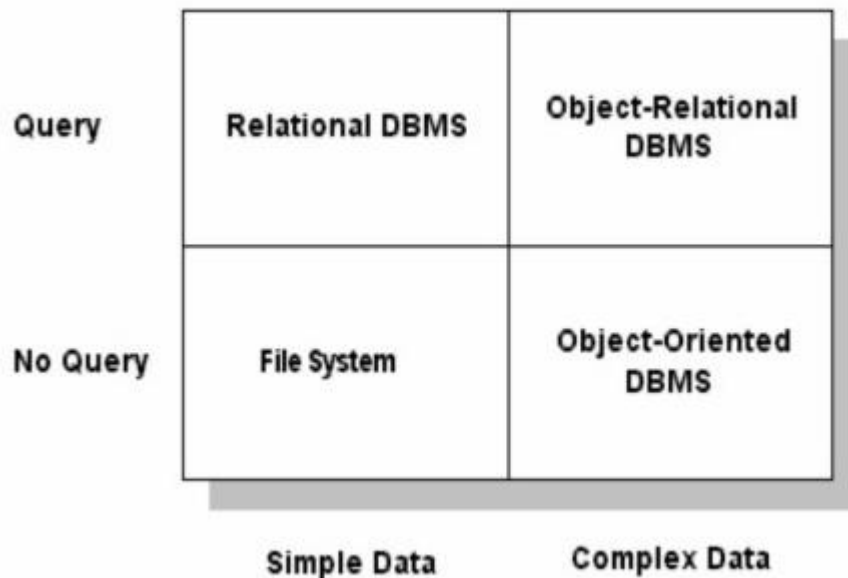


BIC2353 – DATABASE TECHNOLOGY

1. Discuss the classification of Michael Stonebraker's Classification of both database applications and systems.



- When it uses **query** and is **simple data**, it is relational DBMS (Upper left quadrant)
 - When it uses **query** and is **complex data**, it is object-relational DBMS (Upper right quadrant)
 - When it uses **no query** and is **simple data**, it is File system (Lower left quadrant)
 - When it uses **no query** and is **complex data**, it is Object-oriented DBMS (Lower right quadrant)
- **Complex data = object**
- **Query = relational**

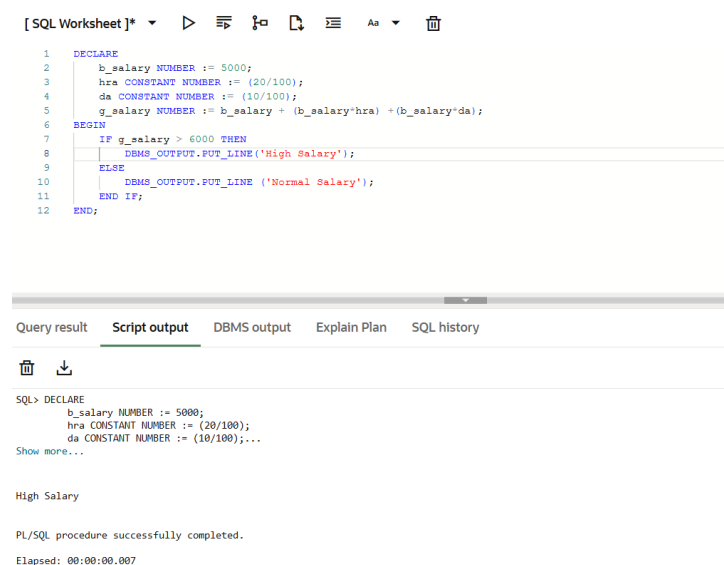
2. Write PL/SQL code for the following :

Question 1: Salary Calculation with IF-ELSE

Write a PL/SQL program to calculate the gross salary of an employee:

- Declare basic salary as a variable and set it to 5000.
- Use a constant for HRA (20%) and DA (10%) of the basic salary.
- If gross salary exceeds 6000, print "High Salary", else print "Normal Salary".

Note : $\text{gross_salary} := \text{basic_salary} + (\text{basic_salary} * \text{hra}) + (\text{basic_salary} * \text{da});$



The screenshot shows an SQL Worksheet interface. At the top, there's a toolbar with icons for running, saving, and other functions. Below the toolbar, the SQL code is displayed in a text area. The code is a PL/SQL program that declares a variable for basic salary, constants for HRA and DA, calculates the gross salary, and prints a message based on whether the gross salary exceeds 6000. The code is as follows:

```
1 DECLARE
2     b_salary NUMBER := 5000;
3     hra CONSTANT NUMBER := (20/100);
4     da CONSTANT NUMBER := (10/100);
5     g_salary NUMBER := b_salary + (b_salary*hra) + (b_salary*da);
6 BEGIN
7     IF g_salary > 6000 THEN
8         DBMS_OUTPUT.PUT_LINE('High Salary');
9     ELSE
10        DBMS_OUTPUT.PUT_LINE ('Normal Salary');
11    END IF;
12 END;
```

Below the code, there's a section for the execution output. It shows the results of the SQL execution, including the output of the PL/SQL procedure and the elapsed time.

