National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computer Science

CS236: Advance Database Systems

Class: BSCS-13

Lab 08: Revision

MAIER ALI

481889

13-A

Date: 18-03-2025

Time: 2:30 to 5:00

Instructor: Dr. Bilal Ali

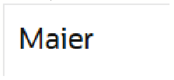
Lab Engineer: Syed Muhammad Ali Musa

Lab 08: Revision

Lab Task

1) **Write a PL/SQL code to print your name**

```
 BEGIN

   DBMS_OUTPUT.PUT_LINE('Maier');
END;
```

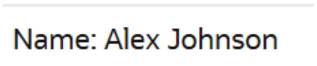


```
Maier
```

2) **Explore datatypes in Pl/SQL and describe at least 5 with their examples. (Add Screen Shots)**

- VARCHAR2 : Stores variable-length character strings.

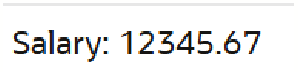
```
DECLARE
   name VARCHAR2(50);
BEGIN
   name := 'Alex Johnson';
   DBMS_OUTPUT.PUT_LINE('Name: ' || name);
END;
/
```



```
Name: Alex Johnson
```

–NUMBER : Used to store integers or real numbers with optional precision and scale.

```
DECLARE
   salary NUMBER(8,2);  -- 8 digits total, 2 after decimal
BEGIN
   salary := 12345.67;
   DBMS_OUTPUT.PUT_LINE('Salary: ' || salary);
END;
/
```

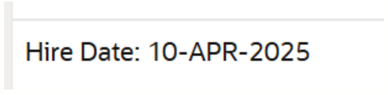


```
Salary: 12345.67
```

- DATE: Used to store date and time values.

```
DECLARE
   hire_date DATE;
```

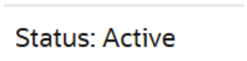
```
BEGIN
   hire_date := TO_DATE('2025-04-10', 'YYYY-MM-DD');
   DBMS_OUTPUT.PUT_LINE('Hire Date: ' || TO_CHAR(hire_date, 'DD-MON-YYYY'));
END;
/
```



```
Hire Date: 10-APR-2025
```

-Boolean : Can store logical values: `TRUE`, `FALSE`, or `NULL`. Only usable in PL/SQL, not in SQL directly.

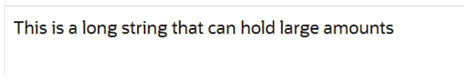
```
 DECLARE

   is_active BOOLEAN;
BEGIN
   is_active := TRUE;
   IF is_active THEN
      DBMS_OUTPUT.PUT_LINE('Status: Active');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Status: Inactive');
   END IF;
END;
/
```



```
Status: Active
```

- CLOB (Character Large Object) : Used for storing large text data.

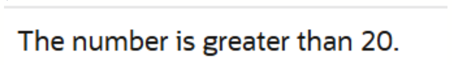
```
DECLARE
   large_text CLOB;
BEGIN
   large_text := 'This is a long string that can hold large amounts of data...';
   DBMS_OUTPUT.PUT_LINE(SUBSTR(large_text, 1, 50));   -- show only first 50 characters
END;
/
```



```
This is a long string that can hold large amounts
```

3) **Write a block of code that checks the num if the num is greater then 20 display a message other wise display a message the num is not greater then 20.**

```
 DECLARE

   num NUMBER := 25;   -- You can change this value to test other cases
BEGIN
   IF num > 20 THEN
      DBMS_OUTPUT.PUT_LINE('The number is greater than 20.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('The number is not greater than 20.');
   END IF;
END;
/
```

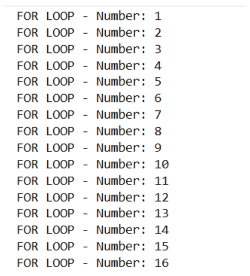


```
The number is greater than 20.
```

4) **Print a num between 1 to 20 using all loop types. (Include Screen Shots for every loop)**

