JOINS: SQL joins are used to query data from two or more tables, based on a relationship between certain columns in these tables.

Syntax:

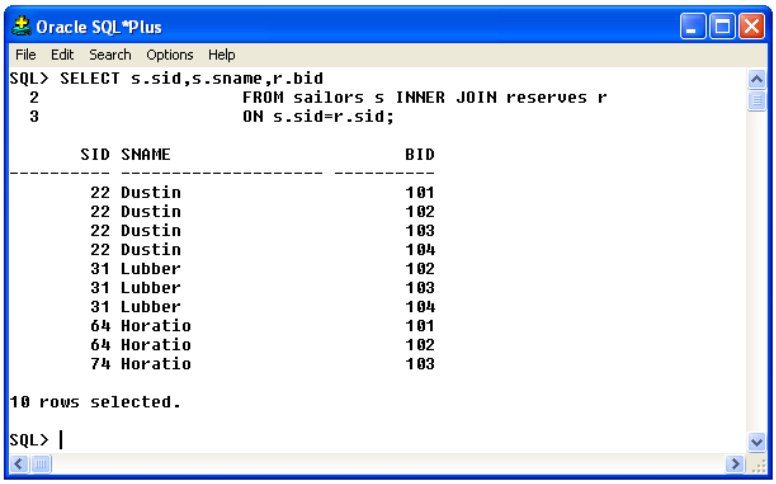
SELECT column_list
FROM table_name
[INNER|{LEFT|RIGHT|FULL OUTER}] JOIN table_name
ON qualification_list
WHERE <qualifier>;

INNER JOINS: Only rows that match the search conditions are returned.

Example:
SELECT s.sid,s.sname,r.bid
FROM sailors s INNER JOIN reserves r
ON s.sid=r.sid;

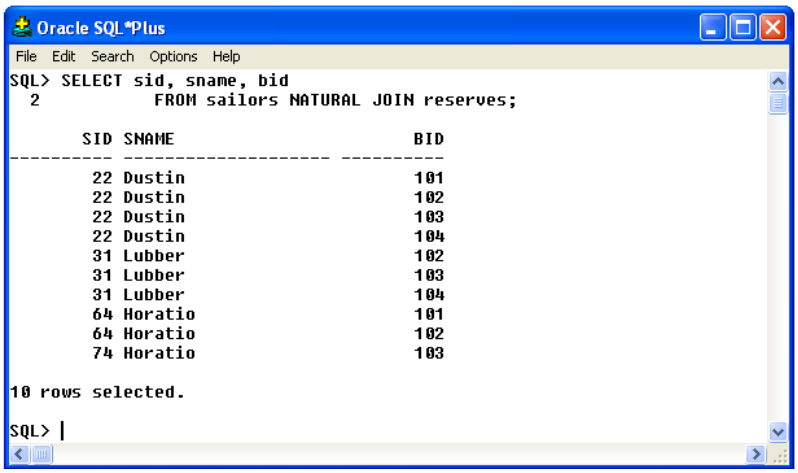
(OR)

SELECT s.sid, s.sname ,r.bid
FROM sailors s, reserves r
WHERE s.sid=r.sid;



NATURAL JOIN
NATURAL means equi-join for each pair of attributes with the same name.

Example:
SELECT sid, sname, bid
FROM sailors NATURAL JOIN reserves;



LEFT OUTER JOIN
Returns all matched rows, plus all unmatched rows from the table on the left of the join clause. Displays nulls in fields of non-matching tuples.

Example:
SELECT s.sid, s.sname ,r.bid
FROM sailors s LEFT OUTER JOIN reserves r
ON s.sid=r.sid;

(OR)

SELECT s.sid, s.sname ,r.bid
FROM sailors s, reserves r
WHERE s.sid=r.sid(+);

Also returns details of sailors who didn't reserve a boat.

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> SELECT s.sid, s.sname ,r.bid
  2             FROM sailors s LEFT OUTER JOIN reserves r
  3             ON s.sid=r.sid;

       SID SNAME                       BID
---------- -------------------- ----------
        22 Dustin                      101
        22 Dustin                      102
        22 Dustin                      103
        22 Dustin                      104
        31 Lubber                      102
        31 Lubber                      103
        31 Lubber                      104
        64 Horatio                     101
        64 Horatio                     102
        74 Horatio                     103
        71 Zorba

       SID SNAME                       BID
---------- -------------------- ----------
        85 Art
        58 Rusty
        32 Andy
        29 Brutus
        95 Bob

16 rows selected.

