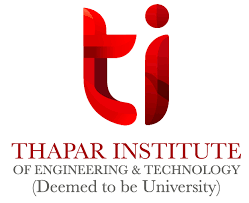
**Project Report On**

**Student Management System (UCS310)**



**Submitted By:**

|  |  |
| --- | --- |
| **DEV SATIJA** | **102116036** |
| **DEEPANSHU THAKUR** | **102296003** |
| **SAMRIDHI WADHWA** | **102116060** |
| **SUKHMANDEEP BRAR** | **102116056** |

# 

# Submitted to:

**Dr Manisha Kaushal**

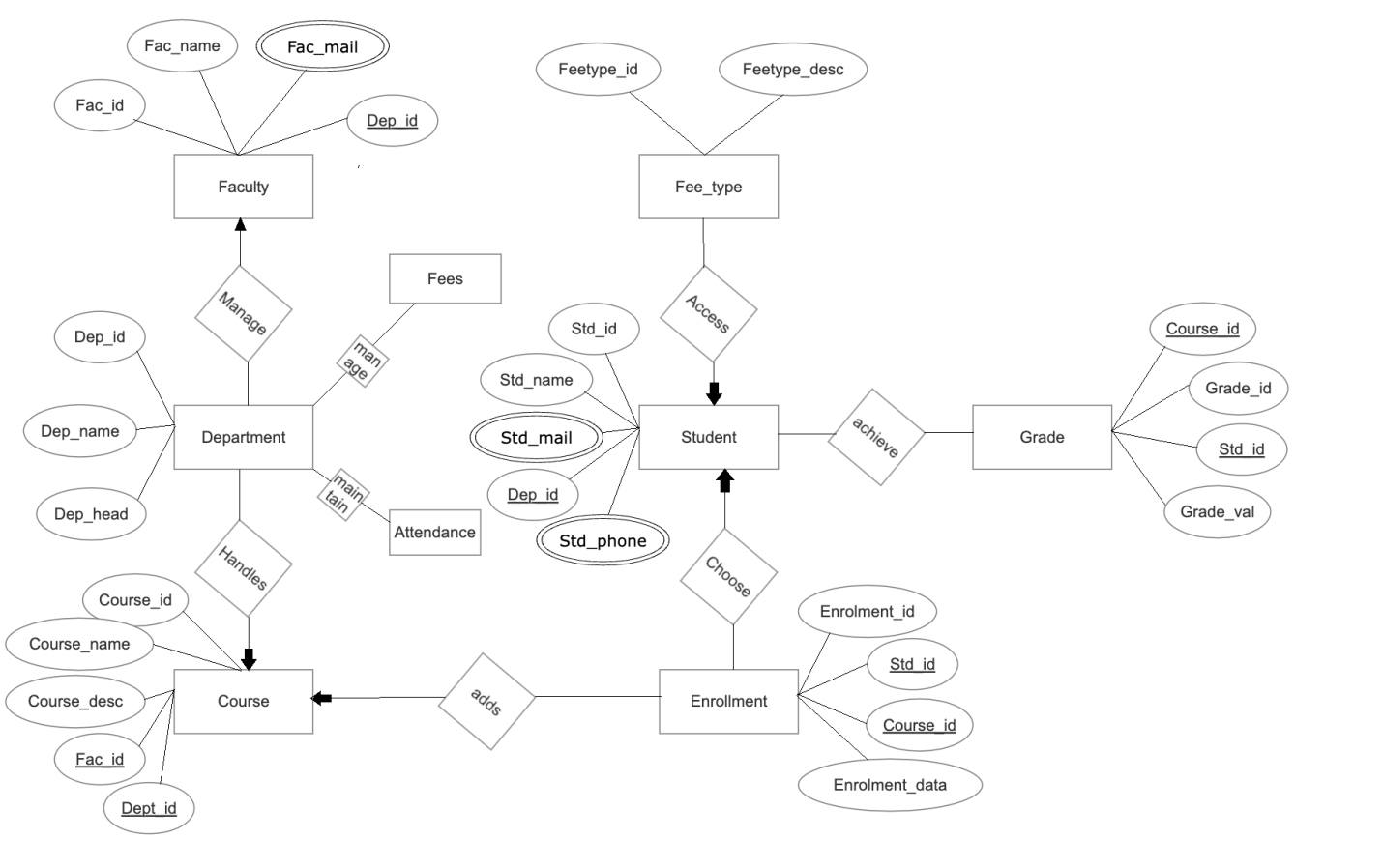
# INDEX

|  |  |
| --- | --- |
| **CONTENTS** | **PAGENUMBER** |
| Problem Statement | 3 |
| ER Diagram | 3 |
| ER Diagram to Table | 4 |
| Normalized Tables | 5 |
| Creating tables | 5 |
| Insertion of data in the tables | 7 |
| SQL Output Screenshots | 13 |
| PL/SQL Codes and screenshots | 16 |

