**NATIONAL INSTITUTE OF BUSINESS MANAGEMENT**

**HIGHER NATIONAL DIPLOMA IN SOFTWARE ENGINEERING**

**COURSEWORK**

**DATABASE MANAGEMENT 2**

**LEARNING MANAGEMENT SYSTEM (LMS)**

**SUBMITTED BY**

**MAHNDSE24.2F-022 K P AMOD MATHEESHA**

**MAHNDSE24.1F-024 T M NANAYAKKARA**

**MAHNDSE24.2F-013 P P J R PATABANDIGE**

**MAHNDSE24.2F-012 A A R WEHARI**

**Date of Submission: 2024-10-21**

# Declaration

“I certify that this project does not incorporate without acknowledgement, any material previously submitted for a Higher National Diploma in any institution and to the best of my knowledge and belief ,it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my project report, if accepted, to be made available for photocopying and for interlibrary loans, and for the title and summary to be made available to outside organizations”

**Index no Name Signature**

MAHNDSE24.2F-022 K P AMOD MATHEESHA ……………………….

MAHNDSE24.1F-024 T M NANAYAKKARA ……………………….

MAHNDSE24.2F-013 P P J R PATABANDIGE ……………………….

MAHNDSE24.2F-012 A A R WEHARI ……………………….

**Table of Contents**

**Table of Figures**

# Chapter 1

## **Introduction**

# Chapter 2

## **Database Design**

### **Logical Design**

### **Physical Design**

# Chapter 3

## **PL/SQL Programs for CRUD Operations**

### **Create Admin User in XE\_System**

CREATE USER ADMIN IDENTIFIED BY ADMIN123

DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp

QUOTA 20M ON users

ACCOUNT UNLOCK;

/

### **Grant All Privileges to Admin User**

GRANT ALL PRIVILEGES TO ADMIN;

A black text on a white background

Description automatically generated

### **Create LECTURER User in XE\_System**

CREATE USER LECTURER IDENTIFIED BY LECTURER123

DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp

QUOTA 20M ON users

ACCOUNT UNLOCK;

/

### **Create STUDENT User in XE\_System**

CREATE USER STUDENT IDENTIFIED BY STUDENT123

DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp

QUOTA 20M ON users

ACCOUNT UNLOCK;

/

A black text on a white background

Description automatically generated

### **Create User Role**

CREATE ROLE USER\_ROLE;

### **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;

A screenshot of a computer error

Description automatically generated

### **Connect ADMIN**

A screenshot of a computer

Description automatically generated

### **Connect LECTURER**

A screenshot of a computer

Description automatically generated

### **Connect STUDENT**

A screenshot of a computer

Description automatically generated

### **CREATE** **departments TABLE**

CREATE TABLE departments (

department\_id NUMBER PRIMARY KEY,

department\_name VARCHAR2(100) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

A close up of a sign

Description automatically generated

### **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');

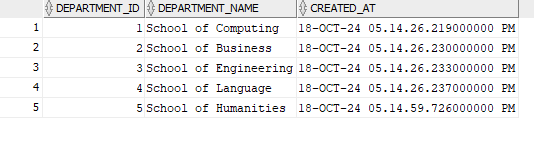
INSERT INTO departments (department\_id, department\_name) VALUES (5, 'School of Humanities');

A white background with black text

Description automatically generated

### **Retrieve All Records from departments Table**

SELECT \* FROM departments;



### **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)

);

A close up of a logo

Description automatically generated

### **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');

A white background with black text

Description automatically generated

### **Retrieve All Records from lecturers Table**

SELECT \* FROM lecturers;

A screenshot of a computer

Description automatically generated

### **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)

);

