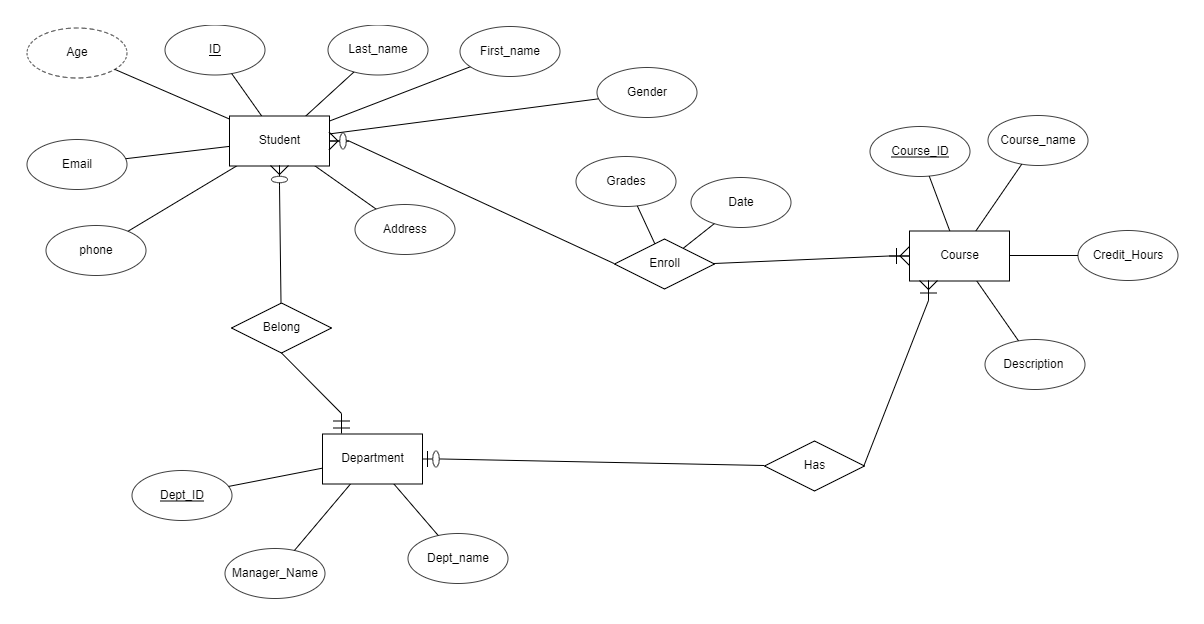
**Database Schema and Mapping:**

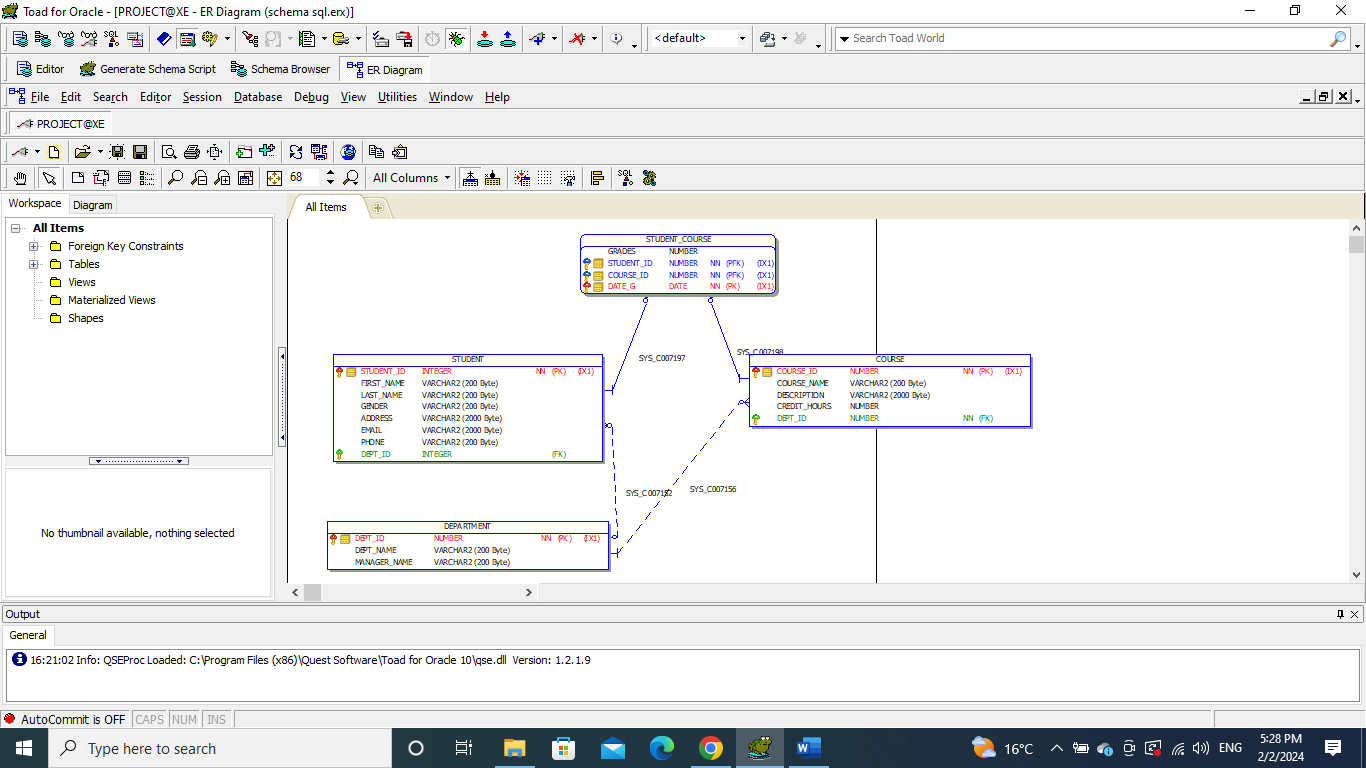
* Designed a relational database schema for a university system with tables for students, courses, departments, and grades.
* Established relationships between entities: Student, Department, Course, and Student\_Course.
* Utilized foreign keys to maintain referential integrity, linking Student and Course tables to the Department table.
* Enforced data consistency through proper normalization of the schema.



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Student | Student\_id | first\_name | Last\_name | address | email | Gender | phone | Dept\_id(fK) |
| Department | Dept\_id | Dept\_name | Manager\_name |
| course | Course\_id | Course\_name | description | Credit\_hours | Dept\_id(fK) |
| Student\_course | Student\_id | Course\_id | date | grades |

**SQL Script:**

1. **DEPARTMENT Table:**
   * Created a DEPARTMENT table with attributes DEPT\_ID, DEPT\_NAME, and MANAGER\_NAME.
   * Defined DEPT\_ID as the primary key to ensure uniqueness.
2. **COURSE Table:**
   * Established a COURSE table containing attributes COURSE\_ID, COURSE\_NAME, DESCRIPTION, CREDIT\_HOURS, and DEPT\_ID.
   * Set COURSE\_ID as the primary key and DEPT\_ID as a foreign key referencing DEPARTMENT.
3. **STUDENT Table:**
   * Developed a STUDENT table with attributes STUDENT\_ID, FIRST\_NAME, LAST\_NAME, GENDER, ADDRESS, EMAIL, PHONE, and DEPT\_ID.
   * Specified STUDENT\_ID as the primary key and DEPT\_ID as a foreign key referencing DEPARTMENT.
4. **STUDENT\_COURSE Table:**
   * Designed a STUDENT\_COURSE table with attributes GRADES, STUDENT\_ID, COURSE\_ID, and DATE\_G.
   * Established a composite primary key using STUDENT\_ID, COURSE\_ID, and DATE\_G.
   * Defined foreign keys for STUDENT\_ID and COURSE\_ID referencing STUDENT and COURSE tables.