# Problem Statement

To develop a system that can efficiently and accurately identify students by their facial features and retrieve their academic and personal information from a database. The system should be able to capture and store the images of students, as well as their personal and academic data, in a secure and organized manner.

# Entity-Relationship Diagram

****

**Normalized Table**

department: department\_id -> department\_name, department\_head

student: student\_id -> student\_name, student\_email, student\_phone, department\_id

faculty: faculty\_id -> faculty\_name, faculty\_email, department\_id

course: course\_id -> course\_name, course\_description, faculty\_id, department\_id

enrollment: enrollment\_id -> student\_id, course\_id, enrollment\_date

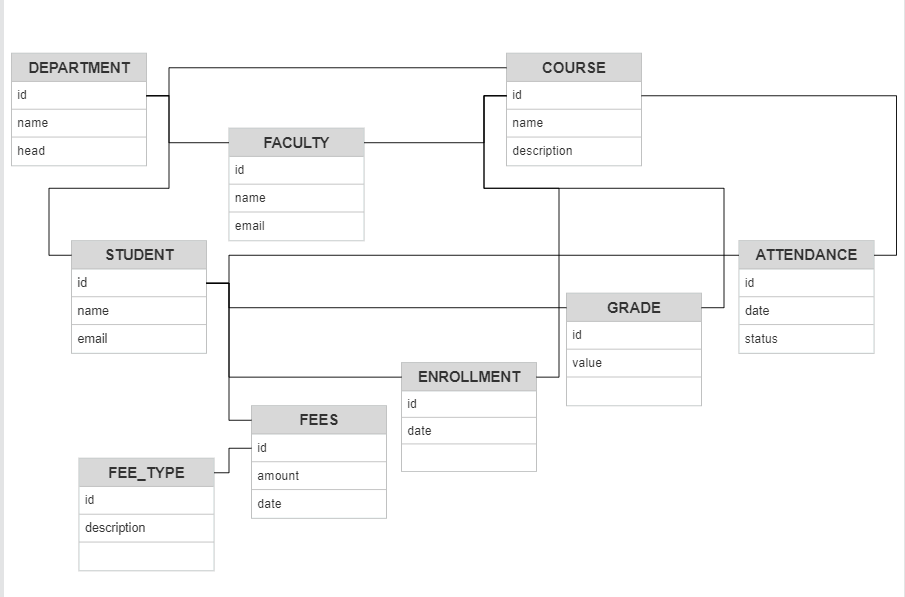
attendance: attendance\_id -> student\_id, course\_id, attendance\_date, attendance\_status

grade: grade\_id -> student\_id, course\_id, grade\_value

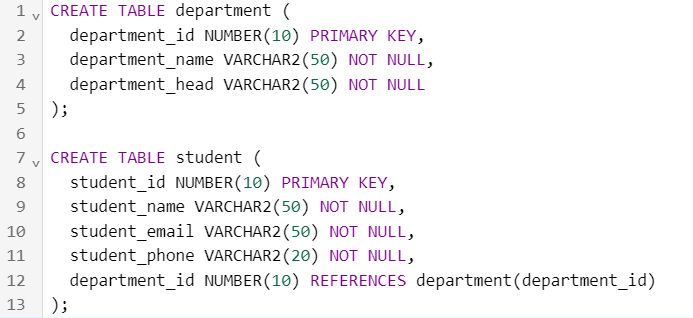
fee\_type: fee\_type\_id -> fee\_type\_description