Query result **Script output** DBMS output Explain Plan SQL history

SQL> DECLARE
b_salary NUMBER := 5000;
hra CONSTANT NUMBER := (20/100);
da CONSTANT NUMBER := (10/100);...
Show more...

High Salary

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007

DECLARE

b_salary NUMBER := 5000;

hra CONSTANT NUMBER := (20/100);

da CONSTANT NUMBER := (10/100);

g_salary NUMBER := b_salary + (b_salary*hra) +(b_salary*da);

BEGIN

IF g_salary > 6000 THEN

DBMS_OUTPUT.PUT_LINE('High Salary');

ELSE

DBMS_OUTPUT.PUT_LINE ('Normal Salary');

END IF;

END;

Question 2: Loop and Arithmetic Operations

Write a PL/SQL program that:

- Uses a loop to calculate the square and cube of numbers from 1 to 5.
- Print the number, its square, and its cube using DBMS_OUTPUT.PUT_LINE.

```
14 DECLARE
15     i NUMBER:= 1;
16     square NUMBER;
17     cube NUMBER;
18 BEGIN
19     LOOP
20         EXIT WHEN i>5;
21         square := i*i;
22         cube := i*i*i;
23         DBMS_OUTPUT.PUT_LINE('i = ' || i || ' square = ' || square || ' cube = ' || cube);
24         i := i + 1;
25     END LOOP;
26 END;
27
28
```

Query result **Script output** DBMS output Explain Plan SQL history



```
i = 1 square = 1 cube = 1
i = 2 square = 4 cube = 8
i = 3 square = 9 cube = 27
i = 4 square = 16 cube = 64
i = 5 square = 25 cube = 125
```

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007

DECLARE

 i NUMBER:= 1;

 square NUMBER;

 cube NUMBER;

BEGIN

 LOOP

 EXIT WHEN i>5;

 square := i*i;

 cube := i*i*i;

 DBMS_OUTPUT.PUT_LINE('i = ' || i || ' square = ' || square || ' cube = ' || cube);

 i := i + 1;

 END LOOP;

END;

Question 3: Character Manipulation Functions

Write a PL/SQL program to:

- Declare a variable containing the string ' welcomeToPLSQL***'.
- Use and display the output of LTRIM, RTRIM, UPPER, LOWER, INITCAP, and LENGTH functions on it.

```
28 DECLARE
29     original_str VARCHAR2(50) := ' welcomeToPLSQL***';
30 BEGIN
31     DBMS_OUTPUT.PUT_LINE('Original string: [' || original_str || ']');
32     DBMS_OUTPUT.PUT_LINE('LTRIM: [' || LTRIM(original_str, ' ') || ']');
33     DBMS_OUTPUT.PUT_LINE('RTRIM: [' || RTRIM(original_str, '*') || ']');
34     DBMS_OUTPUT.PUT_LINE('UPPER: [' || UPPER(original_str) || ']');
35     DBMS_OUTPUT.PUT_LINE('LOWER: [' || LOWER(original_str) || ']');
36     DBMS_OUTPUT.PUT_LINE('INITCAP: [' || INITCAP(original_str) || ']');
37     DBMS_OUTPUT.PUT_LINE('LENGTH: ' || LENGTH(original_str));
38 END;
```

Query result **Script output** DBMS output Explain Plan SQL history



```
Original string: [ welcomeToPLSQL***]
LTRIM: [welcomeToPLSQL***]
RTRIM: [ welcomeToPLSQL]
UPPER: [ WELCOMETOPLSQL***]
LOWER: [ welcometoplsql***]
INITCAP: [ Welcometoplsql***]
LENGTH: 20
```

DECLARE

```
original_str VARCHAR2(50) := ' welcomeToPLSQL***';
```

BEGIN

```
DBMS_OUTPUT.PUT_LINE('Original string: [' || original_str || ']');
```

```
DBMS_OUTPUT.PUT_LINE('LTRIM: [' || LTRIM(original_str, ' ') || ']');
```

```
DBMS_OUTPUT.PUT_LINE('RTRIM: [' || RTRIM(original_str, '*') || ']');
```

```
DBMS_OUTPUT.PUT_LINE('UPPER: [' || UPPER(original_str) || ']');
```

```
DBMS_OUTPUT.PUT_LINE('LOWER: [' || LOWER(original_str) || ']');
```

```
DBMS_OUTPUT.PUT_LINE('INITCAP: [' || INITCAP(original_str) || ']');
```

```
DBMS_OUTPUT.PUT_LINE('LENGTH: ' || LENGTH(original_str));
```

END;