Aggregate Functions_SubQueries_Joins

Presented by

Sagar
Java Consultant

Introduction to Aggregate Functions

- Aggregate functions take a collection (a set or multiset) of values as input and return a single value. These are some features of Aggregate functions:
  - They are used along with the column names in the SELECT statement.
  - They ignore null values.
  - They cannot be used with the WHERE clause.
- Each of these functions perform an action that draws data from a set of rows instead of a single row.

Description of Aggregate Functions

Function Name	Description
SUM()	Adds up the values in the specified column of a numeric data type. Value of the sum must be within the range of that data type.
AVG()	Returns the average of all the values in a specified column. The column must be a numeric data type.
MAX()	Returns the largest value that occurs in the specified column. The column need not be a numeric data type.
MIN()	Returns the smallest value that occurs in the specified column. The column need not be a numeric data type.
COUNT(Qty)	Returns the number of rows that do not have NULL value in the column quantity.
COUNT(*)	Returns the number of rows in the table.

Aggregate Functions: Examples

To display the summation and average of salary from emp table.

Example

```
SELECT SUM (sal) TOTAL, AVG (sal) AVERAGE FROM EMP
```

Sample Output:

```
     TOTAL   AVERAGE
     ================
     29025    2073.21429
```

To display the maximum and minimum salary from emp table.

Example

```
SELECT MAX (sal) Maxsal, MIN (sal) Minsal FROM EMP
```

Sample Output:

```
     MAXSAL   MINSAL
     ================
     5000        800
```

GROUP BY Clause – Introduction

- Any column on which the aggregate function is used is called an aggregate column. Any column on which an aggregate function is not applied is called a non-aggregate column.
  - Example: Find the number of employees belonging to a particular department. Find the total salary for each job.

GROUP BY Clause – Example

- Example:
  - Find the number of employees belonging to a particular department.
  - Find the total salary for each job.

GROUP BY Clause – Output

```
SQL> SELECT JOB, SAL
  2  FROM EMP;

JOB                 SAL
--------- ----------
CLERK               800
SALESMAN           1600
SALESMAN           1250
MANAGER            2975
SALESMAN           1250
MANAGER            2850
MANAGER            2450
ANALYST            3000
PRESIDENT          5000
SALESMAN           1500
CLERK              1100
CLERK               950
ANALYST            3000
CLERK              1300

14 rows selected.
```

```
SQL> SELECT JOB, SUM(SAL)
  2  FROM EMP
  3  GROUP BY JOB;

JOB           SUM(SAL)
--------- ----------
CLERK             4150
SALESMAN          5600
PRESIDENT         5000
MANAGER           8275
ANALYST           6000
```

The example displays using GROUP BY clause with single column.

Grouping More than One Column

- Used to group data in a column, which is grouped within the data of a column.

Example

```
SQL> SELECT count(*), city, description
  2  FROM employee
  3  GROUP BY city, description;

  COUNT(*) CITY       DESCRIPTION
---------- ---------- ---------------
         1 New York   Manager
         4 Vancouver  Tester
         1 Toronto    Programmer
         1 Vancouver  Manager
         1 New York   Tester
```

Using the HAVING Clause

- Features of the HAVING Clause

☑ **The HAVING clause is used to filter data; it is used only with the GROUP BY clause.**

☑ **Aggregate functions can be used with the HAVING clause.**

☑ **Both the WHERE and HAVING clauses are used for filtering data.**

☑ **The WHERE clause is applied *before* the GROUP BY; the HAVING clause is applied *after* GROUP BY.**

☑ **The HAVING clause is used in situations where aggregate columns need to be filtered.**

Example of HAVING Clause

- The HAVING clause is used with GROUP BY to filter records that it returns.

Example

```
SQL> SELECT JOB, SUM(SAL)
  2  FROM EMP
  3  GROUP BY JOB
  4  HAVING SUM(SAL) BETWEEN 1000 AND 5000;

JOB          SUM(SAL)
--------- ----------
CLERK            4150
PRESIDENT        5000

SQL>
```

Example of HAVING Clause (cont.)

To display the DEPTNO and the total number of employees in each department, only those rows should be displayed where three or more employees are working in each department.

