Ex.no: 05

Date:

JOIN OPERATIONS AND SUBQUERIES

AIM:

To perform join operation and subqueries for university database system

Types of Joins :

1. Inner Join
2. Left Join (Or Left Outer Join)
3. Right Join (Or Right Outer Join)
4. Full Outer Join
5. Cross Join
6. Self-Join

Table creation: **create**

table student (

sid int(10), sname
varchar(20), dpmt
varchar(5), email
varchar(25), contactno
numeric(10)
);

Table created.

SQL> **insert into student values**
(101, 'Nisanth', 'AM', 'nisanthg.24mca@kongu.edu', 9345292781),
(102, 'Sivakumar', 'DBT', 'sivakumarp.24mca@kongu.edu', 8072363074),
(103, 'Sachin', 'C', 'sachins.24mca@kongu.edu', 8754681258),
(104, 'Siva', 'IOT', 'sivak.24mca@kongu.edu', 6380603146);

SQL> select * from student;

SID	SNAME	DPMT	EMAIL	CONTACTNO
101	Nisanth	AM	nisanthg.24mca@kongu.edu	9345292781
102	Sivakumar	DBT	sivakumarp.24mca@kongu.edu	8072363074
103	Sachin	C	sachins.24mca@kongu.edu	8754681258
104	Siva	IOT	sivak.24mca@kongu.edu	6380603146

CREATE TABLE department (
did int(10),
dname varchar(30),
head varchar(30)
);
Table created

INSERT INTO department VALUES
(101, 'AM', 'Dr. Smith'),
(102, 'DBT', 'Dr. Johnson'),
(103, 'DS', 'Dr. Lee'),
(104, 'C', 'Dr. Kim'),
(105, 'IoT', 'Dr. Patel'),
(106, 'Blockchain', 'Dr. Brown');

SQL> select * from department;

DID	DNAME	HEAD
101	AM	Dr. Smith
102	DBT	Dr. Johnson
103	DS	Dr. Lee
104	C	Dr. Kim
105	IOT	Dr. Patel
106	BLOCKCHAIN	Dr. Brown

1. Inner Join

An inner join returns only the rows where there is a match in both tables based on the dpmt field in student and dname field in department.

SQL>SELECT student.sid, student.sname, student.dpmt, student.email, student.contactno, department.did, department.dname, department.head
FROM student
INNER JOIN department ON student.dpmt = department.dname;

SID	SNAME	DPMT	EMAIL	CONTACTNO	DID	DNAME	HEAD
101	Nisanth	AM	nisanthg.24mca@kongu.edu	9345292781	101	AM	Dr. Smith
102	Sivakumar	DBT	sivakumarp.24mca@kongu.edu	8072363074	102	DBT	Dr. Johnson
103	Sachin	C	sachins.24mca@kongu.edu	8754681258	104	C	Dr. Kim
104	Siva	IOT	sivak.24mca@kongu.edu	6380603146	105	IoT	Dr. Patel

2.Left Join (Left Outer Join)

A left join returns all rows from the student table and matched rows from the department table. If there is no match, NULL values are returned.

SQL>SELECT student.sid, student.sname, student.dpmt, student.email, student.contactno, department.did, department.dname, department.head
FROM student
LEFT JOIN department ON student.dpmt = department.dname;

SID	SNAME	DPMT	EMAIL	CONTACTNO	DID	DNAME	HEAD

101	Nisanth	AM	nisanthg.24mca@kongu.edu	9345292781	101	AM	Dr. Smith
102	Sivakumar	DBT	sivakumarp.24mca@kongu.edu	8072363074	102	DBT	Dr. Johnson
103	Sachin	C	sachins.24mca@kongu.edu	8754681258	104	C	Dr. Kim
104	Siva	IOT	sivak.24mca@kongu.edu	6380603146	105	IoT	Dr. Patel

3. Right Join (Right Outer Join)

A right join returns all rows from the department table and matched rows from the student table. If there is no match, NULL values are returned.