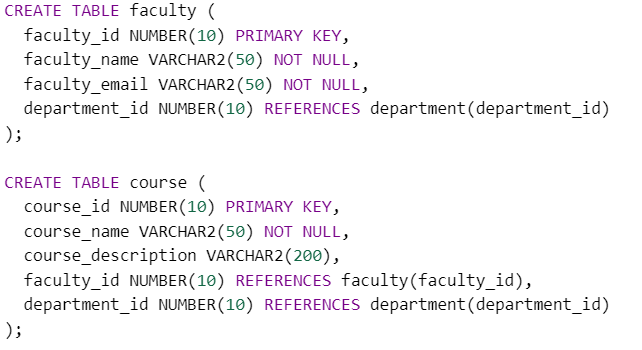
fees: fee\_id -> student\_id, fee\_type\_id, fee\_amount, fee\_date

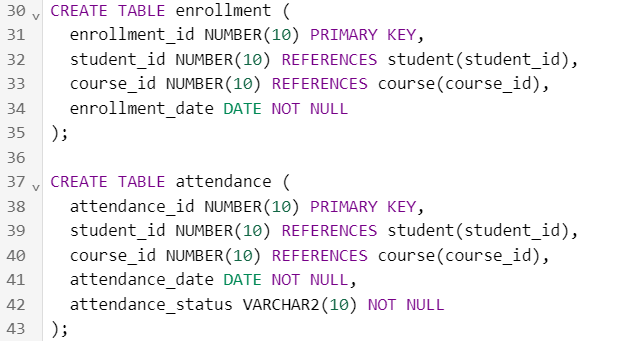
# ER TO TABLE

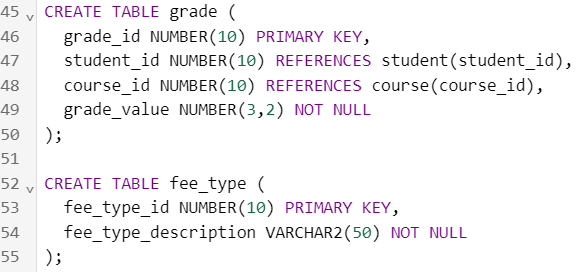
****

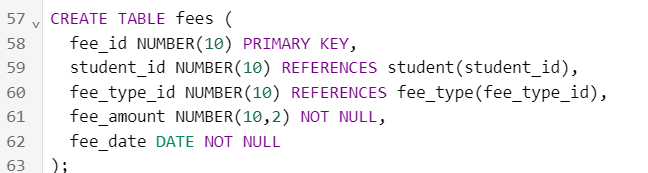
**Creating Table**

****

****

****

****

****

# SQL Output Screenshots

Table-1

COURSE

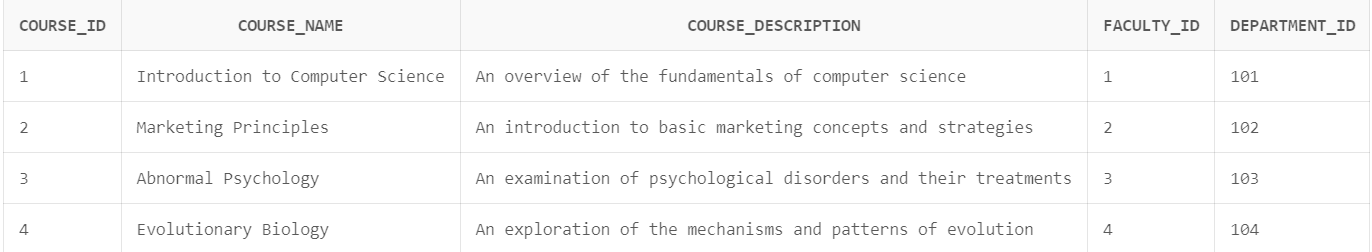


Table-2

DEPARTMENT

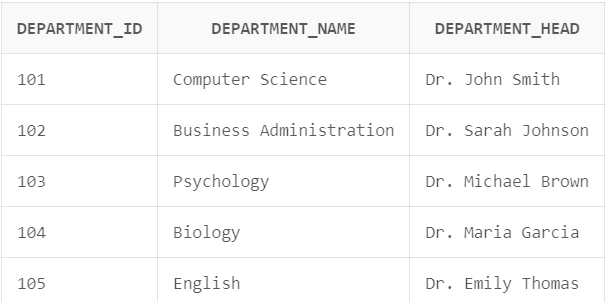


Table-3

STUDENT

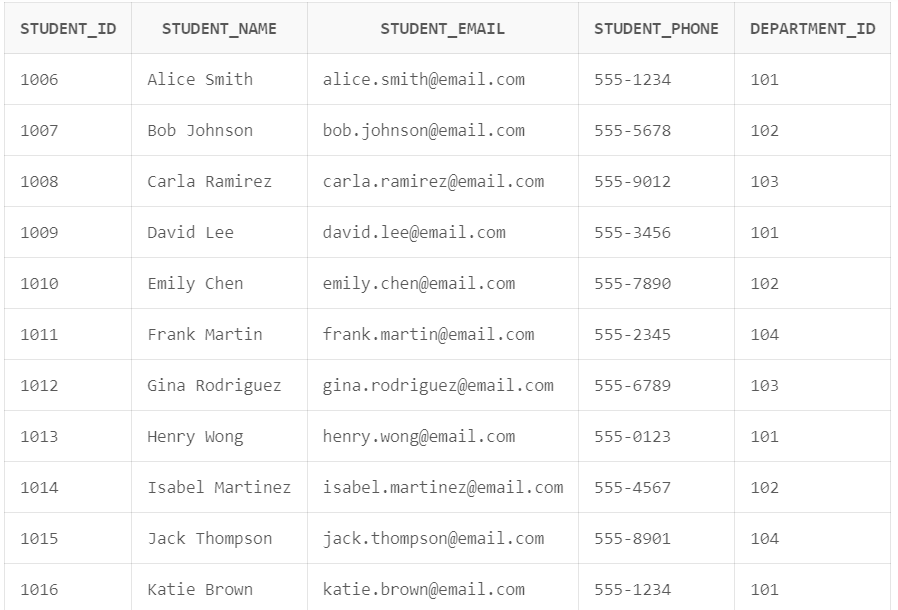


TABLE-4

FACULTY

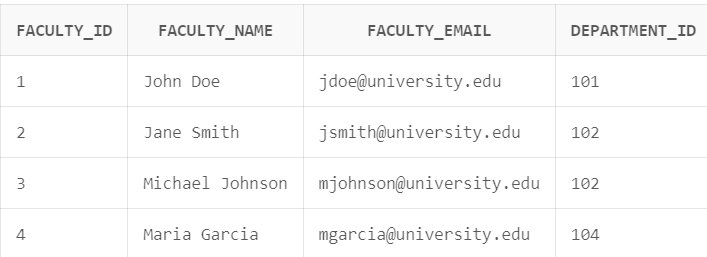


TABLE-5

FEE

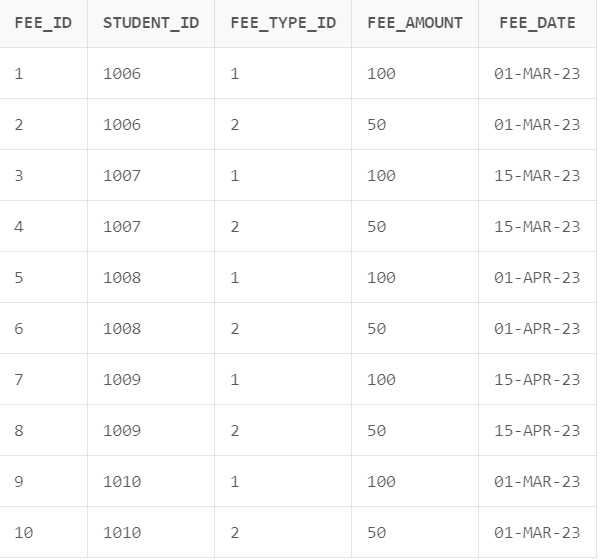


