MENNA ALLAH AMR

# DBMS FOR UNIVERSITY

DOCUMENTATION

# Table Content

1.Database Design (Day1):

- ERD Diagram.
- mapping .
- Normalizing the schema.

2. SQL Implementation (Day2):

- Creating New User.
- Creating Tables to Build up The Schema.
- Inserting Sample Data into Database schema.
- Checking The Correctness Of The Schema.

3. PLSQL Implementation (Day3):

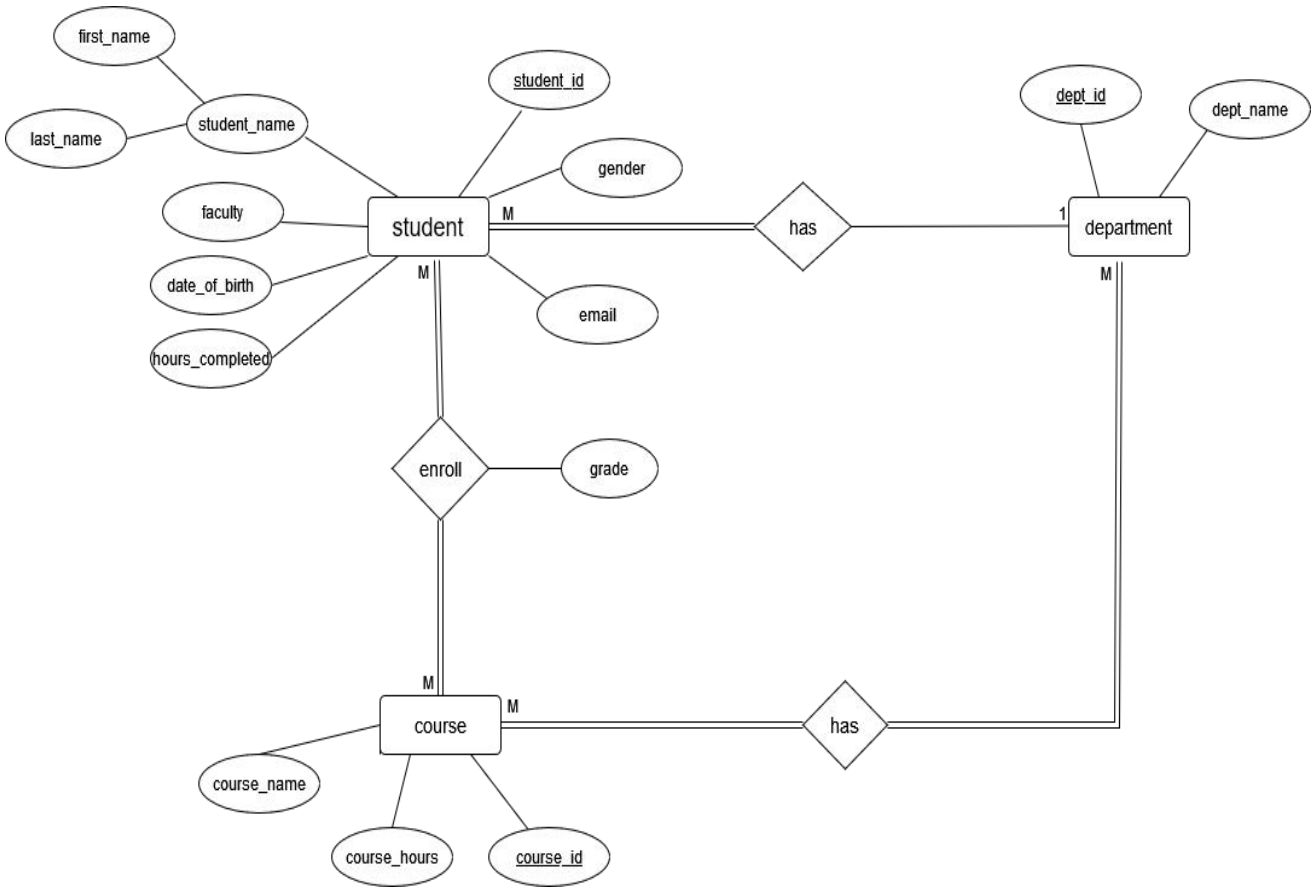
- Creating Procedure To update Student Name.
- Creating Procedure To Update Course Name.
- Creating Procedure To Delete Department from ALL Tables.
- Creating Function To Calculate Student GPA.
- Creating Trigger To Upload Old Data After Udate.
- Creating Trigger To Delete All The Data Of Student.

4. Automation Scripts (Day4);

- Bash Script For Database Backup.
- Bash script For Monitoring Disk Space And Sending Alert.
- Schedule A Script To Check For Memory Consumption And Send Notification.

# • Database Design (Day 1):

## • ER Diagram




1. Student to Department (Many-to-One):
   - Many students can belong to one department.
   - One department can have many students.
2. Student to Courses (Many-to-Many):
   - Many students can enroll in many courses.
   - Many courses can have many students enrolled.
3. Course to Department (Many-to-Many):
   - Many courses can be associated with many departments.
   - Many departments can offer many courses.

# -Mapping

Student table:

(student_id (Primary Key), first_name, last_name, date_of_birth, gender, faculty, email, hours_completed

, dep_id (Foreign Key referencing department.dep_id)

department table:

(dep_id (Primary Key), dept_name)

courses table:

(course_id (Primary Key), course_name, course_hours)

enrollment table:

(student_id (Composite Foreign Key referencing student.student_id),

course_id (Composite Foreign Key referencing courses.course_id),

grade)

Dept_course table:

(dept_id (Composite Foreign Key referencing department.dep_id),

course_id (Composite Foreign Key referencing courses.course_id))

# -Normalization

1. First Normal Form (1NF):

- No changes are needed for 1NF as all columns seem to contain atomic values.

2. Second Normal Form (2NF):

- No partial dependencies in the given schema.

3. Third Normal Form (3NF):

- No transitive dependencies.

# • SQL Implementation (Day 2):

- Creating new user by Run SQL command line and connecting it to toad.