SQL>SELECT student.sid, student.sname, student.dpmt, student.email, student.contactno, department.did, department.dname, department.head
FROM student
RIGHT JOIN department ON student.dpmt = department.dname;

SID	SNAME	DPMT	EMAIL	CONTACTNO	DID	DNAME	HEAD
101	Nisanth	AM	nisanthg.24mca@kongu.edu	9345292781	101	AM	Dr. Smith
102	Sivakumar	DBT	sivakumarp.24mca@kongu.edu	8072363074	102	DBT	Dr. Johnson
103	Sachin	C	sachins.24mca@kongu.edu	8754681258	104	C	Dr. Kim
104	Siva	IOT	sivak.24mca@kongu.edu	6380603146	105	IoT	Dr. Patel
NULL	NULL	NULL	NULL	NULL	103	DS	Dr. Lee

NULL	NULL	NULL	NULL	NULL	106	Blockchain	Dr. Brown

4. Full Outer Join

A full outer join returns all rows where there is a match in either table, or NULL where no match exists in either table.

SQL>department.did, department.dname, department.head

FROM student

FULL OUTER JOIN department ON student.dpmt = department.dname;

SID	SNAME	DPMT	EMAIL	CONTACTNO	DID	DNAME	HEAD
101	Nisanth	AM	nisanthg.24mca@kongu.edu	9345292781	101	AM	Dr. Smith
102	Sivakumar	DBT	sivakumarp.24mca@kongu.edu	8072363074	102	DBT	Dr. Johnson
103	Sachin	C	sachins.24mca@kongu.edu	8754681258	104	C	Dr. Kim
104	Siva	IOT	sivak.24mca@kongu.edu	6380603146	105	IoT	Dr. Patel
NULL	NULL	NULL	NULL	NULL	103	DS	Dr. Lee
NULL	NULL	NULL	NULL	NULL	106	Blockchain	Dr. Brown

5. Cross Join

A cross join returns the Cartesian product of the two tables, combining each row of the student table with each row of the department table.

SQL> SELECT s.sname, d.dname

FROM student s

CROSS JOIN department d;

SNAME	**DNAME**
Nisanth	AM
Nisanth	DBT
Nisanth	DS
Nisanth	C
Nisanth	IoT
Nisanth	Blockchain
Sivakumar	AM
Sivakumar	DBT
Sivakumar	DS
Sivakumar	C
Sivakumar	IoT
Sivakumar	Blockchain
Sachin	AM
Sachin	DBT
Sachin	DS
Sachin	C
Sachin	IoT
Sachin	Blockchain
Siva	AM
Siva	DBT
Siva	DS
Siva	C
Siva	IoT
Siva	Blockchain

6. Self Join

A self-join combines rows within the same table

1. Self-Join on the student Table to Find Students in the Same Department

SQL>SELECT student.sid AS student1_id, student.sname AS student1_name, student2.sid AS student2_id, student2.sname AS student2_name, student.dpmt
FROM student
JOIN student AS student2 ON student.dpmt = student2.dpmt AND student.sid < student2.sid;

Expected Output:

Based on the current data, no two students share the same department. Therefore, the result of this query would be an **empty set**.

2. Self-Join on the department Table to Find Departments with the Same Head SQL>SELECT department.did AS department1_id, department.dname AS department1_name, department2.did AS department2_id, department2.dname AS department2_name, department.head
FROM department
JOIN department AS department2 ON department.head = department2.head AND department.did < department2.did;

Expected Output:

Each department has a unique head in the current data, so this query would also return an **empty set**.

GROUP BY, ORDER BY, and HAVING clauses

1. Count Students in Each Department

SQL>SELECT dpmt, COUNT(*) AS student_count
FROM student
GROUP BY dpmt
ORDER BY dpmt;

Explanation: This query counts the number of students in each department and orders the results alphabetically by the department code (dpmt).

Expected Output:

DPMT	STUDENT_COUNT
AM	1
C	1
DBT	1
IOT	1

2.List All Departments with Their Heads

SQL>SELECT dname, head

FROM department

ORDER BY dname;

Explanation: This retrieves all departments along with their heads and orders the results alphabetically by department name (dname).