CREATE TABLE DEPARTMENT

(

DEPT\_ID NUMBER NOT NULL,

DEPT\_NAME VARCHAR2(200 BYTE),

MANAGER\_NAME VARCHAR2(200 BYTE),

PRIMARY KEY (DEPT\_ID)

);

CREATE TABLE COURSE

(

COURSE\_ID NUMBER NOT NULL,

COURSE\_NAME VARCHAR2(200 BYTE),

DESCRIPTION VARCHAR2(2000 BYTE),

CREDIT\_HOURS NUMBER,

DEPT\_ID NUMBER NOT NULL,

PRIMARY KEY (COURSE\_ID),

FOREIGN KEY (DEPT\_ID) REFERENCES DEPARTMENT (DEPT\_ID)

);

CREATE TABLE STUDENT

(

STUDENT\_ID INTEGER NOT NULL,

FIRST\_NAME VARCHAR2(200 BYTE),

LAST\_NAME VARCHAR2(200 BYTE),

GENDER VARCHAR2(200 BYTE),

ADDRESS VARCHAR2(2000 BYTE),

EMAIL VARCHAR2(2000 BYTE),

PHONE VARCHAR2(200 BYTE),

DEPT\_ID INTEGER,

PRIMARY KEY (STUDENT\_ID),

FOREIGN KEY (DEPT\_ID) REFERENCES DEPARTMENT (DEPT\_ID)

);

CREATE TABLE STUDENT\_COURSE

(

GRADES NUMBER,

STUDENT\_ID NUMBER NOT NULL,

COURSE\_ID NUMBER NOT NULL,

DATE\_G DATE,

PRIMARY KEY (STUDENT\_ID, COURSE\_ID, DATE\_G),

FOREIGN KEY (STUDENT\_ID) REFERENCES STUDENT (STUDENT\_ID),

FOREIGN KEY (COURSE\_ID) REFERENCES COURSE (COURSE\_ID)

);

**PLSQL Procedure and Function:**

1. **PLSQL Procedure (PROJECT.update\_info):**
   * Developed a procedure for updating student information, taking parameters for student details (ID, name, gender, address, email, phone, department).
   * Utilized an UPDATE statement to modify the corresponding student record in the STUDENT table.
2. **PLSQL Function (PROJECT.calc\_gpa):**
   * Implemented a function to calculate GPA based on the provided student ID.
   * Utilized a SELECT statement with JOINs and aggregation to calculate grade points and total credit hours.
   * Employed conditional logic to determine the GPA range and return the corresponding letter grade ('A', 'B', 'C', or 'F').

**Procedure :**

CREATE OR REPLACE procedure PROJECT.update\_info ( v\_student\_id int , v\_first\_name varchar2,v\_last\_name varchar2,v\_gender varchar2 , v\_address varchar2 , v\_email varchar2 , v\_phone varchar2 , v\_dept\_id number )

is

begin

UPDATE Student

SET

first\_name =v\_first\_name,

last\_name=v\_last\_name,

gender=v\_gender,

address=v\_address,

email=v\_email,

phone=v\_phone,

dept\_id=v\_dept\_id

WHERE student\_id = v\_student\_id;

end;

/

**Function :**

CREATE OR REPLACE function PROJECT.calc\_gpa (v\_student\_id number )

return varchar2

is

v\_credit\_hours number;

v\_gpa number;

v\_grade\_points number;

begin

select sum (c.credit\_hours \* sc.grades) as grade\_points ,

sum(credit\_hours) as total\_credit\_hours

into v\_grade\_points , v\_credit\_hours

from course c

join student\_course sc

on c.course\_id = sc.course\_id

where sc.student\_id = v\_student\_id

group by sc.student\_id ;

v\_gpa := v\_grade\_points / v\_credit\_hours ;

IF v\_gpa between 3.5 and 4 THEN

RETURN 'A';

ELSIF v\_gpa between 2.5 and 3.5 THEN

RETURN 'B';

ELSIF v\_gpa between 1.5 and 2.5 THEN

RETURN 'C';