Select Run SQL Command Line

```
SQL*Plus: Release 11.2.0.2.0 Production on Mon Feb 5 16:39:58 2024

Copyright (c) 1982, 2010, Oracle.  All rights reserved.

SQL> conn sys as sysdba;
Enter password:
Connected.
SQL> create user university identified by 123;

User created.

SQL> grant connect, resource to university;

Grant succeeded.

SQL> grant dba to university;

Grant succeeded.

SQL>
```

Run SQL Command Line

```
SQL*Plus: Release 11.2.0.2.0 Production on Mon Feb 5 16:48:32 2024

Copyright (c) 1982, 2010, Oracle.  All rights reserved.

SQL> conn university/123;
Connected.
SQL>
```

- Create tables to build up schema

```sql
-- Create department table
CREATE TABLE department (
   dep_id NUMBER PRIMARY KEY,
   dept_name VARCHAR2(255) NOT NULL
);
-- Create student table
CREATE TABLE student (
   student_id NUMBER PRIMARY KEY,
   first_name VARCHAR2(255) NOT NULL,
   last_name VARCHAR2(255) NOT NULL,
   date_of_birth DATE NOT NULL,
   gender VARCHAR2(1) NOT NULL,
   faculty VARCHAR2(255) NOT NULL,
   email VARCHAR2(255) UNIQUE NOT NULL,
   hours_completed NUMBER NOT NULL,
   dep_id NUMBER,
   FOREIGN KEY (dep_id) REFERENCES department(dep_id)
);
-- Create courses table
CREATE TABLE courses (
   course_id NUMBER PRIMARY KEY,
   course_name VARCHAR2(255) NOT NULL,
   course_hours NUMBER NOT NULL
);
-- Create enrollment table
CREATE TABLE enrollment (
   student_id NUMBER,
   course_id NUMBER,
   grade VARCHAR2(2),
   PRIMARY KEY (student_id, course_id),
   FOREIGN KEY (student_id) REFERENCES student(student_id),
   FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

```sql
-- Create dept_course table
CREATE TABLE dept_course (
   dep_id NUMBER,
   course_id NUMBER,
   PRIMARY KEY (dep_id, course_id),
   FOREIGN KEY (dep_id) REFERENCES department(dep_id),
   FOREIGN KEY (course_id) REFERENCES courses(course_id)
);
```

- Inserting data into database

```sql
-- Insert data into enrollment table
-- Enroll John Doe in Introduction to Programming
INSERT INTO enrollment (student_id, course_id, grade) VALUES (1, 101, 'A');
-- Enroll Jane Smith in Physics I
INSERT INTO enrollment (student_id, course_id, grade) VALUES (2, 102, 'B');
-- Enroll Bob Johnson in Marketing Principles
INSERT INTO enrollment (student_id, course_id, grade) VALUES (3, 103, 'C');
-- Enroll Alice Johnson in Calculus I
INSERT INTO enrollment (student_id, course_id, grade) VALUES (4, 104, 'B+');
-- Enroll Charlie Brown in Shakespearean Literature
INSERT INTO enrollment (student_id, course_id, grade) VALUES (5, 105, 'A-');
-- Insert Menna's enrollment for the Principle of Applied Chemistry (at Microbiology/Chemistry department)
INSERT INTO enrollment (student_id, course_id, grade)
VALUES (6, 106, 'A');
-- Insert Menna's enrollment for bacteria and chronic infections (at Microbiology/Chemistry department)
INSERT INTO enrollment (student_id, course_id, grade)
VALUES (6, (SELECT course_id FROM courses WHERE course_name = 'Bacteria and Chronic Infections'), 'B');
-- Insert Menna's enrollment for history course (at Literature department)
INSERT INTO enrollment (student_id, course_id, grade)
VALUES (6, (SELECT course_id FROM courses WHERE course_name = 'History'), 'A');
```

```sql
--insert into department table
INSERT INTO department (dep_id, dept_name) VALUES (1, 'Computer Science');
INSERT INTO department (dep_id, dept_name) VALUES (2, 'Physics');
INSERT INTO department (dep_id, dept_name) VALUES (3, 'Business Administration');
INSERT INTO department (dep_id, dept_name) VALUES (4, 'Mathematics');
INSERT INTO department (dep_id, dept_name) VALUES (5, 'English Literature');
INSERT INTO department (dep_id, dept_name) VALUES (6, 'Microbiology/Chemistry');
INSERT INTO department (dep_id, dept_name) VALUES (7, 'History literature');

-- Insert data into student table
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id)
VALUES (1, 'John', 'Doe', TO_DATE('2007-02-05', 'YYYY-MM-DD'), 'M', 'Engineering', 'john.doe@email.com', 100, 1);
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id)
VALUES (2, 'Jane', 'Smith', TO_DATE('2006-02-05', 'YYYY-MM-DD'), 'F', 'Science', 'jane.smith@email.com', 120, 2);
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id)
VALUES (3, 'Bob', 'Johnson', TO_DATE('2005-02-05', 'YYYY-MM-DD'), 'M', 'Business', 'bob.johnson@email.com', 80, 3);
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id)
VALUES (4, 'Alice', 'Johnson', TO_DATE('2004-02-05', 'YYYY-MM-DD'), 'F', 'Science', 'alice.johnson@email.com', 90, 4);
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id)
VALUES (5, 'Charlie', 'Brown', TO_DATE('2003-02-05', 'YYYY-MM-DD'), 'M', 'Arts', 'charlie.brown@email.com', 110, 5);
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id)
VALUES (6, 'Menna', 'Gabr', TO_DATE('2003-05-15', 'YYYY-MM-DD'), 'F', 'Science', 'menna.gabr@email.com', 60, 6);

-- Insert data into courses table
INSERT INTO courses (course_id, course_name, course_hours) VALUES (101, 'Introduction to Programming', 3);
INSERT INTO courses (course_id, course_name, course_hours) VALUES (102, 'Physics I', 4)
INSERT INTO courses (course_id, course_name, course_hours) VALUES (103, 'Marketing Principles', 3);
INSERT INTO courses (course_id, course_name, course_hours) VALUES (104, 'Calculus I', 4);
INSERT INTO courses (course_id, course_name, course_hours) VALUES (105, 'Shakespearean Literature', 3);
INSERT INTO courses (course_id, course_name, course_hours) VALUES (106, 'Principle of Applied Chemistry', 3);
INSERT INTO courses (course_id, course_name, course_hours) VALUES (107, 'Bacteria and Chronic Infections', 4);
INSERT INTO courses (course_id, course_name, course_hours) VALUES (108, 'History', 3);
```

```sql
-- Insert data into dept_course table
-- Associate Computer Science department with Introduction to Programming
INSERT INTO dept_course (dep_id, course_id) VALUES (1, 101);
-- Associate Physics department with Physics I
INSERT INTO dept_course (dep_id, course_id) VALUES (2, 102);
-- Associate Business Administration department with Marketing Principles
INSERT INTO dept_course (dep_id, course_id) VALUES (3, 103);
-- Associate Mathematics department with Calculus I
INSERT INTO dept_course (dep_id, course_id) VALUES (4, 104);
-- Associate English Literature department with Shakespearean Literature
INSERT INTO dept_course (dep_id, course_id) VALUES (5, 105);
-- Associate Microbiology/Chemistry department with Principle of Applied Chemistry
INSERT INTO dept_course (dep_id, course_id) VALUES (6, 106);
-- Associate Microbiology/Chemistry department with Bacteria and Chronic Infections
INSERT INTO dept_course (dep_id, course_id) VALUES (6, 107);
-- Associate History Literature department with History
INSERT INTO dept_course (dep_id, course_id) VALUES (7, 108);
```

-checking the correctness of schema

- checking the content of the tables