Example

```
SELECT DEPTNO,COUNT(*) FROM EMP
GROUP BY DEPTNO HAVING COUNT(*)>3;
```

Sample Output

DEPTNO	COUNT(*)
20	5
30	6

Excluding Group Results: Using the HAVING Clause

- The HAVING clause is used in combination with the GROUP BY clause.

To display the deptno and the total number of employees in each department, only those rows

should be displayed where 3 or more employees are working in each department and their deptno is 30.

Example

```
SELECT DEPTNO,COUNT(*) FROM EMP where deptno=30
GROUP BY DEPTNO HAVING COUNT(*)>3;
```

```
 1  SELECT DEPTNO,COUNT(*) FROM EMP
 2    WHERE DEPTNO=30
 3  GROUP BY DEPTNO
 4* HAVING COUNT(*)>3
QL> /

    DEPTNO   COUNT(*)
---------- ----------
        30          6
```

TO_CHAR Functions With Dates

- The To_CHAR functions must be enclosed in single quotation marks.
- They are case sensitive.
- They can include any valid date format element.
- They have a format mask element to remove padded blanks or suppress leading zeros.
- They are separated from the date value by a comma.

Syntax

```
TO_CHAR(date, 'format_mask')
```

Commonly Used Format Masks in TO_CHAR Functions With Dates

Format_mask	Explanation
MM	Month (1 – 12)
MON	Abbreviated name of the month
D	Day of week (1 – 7)
DAY	Name of the day
DD	Day of the month (1 – 31)
HH	Hours of the day (1 – 12)
HH12	Hours of the day (1 – 12)
HH24	Hours of the day (0 – 23)
MI	Minute (0 – 59)
SS	Seconds (0 – 59)

TO_CHAR Functions with Dates: Example

```
SQL> SELECT ENAME NAME, HIREDATE, TO_CHAR(HIREDATE, 'YYYY') YEAR
  2  FROM EMP;

NAME       HIREDATE  YEAR
---------- --------- ----
SMITH      17-DEC-80 1980
ALLEN      20-FEB-81 1981
WARD       22-FEB-81 1981
JONES      02-APR-81 1981
MARTIN     28-SEP-81 1981
BLAKE      01-MAY-81 1981
CLARK      09-JUN-81 1981
SCOTT      19-APR-87 1987
KING       17-NOV-81 1981
TURNER     08-SEP-81 1981
ADAMS      23-MAY-87 1987
JAMES      03-DEC-81 1981
FORD       03-DEC-81 1981
MILLER     23-JAN-82 1982
```

Only the year portion of the date is shown. To show only the Month, use ‘MON’ and for the day use ‘DAY’.

TO_CHAR Functions with Dates: Example (cont.)

```
SQL> SELECT ENAME NAME, TO_CHAR(HIREDATE, 'MON DDth YYYY') YEAR
  2  FROM EMP
  3  WHERE EMPNO IN (7369, 7566, 7844);

NAME       YEAR
---------- -------------
SMITH      DEC 17TH 1980
JONES      APR 02ND 1981
TURNER     SEP 08TH 1981

SQL> SELECT ENAME NAME, TO_CHAR(HIREDATE, 'MON fmDDth YYYY') YEAR
  2  FROM EMP
  3   WHERE EMPNO IN (7369, 7566, 7844);

NAME       YEAR
---------- -------------
SMITH      DEC 17TH 1980
JONES      APR 2ND 1981
TURNER     SEP 8TH 1981
```

Note the use of ‘fm’ to suppress the ‘0’ in the day portion of the day. After the use of ‘fm’, day is shown only as 8th.

TO_CHAR Functions with Numbers

Syntax

```
to_char(number,'format_mask')
```

Format_mask	Explanation
9	Represents a number
0	Forces a zero to be displayed
$	Displays a dollar symbol before the number
,	Thousand separator
.	Prints a decimal point

TO_CHAR Functions with Numbers: Example

```
SQL> SELECT ENAME, TO_CHAR(SAL, '$999,999,999.99') SALARY
  2  FROM EMP;

ENAME      SALARY
---------- ----------------
SMITH                $800.00
ALLEN              $1,600.00
WARD               $1,250.00
JONES              $2,975.00
MARTIN             $1,250.00
BLAKE              $2,850.00
CLARK              $2,450.00
SCOTT              $3,000.00
KING               $5,000.00
TURNER             $1,500.00
ADAMS              $1,100.00
JAMES                $950.00
FORD               $3,000.00
MILLER             $1,300.00
```

Displays a dollar symbol before the number

TO_DATE Functions

Syntax

```
to_date(string1, [ format_mask ])
```

- string1 will be converted to a date.
- format_mask is optional.