Expected Output:

DNAME	HEAD
AM	Dr. Smith
Blockchain	Dr. Brown
C	Dr. Kim
DBT	Dr. Johnson
DS	Dr. Lee
IoT	Dr. Patel

3.Finding Departments with HAVING Clause

SQL>SELECT dname, head

FROM department

HAVING head LIKE 'Dr.%';

Explanation: This retrieves all departments where the head's name starts with "Dr.".

Expected Output:

DNAME	HEAD
AM	Dr. Smith
DBT	Dr. Johnson
DS	Dr. Lee
C	Dr. Kim
IoT	Dr. Patel
Blockchain	Dr. Brown

COE (30)	
RECORD (20)	
VIVA (10)	
TOTAL (60)	

RESULT:

The execution of join operations, subqueries, and the use of aggregate functions has been performed successfully on the university database system

Ex No: 06

Date:

PL/SQL

AIM:

To execute PL/SQL procedures for University Database Management System.

QUERIES

1.Write a PL/SQL code block to declare a basic variable, assign it a value, and then display that value using `DBMS_OUTPUT.PUT_LINE`.

```
SQL> set serveroutput on;
SQL> declare
2 var varchar2(50);
3 begin
4 var:='hello world';
5 dbms_output.put_line(var);
6 end;
7 /
```

O/P

Database Technologies

PL/SQL procedure successfully completed.

2.Write a PL/SQL code block that takes two numbers as input from the user, calculates their sum, and displays the result. Use DBMS_OUTPUT.PUT_LINE to print the output in the following format:

The sum of <a> and <b> is <sum>

```
DECLARE
    a NUMBER := &a;
    b NUMBER := &b;
    sum NUMBER;
BEGIN
    sum := a + b;
    dbms_output.put_line('The sum of ' || a || ' and ' || b || ' is ' || sum);
END;
/
```

O/P

The sum of 5 and 10 is 15

PL/SQL procedure successfully completed.

3.Write a SQL script to create a table named student with the following columns: sid (a number for student ID), sname (a string for student name), and sdept (a string for student department). Then, insert the following records into the table:

1. **Student ID: 5, Name: Nisanth, Department: MCA**
2. **Student ID: 6, Name: Sivakumar, Department: MCA**
3. **Student ID: 7, Name: Sachin, Department: MCA**
4. **Student ID: 8, Name: Siva, Department: MCA**

After inserting the records, display the contents of the student table.

```
CREATE TABLE student (
    sid NUMBER(5),
    sname VARCHAR2(10),
    sdept VARCHAR2(5)
);

INSERT INTO student VALUES (5, 'Nisanth', 'MCA');
INSERT INTO student VALUES (6, 'Sivakumar', 'MCA');
INSERT INTO student VALUES (7, 'Sachin', 'MCA');
INSERT INTO student VALUES (8, 'Siva', 'MCA');
```

O/P

SELECT * FROM student;

SID	SNAME	SDEPT
5	**Nisanth**	**MCA**
6	**Sivakumar**	**MCA**
7	**Sachin**	**MCA**
8	**Siva**	**MCA**

4. Write a PL/SQL block that retrieves and displays the name and department of a student from the student table based on a specified student ID. Use the following details:

- **Specify a student ID to search for (e.g., 1).**
- **Retrieve the student's name (sname) and department (sdept).**
- **Use DBMS_OUTPUT.PUT_LINE to print the output in the following format:**

Name: <name>, Department: <department>

```
DECLARE
    id NUMBER := 5; -- specify student ID to search
    name VARCHAR2(10);
    dept VARCHAR2(5);
BEGIN
    SELECT sname, sdept INTO name, dept FROM student WHERE sid = id;
    dbms_output.put_line('Name: ' || name || ', Department: ' || dept);
END;
/
```

O/P

Name: Alice, Department: CS

PL/SQL procedure successfully completed.

5.Write a PL/SQL block that takes a number as input from the user and determines whether the number is even or odd. Use a substitution variable (&num) to prompt the user for the input. The block should use the MOD function to check if the number is even or odd and display the result using DBMS_OUTPUT.PUT_LINE in the following format: <num> is even

```
DECLARE
    num NUMBER := &num;
BEGIN
    IF MOD(num, 2) = 0 THEN
        dbms_output.put_line(num || ' is even');
    ELSE
        dbms_output.put_line(num || ' is odd');
    END IF;
```

```
END;
/
```

input num = 7:

O/P

7 is odd

PL/SQL procedure successfully completed.