```sql
--Check the contents of the department table:
SELECT * FROM department;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert

Output | Grid 1 | Environment

| DEP_ID | DEPT_NAME |
|---|---|
| 1 | Computer Science |
| 2 | Physics |
| 3 | Business Administration |
| 4 | Mathematics |
| 5 | English Literature |
| 6 | Microbiology/Chemistry |
| 7 | History literature |

```sql
--Check the contents of the student table:
SELECT * FROM student;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler | REF CURSOR Results | Script Output | Data Grid

Output | Grid 1 | Environment

| STUDENT_ID | FIRST_NAME | LAST_NAME | DATE_OF_BIRTH | GENDER | FACULTY | EMAIL | HOURS_COMPLETED | DEP_ID |
|---|---|---|---|---|---|---|---|---|
| 1 | John | Doe | 05-FEB-07 | M | Engineering | john.doe@email.com | 100 | 1 |
| 2 | Jane | Smith | 05-FEB-06 | F | Science | jane.smith@email.com | 120 | 2 |
| 3 | Bob | Johnson | 05-FEB-05 | M | Business | bob.johnson@email.com | 80 | 3 |
| 4 | Alice | Johnson | 05-FEB-04 | F | Science | alice.johnson@email.com | 90 | 4 |
| 5 | Charlie | Brown | 05-FEB-03 | M | Arts | charlie.brown@email.com | 110 | 5 |
| 6 | Menna | Gabr | 15-MAY-03 | F | Science | menna.gabr@email.com | 60 | 6 |

```sql
--Check for students who have enrolled in courses from their respective departments:
SELECT s.student_id, s.first_name, s.last_name, s.dep_id, d.dept_name,
       e.course_id, c.course_name, e.grade
FROM student s
JOIN enrollment e ON s.student_id = e.student_id
JOIN dept_course dc ON s.dep_id = dc.dep_id AND e.course_id = dc.course_id
JOIN department d ON s.dep_id = d.dep_id
JOIN courses c ON e.course_id = c.course_id;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler | REF CURSOR Results | Script Output | Data G

Output | Grid 1 | Environment

| STUDENT_ID | FIRST_NAME | LAST_NAME | DEP_ID | DEPT_NAME | COURSE_ID | COURSE_NAME | GRADE |
|---|---|---|---|---|---|---|---|
| 1 | John | Doe | 1 | Computer Science | 101 | Introduction to Programming | A |
| 2 | Jane | Smith | 2 | Physics | 102 | Physics I | B |
| 3 | Bob | Johnson | 3 | Business Administration | 103 | Marketing Principles | C |
| 4 | Alice | Johnson | 4 | Mathematics | 104 | Calculus I | B+ |
| 5 | Charlie | Brown | 5 | English Literature | 105 | Shakespearean Literature | A- |
| 6 | Menna | Gabr | 6 | Microbiology/Chemistry | 106 | Principle of Applied Chemistry | A |
| 6 | Menna | Gabr | 6 | Microbiology/Chemistry | 107 | Bacteria and Chronic Infections | B |

```sql
--Check the total hours completed by each student:
SELECT student_id, SUM(course_hours) AS total_hours_completed
FROM  enrollment
JOIN courses ON enrollment.course_id = courses.course_id
GROUP BY student_id;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profi

Output | Grid 1 | Environment

| STUDENT_ID | TOTAL_HOURS_COMPLETED |
|---|---|
| 1 | 3 |
| 6 | 10 |
| 2 | 4 |
| 4 | 4 |
| 5 | 3 |
| 3 | 3 |

- ADD additional constraint

```sql
ALTER TABLE enrollment
ADD CONSTRAINT check_grade CHECK (regexp_like(grade, '^[^0-9]+$'));
ALTER TABLE student
ADD CONSTRAINT unique_student_name UNIQUE (first_name, last_name);
ALTER TABLE student
ADD CONSTRAINT check_hours_completed CHECK (regexp_like(hours_completed, '^[0-9]+$'));
-- use regexp_like function for pattern matching.
```

- checking the constraint

```sql
-- Attempt to violate the constraint on grade by inserting a number
-- This will fail because '2' contains a number
INSERT INTO enrollment (student_id, course_id, grade) VALUES
(1, 102, '2');
```

TOAD Script Error

```
ORA-02290: check constraint (UNIVERSITY.
CHECK_GRADE) violated
```

Ignore | Ignore All | Cancel | Clipboard | Details >>

```
-- Attempt to violate the unique constraint on student name
-- This will fail because 'John Doe' already exists in the table
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id) VALUES
(3, 'John', 'Doe', '2002-03-03', 'M', 'Physics', 'john.doe2@example.com', 60, 1);
```

TOAD Script Error

ORA-01861: literal does not match format string

Ignore | Ignore All | Cancel | Clipboard | Details >>

```
--Try to insert a student with a non-existing department:
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id)
VALUES (7, 'New', 'Student', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'Science', 'new.student@email.com', 80, 8);
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler | REF CURSOR Results | Script Output | Data Grid

Output | Environment

```
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id)
VALUES (7, 'New', 'Student', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'Science', 'new.student@email.com', 80, 8)
Error at line 2
ORA-02291: integrity constraint (DATA.SYS_C007250) violated - parent key not found
Script Terminated on line 27.
```

```
-- Try to insert a new student with an existing email
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id)
VALUES (7, 'New', 'Student', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'Science', 'john.doe@email.com', 80, 1);
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler | REF CURSOR Results | Script Output | Data Grid

Output | Environment

```
INSERT INTO student (student_id, first_name, last_name, date_of_birth, gender, faculty, email, hours_completed, dep_id)
VALUES (7, 'New', 'Student', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'Science', 'john.doe@email.com', 80, 1)
Error at line 1
ORA-00001: unique constraint (DATA.SYS_C007249) violated
Script Terminated on line 35.
```

```
--Try to insert a department-course association with a non-existing department or course:
INSERT INTO dept_course (dep_id, course_id) VALUES (8, 109);
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler | REF CURSOR Results | Script Output

Output | Environment

```
INSERT INTO dept_course (dep_id, course_id) VALUES (8, 109)
Error at line 2
ORA-02291: integrity constraint (DATA.SYS_C007259) violated - parent key not found
Script Terminated on line 32.
```