TABLE-6

FEE TYPE

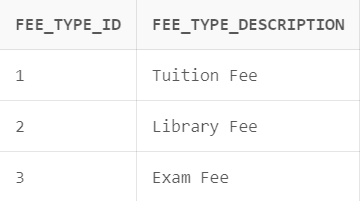


TABLE-7

ENROLLMENT

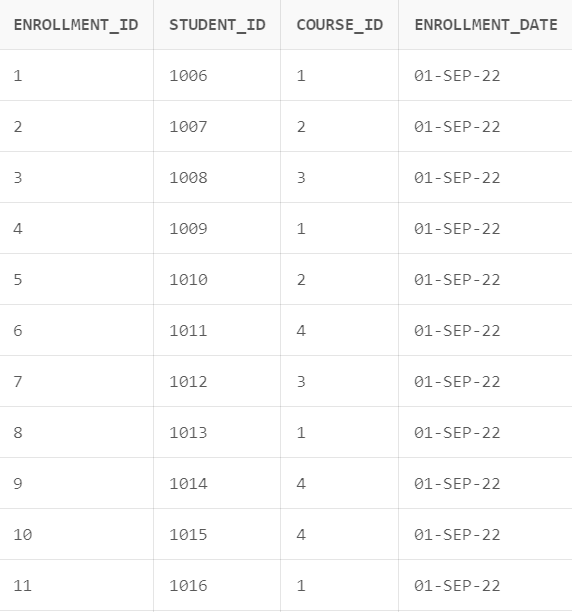
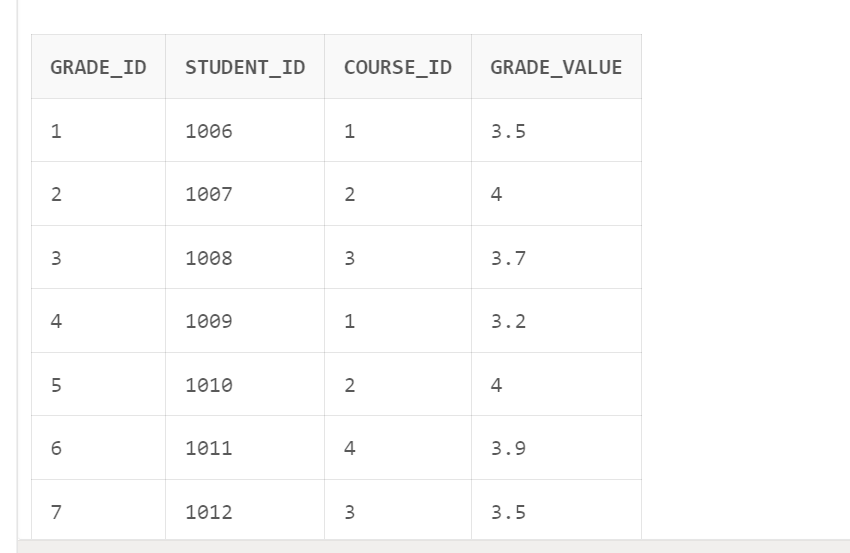


TABLE-8

GRADE



**PL/SQL Codes and Screenshots**

1.TRIGGER-

CREATE TABLE student (

student\_id NUMBER(10) PRIMARY KEY,

student\_name VARCHAR2(50) NOT NULL,

student\_email VARCHAR2(50) NOT NULL,

student\_phone VARCHAR2(20) NOT NULL,

department\_id NUMBER(10) REFERENCES department(department\_id)

);

CREATE OR REPLACE TRIGGER trg\_student\_name\_capitalized

BEFORE INSERT OR UPDATE ON student

FOR EACH ROW

BEGIN

:new.student\_name := INITCAP(:new.student\_name);