A close up of a logo

Description automatically generated

### **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

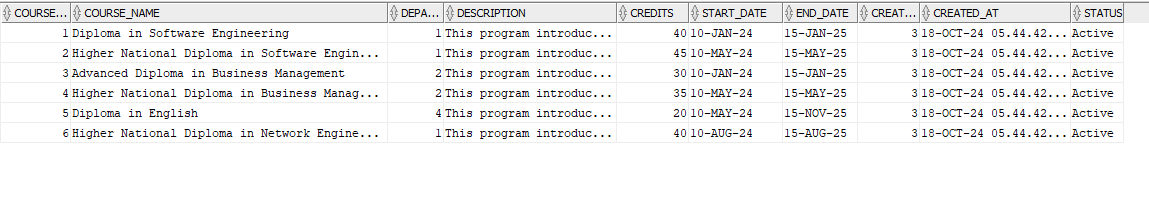
(6, 'Higher National Diploma in Network Engineering', 1, 'This program introduces students to the fundamentals of Network', 40, TO\_DATE('2024-08-10', 'YYYY-MM-DD'), TO\_DATE('2025-08-15', 'YYYY-MM-DD'), 3);

A white background with black text

Description automatically generated

### **Retrieve All Records from courses Table**

SELECT \* FROM courses;



### **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)

);

A black text on a white background

Description automatically generated

### **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);

A white background with black text

Description automatically generated

### **Retrieve All Records from lessons Table**

SELECT \* FROM lessons;

A screenshot of a computer

Description automatically generated

### **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

);

A close up of a logo

Description automatically generated

### **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)

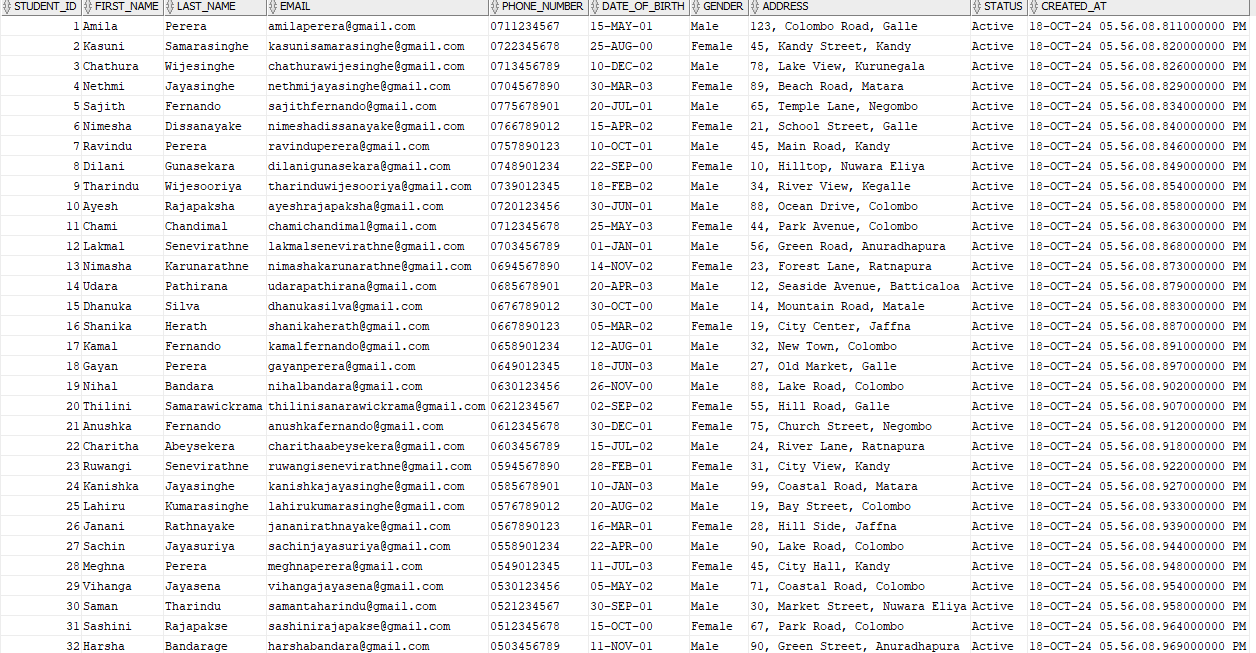
VALUES (40, 'Chamila', 'Gunarathne', 'chamilagunarathne@gmail.com', '0421234567', TO\_DATE('2000-07-29', 'YYYY-MM-DD'), 'Female', '12, Lake View, Kandy');