6. Write a PL/SQL block that takes two numbers as input from the user and determines which number is greater. Use substitution variables (&a and &b) to prompt the user for the two input values. The block should compare the numbers using an IF statement and display the result using DBMS_OUTPUT.PUT_LINE in the following format:

<greater_number> is greater

```
DECLARE
    a NUMBER := &a;
    b NUMBER := &b;
BEGIN
    IF a > b THEN
        dbms_output.put_line(a || ' is greater');
    ELSE
        dbms_output.put_line(b || ' is greater');
    END IF;
END;
/
```

inputs a = 15 and b = 10

O/P

15 is greater

PL/SQL procedure successfully completed.

7. Write a PL/SQL block that checks whether a given number is divisible by 3. Use a substitution variable (&a) to prompt the user for the input number. The block should perform the following tasks:

1. **Take a number as input from the user.**
2. **Use the MOD function to determine if the number is divisible by 3.**
3. **Display a message using DBMS_OUTPUT.PUT_LINE indicating whether the number is divisible by 3 or not. The output should be in the following format: <number> is divisible by 3**

```
DECLARE
   a NUMBER := &a;  -- Take input from the user
BEGIN
   IF MOD(a, 3) = 0 THEN
      dbms_output.put_line(a || ' is divisible by 3');
   ELSE
      dbms_output.put_line(a || ' is not divisible by 3');
   END IF;
END;
/
```

O/p

If the user inputs 9, the output will be: 9 is divisible by 3

If the user inputs 10, the output will be: 10 is not divisible by 3

8. Write a PL/SQL block that uses a loop to print the numbers from 1 to 10. The block should perform the following tasks:

1. **Initialize a variable i with a value of 1.**

2. Use a LOOP to repeatedly execute the block of code.

3. Within the loop, use DBMS_OUTPUT.PUT_LINE to display the current value of i.

4. Increment the value of i by 1 in each iteration.

5. Include an EXIT condition to terminate the loop when i exceeds 10.

```
DECLARE
   i NUMBER := 1;
BEGIN
   LOOP
      EXIT WHEN i > 10;
      dbms_output.put_line(i);
      i := i + 1;
   END LOOP;
END;
/
```

O/P

1

2

3

4

5

6

7

8

9

10

PL/SQL procedure successfully completed.

9. Write a PL/SQL block to generate and display the Fibonacci series up to userspecified number of terms. Use a substitution variable (`&n`) to take the input for the number of terms and print each Fibonacci number using DBMS_OUTPUT.PUT_LINE`.

```
SET SERVEROUTPUT ON;
DECLARE
   n NUMBER;
   a NUMBER := 0;
   b NUMBER := 1;
   temp NUMBER;
BEGIN
   n := &n;
   dbms_output.put_line('Fibonacci Series:');
   FOR i IN 1..n LOOP
      dbms_output.put_line(a);
      temp := a + b;
      a := b;
      b := temp;
   END LOOP;
END;
/
```

O/P

If the user enters 7, the output will be:

Fibonacci Series:

0

1

1

2

3

5

8

10.What does the following PL/SQL code produce, and how does it create a pattern based on user input? Describe the roles of the variables n, i, and j in the nested loops.

```
DECLARE
   n NUMBER := &n;  -- Take input from the user for the number of rows
   i NUMBER;
   j NUMBER;
BEGIN
   FOR i IN 1..n LOOP          -- Loop through rows
      FOR j IN 1..i LOOP       -- Loop through columns
         dbms_output.put(j); -- Print the current row number
      END LOOP;
      dbms_output.new_line;    -- Move to the next line after each row
   END LOOP;
END;
/
```

O/P

1

22

333

4444

55555

COE (30)	
RECORD (20)	
VIVA (10)	
TOTAL (60)	

RESULT:

Thus, the execution of basic PL/SQL queries for University Database Management System.

has been done successfully.