# PLSQL Implementation (Day 3):

- Creating procedure to update the student name by student_id at one step without the need to update the first name and the last name at each column.

```oracle_plsql
-- Creating or replacing a procedure to update student's first and last name
CREATE OR REPLACE PROCEDURE update_student_name(
   p_student_id IN NUMBER,          -- Input parameter: Student ID
   p_new_name IN VARCHAR2           -- Input parameter: New full name
) AS
BEGIN
   -- Extracting first and last names from the full name
   UPDATE student
   SET first_name = SUBSTR(p_new_name, 1, INSTR(p_new_name, ' ') - 1),
      last_name = SUBSTR(p_new_name, INSTR(p_new_name, ' ') + 1)
   WHERE student_id = p_student_id;

   -- Displaying a success message if the update is successful
   DBMS_OUTPUT.PUT_LINE('Student name updated successfully.');
EXCEPTION
   -- Handling any exceptions and displaying an error message
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error updating student name: ' || SQLERRM);
END;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler | REF CU

Output | Environment

```
Procedure created.
```

- Calling the procedure to test the correctness of the code

```oracle_plsql
-- Testing the procedure
SET SERVEROUTPUT ON;

-- Selecting the student's first and last name before the update
SELECT first_name, last_name FROM student WHERE student_id = 6;

-- Declaring variables for the test
DECLARE
    v_student_name VARCHAR2(100) := 'Menna zaki';
BEGIN
    -- Calling the update_student_name procedure
    update_student_name(6, v_student_name);

    -- Displaying the new student name after the update
    DBMS_OUTPUT.PUT_LINE('Student new name: ' || v_student_name);
END;

-- Selecting the student's first and last name after the update
SELECT first_name, last_name FROM student WHERE student_id = 6;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler | REF CUR

Output | Grid 1 | Grid 2 | Environment

```
LAST_NAME
--------------------------------------------------------------------------------
Menna
zaki

1 row selected.
```

- Check the table before updating

Output | Grid 1 | Grid 2 | Environment

| FIRST_NAME | LAST_NAME |
|---|---|
| Menna | Amr |

- Check the table after updating

Output | Grid 1 | Grid 2 | Environment

| FIRST_NAME | LAST_NAME |
|---|---|
| Menna | zaki |

- Creating procedure to update the course name by course_id.

```oracle
-- Creating or replacing a procedure to update the course name
CREATE OR REPLACE PROCEDURE UpdateCourseName (
   p_course_id NUMBER,          -- Input parameter: Course ID to be updated
   p_new_course_name VARCHAR2     -- Input parameter: New course name
)
AS
   v_course_count NUMBER;        -- Variable to store the count of courses with the given ID
BEGIN
   -- Check if the course_id exists
   SELECT COUNT(*)
   INTO v_course_count
   FROM courses
   WHERE course_id = p_course_id;

   IF v_course_count = 0 THEN
      -- Course ID not found, raise an exception or handle it as needed
      RAISE_APPLICATION_ERROR(-20001, 'Course ID not found.');
   ELSE
      -- Update the course name
      UPDATE courses
      SET course_name = p_new_course_name
      WHERE course_id = p_course_id;

      -- Display success message
      DBMS_OUTPUT.PUT_LINE('Course name updated successfully.');
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      -- Handle exceptions
      DBMS_OUTPUT.PUT_LINE('Error updating course name: ' || SQLERRM);
      -- Optionally, rollback the transaction if an error occurs
      ROLLBACK;
END;
```

- Calling the procedure to check the correctness of the code.

```oracle
-- Testing the procedure
SET SERVEROUTPUT ON;

-- Selecting the course information before the update
SELECT * FROM courses WHERE course_id = 106;

-- Declaring variables for the test
DECLARE
  v_course_name VARCHAR2(100) := 'Chemistry I';
BEGIN
   -- Calling the UpdateCourseName procedure
   UpdateCourseName(106, v_course_name);

   -- Displaying the new course name after the update
   DBMS_OUTPUT.PUT_LINE('Course new name: ' || v_course_name);
END;

-- Selecting the course information after the update
SELECT * FROM courses WHERE course_id = 106;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler

Output | Grid 1 | Grid 2 | Environment

```
----------
COURSE_NAME
--------------------------------------------------------------
COURSE_HOURS
------------
       106
Chemistry I
           3

1 row selected.
```

- Check the table before updating

Output | Grid 1 | Grid 2 | Environment

| COURSE_ID | COURSE_NAME | COURSE_HOURS |
|---|---|---|
| 106 | Principle of Applied Chemistry | 3 |

- Check the table after updating.

Output | Grid 1 | Grid 2 | Environment

| COURSE_ID | COURSE_NAME | COURSE_HOURS |
|---|---|---|
| 106 | Chemistry I | 3 |

- Creating procedure delete department not just in department table but the whole tables without the need to get each table and checking if exist or not and then delete it.

```
CREATE OR REPLACE PROCEDURE delete_department (
    old_dep_id   IN NUMBER
) AS
    CURSOR CHANGE_DEPT_CURSOR IS
        SELECT TABLE_NAME, COLUMN_NAME, OBJECT_TYPE
        FROM USER_TAB_COLUMNS, USER_OBJECTS
        WHERE USER_TAB_COLUMNS.TABLE_NAME = USER_OBJECTS.OBJECT_NAME
        AND COLUMN_NAME = 'DEP_ID'
        AND OBJECT_TYPE = 'TABLE';
BEGIN
    FOR DEPT_RECORD IN CHANGE_DEPT_CURSOR LOOP
        EXECUTE IMMEDIATE 'DELETE FROM ' || DEPT_RECORD.TABLE_NAME || ' WHERE dep_id = :1'
            USING old_dep_id;
    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        -- Handle exceptions if needed
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
        RAISE;
END;
```

- Calling the procedure to check the correctness of the code.

```
select* from department;
select* from student;
select* from dept_course;
DECLARE
BEGIN
   delete_department(6);
END;
select* from department;
select* from student;
select* from dept_course;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoin

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Grid 6 | Environment

```
    DEP_ID  COURSE_ID
---------- ----------
         1        101
         2        102
         3        103
         4        104
         5        105

5 rows selected.
```

- Check the table before deleting.