Example

```
SQL> SELECT TO_DATE('April 12 1991','Month DD YYYY')
  2  FROM dual;

TO_DATE('
---------
12-APR-91

SQL> SELECT TO_DATE('2006/07/09','yyyy/mm/dd')
  2  FROM dual;

TO_DATE('
---------
09-JUL-06

SQL> SELECT TO_DATE('02 09 06','DD MM YY')
  2  FROM dual;

TO_DATE('
---------
02-SEP-06
```

TO_NUMBER Functions

Syntax

```
to_number(string1, [ format_mask ])
```

```
SQL> SELECT TO_NUMBER('123,456','999999') converted_number
  2  FROM dual;

CONVERTED_NUMBER
----------------
          123456
```

Sub-queries

- Sub-queries are also called ‘nested queries,’ which means that one SELECT statement can be nested inside another.

They are placed in the WHERE clause of a SELECT statement.

They allow users to group multiple SELECT statements together.

There are 2 types of sub-queries:
- ⇨ **Independent/non-correlated sub-queries**
- ⇨ **Correlated sub-queries**

They can return either a single row or multiple rows.

Syntax

```
SELECT select_list          } Outer
FROM         table          } Query
WHERE expr[operator]
                    (SELECT select_list   } Inner
                     FROM table);         } Query
```

Example: Single Row Sub-queries

```
SQL> SELECT emp_id,first_na||','||last_na name, job_id, salary
  2  FROM   emp
  3  WHERE  job_id =
  4                  (SELECT job_id
  5                   FROM   emp
  6                   WHERE  emp_id = 34);

    EMP_ID NAME                                               JOB_ID         SALARY
---------- ---------------------------------------------- ---------- ----------
        10 John,King                                      CLERK
        40 kwalker,blewis                                 CLERK             800
       103 David,louis                                    CLERK            5500
        34 PHILIP,WATSON                                  CLERK            8000

SQL>
SQL> SELECT emp_id,first_na||','||last_na name, job_id, salary
  2  FROM   emp
  3  WHERE  salary >
  4                  (SELECT salary
  5                   FROM   emp
  6                   WHERE  emp_id = 34);

    EMP_ID NAME                                               JOB_ID         SALARY
---------- ---------------------------------------------- ---------- ----------
        50 john,smith                                     SALES_REP       19800
       104 steven,king                                    IT_PROG         10000
       102 ALEXANDER,KING                                 IT_PROG         22000
        55 CARL,HARY                                      SALES_REP       12000
```

Sub-query: Examples

- The inner query executes once, which returns the salary.
  - Salary is used in the outer query.

Example

```
SQL> SELECT EMPNO, ENAME, SAL
  2       FROM EMP    WHERE SAL >
  3            (
  4              SELECT SAL
  5               FROM EMP
  6               WHERE EMPNO=7369 );

     EMPNO ENAME                SAL
---------- ---------- ----------
      7499 ALLEN                1600
      7521 WARD                 1250
      7566 JONES                2975
      7654 MARTIN               1250
      7698 BLAKE                2850
      7782 CLARK                2450
      7788 SCOTT                3000
      7839 KING                 5000
      7844 TURNER               1500
      7876 ADAMS                1100
      7900 JAMES                 950

     EMPNO ENAME                SAL
---------- ---------- ----------
      7902 FORD                 3000
      7934 MILLER               1300

13 rows selected.
```

Independent/Non-Correlated Sub-queries

- A sub-query is also called an inner query.
- The query that is placed before the inner query is called the ‘outer query’ or ‘parent query.’
- In independent sub-queries:
  - The inner query is executed independent of the outer query.
  - The inner query executes only once.
  - The inner query is executed first and the results replace the inner SELECT statement.
  - The outer query executes based on the results provided by the inner query.

