**SET OPERATORS – good notes**

With 2 tables and examples.

Run this query first to build and load 2 tables.

DROP TABLE emp PURGE;

DROP TABLE dept PURGE;

CREATE TABLE dept (

department\_id NUMBER(2)

CONSTRAINT departments\_pk PRIMARY KEY,

department\_name VARCHAR2(14),

location VARCHAR2(13)

);

INSERT INTO dept VALUES (10,'ACCOUNTING','NEW YORK');

INSERT INTO dept VALUES (20,'RESEARCH','DALLAS');

INSERT INTO dept VALUES (30,'SALES','CHICAGO');

INSERT INTO dept VALUES (40,'OPERATIONS','BOSTON');

COMMIT;

CREATE TABLE emp (

employee\_id NUMBER(4)

CONSTRAINT employees\_pk PRIMARY KEY,

employee\_name VARCHAR2(10),

job VARCHAR2(9),

manager\_id NUMBER(4),

hiredate DATE,

salary NUMBER(7,2),

commission NUMBER(7,2),

department\_id NUMBER(2)

CONSTRAINT emp\_department\_id\_fk REFERENCES dept(department\_id)

);

INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,to\_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);

INSERT INTO emp VALUES (7499,'ALLEN','SALESMAN',7698,to\_date('20-2-1981','dd-mm-yyyy'),1600,300,30);

INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,to\_date('22-2-1981','dd-mm-yyyy'),1250,500,30);

INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,to\_date('2-4-1981','dd-mm-yyyy'),2975,NULL,20);

INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,to\_date('28-9-1981','dd-mm-yyyy'),1250,1400,30);

INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,to\_date('1-5-1981','dd-mm-yyyy'),2850,NULL,30);

INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,to\_date('9-6-1981','dd-mm-yyyy'),2450,NULL,10);

INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,to\_date('13-JUL-87','dd-mm-rr')-85,3000,NULL,20);

INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,to\_date('17-11-1981','dd-mm-yyyy'),5000,NULL,10);

INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,to\_date('8-9-1981','dd-mm-yyyy'),1500,0,30);

INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,to\_date('13-JUL-87', 'dd-mm-rr')-51,1100,NULL,20);

INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,to\_date('3-12-1981','dd-mm-yyyy'),950,NULL,30);

INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,to\_date('3-12-1981','dd-mm-yyyy'),3000,NULL,20);

INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,to\_date('23-1-1982','dd-mm-yyyy'),1300,NULL,10);

COMMIT;

NOTES

Here are the starting queries that will be used

-- Department 10, 20 and 30.

SELECT department\_id, department\_name

FROM dept

WHERE department\_id <= 30;

DEPARTMENT\_ID DEPARTMENT\_NAM

------------- --------------

10 ACCOUNTING

20 RESEARCH

30 SALES

3 rows selected.

--Department 20, 30 and 40.

SELECT department\_id, department\_name

FROM dept

WHERE department\_id >= 20;

DEPARTMENT\_ID DEPARTMENT\_NAM

------------- --------------

20 RESEARCH

30 SALES

40 OPERATIONS

3 rows selected.

## UNION

The UNION set operator returns all distinct rows selected by either query. That means any duplicate rows will be removed.

Shape

Description automatically generated

Look at the results above you can see that department 20 and 30 are in both sets of output. That mean the UNION will only show it once. 🡺 10, 20 , 30 40

In the example below, notice there is only a single row each for departments 20 and 30, rather than two each.

**SELECT department\_id, department\_name**

**FROM dept**

**WHERE department\_id <= 30**

**UNION**

**SELECT department\_id, department\_name**

**FROM dept**

**WHERE department\_id >= 20**

**ORDER BY 1;**

DEPARTMENT\_ID DEPARTMENT\_NAM

------------- --------------

10 ACCOUNTING

20 RESEARCH

30 SALES

40 OPERATIONS

The removal of duplicates requires extra processing, so you might want to consider using UNION ALL if possible.

## UNION ALL

The UNION ALL set operator returns all rows selected by either query. That means any duplicates will remain in the result set.

A picture containing chart

Description automatically generated

In the example below, notice there are two rows each for departments 20 and 30.