ELSE

RETURN 'F';

END IF;

end;

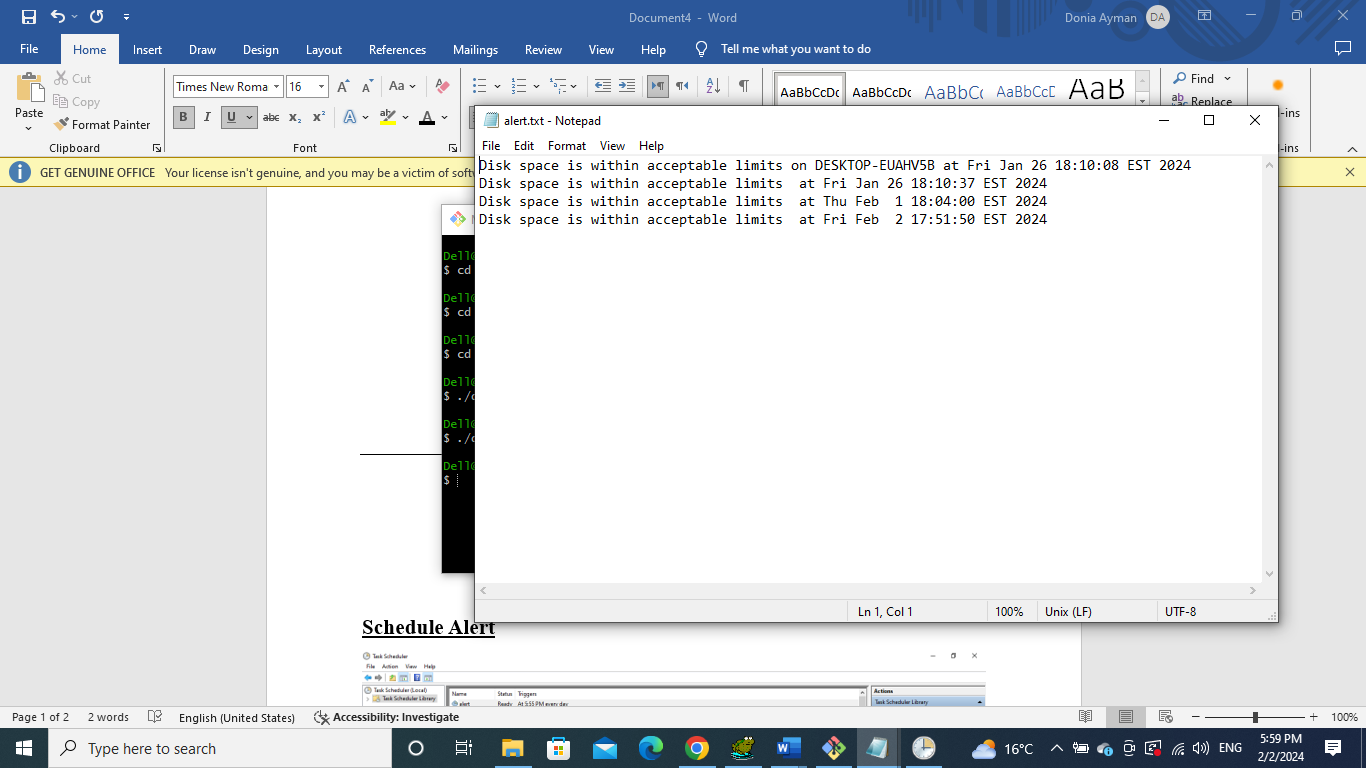
/

**So as conclusion :**

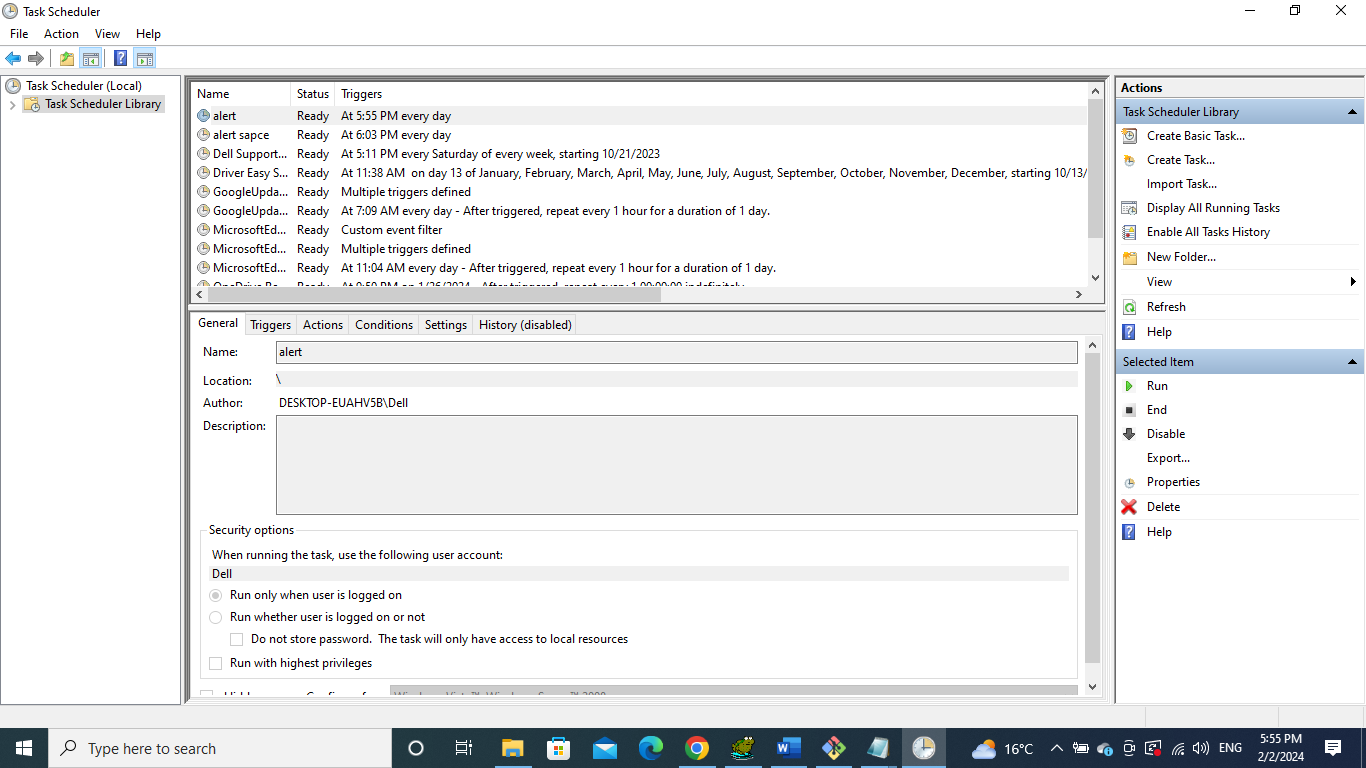
* Established a robust relational database schema with tables designed to handle student, course, department, and grading information.
* Implemented SQL scripts to create tables with appropriate primary and foreign key relationships.
* Developed a PLSQL procedure to update student information and a function to calculate GPA.
* Ensured referential integrity by enforcing foreign key relationships between tables.
* Documented each component for clarity and ease of understanding in the context of the university data management system.

