database Service (RDS). AWS RDS is a highly reliable, scalable, and cost-effective solution for managing databases in the cloud. It supports several database engines, such as Oracle, MySQL, PostgreSQL, and Microsoft SQL Server, allowing the business to choose the most appropriate engine for its needs.

6.2 Justification for Choosing Amazon Web Services (AWS) Relational Database Service (RDS)

6.2.1 Scalability

AWS RDS provides automatic scaling, allowing the database to handle increasing amounts of data as the LMS grows.

6.2.2 Reliability

AWS RDS offers automated backups, patching, and replication, ensuring data availability and minimizing the risk of downtime.

6.2.3 Security

AWS RDS comes with built-in security features like encryption at rest and in transit, along with fine-grained access control using AWS Identity and Access Management.

6.2.3 Cost-Efficiency

AWS RDS is a fully managed service, reducing the need for database administration and maintenance.

6.3 Database Migration Plan

6.3.1 Assessment and Planning

Analyze the current database structure and data size.

Identify any application dependencies or features that might need adjustment during migration.

6.3.2 Select the Appropriate RDS Instance

Choose the right instance type and database engine in AWS RDS that aligns with current database requirements. AWS RDS supports Oracle, which simplifies the migration process without needing to change the database engine.

6.3.3 Database Backup and Export

Perform a full backup of the existing Oracle database and export all relevant data using Oracle's Data Pump utility or other backup tools. Ensure that a full export of the database is available to minimize downtime during migration.

6.3.4 Create an AWS RDS Instance

In the AWS Management Console, create a new Oracle RDS instance, configuring it for the required storage, security groups, and backup policies. Enable encryption and set up access control with IAM roles to ensure secure management.

6.3.5 Migrate Data

Use AWS Database Migration Service (DMS) to migrate the data from the on-premises Oracle database to the new RDS instance. This service allows for continuous replication, ensuring minimal downtime during migration.

6.3.6 Testing

After migration, conduct thorough testing of the RDS instance to ensure data integrity, performance, and compatibility with the LMS. Test all application features that rely on the database, such as user authentication, course management, and reporting.

6.3.7 Switch to Production

Once testing is complete and all issues have been resolved, switch the production environment to use the new AWS RDS database. Ensure that the application and services are pointing to the new RDS instance.

6.3.8 Monitor and Optimize

Monitor the performance of the RDS instance using AWS CloudWatch and other monitoring tools. Adjust the instance size and settings based on performance metrics and optimize the database as necessary.

07. Conclusion

In conclusion, the design of a Learning Management System (LMS) for a university requires careful consideration of various components such as lessons, schedules, assessments, student submissions, courses, and course enrollments. By constructing well-defined tables, implementing procedures, and utilizing functions and views, the LMS can efficiently manage and analyze academic data to support the university's operations. The entity-relationship diagram provides a clear visual representation of how different entities, including admin, lecturers, and students, interact within the system. Through the use of SQL queries, relevant information can be retrieved, updated, and presented to enhance academic decision-making processes. With a well-structured database and optimized queries, the LMS can streamline educational workflows, improve student and faculty experiences, and contribute to the overall efficiency of the university's academic management system.