A white screen with black text

Description automatically generated

### **Retrieve All Records from students Table**

SELECT \* FROM students;



### **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)

);

A close up of a sign

Description automatically generated

### **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);

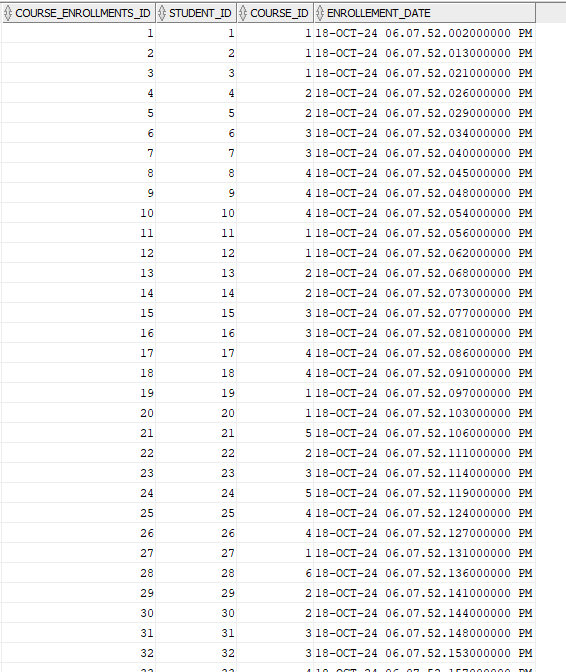
INSERT INTO course\_enrollments (course\_enrollments\_id, student\_id, course\_id) VALUES (40, 40, 3);

A white background with black text

Description automatically generated

### **Retrieve All Records from course\_enrollments Table**

SELECT \* FROM course\_enrollments;



### **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)

);

A black and red text

Description automatically generated

### **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'));

A black and white text

Description automatically generated

### **Retrieve All Records from assignment Table**

SELECT \* FROM assignment;

A screenshot of a computer

Description automatically generated

### **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)

);

A black text on a white background

Description automatically generated

### **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);

A black and white text

Description automatically generated

### **Retrieve All Records from assignment\_students Table**

SELECT \* FROM assignment\_students;



### **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)

);

A black text on a white background

Description automatically generated

### **Insert Records into students\_feedback** **Table**

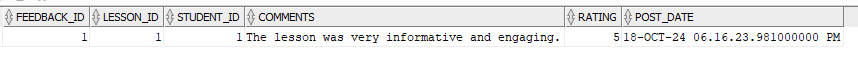
INSERT INTO students\_feedback (feedback\_id, lesson\_id, student\_id, comments, rating) VALUES (1, 1, 1, 'The lesson was very informative and engaging.', 5);

A black and white text

Description automatically generated

### **Retrieve All Records from students\_feedback Table**

SELECT \* FROM students\_feedback;



### **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)

);



### **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');

A screen shot of a computer

Description automatically generated

### **Retrieve All Records from schedule Table**

SELECT \* FROM schedule;

A screenshot of a schedule

Description automatically generated

### **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 ;

A close up of a sign

Description automatically generated

### **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);

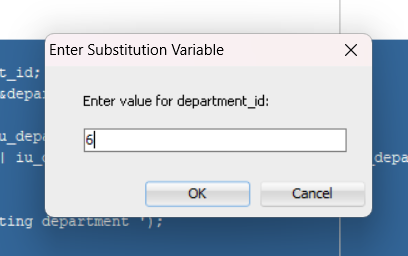
DBMS\_OUTPUT.PUT\_LINE('Department "' || iu\_department\_name || '" with ID ' || iu\_department\_id || ' has been successfully added.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error inserting department ');

END;



A screenshot of a computer error

Description automatically generated

SELECT \* FROM departments;



### **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;

A close up of a sign

Description automatically generated

### **Update Department Using Variables**

SET SERVEROUTPUT ON;

DECLARE

v\_department\_id NUMBER := &department\_id;

v\_department\_name VARCHAR2(100) := '&department\_name';