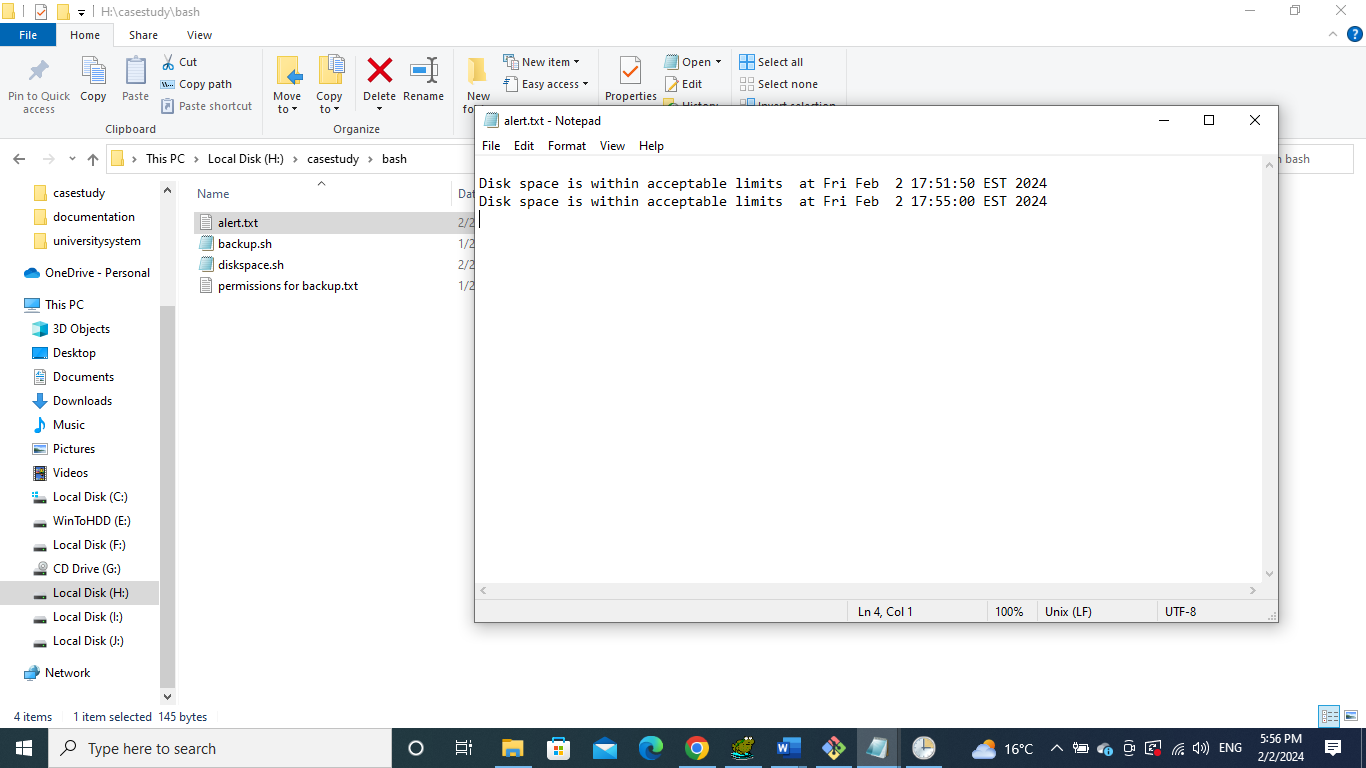
**Bash Scripts for Automation:**

1. **Disk Space Monitoring Script:**
   * Created a Bash script for monitoring disk space using **df** and **awk**.
   * Implemented conditional checks to send alerts when disk space exceeds a predefined threshold.