-For loop

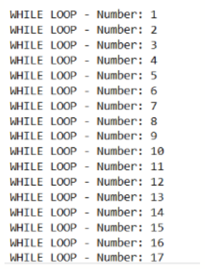
```
BEGIN
   FOR i IN 1..20 LOOP
      DBMS_OUTPUT.PUT_LINE('FOR LOOP - Number: ' || i);
   END LOOP;
END;
/
```



```
FOR LOOP - Number: 1
FOR LOOP - Number: 2
FOR LOOP - Number: 3
FOR LOOP - Number: 4
FOR LOOP - Number: 5
FOR LOOP - Number: 6
FOR LOOP - Number: 7
FOR LOOP - Number: 8
FOR LOOP - Number: 9
FOR LOOP - Number: 10
FOR LOOP - Number: 11
FOR LOOP - Number: 12
FOR LOOP - Number: 13
FOR LOOP - Number: 14
FOR LOOP - Number: 15
FOR LOOP - Number: 16
```

-While loop

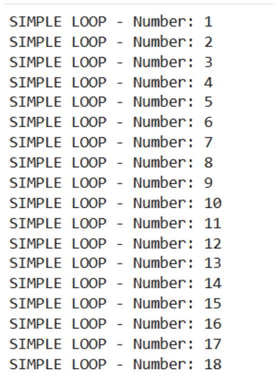
```
DECLARE
   i NUMBER := 1;
BEGIN
   WHILE i <= 20 LOOP
      DBMS_OUTPUT.PUT_LINE('WHILE LOOP - Number: ' || i);
      i := i + 1;
   END LOOP;
```



```
WHILE LOOP - Number: 1
WHILE LOOP - Number: 2
WHILE LOOP - Number: 3
WHILE LOOP - Number: 4
WHILE LOOP - Number: 5
WHILE LOOP - Number: 6
WHILE LOOP - Number: 7
WHILE LOOP - Number: 8
WHILE LOOP - Number: 9
WHILE LOOP - Number: 10
WHILE LOOP - Number: 11
WHILE LOOP - Number: 12
WHILE LOOP - Number: 13
WHILE LOOP - Number: 14
WHILE LOOP - Number: 15
WHILE LOOP - Number: 16
WHILE LOOP - Number: 17
```

– LOOP (Simple/Infinite Loop with EXIT Condition)

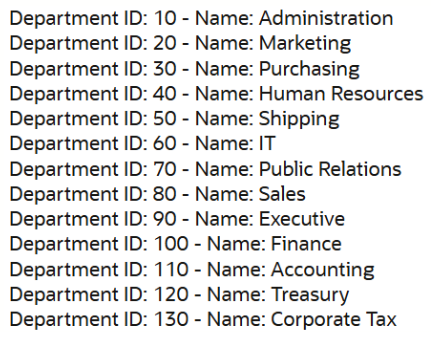
```
DECLARE
   i NUMBER := 1;
BEGIN
   LOOP
      EXIT WHEN i > 20;
      DBMS_OUTPUT.PUT_LINE('SIMPLE LOOP - Number: ' || i);
      i := i + 1;
   END LOOP;
END;
/
```



```
SIMPLE LOOP - Number: 1
SIMPLE LOOP - Number: 2
SIMPLE LOOP - Number: 3
SIMPLE LOOP - Number: 4
SIMPLE LOOP - Number: 5
SIMPLE LOOP - Number: 6
SIMPLE LOOP - Number: 7
SIMPLE LOOP - Number: 8
SIMPLE LOOP - Number: 9
SIMPLE LOOP - Number: 10
SIMPLE LOOP - Number: 11
SIMPLE LOOP - Number: 12
SIMPLE LOOP - Number: 13
SIMPLE LOOP - Number: 14
SIMPLE LOOP - Number: 15
SIMPLE LOOP - Number: 16
SIMPLE LOOP - Number: 17
SIMPLE LOOP - Number: 18
```