Operators Used in Sub-queries

Operator	Description
Comparison Operator	(=,<>,<,>,<=,>=). Used when a subquery returns a single value.
IN	Used to select rows that match the value in a list; usually when a subquery returns multiple rows.
ANY or SOME ALL	Compares a value to each value returned by a subquery, or all values returned by the subquery. Used when a subquery returns multiple rows along with comparison operators.
EXISTS	Always returns data in terms of True or False values. Used with co-related sub-queries.

Non-Correlated Sub-queries (1 of 3)

- Sub-queries returning a single row.

Example

```
SELECT  ENAME FROM EMP WHERE  SAL
< (SELECT  AVG(SAL) FROM EMP);
```

To display the names of employees whose salary is less than the average.

Sample Output

```
ENAME
=====
SMITH
ALLEN
WARD
MARTIN
TURNER
ADAMS
JAMES
MILLER
```

Non-Correlated Sub-queries (2 of 3)

- Sub-queries returning multiple rows.

Example

```
SELECT ENAME,JOB,SAL FROM EMP WHERE
DEPTNO IN (SELECT  DEPTNO FROM DEPT
WHERE  DNAME IN
('ACCOUNTING','SALES'))
```

To display the names of the employees working in departments ACCOUNTING and SALES.

Sample Output

ENAME	JOB	SAL
ALLEN	SALESMAN	1600
WARD	SALESMAN	1250
MARTIN	SALESMAN	1250
BLAKE	MANAGER	2850
CLARK	MANAGER	2450
KING	PRESIDENT	5000
TURNER	SALESMAN	1500
JAMES	CLERK	950
MILLER	CLERK	1300

Non-Correlated Sub-queries (3 of 3)

- Sub-queries with ANY/ALL operators.

Example

```
SELECT ENAME,JOB, SAL FROM EMP
WHERE SAL <ALL
(SELECT AVG(SAL)FROM EMP GROUP
BY JOB)
```

Sample Output

```
ENAME      JOB          SAL
=======================
SMITH      CLERK        800
JAMES      CLERK        950
```

To find the names of employees whose salaries are less than the average for each job description.

Example

```
SELECT ENAME,JOB, SAL FROM EMP
WHERE SAL <ANY
(SELECT AVG(SAL)FROM EMP GROUP
BY JOB)
```

Sample Output

```
ENAME      JOB          SAL
=========================
SMITH      CLERK        800
ALLEN      SALESMAN     1600
……..       ……..         ……..
```

To find the names of the employees who receive salary less than either of average salary for each type of job.

Correlated Sub-queries

- A correlated sub-query references the outer query.
- For correlated sub-queries:
  - The inner query is dependent upon the outer query.
  - The inner query is executed as many times as the outer query.

Correlated Sub-query: Example

Example

```
SELECT  EMPNO,ENAME,SAL,DEPTNO FROM EMP E1 WHERE SAL
>(SELECT AVG(SAL) FROM EMP E2
WHERE E1.DEPTNO=E2.DEPTNO);
```

To get the employee details of those employees whose salaries are greater than the average within their department.

Sample Output

EMPNO	ENAME	SAL	DEPTNO
7499	ALLEN	1600	30
7566	JONES	2975	20
7698	BLAKE	2850	30
7788	SCOTT	3000	20
7839	KING	5000	10
7902	FORD	3000	20

Correlated Sub-query: Example (cont.)

Example

```
SELECT ENAME FROM EMP E1
WHERE MGR =(SELECT
EMPNO FROM EMP E2
WHERE E1.MGR=E2.EMPNO);
```

To get the names of employees who have a reporting authority.

Sample Output

```
ENAME
========
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
TURNER
ADAMS
JAMES
FORD
MILLER
```

Using EXISTS

- The EXISTS clause checks for the existence of rows and does not compare columns or its values.
- It is used to check the existence of data rows according to the condition specified in the inner query, and passes the existence status to the outer query to produce the result set.
- The EXISTS clause:
  - Returns data in terms of a TRUE or FALSE value.
  - Can be used with the NOT operator.
- The inner query need not specify any columns in the SELECT statement.

Using EXISTS: Example

Example

```
SELECT ENAME FROM EMP E1 WHERE EXISTS
(SELECT * FROM  EMP E2 WHERE  E1.MGR=E2.EMPNO
AND E2.ENAME='KING');
```

To list the employee details who directly report to manager 'KING'.