END;

INSERT INTO student (student\_id, student\_name, student\_email, student\_phone, department\_id)

VALUES (1006, 'Alice Smith', 'alice.smith@email.com', '555-1234', 101);

INSERT INTO student (student\_id, student\_name, student\_email, student\_phone, department\_id)

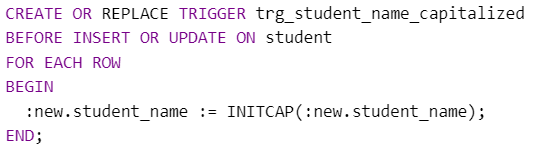
VALUES (1007, 'Bob Johnson', 'bob.johnson@email.com', '555-5678', 102);

INSERT INTO student (student\_id, student\_name, student\_email, student\_phone, department\_id)

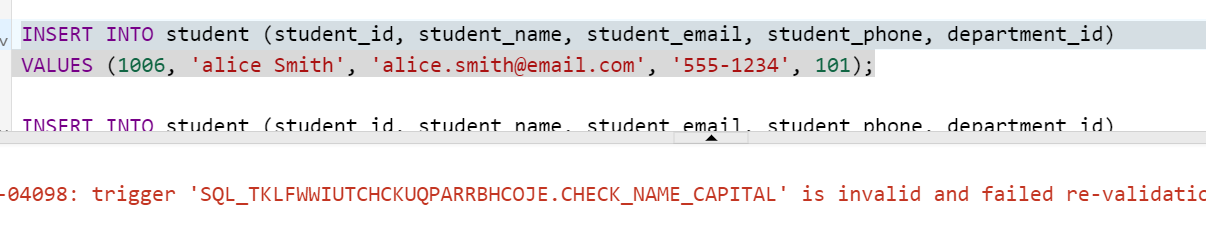
VALUES (1008, 'Carla Ramirez', 'carla.ramirez@email.com', '555-9012', 103);