5) **Use while loop to print departments and their name.**

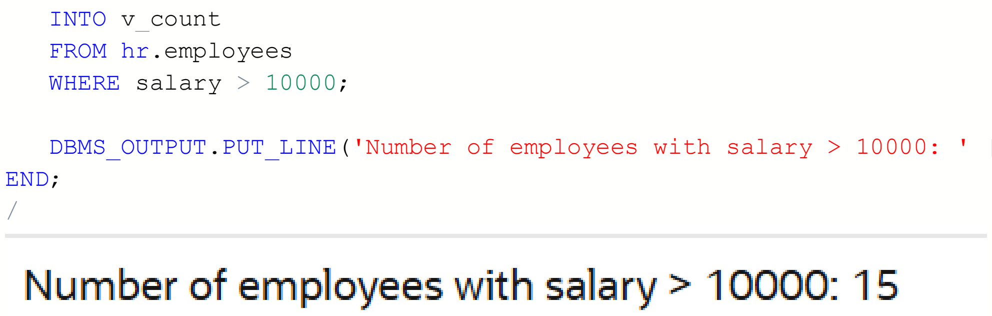
```
 DECLARE

   CURSOR dept_cursor IS
      SELECT department_id, department_name FROM hr.departments;

   v_dept_id   hr.departments.department_id%TYPE;
   v_dept_name hr.departments.department_name%TYPE;
BEGIN
   OPEN dept_cursor;

   FETCH dept_cursor INTO v_dept_id, v_dept_name;
   WHILE dept_cursor%FOUND LOOP
      DBMS_OUTPUT.PUT_LINE('Department ID: ' || v_dept_id || ' - Name: ' ||
v_dept_name);
      FETCH dept_cursor INTO v_dept_id, v_dept_name;
   END LOOP;

   CLOSE dept_cursor;
END;
```



```
Department ID: 10 - Name: Administration
Department ID: 20 - Name: Marketing
Department ID: 30 - Name: Purchasing
Department ID: 40 - Name: Human Resources
Department ID: 50 - Name: Shipping
Department ID: 60 - Name: IT
Department ID: 70 - Name: Public Relations
Department ID: 80 - Name: Sales
Department ID: 90 - Name: Executive
Department ID: 100 - Name: Finance
Department ID: 110 - Name: Accounting
Department ID: 120 - Name: Treasury
Department ID: 130 - Name: Corporate Tax
```

6) **Provide a count of All Employees with a salary greater then 10000**

```
 DECLARE

   v_count NUMBER;
BEGIN
   SELECT COUNT(*)
```



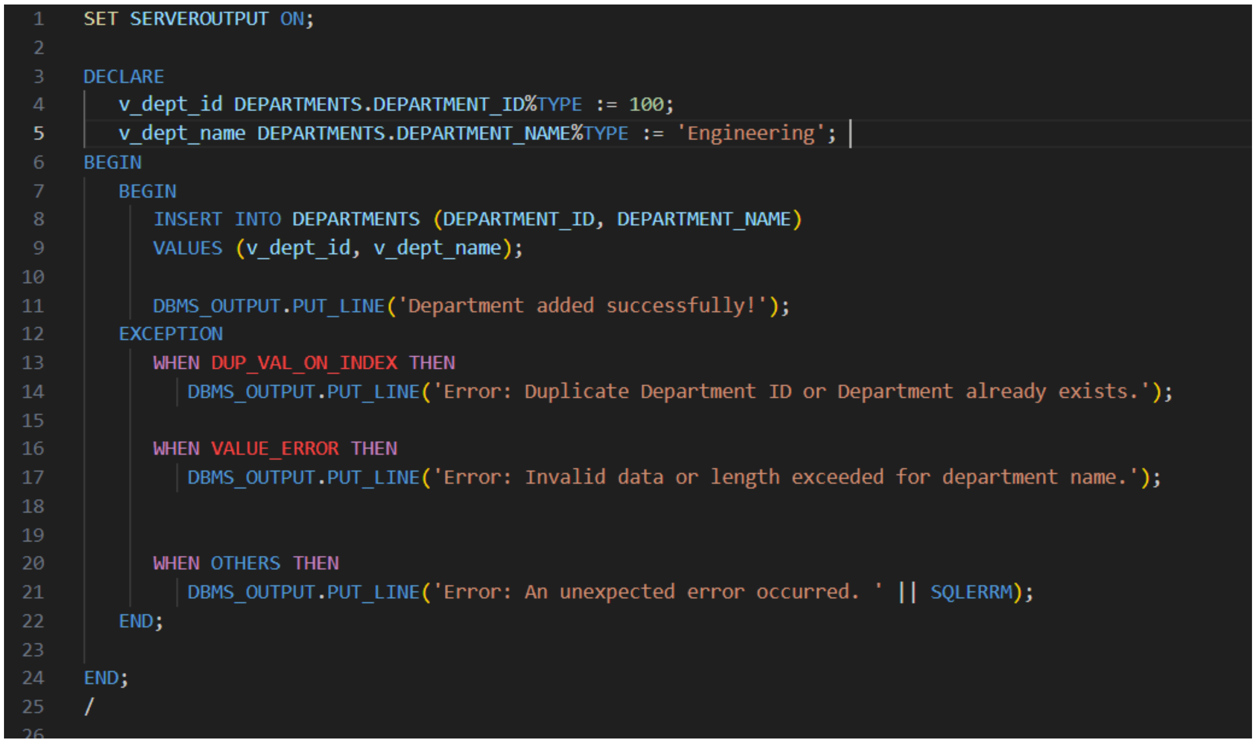
```
   INTO v_count
   FROM hr.employees
   WHERE salary > 10000;

   DBMS_OUTPUT.PUT_LINE('Number of employees with salary > 10000: ' || v_count);
END;
/
```

```
Number of employees with salary > 10000: 15
```