Sample Output

```
ENAME
=======
JONES
BLAKE
CLARK
```

Using EXISTS Operator

- An EXISTS condition:
  - Tests for existence of rows in a subquery.
  - Is considered ‘to be met’ if the subquery returns at least one row.
- The EXISTS operator checks if the inner query returns any rows.
  - If it does, then the outer query is processed.
  - If it does not, the outer query does not execute and the entire SQL statement returns nothing.

Using EXISTS Operator: Example

Example

```
SQL> SELECT EMPNO, ENAME, DEPTNO
  2  FROM EMP E
  3  WHERE EXISTS
  4                (
  5                 SELECT * FROM DEPT D
  6                 WHERE E.DEPTNO=D.DEPTNO
  7                 );

     EMPNO ENAME           DEPTNO
---------- ---------- ----------
      7369 SMITH              20
      7499 ALLEN              30
      7521 WARD               30
      7566 JONES              20
      7654 MARTIN             30
      7698 BLAKE              30
      7782 CLARK              10
      7788 SCOTT              20
      7839 KING               10
      7844 TURNER             30
      7876 ADAMS              20

     EMPNO ENAME           DEPTNO
---------- ---------- ----------
      7900 JAMES              30
      7902 FORD               20
      7934 MILLER             10

14 rows selected.

SQL>
```

This subquery looks for the values in the department table.

Here, the values are returned. As a result, the outer query executes.

NOT EXISTS Operator

- The EXISTS condition can also be combined with the NOT operator.
- The NOT EXISTS operator returns Boolean value.

Example

```
SELECT E.EMPNO, E.ENAME, E.DEPTNO
FROM EMP E
WHERE NOT EXISTS
        (
         SELECT * FROM DEPT D
         WHERE E.DEPTNO=D.DEPTNO
         );
```

JOINS

- Joins are normally used to retrieve data from more than one table.
- The keyword JOIN, joins one or more tables together in a results set.
- To perform a join, there must be a common key that defines how the rows in the table correspond to each other.
- All sub-queries can be converted to joins; however, all joins cannot be converted to sub-queries.

Cartesian Join

- A Cartesian Join is also known as Cross Join.
- If there are two tables, then the Cartesian Join is obtained when every row of one table is joined to a row in another table.
- Example:

```
SELECT * FROM a, b;
```

Example: Usage of Cartesian Join

```
SQL> select * from emp,dept;

    EMP_ID FIRST_NA             LAST_NA                   EMAIL                     PHONE_NO
---------- -------------------- ------------------------- ------------------------- ---------------
        80 dfg                  ghgj                                                778
        40 kwalker              blewis                    kblewis                   6000
        50 john                 simith                    jsmith                    3245689
       103 David                louis                     dlouis                    515.216.5678
       104 steven               king                      sking                     33258791
      1060 pad                  ran                       p.r                       999
       100 TINA                 RAJ                       TRAJ                      515.789.2006
       101 ABHEY                KELKAR                    AKELHAR                   515.123.6789
       102 ALEXANDER            KING                      AKING                     650.121.4567
        55 CARL                 HARY                      CHARY                     209.789.3675
        34                      WATSON                    PWATSON

    EMP_ID FIRST_NA             LAST_NA                   EMAIL                     PHONE_NO
---------- -------------------- ------------------------- ------------------------- ---------------
        80 dfg                  ghgj                                                778
        40 kwalker              blewis                    kblewis                   6000
        50 john                 simith                    jsmith                    3245689
       103 David                louis                     dlouis                    515.216.5678
       104 steven               king                      sking                     33258791
      1060 pad                  ran                       p.r                       999
       100 TINA                 RAJ                       TRAJ                      515.789.2006
       101 ABHEY                KELKAR                    AKELHAR                   515.123.6789
       102 ALEXANDER            KING                      AKING                     650.121.4567
        55 CARL                 HARY                      CHARY                     209.789.3675
        34                      WATSON                    PWATSON
121 rows selected.

SQL>
```

Using Aliases

- Aliases are of two types:
  - Column aliases
  - Table aliases
- By using aliases, table prefixes can be used to:
  - Qualify column names that are in multiple tables.
  - Improve performance.
  - Distinguish columns that have identical names, but reside in different tables.
    - By using column aliases

```
SELECT emp.emp_id employee_number,
       emp.first_na first_name,
       emp.first_na last_name,
FROM emp e,dept d
WHERE e.dept_id = d.dept_id
AND e.dept_id=10;
```

'Employee_number', 'first_name', and 'last_name' are known as Column Aliases.