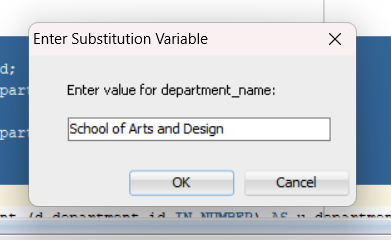
BEGIN

update\_department(v\_department\_id, v\_department\_name);

END;

A screenshot of a computer error

Description automatically generated



SELECT \* FROM departments;



A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A close up of a text

Description automatically generated

### **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;

A close up of a sign

Description automatically generated

### **Delete Department Using Variables**

SET SERVEROUTPUT ON;

DECLARE

v\_department\_id NUMBER := &department\_id;

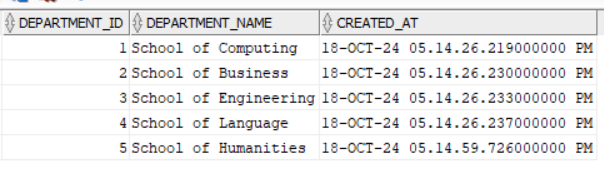
BEGIN

delete\_department(v\_department\_id);

END;



SELECT \* FROM departments;



A screenshot of a computer error

Description automatically generated

A black and white text

Description automatically generated

### **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;

A black text on a white background

Description automatically generated

### **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');

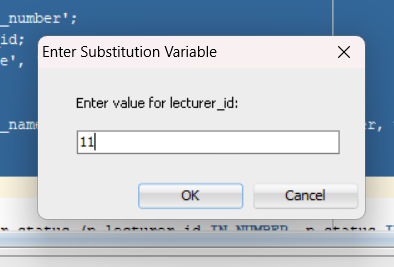
v\_salary NUMBER := &salary;

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;

/

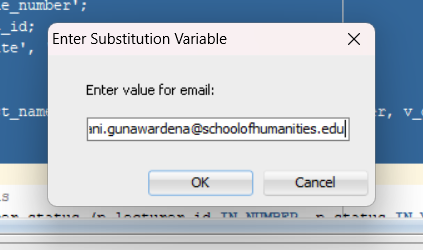


A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

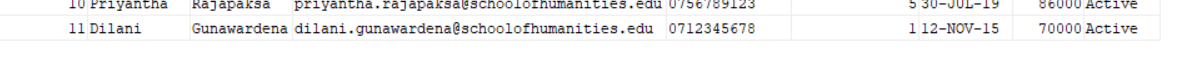
A screenshot of a computer error

Description automatically generated

A screenshot of a computer screen

Description automatically generated

SELECT \* FROM lecturers;



### **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;

A black text on a white background

Description automatically generated

### **Update Lecturer Using Variables**

SET SERVEROUTPUT ON;

DECLARE

v\_lecturer\_id NUMBER := &lecturer\_id;

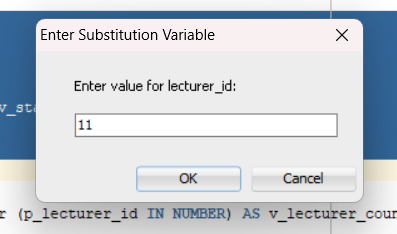
v\_status VARCHAR2(20) := '&status';

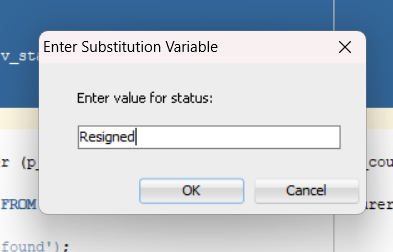
BEGIN

update\_lecturer\_status(v\_lecturer\_id, v\_status);

END;

/





SELECT \* FROM lecturers;



A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated



### **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;

A black and white text

Description automatically generated

### **Delete Lecturer Using Variable**

SET SERVEROUTPUT ON;

DECLARE

v\_lecturer\_id NUMBER := &lecturer\_id;

BEGIN

delete\_lecturer(v\_lecturer\_id);

END;

/

A screenshot of a computer error

Description automatically generated

SELECT \* FROM lecturers;