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Grid 6 | Environment

| DEP_ID | DEPT_NAME |
|---|---|
| 1 | Computer Science |
| 2 | Physics |
| 3 | Business Administration |
| 4 | Mathematics |
| 5 | English Literature |
| 6 | Microbiology/Chemistry |
| 8 | Historic literaturet |

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Grid 6 | Environment

| STUDENT_ID | FIRST_NAME | LAST_NAME | DATE_OF_BIRTH | GENDER | FACULTY | EMAIL | HOURS_COMPLETED | DEP_ID |
|---|---|---|---|---|---|---|---|---|
| 1 | John | Doe | 05-FEB-07 | M | Engineering | john.doe@email.com | 100 | 1 |
| 2 | Jane | Smith | 05-FEB-06 | F | Science | jane.smith@email.com | 120 | 2 |
| 3 | Bob | Johnson | 05-FEB-05 | M | Business | bob.johnson@email.com | 80 | 3 |
| 4 | Alice | Johnson | 05-FEB-04 | F | Science | alice.johnson@email.com | 90 | 4 |
| 5 | Charlie | Brown | 05-FEB-03 | M | Arts | charlie.brown@email.com | 110 | 5 |
| 6 | Menna | zaki | 15-MAY-03 | F | Science | menna.gabr@email.com | 60 | 6 |

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Grid 6 | Environment

| DEP_ID | COURSE_ID |
|---|---|
| 1 | 101 |
| 2 | 102 |
| 3 | 103 |
| 4 | 104 |
| 5 | 105 |
| 6 | 106 |
| 6 | 107 |

- Checking the tables after the deleting.

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Grid 6 | Environment

| DEP_ID | DEPT_NAME |
|---|---|
| 1 | Computer Science |
| 2 | Physics |
| 3 | Business Administration |
| 4 | Mathematics |
| 5 | English Literature |
| 8 | Historic literaturet |

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Grid 6 | Environment

| STUDENT_ID | FIRST_NAME | LAST_NAME | DATE_OF_BIRTH | GENDER | FACULTY | EMAIL | HOURS_COMPLETED | DEP_ID |
|---|---|---|---|---|---|---|---|---|
| 1 | John | Doe | 05-FEB-07 | M | Engineering | john.doe@email.com | 100 | 1 |
| 2 | Jane | Smith | 05-FEB-06 | F | Science | jane.smith@email.com | 120 | 2 |
| 3 | Bob | Johnson | 05-FEB-05 | M | Business | bob.johnson@email.com | 80 | 3 |
| 4 | Alice | Johnson | 05-FEB-04 | F | Science | alice.johnson@email.com | 90 | 4 |
| 5 | Charlie | Brown | 05-FEB-03 | M | Arts | charlie.brown@email.com | 110 | 5 |

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Grid 6 | Environment

| DEP_ID | COURSE_ID |
|---|---|
| 1 | 101 |
| 2 | 102 |
| 3 | 103 |
| 4 | 104 |
| 5 | 105 |

- Creating function to calculate the student GPA by the student_id.

```
CREATE OR REPLACE FUNCTION calculate_gpa(p_student_id IN NUMBER) RETURN NUMBER IS
   v_total_grade_points NUMBER := 0;  -- Total grade points accumulated
   v_total_credit_hours NUMBER := 0;  -- Total credit hours for all courses
   v_course_grade CHAR(1);            -- Grade for each course
   v_course_hours NUMBER;             -- Credit hours for each course

   -- Cursor to retrieve grades and credit hours for enrolled courses
   CURSOR c_enrollments IS
      SELECT e.grade, c.course_hours
      FROM enrollment e
      JOIN courses c ON e.course_id = c.course_id
      WHERE e.student_id = p_student_id;

BEGIN
   -- Loop through each enrolled course
   FOR enrollment_rec IN c_enrollments LOOP
      v_course_grade := enrollment_rec.grade;   -- Retrieve grade for the course
      v_course_hours := enrollment_rec.course_hours;  -- Retrieve credit hours for the course

      -- Assigning grade points based on the grade
      CASE v_course_grade
         WHEN 'A' THEN v_total_grade_points := v_total_grade_points + (4 * v_course_hours);
         WHEN 'B' THEN v_total_grade_points := v_total_grade_points + (3 * v_course_hours);
         WHEN 'C' THEN v_total_grade_points := v_total_grade_points + (2 * v_course_hours);
         WHEN 'D' THEN v_total_grade_points := v_total_grade_points + (1 * v_course_hours);
         WHEN 'F' THEN v_total_grade_points := v_total_grade_points + (0 * v_course_hours);
         -- Add more cases if needed for other grades
      END CASE;

      v_total_credit_hours := v_total_credit_hours + v_course_hours;  -- Accumulate total credit hours
   END LOOP;
```

```
  -- Avoid division by zero
  IF v_total_credit_hours = 0 THEN
     RETURN NULL;  -- Return NULL if there are no credit hours to avoid division by zero
  END IF;

  -- Calculating GPA by dividing total grade points by total credit hours
  RETURN v_total_grade_points / v_total_credit_hours;
END;
```

- Calling the function to test the code.

```
-----test for function
DECLARE
  v_gpa NUMBER;
BEGIN
 v_gpa := calculate_gpa(6);  -- add the student_id that you want to calc his gpa
   DBMS_OUTPUT.PUT_LINE('GPA: ' || TO_CHAR(v_gpa, '0.00'));
END;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler | REF CURS

Output | Environment

```
GPA:   3.60
PL/SQL procedure successfully completed.
```

- Creating table student_course_history so that any update in the course the old data would automatically uploaded in the student_course_history by using trigger.

```
CREATE TABLE student_course_history (
        student_id NUMBER,
          old_course_name VARCHAR2(255 CHAR),
          old_grade VARCHAR2(10 CHAR),
          updated_at DATE
        );
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Brea

Output | Environment

```
Table created.
```

```
CREATE OR REPLACE TRIGGER student_update_trigger
after UPDATE ON courses
FOR EACH ROW
DECLARE
   CURSOR enrollment_cursor IS
      SELECT student_id, grade
      FROM enrollment
      WHERE course_id = :old.course_id;

   v_enrollment_rec enrollment_cursor%ROWTYPE;
BEGIN
   FOR v_enrollment_rec IN enrollment_cursor
   LOOP
      INSERT INTO student_course_history (STUDENT_ID, OLD_COURSE_NAME, OLD_GRADE, UPDATED_AT)
      VALUES (v_enrollment_rec.student_id, :old.course_name, v_enrollment_rec.grade, SYSDATE);
   END LOOP;
END;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler | REF CURSOR Results | Script Output

Output | Environment

```
Trigger created.
```

- Testing the trigger by updating the course name and select all data about the table to check if the old data uploaded or not.