'e' and 'd' are Table Aliases.

Types of Joins

- The different types of joins are:
  - Inner Join
  - Outer Join
    - Left Outer Join
    - Right Outer Join
    - Full Outer Join
  - Self Join

Inner Join (Equijoin)

- The Inner Join joins two or more tables, returning only matched rows.
- It requires that both tables involved in the join must have a primary-foreign key relationship.

Inner Join: Example

- In this example:
  - emp_id is the primary key in the emp table.
  - loc_id is the primary key in the office_loc table.
  - emp_id in office_loc table is used to refer to people in the 'Employee' table.

Tables used in this example

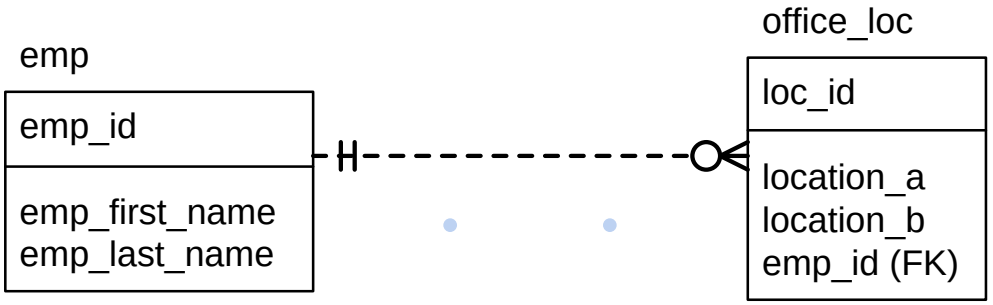
Employee Table

Emp_id	EName
10001	Ram
10002	Nitin
10003	Siddharth
10004	Vivek

Office_loc table

Loc_id	Location	Emp_id
25	Mumbai	10001
30	Pune	10002
35	Hyderabad	10003
40	Delhi	10004

```
SELECT a.emp_id,
a.ename,b.loc_id,b.location
FROM employee a, office_loc b
WHERE a.emp_id = b.emp_id;
```



Additional Search Conditions: AND Operator

The AND operator tells Oracle to return only those values that meet both conditions – before and after AND. It adds further conditions / constraints to the query. An example is provided below.

Example

```
SELECT ENAME,
JOB,SAL, DNAME
FROM DEPT D,EMP E
WHERE
D.DEPTNO=E.DEPTNO
and E.DEPTNO=10
```

To search for details of employees who belong to dept id = 10;

Outer Join

- The SQL OUTER JOIN command is used to display the elements in a table, regardless of whether they are present in the second table. The OUTER JOIN:
  - Returns not only the common rows from the tables, but also returns the rows that are unique within the tables.
  - Returns rows even if the rows from one table are not matching with those of another table.
- Syntactically, place ‘(+)’ in the WHERE clause, on the other side of the table, for which all the rows need to be included.
- There are three types of Outer Joins:
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join

Left Outer Join

- In the Left Outer Join, all the records from the table on the left of the OUTER JOIN statement are returned.

Syntax

```
SELECT column_list FROM left_table LEFT [OUTER]
JOIN right_table ON condition
```

Syntax

```
SELECT column_list FROM table t1,table t2 where
t1.column_name=t2.column_name (+)
```

Left Outer Join: Example

Example

To display the ENAME, JOB, SAL, and DNAME in which the employees are working along with the DNAME in which no employees are working.

```
SELECT ENAME,
JOB,SAL, DNAME
FROM DEPT D
LEFT JOIN  EMP E
ON D.DEPTNO =
E.DEPTNO;
```

Sample Output

ENAME	JOB	SAL	DNAME
SMITH	CLERK	800	RESEARCH
ALLEN	SALESMAN	1600	SALES
WARD	SALESMAN	1250	SALES
JONES	MANAGER	2975	RESEARCH
MARTIN	SALESMAN	1250	SALES
BLAKE	MANAGER	2850	SALES
CLARK	MANAGER	2450	ACCOUNTING
SCOTT	ANALYST	3000	RESEARCH
KING	PRESIDENT	5000	ACCOUNTING
TURNER	SALESMAN	1500	SALES
ADAMS	CLERK	1100	RESEARCH
JAMES	CLERK	950	SALES
FORD	ANALYST	3000	RESEARCH
MILLER	CLERK	1300	ACCOUNTING

Right Outer Join

- In a Right Outer Join, all the records from the table on the right of the OUTER JOIN are returned.