A screen shot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated



### **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;

A black text on a white background

Description automatically generated

### **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;

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

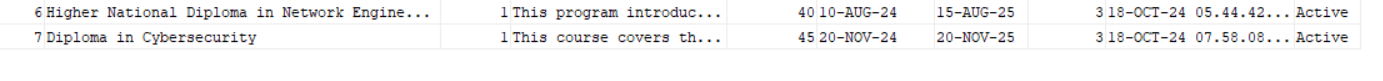
A screenshot of a computer error

Description automatically generated

A black text on a white background

Description automatically generated

SELECT \* FROM courses;



### **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;

A close up of a text

Description automatically generated

### **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;

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

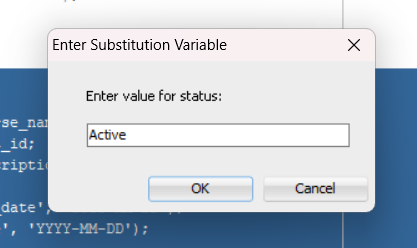
Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

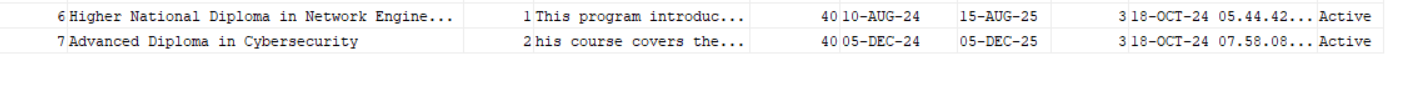
Description automatically generated



A black and white text

Description automatically generated

SELECT \* FROM courses;



### **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;

A close up of a text

Description automatically generated

### **Delete Course Using Variables**

SET SERVEROUTPUT ON;

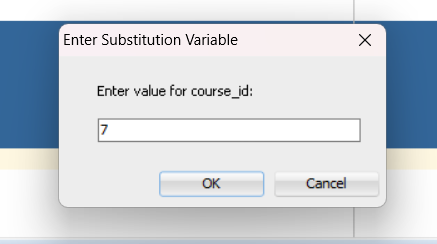
DECLARE

v\_course\_id NUMBER := &course\_id;

BEGIN

delete\_course(v\_course\_id);

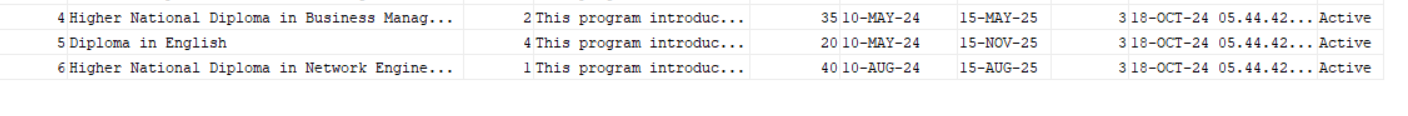
END;



A close up of a word

Description automatically generated

SELECT \* FROM courses;



### **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;

A black and white text

Description automatically generated

### **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;

/

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

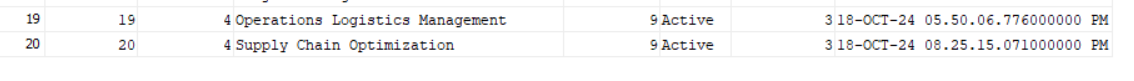
A screenshot of a computer

Description automatically generated

A close up of a word

Description automatically generated

SELECT \* FROM lessons;



### **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;

A black text on a white background

Description automatically generated

### **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;

/

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black and white text

Description automatically generated

SELECT \* FROM lessons;



### **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;

A black text on a white background

Description automatically generated

### **Delete Lesson Using Variables**

SET SERVEROUTPUT ON;

DECLARE

v\_lesson\_id NUMBER := &lesson\_id;

BEGIN

delete\_lesson(v\_lesson\_id);

END;

/

A screenshot of a computer error

Description automatically generated

A black text on a white background

Description automatically generated

### **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;

A close up of a text

Description automatically generated

### **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;

A black text on a white background

Description automatically generated

### **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');

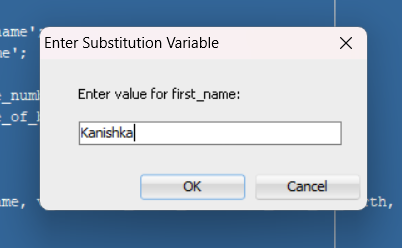
v\_gender VARCHAR2(15) := '&gender';