SQL>
```

RIGHT OUTER JOIN

Returns all matched rows, plus all unmatched rows from the table on the right of the join clause. Displays nulls in fields of non-matching tuples.

Example:

SELECT r.sid,b.bid,b.bname
FROM reserves r RIGHT OUTER JOIN boats b
ON r.bid=b.bid;

(OR)

SELECT r.sid,b.bid,b.bname
FROM reserves r,boats b
WHERE r.bid(+)=b.bid;

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> SELECT r.sid,b.bid,b.bname
  2    FROM reserves r,boats b
  3    WHERE  r.bid(+)=b.bid;

       SID        BID BNAME
---------- ---------- --------------------
        22        101 Interlake
        22        102 Interlake
        22        103 Clipper
        22        104 Marine
        31        102 Interlake
        31        103 Clipper
        31        104 Marine
        64        101 Interlake
        64        102 Interlake
        74        103 Clipper

10 rows selected.

SQL>
```

Also returns details of boats which don't have any reservations

FULL OUTER JOIN

Returns all matched or unmatched rows from the tables on both sides of the join clause

Example:

SELECT r.sid,b.bid,b.bname
FROM reserves r FULL OUTER JOIN boats b
ON r.bid=b.bid;

Note: In this case it is the same as the right outer join because bid is a foreign key in reserves, so all reservations must have a corresponding tuple in boats.

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> SELECT r.sid,b.bid,b.bname
  2    FROM reserves r FULL OUTER JOIN boats b
  3    ON r.bid=b.bid;

       SID        BID BNAME
---------- ---------- --------------------
        22        101 Interlake
        22        102 Interlake
        22        103 Clipper
        22        104 Marine
        31        102 Interlake
        31        103 Clipper
        31        104 Marine
        64        101 Interlake
        64        102 Interlake
        74        103 Clipper

10 rows selected.

SQL> |
```

Exercise for today

1. Find the names of Sailors who have reserved boat number 103
2. Find the names of the Sailors who have reserved a red boat
3. Find the names and colors of the boats reserved by the sailor named Lubber
4. Find the Sailors who have reserved at least one boat
5. Compute Increments for the ratings of sailors who have sailed two different boats on the same day.
6. Find the sailors who have reserved both a red and a green boat.

SQL OPERATORS

Arithmetic: + , - , * , /
Logical: AND, OR, NOT
Relational: = , != , < , > , <= , >= ; <> , ^= (same as !=)
Set operators: UNION [ALL],
INTERSECT,
MINUS
SQL Operators:
BETWEEN,
IN,
LIKE,
IS NULL,
UNIQUE,
EXISTS,
ANY,
SOME,
ALL

Note: ALL and ANY should be preceded by a relational operator

1) SET Operators are used to combine information of similar type from one or more tables. They combine two or more queries into one result.
2) Number of columns should match in every select.
3) Order of data-types of columns should match in every select statement.
4) Can use any clause in any select statement but order by should be used at the end only.
Automatically sorts by the first column

- **UNION :** Rows of first query plus rows of second query, does not include the duplicate rows.
- **UNION ALL:** Rows of first query plus rows of second query, includes the duplicate rows.

Syntax:

```
SELECT <stmt-2>
UNION [ALL]
SELECT <stmt-2>
[ORDER BY clause]
```

Examples:

SELECT sname, age FROM sailors
WHERE rating=7
UNION
SELECT sname, age FROM sailors
WHERE rating=10;

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> SELECT sname, age FROM sailors WHERE rating=7
  2  UNION
  3  SELECT sname, age FROM sailors WHERE rating=10;

SNAME                       AGE
-------------------- ----------
Dustin                       45
Horatio                      35
Rusty                        35
Zorba                        16

SQL> |
```

SELECT bid FROM boats
UNION ALL
SELECT bid FROM reserves;

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> SELECT bid FROM boats
  2  UNION ALL
  3  SELECT bid FROM reserves;

       BID
----------
       101
       102
       103
       104
       101
       102
       103
       104
       102
       103
       104

       BID
----------
       101
       102
       103

14 rows selected.

SQL> |
```

INTERSECT : returns common rows between two sets of rows

Syntax:
SELECT <stmt-2>
INTERSECT
SELECT <stmt-2>
[ORDER BY clause>

Example:

SELECT bname FROM boats
WHERE color='red'
INTERSECT
SELECT bname FROM boats WHERE color='blue';

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> SELECT bname FROM boats WHERE color='red'
  2  INTERSECT
  3  SELECT bname FROM boats WHERE color='blue';

BNAME
--------------------
Interlake

SQL>
```

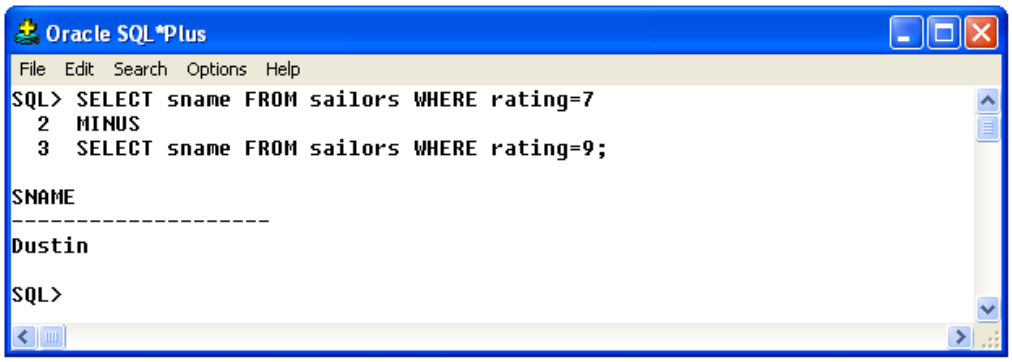
MINUS : returns the rows unique to the first query