Syntax

```
SELECT column_list FROM left_table RIGHT [OUTER] JOIN right_table
ON condition
```

Syntax

```
SELECT column_list FROM table t1,table t2 where t1.coluumn_name
(+) =t2.column_name
```

Right Outer Join: Example

Example

To display the ENAME, JOB, SAL, and DNAME in which the employees are working along with the DNAME in which no employees are working.

```
SELECT ENAME,
JOB,SAL, DNAME
FROM EMP E
RIGHT JOIN  DEPT D
ON E.DEPTNO =
D.DEPTNO;
```

Sample Output

ENAME	JOB	SAL	DNAME
SMITH	CLERK	800	RESEARCH
ALLEN	SALESMAN	1600	SALES
WARD	SALESMAN	1250	SALES
JONES	MANAGER	2975	RESEARCH
MARTIN	SALESMAN	1250	SALES
BLAKE	MANAGER	2850	SALES
CLARK	MANAGER	2450	ACCOUNTING
SCOTT	ANALYST	3000	RESEARCH
KING	PRESIDENT	5000	ACCOUNTING
TURNER	SALESMAN	1500	SALES
ADAMS	CLERK	1100	RESEARCH
JAMES	CLERK	950	SALES
FORD	ANALYST	3000	RESEARCH
MILLER	CLERK	1300	ACCOUNTING
			OPERATIONS

Full Outer Join

- A Full Outer Join is essentially a combination of Left and Right outer joins, i.e.:
  - The records from the table on left are included even if there are no matching records on the right.
  - The records from the table on the right are included even if there are no matching records on the left.

Full Outer Join: Example

Example

```
SELECT ENAME,
JOB,SAL, DNAME
FROM EMP E FULL
JOIN  DEPT D
ON E.DEPTNO =
D.DEPTNO;
```

Sample Output

```
ENAME          JOB       SAL       DNAME
=====================================
SMITH          CLERK      800    RESEARCH
ALLEN      SALESMAN      1600       SALES
WARD       SALESMAN      1250       SALES
JONES       MANAGER      2975    RESEARCH
MARTIN     SALESMAN      1250       SALES
BLAKE       MANAGER      2850       SALES
CLARK       MANAGER      2450  ACCOUNTING
SCOTT       ANALYST      3000    RESEARCH
KING      PRESIDENT      5000  ACCOUNTING
TURNER     SALESMAN      1500       SALES
ADAMS          CLERK      1100    RESEARCH
JAMES          CLERK       950       SALES
FORD         ANALYST      3000    RESEARCH
MILLER         CLERK      1300  ACCOUNTING
                                 OPERATIONS
```

Self Join

- A Self Join is a query in which a table is joined (compared) to itself.
- It is used to compare values in a column, with other values in the same column in the same table.
- Self Joins are also very useful in conjunction with sub-queries.

Self Join: Example

Example

To find out the manager names for each of the employee.

```
SELECT E1.ENAME||' REPORTS
TO '|| E2.ENAME AS REPORTS
FROM EMP E1, EMP  E2 WHERE
E1.MGR=E2.EMPNO;
```

Sample Output

```
REPORTS
==============
SMITH REPORTS TO FORD
ALLEN REPORTS TO BLAKE
WARD REPORTS TO BLAKE
JONES REPORTS TO KING
MARTIN REPORTS TO BLAKE
BLAKE REPORTS TO KING
CLARK REPORTS TO KING
SCOTT REPORTS TO JONES
TURNER REPORTS TO BLAKE
ADAMS REPORTS TO SCOTT
JAMES REPORTS TO BLAKE
FORD REPORTS TO JONES
MILLER REPORTS TO CLARK
```


Thank you!