INSERT INTO student (student\_id, student\_name, student\_email, student\_phone, department\_id)

VALUES (1009, 'David Lee', 'david.lee@email.com', '555-3456', 101);



**OUTPUT-**



**2.PROCEDURE-**

CREATE OR REPLACE PROCEDURE get\_course\_grades(

course\_id IN NUMBER,

grades OUT SYS\_REFCURSOR

)

AS

BEGIN

OPEN grades FOR

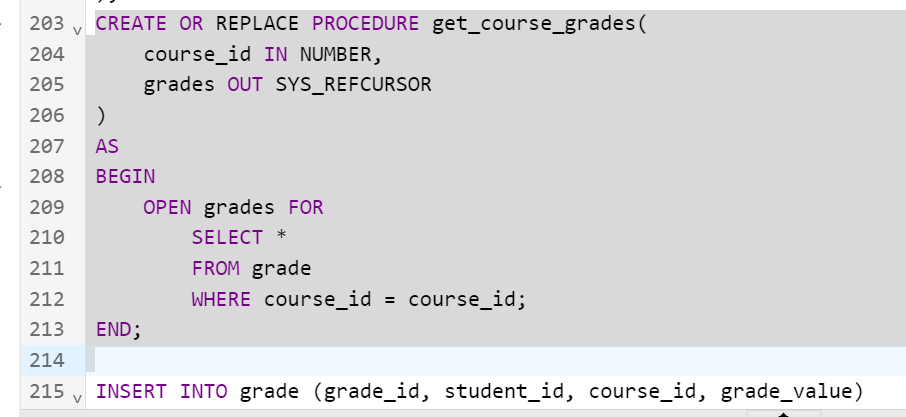
SELECT \*

FROM grade

WHERE course\_id = course\_id;

END;

**OUTPUT-**

****

**3.EXCEPTION-**

**BEGIN**

**INSERT INTO department (department\_id, department\_name, department\_head)**

**VALUES (106, 'Mathematics', 'Dr. William Brown');**

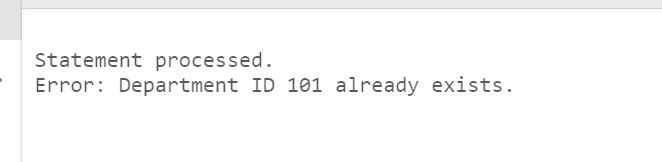
**EXCEPTION**

**WHEN DUP\_VAL\_ON\_INDEX THEN**

**DBMS\_OUTPUT.PUT\_LINE('Error: Department ID 106 already exists.');**

**END;**

**OUTPUT-**

****

**4.CURSOR-**

**DECLARE**

**CURSOR dept\_faculty\_cursor IS**

**SELECT d.department\_name, f.faculty\_name**

**FROM department d**

**INNER JOIN faculty f ON d.department\_id = f.department\_id**

**ORDER BY d.department\_name;**

**dept\_name department.department\_name%TYPE;**

**faculty\_name faculty.faculty\_name%TYPE;**

**BEGIN**

**OPEN dept\_faculty\_cursor;**

**LOOP**

**FETCH dept\_faculty\_cursor INTO dept\_name, faculty\_name;**

**EXIT WHEN dept\_faculty\_cursor%NOTFOUND;**

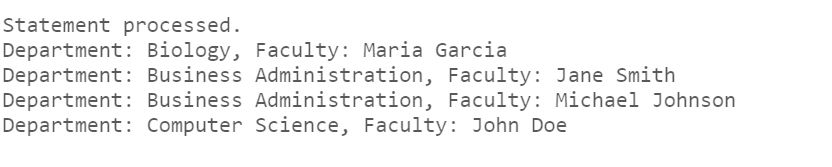
**DBMS\_OUTPUT.PUT\_LINE('Department: ' || dept\_name || ', Faculty: ' || faculty\_name);**

**END LOOP;**

**CLOSE dept\_faculty\_cursor;**

**END;**

**OUTPUT-**

****