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;



A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A close up of a word

Description automatically generated

SELECT \* FROM students;



### **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;

A close up of a text

Description automatically generated

### **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';

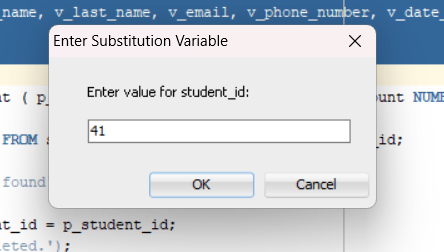
v\_address VARCHAR2(200) := '&address';

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;



A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

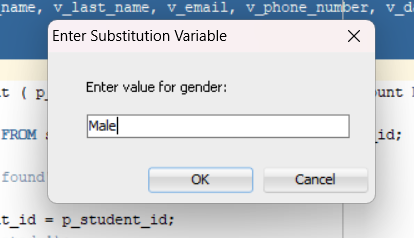
Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

A close up of a text

Description automatically generated

SELECT \* FROM students;



### **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;

A black text on a white background

Description automatically generated

### **Delete Student Using Variable**

SET SERVEROUTPUT ON;

DECLARE

v\_student\_id NUMBER := &student\_id;

BEGIN

delete\_student(v\_student\_id);

END;

A screenshot of a computer error

Description automatically generated

A close up of words

Description automatically generated

### **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;

A black text on a white background

Description automatically generated

### **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 ;

A black text on a white background

Description automatically generated

### **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;

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A close up of a text

Description automatically generated

SELECT \* FROM course\_enrollments;



### **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;

A close up of a sign

Description automatically generated

### **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;

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A black text on a white background

Description automatically generated

SELECT \* FROM course\_enrollments;



### **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;

A close up of a sign

Description automatically generated

### **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;

A screenshot of a computer error

Description automatically generated

A black and white text

Description automatically generated

### **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;

/

A black text on a white background

Description automatically generated

### **Create Sequence for Assignment Students IDs**

CREATE SEQUENCE assignment\_students\_seq

START WITH 1

INCREMENT BY 1;

A black text on a white background

Description automatically generated

### **Trigger to Add Students to Assignment After Insertion**

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;

/

A close up of a text

Description automatically generated

### **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;

/



DELETE FROM assignment\_students WHERE assignment\_students\_id = 1;



SELECT \* FROM assignment\_students;



DELETE FROM assignment WHERE assignment\_id = 1;

A black text on a white background

Description automatically generated

SELECT \* FROM assignment;

A computer screen shot of a computer

Description automatically generated

### **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;

/

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A white screen with black text

Description automatically generated

SELECT \* FROM assignment;

A screenshot of a computer

Description automatically generated

SELECT \* FROM assignment\_students;

A screenshot of a computer

Description automatically generated

### **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;

/

A computer screen shot of a computer

Description automatically generated

### **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.');

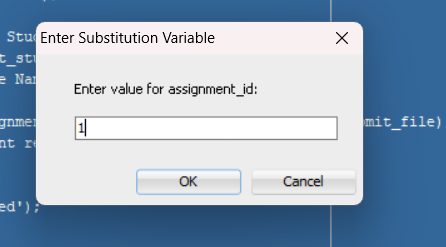
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred');

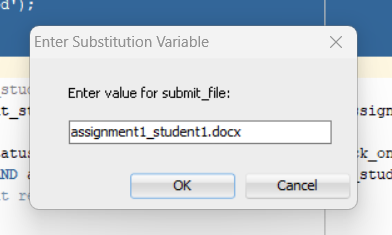
END;