7) **Insert data into department table and use PL/SQL block for this instead of simple insert statement. Handle the incorrect data in exception block.**
Handle following scenarios
- **Duplicate Departments**
- **Invalid data or length exceeded**
- **Any other Error**

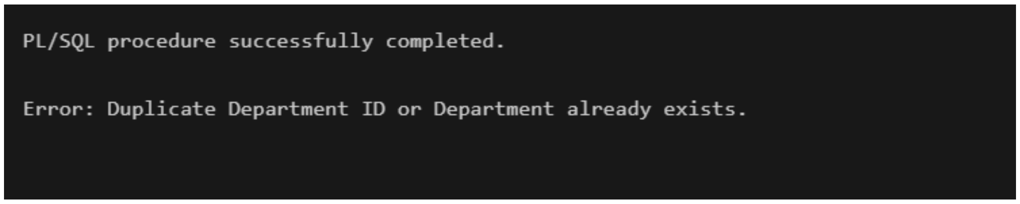
Test your exceptions by putting the incorrect data

- **Doing this lab in vs code as live sql server does not support insertion in data**



```
SET SERVEROUTPUT ON;

DECLARE
    v_dept_id DEPARTMENTS.DEPARTMENT_ID%TYPE := 100;
    v_dept_name DEPARTMENTS.DEPARTMENT_NAME%TYPE := 'Engineering';
BEGIN
    BEGIN
        INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME)
        VALUES (v_dept_id, v_dept_name);

        DBMS_OUTPUT.PUT_LINE('Department added successfully!');
    EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
            DBMS_OUTPUT.PUT_LINE('Error: Duplicate Department ID or Department already exists.');

        WHEN VALUE_ERROR THEN
            DBMS_OUTPUT.PUT_LINE('Error: Invalid data or length exceeded for department name.');


        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error: An unexpected error occurred. ' || SQLERRM);
    END;

END;
/
```