```
--for testing
--update courses set course_name = 'principle of Applied chemistry' where course_id = 106;
select* from student_course_history;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert | Breakpoints | Profiler | REF CURSOR Results

Output | Grid 1 | Environment

| STUDENT_ID | OLD_COURSE_NAME | OLD_GRADE | UPDATED_AT |
|---|---|---|---|
| 6 | Chemistry I | A | 05-FEB-24 |

- Creating trigger so that in case of removing student from university we can just delete the student_id from students and all data about that student will be deleted from all other tables

```oracle_plsql
CREATE OR REPLACE PROCEDURE delete_student_id(
   v_old_student_id IN student.student_id%TYPE
)
AS
BEGIN
   FOR rec IN (
      SELECT table_name, column_name, object_type
      FROM user_tab_columns c
      JOIN user_objects o ON c.table_name = o.object_name
      WHERE c.column_name = 'STUDENT_ID' AND object_type = 'TABLE'
   )
   LOOP
      -- Construct dynamic SQL to delete records with the specified student_id
      EXECUTE IMMEDIATE 'DELETE FROM ' || rec.table_name || ' WHERE student_id = ' || v_old_student_id;

      -- Print information
      DBMS_OUTPUT.PUT_LINE('Student ID ' || v_old_student_id || ' records deleted from ' || rec.table_name);
   END LOOP;
EXCEPTION
   WHEN OTHERS THEN
      -- Handle exceptions
      DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/

CREATE OR REPLACE TRIGGER trg_delete_student
AFTER DELETE ON student
FOR EACH ROW
DECLARE
BEGIN
   -- Call the delete_student_id procedure to delete related records
   delete_student_id(1);
END;
/
```

- Testing the trigger by calling the function and deleting the desired student_id.

```
-- Check existing data before deletion
SELECT* FROM student;
SELECT* from enrollment;
select* from student_course_history;
BEGIN
    -- Call the delete_student_id procedure
    delete_student_id(6);
END;
-- Check existing data after deletion
SELECT* FROM student;
SELECT* from enrollment;
select* from student_course_history;
```

Script Output

Call Stack | DBMS Output (disabled) | CodeXpert

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Environm

```
STUDENT_ID  COURSE_ID GRADE
---------- ---------- -----
         1        101 A
         2        102 B
         3        103 C
         4        104 B
         5        105 C

5 rows selected.
no rows selected.
```

- Check the tables before deleting.

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Environment

| STUDENT_ID | FIRST_NAME | LAST_NAME | DATE_OF_BIRTH | GENDER | FACULTY | EMAIL | HOURS_COMPLETED | DEP_ID |
|---|---|---|---|---|---|---|---|---|
| 1 | John | Doe | 05-FEB-07 | M | Engineering | john.doe@email.com | 100 | 1 |
| 2 | Jane | Smith | 05-FEB-06 | F | Science | jane.smith@email.com | 120 | 2 |
| 3 | Bob | Johnson | 05-FEB-05 | M | Business | bob.johnson@email.com | 80 | 3 |
| 4 | Alice | Johnson | 05-FEB-04 | F | Science | alice.johnson@email.com | 90 | 4 |
| 5 | Charlie | Brown | 05-FEB-03 | M | Arts | charlie.brown@email.com | 110 | 5 |
| 6 | Menna | zaki | 15-MAY-03 | F | Science | menna.gabr@email.com | 60 | 6 |

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Environment

| STUDENT_ID | COURSE_ID | GRADE |
|---|---|---|
| 1 | 101 | A |
| 2 | 102 | B |
| 3 | 103 | C |
| 4 | 104 | B |
| 5 | 105 | C |
| 6 | 106 | C |
| 6 | 107 | B |
| 6 | 108 | A |

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Environment

| STUDE... | OLD_COURSE_NAME | OLD_GRADE | UPDATED_AT |
|---|---|---|---|
| 6 | Chemistry I | A | 05-FEB-24 |

- Check the tables after deleting.

Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Environment

| STUDENT_ID | FIRST_NAME | LAST_NAME | DATE_OF_BIRTH | GENDER | FACULTY | EMAIL | HOURS_COMPLETED | DEP_ID |
|---|---|---|---|---|---|---|---|---|
| 1 | John | Doe | 05-FEB-07 | M | Engineering | john.doe@email.com | 100 | 1 |
| 2 | Jane | Smith | 05-FEB-06 | F | Science | jane.smith@email.com | 120 | 2 |
| 3 | Bob | Johnson | 05-FEB-05 | M | Business | bob.johnson@email.com | 80 | 3 |
| 4 | Alice | Johnson | 05-FEB-04 | F | Science | alice.johnson@email.com | 90 | 4 |
| 5 | Charlie | Brown | 05-FEB-03 | M | Arts | charlie.brown@email.com | 110 | 5 |

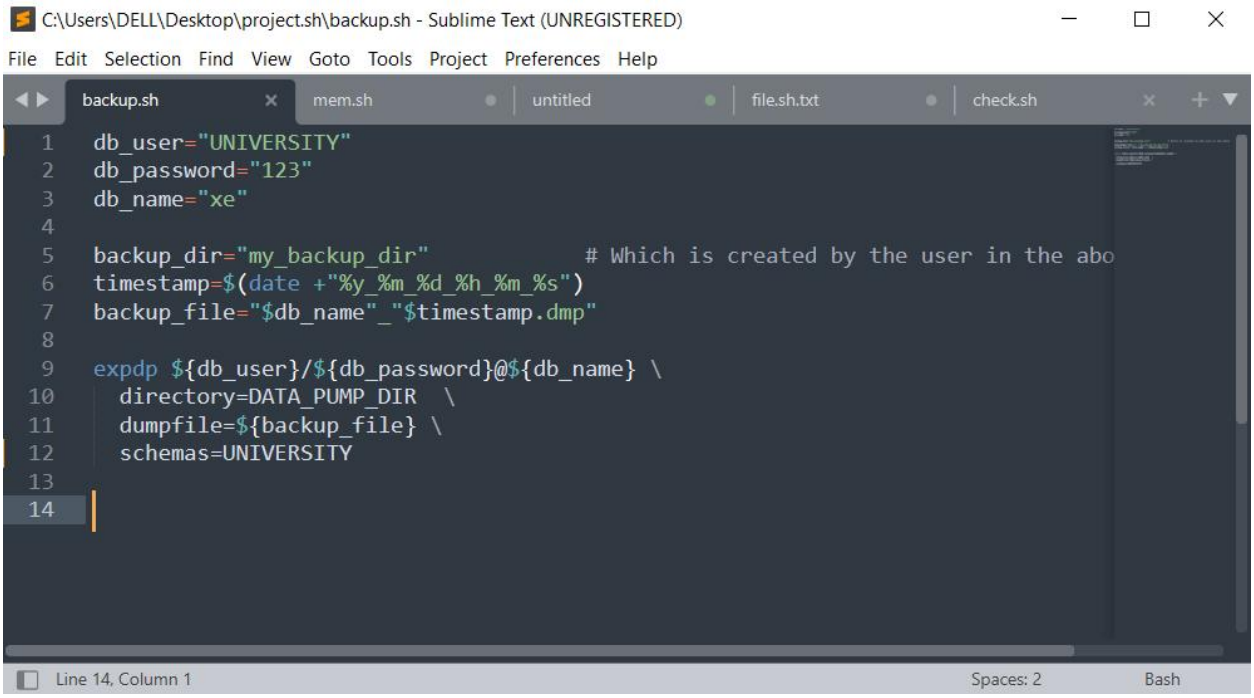
Output | Grid 1 | Grid 2 | Grid 3 | Grid 4 | Grid 5 | Environment

| STUDENT_ID | COURSE_ID | GRADE |
|---|---|---|
| 1 | 101 | A |
| 2 | 102 | B |
| 3 | 103 | C |
| 4 | 104 | B |
| 5 | 105 | C |

# • Automation Scripts (Day 4):

- Bash script for database backup.