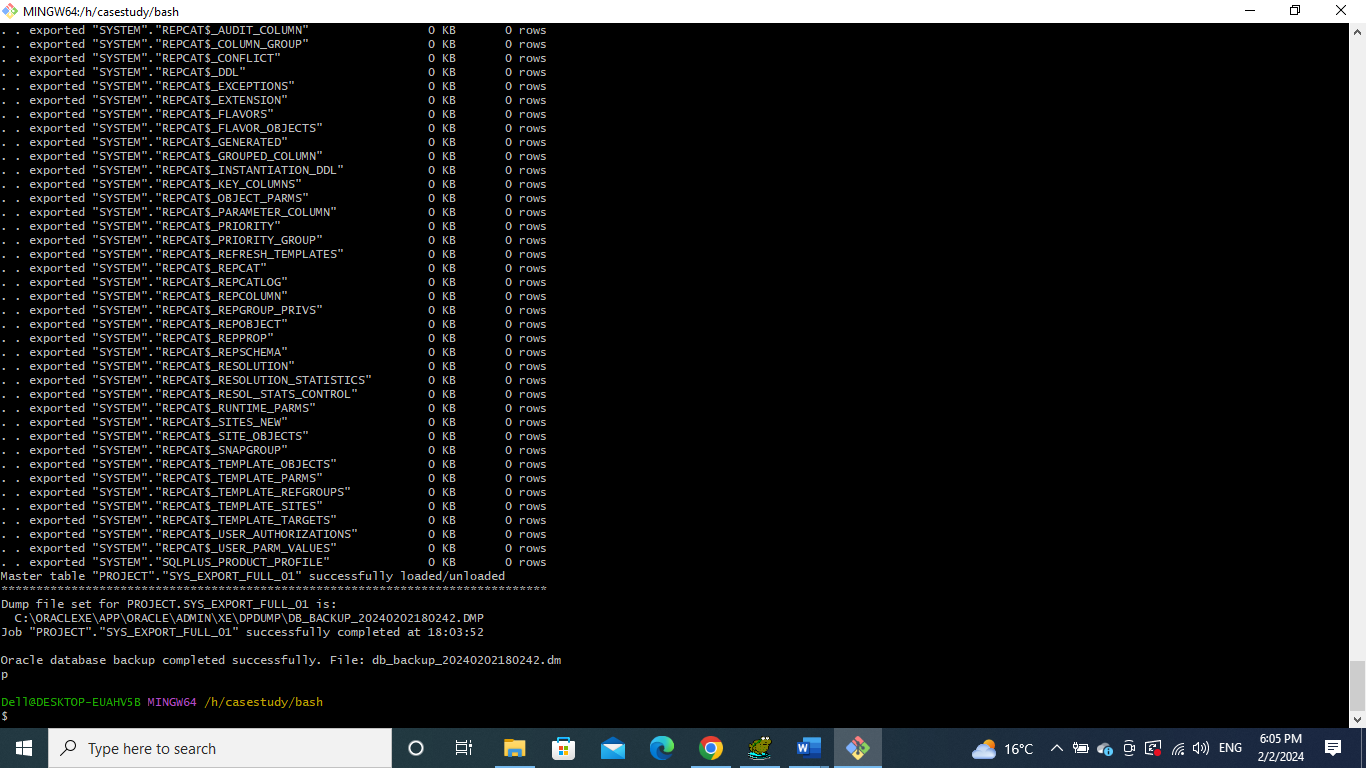


1. **Scheduled Anomaly Checking Script:**
   * Utilized Windows Task Scheduler to schedule the anomaly checking script.
   * Configured a scheduled task for regular execution at specified intervals.





1. **Oracle Database Backup Script:**
   * Designed a Bash script for Oracle database backup with **exe**.
   * Ensured backup file uniqueness with a timestamp.
   * Give permission from sys to can backup   
     GRANT READ, WRITE ON DIRECTORY DATA\_PUMP\_DIR TO project;



**Java Application :**

1. **Student Management:**
   * Implemented methods for adding, updating, and deleting students.
   * Maintained a TableView for displaying student information.
   * Validated input, checked for duplicates.
2. **Course Management:**
   * Incorporated functionality for adding, updating, and deleting courses.
   * Ensured data integrity through SQL queries and user interactions.
3. **Department Management:**
   * Developed methods for department-related operations.
   * Executed SQL queries for adding, updating, and deleting departments.
4. **Grades Management:**
   * Established procedure for managing student grades.
   * Utilized SQL queries to handle grade-related functionalities.
5. **Reports:**
   * Implemented a detailed student report generation.
     + Executed complex SQL queries with JOIN operations.
     + Fetched and displayed data in a TableView.
     + Obtained GPA information using a PLSQL function.
   * Designed a department-wise report generation.
     + Utilized SQL queries with LEFT JOIN operations.
     + Populated a TableView with department-specific data.