/



A screenshot of a computer error

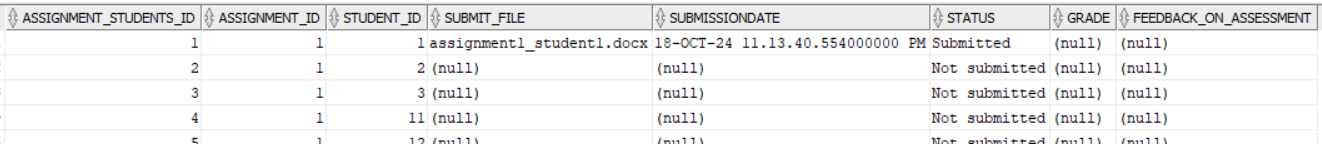
Description automatically generated



A white screen with black text

Description automatically generated

SELECT \* FROM assignment\_students;



### **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;

/

A black and white text

Description automatically generated

### **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;

/

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

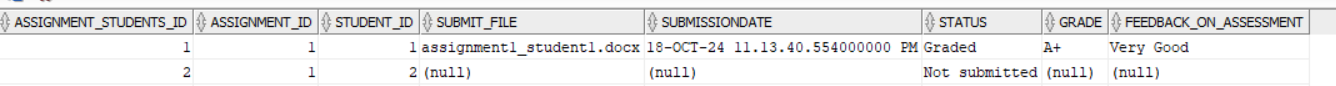
A screenshot of a computer

Description automatically generated

A white screen with black text

Description automatically generated

SELECT \* FROM assignment\_students;



### **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;

/

A close up of a text

Description automatically generated

### **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;

/

A close up of a text

Description automatically generated

### **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;

/

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error message

Description automatically generated

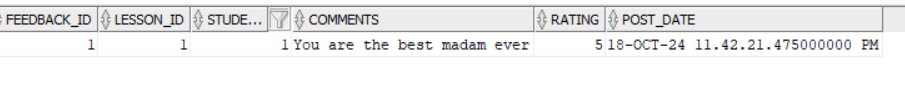
A screenshot of a computer error

Description automatically generated

A black text on a white background

Description automatically generated

SELECT \* FROM students\_feedback;



### **Insert Student Lesson Feedback with Invalid Student ID**

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black text on a white background

Description automatically generated

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);

A white background with black text

Description automatically generated

### **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;

/

A close up of a text

Description automatically generated

### **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;

/

**A screenshot of a computer error

Description automatically generated**

**A computer program on a white background

Description automatically generated**

**A screenshot of a computer error

Description automatically generated**

**A close up of black text

Description automatically generated**

### **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;

/



### **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;

/



### **Execute Schedule Addition Procedure**

BEGIN

add\_schedule\_proc(1, TO\_DATE('2024-10-26', 'YYYY-MM-DD'), 'Lecture Hall 05');

END;

/



select \* from schedule;

A screenshot of a computer

Description automatically generated

### **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;

/



### **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 USER\_ROLE TO STUDENT;

GRANT USER\_ROLE TO LECTURER;

A screenshot of a computer

Description automatically generated

### **Grant DML and Execute Permissions to Lecturers**

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

GRANT UPDATE ON ADMIN.assignment\_students TO LECTURER;

GRANT EXECUTE ON ADMIN.insert\_assignment TO LECTURER;

GRANT EXECUTE ON ADMIN.grade\_assignment\_student TO LECTURER;

GRANT EXECUTE ON ADMIN.get\_feedback\_by\_lecturer TO LECTURER;

A screenshot of a computer

Description automatically generated

### **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;

A screen shot of a computer

Description automatically generated

### **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;

A screenshot of a computer

Description automatically generated

### **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;

A screenshot of a computer

Description automatically generated

### **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 ';

A screenshot of a computer

Description automatically generated

### **All Assignments Due in Next 7 Days**

INSERT INTO assignment (assignment\_title, assignment\_fiies, lesson\_id, end\_date)