- Creating file.sh and open it at the sublime to type the code

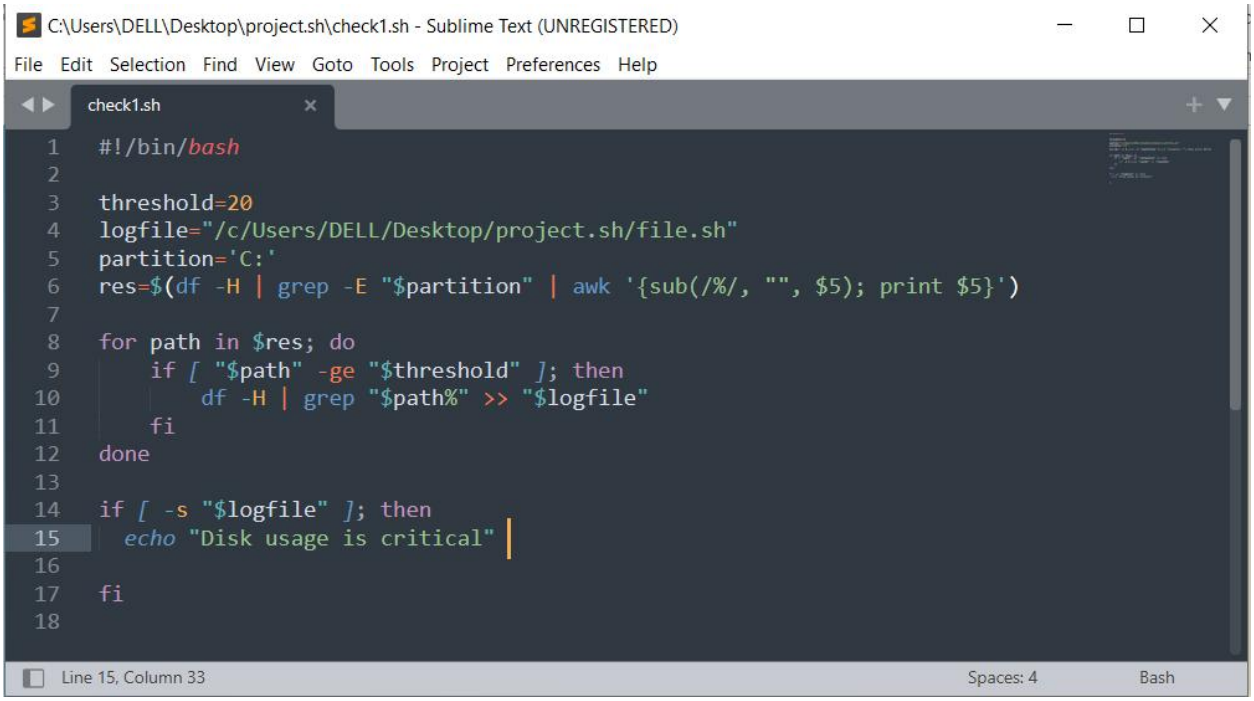


- Running the file at the git bash to create the backup.

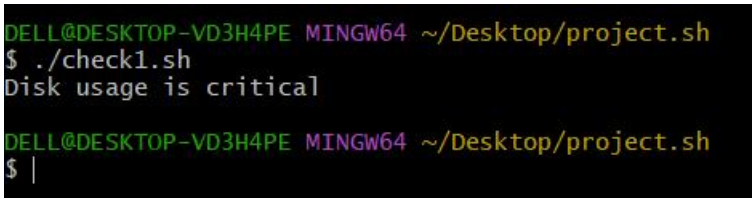
MINGW64:/c/Users/DELL/Desktop/project.sh

```
DELL@DESKTOP-VD3H4PE MINGW64 ~/Desktop/project.sh
$ ./backup

Export: Release 11.2.0.2.0 - Production on Tue Feb 6 03:31:56 2024

Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.

UDE-01017: operation generated ORACLE error 1017
ORA-01017: invalid username/password; logon denied

Username: university
Password: 123

Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 - Productio
n
Starting "UNIVERSITY"."SYS_EXPORT_SCHEMA_01":  university/********
Estimate in progress using BLOCKS method...
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 320 KB
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/FUNCTION/FUNCTION
Processing object type SCHEMA_EXPORT/PROCEDURE/PROCEDURE
Processing object type SCHEMA_EXPORT/FUNCTION/ALTER_FUNCTION
Processing object type SCHEMA_EXPORT/PROCEDURE/ALTER_PROCEDURE
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/TRIGGER
. . exported "UNIVERSITY"."COURSES"                      6.070 KB       8 rows
. . exported "UNIVERSITY"."DEPARTMENT"                   5.585 KB       7 rows
. . exported "UNIVERSITY"."DEPT_COURSE"                  5.492 KB       8 rows
. . exported "UNIVERSITY"."ENROLLMENT"                   5.914 KB       8 rows
. . exported "UNIVERSITY"."STUDENT"                      8.601 KB       6 rows
. . exported "UNIVERSITY"."STUDENT_COURSE_HISTORY"           0 KB       0 rows
Master table "UNIVERSITY"."SYS_EXPORT_SCHEMA_01" successfully loaded/unloaded
******************************************************************************
Dump file set for UNIVERSITY.SYS_EXPORT_SCHEMA_01 is:
  D:\APP\ORACLE\ADMIN\XE\DPDUMP\EXPDAT.DMP
Job "UNIVERSITY"."SYS_EXPORT_SCHEMA_01" successfully completed at 03:35:03


DELL@DESKTOP-VD3H4PE MINGW64 ~/Desktop/project.sh
$ |
```

- Bash script for monitoring disk space and sending alerts.
  - Open the file at sublime to type the code to get the disk usage and if the disk usage larger than threshold it should send alert.




  - Running the file at git bash to check the code.



- Schedule a script to check for anomalies and send notifications.
  - Bash script for checking sudden increase in memory consumptions:
  - create script on Linux and execute the file and then run on terminal to check if the memory usage exceeded the threshold then it will send mail.