**SELECT department\_id, department\_name**

**FROM dept**

**WHERE department\_id <= 30**

**UNION ALL**

**SELECT department\_id, department\_name**

**FROM dept**

**WHERE department\_id >= 20**

**ORDER BY 1;**

DEPARTMENT\_ID DEPARTMENT\_NAM

------------- --------------

10 ACCOUNTING

**20 RESEARCH**

**20 RESEARCH**

**30 SALES**

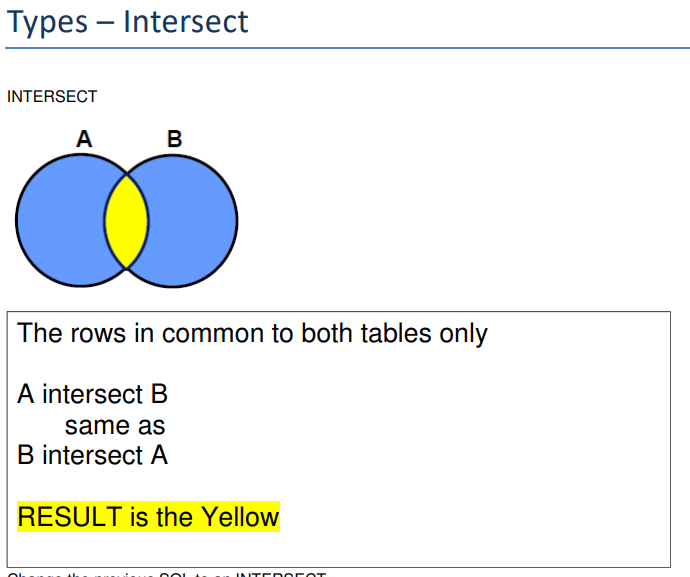
**30 SALES**

40 OPERATIONS

6 rows selected.

INTERSECT

The INTERSECT set operator returns all distinct rows selected by both queries. That means only those rows common to both queries will be present in the result set.



In the example below, notice there is one row each for departments 20 and 30, as both these appear in the result sets for their respective queries.

**SELECT department\_id, department\_name**

**FROM dept**

**WHERE department\_id <= 30**

**INTERSECT**

**SELECT department\_id, department\_name**

**FROM dept**

**WHERE department\_id >= 20**

**ORDER BY 1;**

DEPARTMENT\_ID DEPARTMENT\_NAM

------------- --------------

20 RESEARCH

30 SALES

2 rows selected.

## MINUS

The MINUS set operator returns all distinct rows selected by the first query but not the second. This is functionally equivalent to the ANSI set operator EXCEPT DISTINCT.

Shape

Description automatically generated

In the example below, the first query would return departments 10, 20, 30, but departments 20 and 30 are removed because they are returned by the second query. This leaves a single row for department 10.

**SELECT department\_id, department\_name**

**FROM dept**

**WHERE department\_id <= 30**

**MINUS**

**SELECT department\_id, department\_name**

**FROM dept**

**WHERE department\_id >= 20**

**ORDER BY 1;**

DEPARTMENT\_ID DEPARTMENT\_NAM

------------- --------------

10 ACCOUNTING

1 row selected.

## ORDER BY

The ORDER BY clause is applied to all rows returned in the final result set. Columns in the ORDER BY clause can be referenced by column names or column aliases present in the first query of the statement, as these carry through to the result set.

Typically, you will see people use the **column position** as it is less confusing when the data is sourced from different locations for each query block.

-- Using Column name.

SELECT employee\_id, employee\_name

FROM emp

WHERE department\_id = 10

UNION ALL

SELECT department\_id, department\_name

FROM dept

WHERE department\_id >= 20

ORDER BY employee\_id;

EMPLOYEE\_ID EMPLOYEE\_NAME

----------- --------------

20 RESEARCH

30 SALES

40 OPERATIONS

7782 CLARK

7839 KING

7934 MILLER

6 rows selected.

-- Using Column Alias

SELECT employee\_id AS emp\_id, employee\_name

FROM emp

WHERE department\_id = 10

UNION ALL

SELECT department\_id, department\_name

FROM dept

WHERE department\_id >= 20

ORDER BY emp\_id;

EMP\_ID EMPLOYEE\_NAME

---------- --------------

20 RESEARCH

30 SALES

40 OPERATIONS

7782 CLARK

7839 KING

7934 MILLER

6 rows selected.

-- Using Column position

SELECT employee\_id, employee\_name

FROM emp

WHERE department\_id = 10

UNION ALL

SELECT department\_id, department\_name

FROM dept

WHERE department\_id >= 20

ORDER BY 1;

EMPLOYEE\_ID EMPLOYEE\_NAME

----------- --------------

20 RESEARCH

30 SALES

40 OPERATIONS

7782 CLARK

7839 KING

7934 MILLER

6 rows selected.