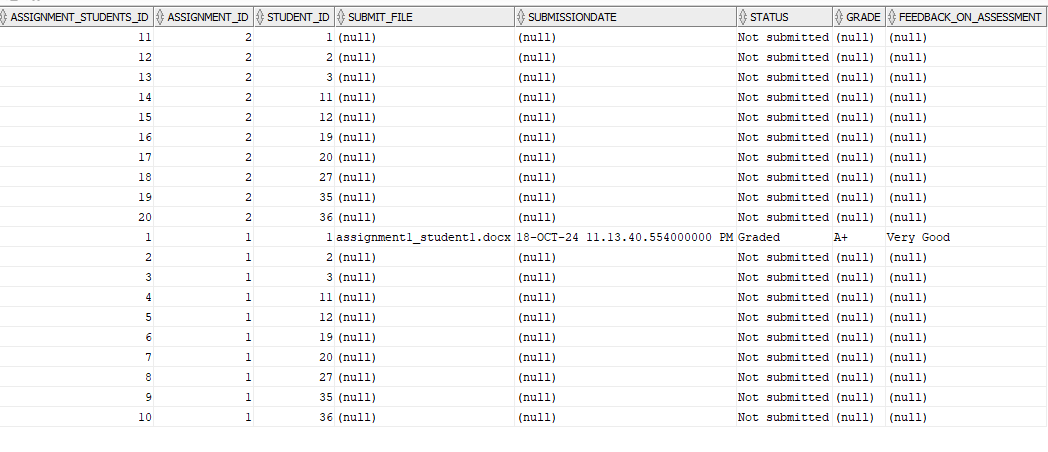
VALUES (' Database Management System Assignment', 'dbms\_assignment.docx', 3, TO\_DATE('2024-10-21', 'YYYY-MM-DD'));

SELECT \* from assignment;

A screen shot of a computer

Description automatically generated

SELECT \* from assignment\_students;



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;

A close up of a computer screen

Description automatically generated

### **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;

A screenshot of a graph

Description automatically generated

### **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;

A screenshot of a computer

Description automatically generated

### **Update Course Enrollment for Students**

UPDATE course\_enrollments SET course\_id = 5 WHERE

student\_id IN (SELECT student\_id FROM course\_enrollments WHERE course\_id = 6);



### **Courses Without any enrollments**

SELECT c.\* FROM courses c

LEFT JOIN course\_enrollments ce ON c.course\_id = ce.course\_id

WHERE ce.course\_id IS NULL;



### **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;

### **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 ;

### **Fetch Students Based on Name, Email, and Phone**

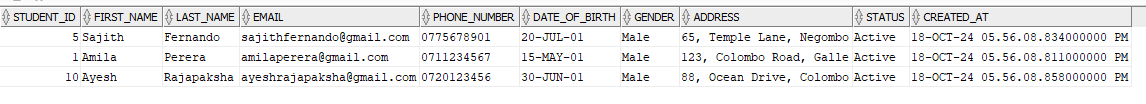
SELECT \* FROM students WHERE

(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;



### **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

A black and white text

Description automatically generated

### **Retrieve Upcoming Assignments from View**

SELECT \* FROM upcoming\_assignments;

A screen shot of a computer

Description automatically generated

### **Create 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;

A black and white text

Description automatically generated

### **Retrieve department course lesson from View**

SELECT \* FROM department\_course\_lesson\_stats;

A screenshot of a computer

Description automatically generated

### **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

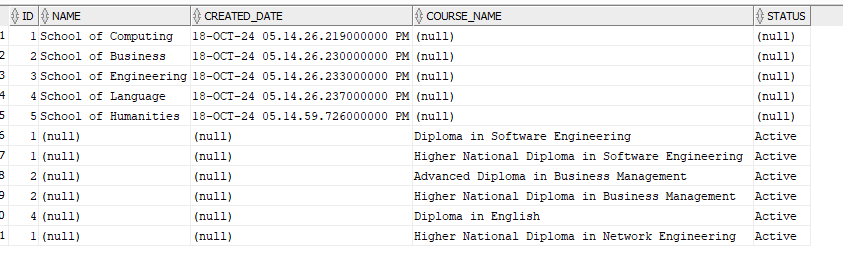
FROM departments

UNION ALL

SELECT department\_id AS id,NULL AS name,NULL AS created\_date,

course\_name AS course\_name,status

FROM courses;



### **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;

/

A black and white text

Description automatically generated

### **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;

/

A screenshot of a computer

Description automatically generated



# Chapter 5

## **Database Administration**

# Chapter 6

## **Backup Plan**

# Chapter 7

## **Cloud Platform Deployment**