Open ▾ memory ~/ Save

```
#!/bin/bash
hostname=$(hostname)
mailto="mzaki"

# Function to get current memory usage
get_memory_usage() {
    ps -p $$ -o rss | awk 'NR==2{print $1}'
}

# Set an initial memory threshold
threshold=44

# Main loop
while true; do
    current_memory=$(get_memory_usage)

    echo "Current Memory Usage: $current_memory KB"

    # Check if memory usage exceeds the threshold
    if [ "$current_memory" -gt "$threshold" ]; then
        mail -s  "$hostname Sudden increase in memory consumption detected!" $mailto < /tmp/temp
        break
    fi

    sleep 1
done
```

sh ▾ Tab Width: 8 ▾ Ln 21, Col 29 ▾ INS

- Running the script on terminal to get the memory usage and then check the mail.

mzaki@localhost:~

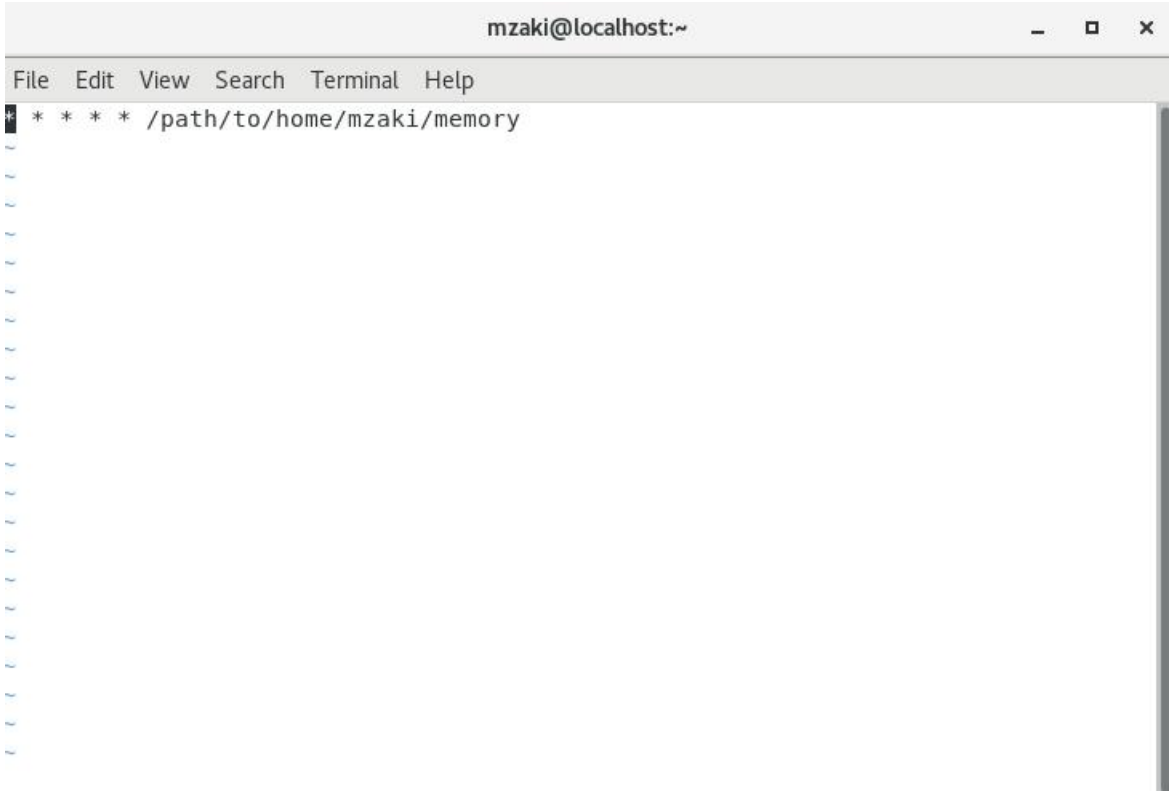
File Edit View Search Terminal Help

```
[mzaki@localhost ~]$ ./memory
Current Memory Usage: 1380 KB
```

- Check the mail on terminal and exit by entering “q”.

```
mzaki@localhost:~
File  Edit  View  Search  Terminal  Help
[mzaki@localhost ~]$ mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/mzaki": 2 messages
>  1 menna zaki          Tue Feb  6 03:00  20/720   "localhost.localdomain Sudden increase in memory consumption detected!"
   2 menna zaki          Tue Feb  6 03:06  20/720   "localhost.localdomain Sudden increase in memory consumption detected!"
& p
Message  1:
From mzaki@localhost.localdomain  Tue Feb  6 03:00:35 2024
Return-Path: <mzaki@localhost.localdomain>
X-Original-To: mzaki
Delivered-To: mzaki@localhost.localdomain
Date: Tue, 06 Feb 2024 03:00:35 +0200
To: mzaki@localhost.localdomain
Subject: localhost.localdomain Sudden increase in memory consumption
 detected!
User-Agent: Heirloom mailx 12.5 7/5/10
Content-Type: text/plain; charset=us-ascii
From: mzaki@localhost.localdomain (menna zaki)
Status: RO

"/home/mzaki/dead.letter" 1/1

& d
& q
Held 1 message in /var/spool/mail/mzaki
[mzaki@localhost ~]$ ^C
```

- scheduling the script to run every minute by using command Crontab –e and then check the mails after sending notification.



mzaki@localhost:~

File Edit View Search Terminal Help

```
[mzaki@localhost ~]$ crontab -e
crontab: installing new crontab
You have new mail in /var/spool/mail/mzaki
[mzaki@localhost ~]$ mail
Heirloom Mail version 12.5 7/5/10.  Type ? for help.
"/var/spool/mail/mzaki": 8 messages 5 new
    1 menna zaki          Tue Feb  6 03:06  20/720   "localhost.localdomain Sudden increase in memory consumption detected!"
    2 (Cron Daemon)       Tue Feb  6 04:07  26/930   "Cron <mzaki@localhost> /path/to/memory"
    3 (Cron Daemon)       Tue Feb  6 04:08  26/952   "Cron <mzaki@localhost> /path/to/home/mzaki/memory"
>N  4 (Cron Daemon)       Tue Feb  6 04:09  25/941   "Cron <mzaki@localhost> /path/to/home/mzaki/memory"
 N  5 (Cron Daemon)       Tue Feb  6 04:10  25/941   "Cron <mzaki@localhost> /path/to/home/mzaki/memory"
 N  6 (Cron Daemon)       Tue Feb  6 04:11  25/941   "Cron <mzaki@localhost> /path/to/home/mzaki/memory"
 N  7 (Cron Daemon)       Tue Feb  6 04:12  25/941   "Cron <mzaki@localhost> /path/to/home/mzaki/memory"
 N  8 (Cron Daemon)       Tue Feb  6 04:13  25/941   "Cron <mzaki@localhost> /path/to/home/mzaki/memory"
& q
Held 8 messages in /var/spool/mail/mzaki
```