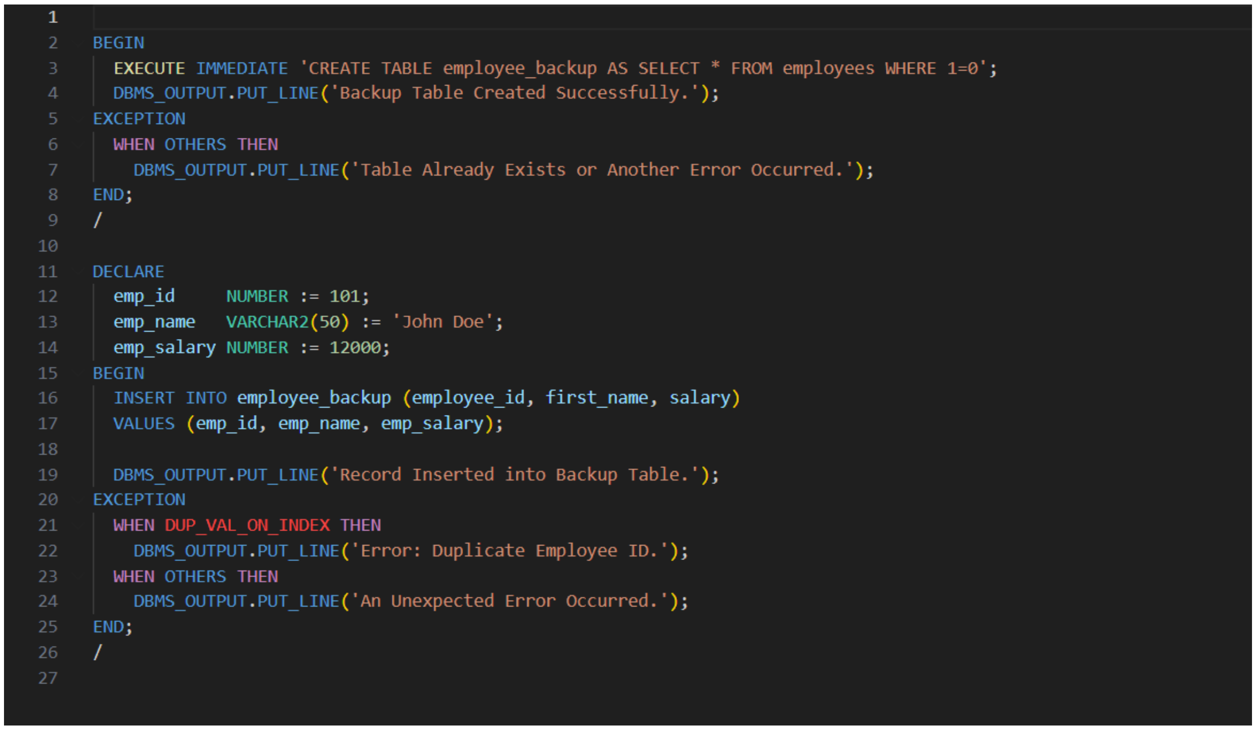
Output :-



```
PL/SQL procedure successfully completed.

Error: Duplicate Department ID or Department already exists.
```

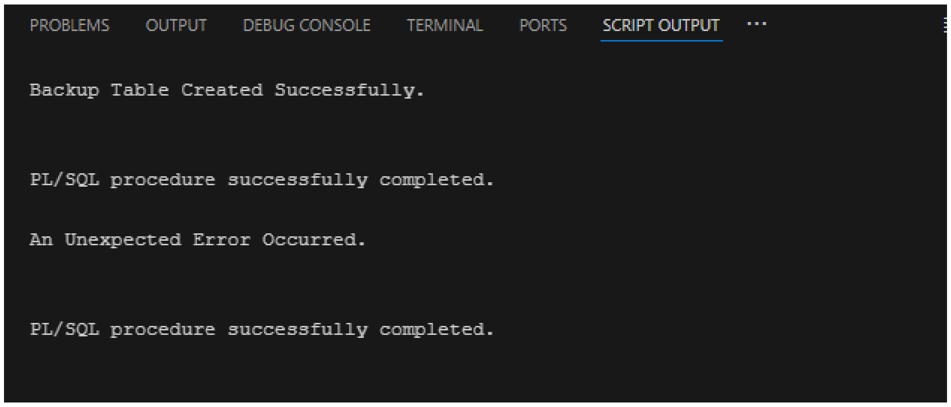
8) **Create a new table (employee backup) in your HR Schema and create a backup table for employee table. Incase of any duplicate entry your code handle this in exception block.**