Syntax:

SELECT <stmt-2>
MINUS
SELECT <stmt-2>

Example:

SELECT sname FROM sailors
WHERE rating=7
MINUS
SELECT sname FROM sailors
WHERE rating=9;



SQL OPERATORS

1) BETWEEN- The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value. (including the boundary values)

Example: SELECT * FROM sailors WHERE age BETWEEN 25 AND 35;

2) IN- The IN operator is used to compare a value to a list of literal values that have been specified.

Example: SELECT * FROM sailors WHERE rating IN (7,10);

3) LIKE-The LIKE operator is used to compare a value to similar values using wildcard operators.

4) IS NULL - The IS NULL operator is used to compare a value with a NULL value (= NULL is not valid)

Example: UPDATE sailors SET age=NULL WHERE rating=7;
SELECT * FROM sailors WHERE age IS NULL;

5) UNIQUE- The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates) //works like DISTINCT
Example: SELECT UNIQUE rating FROM sailors;

6) EXISTS –
- This operator is used to search for the presence of a row in a specified table that meets certain criteria.
- It produces a Boolean result
- It takes a sub-query as an argument and evaluates it to True if the sub-query produces any **output and False, if the sub-query does not produce any output**

Set Comparison Operators
ANY/SOME, ALL Operators –
- Used along with the relational operators
- Similar to IN operator, but only used in sub-queries
- The SOME and ANY operators can be used interchangeably

ANY/SOME – compares the lowest value from the set
ALL – the predicate is true if every value selected by the sub-query satisfies the condition in the **predicate of the outer query.**

	ANY(SOME)	**ALL**
>	More than minimum	More than maximum
<	**Less than maximum**	**Less than minimum**

Examples:
1) Find the names and ratings of sailor whose rating is better than some Sailor called Horatio.
Query:
SELECT S.sname, S.rating
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
FROM Sailors S2
WHERE S2.sname = 'Horatio');

2) Find Sailors whose rating is better than every Sailor called Horatio.
Query:
SELECT S.sname, S.rating
FROM Sailors S
WHERE S.rating > ALL (SELECT S2.rating
FROM Sailors S2
WHERE S2.sname = 'Horatio');

Note that IN and NOT IN are equivalent to = ANY and < > ALL, respectively.

Sub-Query/ Inner Query:
1) A nested query is a query that has another query embedded within it; the embedded query is called a sub-query.

2) Subquery may appear in FROM, WHERE and HAVING clause.

Types of Sub-queries:

1. Single row sub-query: always returns a single value
Sub-query executes only once
Uses operators = , > , < , >= , <= , !=

2. Multiple row sub-query: returns more than one value
Sub-query executes only once
Uses operators :
IN and NOT IN
EXISTS and NOT EXISTS
UNIQUE and NOT UNIQUE
op ANY
op ALL

3. Correlated sub-query: Sub-query executes repeatedly
Uses any of the operators

Note: When the condition includes one column from inner query and one from outer query then it is a correlated sub-query.

SET-COMPARISON OPERATORS:

Op ANY and op ALL, where op is one of the comparison operators **{<, <=, >, >=, =, < >}**

EXAMPLES

1. Find the id, name and the age of the youngest sailor.
Query:
SELECT S.sid, S.sname, S.age
FROM Sailors S
WHERE S.age <= ALL (SELECT age
FROM Sailors);

2. Find the names and ratings of sailor whose rating is better than some sailor called Horatio.
Query:
SELECT S.sname, S.rating
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
FROM Sailors S2
WHERE S2.sname = 'Horatio');

Note that IN and NOT IN are equivalent to = ANY and < > ALL, respectively.

Exercise for today

1. Find the names of sailors who have reserved boat 103.
2.Find the names of the sailors who have reserved a red boat
3. Find the names of the sailors who have not reserved a red boat.
4. Find the names of Sailors who have reserved boat#103.
5. Find Sailors whose rating is better than every Sailor called Horatio.
6. Find the Sailor with the highest rating
7. Find the names of sailors who have reserved all boats.
Query:
SELECT S.sname
FROM Sailors S

```
WHERE NOT EXISTS ((SELECT B.bid
FROM Boats B)
MINUS
(SELECT R.bid
FROM Reserves R
WHERE R.sid = S.sid));
```

8.Find the average age of all sailors.

9. Find the average age of sailors with a rating of 10.

10. Find the name and the age of the oldest sailor.

11. Count the number of sailors.

12. Count the number of different sailor names.

13. Find the names of Sailors who are older than the oldest sailor with a rating of 10.