Test your exception by putting the incorrect data



```

BEGIN
  EXECUTE IMMEDIATE 'CREATE TABLE employee_backup AS SELECT * FROM employees WHERE 1=0';
  DBMS_OUTPUT.PUT_LINE('Backup Table Created Successfully.');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Table Already Exists or Another Error Occurred.');
END;
/

DECLARE
  emp_id     NUMBER := 101;
  emp_name   VARCHAR2(50) := 'John Doe';
  emp_salary NUMBER := 12000;
BEGIN
  INSERT INTO employee_backup (employee_id, first_name, salary)
  VALUES (emp_id, emp_name, emp_salary);

  DBMS_OUTPUT.PUT_LINE('Record Inserted into Backup Table.');
EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('Error: Duplicate Employee ID.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An Unexpected Error Occurred.');
END;
/

```

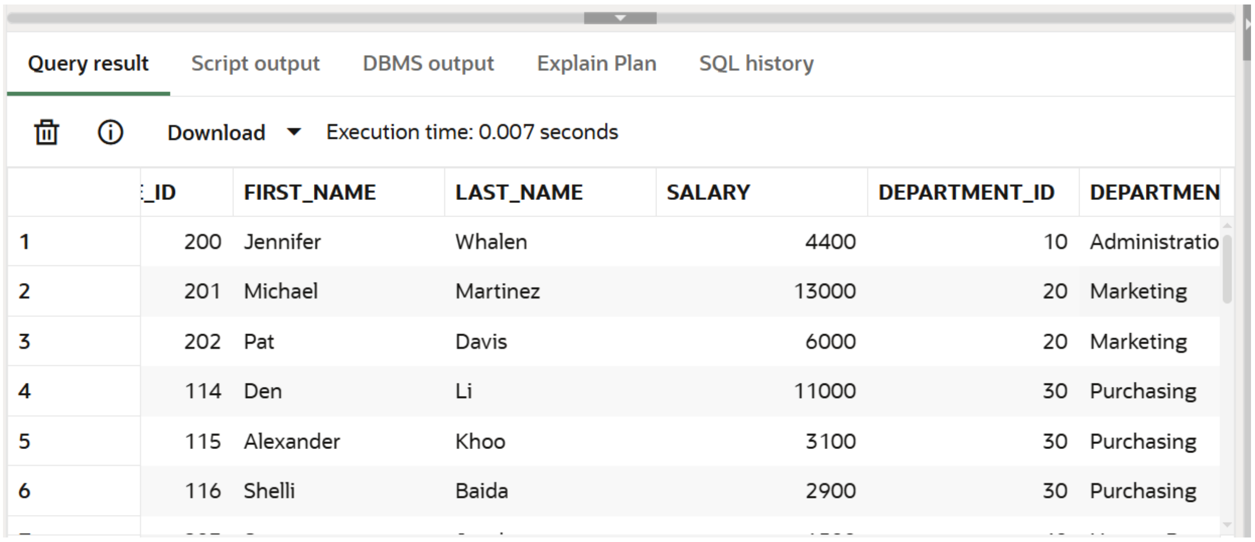
Output :-



```
Backup Table Created Successfully.


PL/SQL procedure successfully completed.

An Unexpected Error Occurred.


PL/SQL procedure successfully completed.
```

9) **For each department in the** HR Schema**, I want to display the** top 3 employees **who have the** highest salaries**.**

```
 SELECT *

FROM (
    SELECT
        e.employee_id,
        e.first_name,
        e.last_name,
        e.salary,
        e.department_id,
        d.department_name,
        RANK() OVER (PARTITION BY e.department_id ORDER BY e.salary
DESC) AS salary_rank
    FROM hr.employees e
    JOIN hr.departments d ON e.department_id = d.department_id
)
WHERE salary_rank <= 3
ORDER BY department_id, salary_rank;
```



	_ID	FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT_ID	DEPARTMEN
1	200	Jennifer	Whalen	4400	10	Administratio
2	201	Michael	Martinez	13000	20	Marketing
3	202	Pat	Davis	6000	20	Marketing
4	114	Den	Li	11000	30	Purchasing
5	115	Alexander	Khoo	3100	30	Purchasing
6	116	Shelli	Baida	2900	30	Purchasing

Deliverables:

Submit a PDF document including the PL/SQL code to answer above-mentioned information needs as well as snapshot of their outcome when executed.