Lab – 07

**Sql Joins**

# Objective:

* Students will be able to retrieve data from multiple tables and display it as single view.

# Cartesian Product:

A Cartesian product is formed when:

* A join condition is omitted
* A join condition is invalid
* All rows in the first table are joined to all rows in the second table

To avoid a Cartesian product, always include a valid join condition in a WHERE clause.

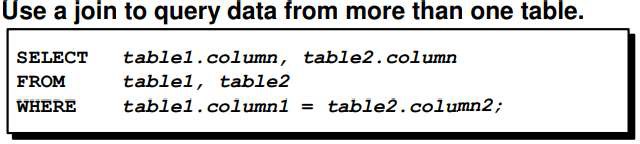
# Types of Joins:

Compliant Joins:

* Cross joins
* Natural joins
* Using clause
* Full or two sided outer joins
* Arbitrary join conditions for outer joins

Oracle Proprietary Joins:

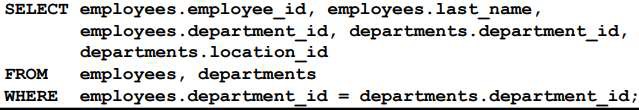
* Equi-join
* Non-equi-join
* Outer join
* Self-join



# Equi Joins:

To determine an employee’s department name, you compare the value in the DEPARTMENT\_ID column in the EMPLOYEES table with the DEPARTMENT\_ID values in the DEPARTMENTS table. The relationship between the EMPLOYEES and DEPARTMENTS tables is an equijoin, that is, values in the DEPARTMENT\_ID column on both tables must be equal. Frequently, this type of join involves primary and foreign key complements.

# Equijoins are also called simple joins or inner joins Example:



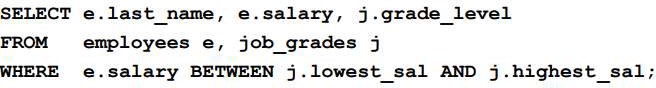
**Additional Search Conditions**

Sometimes you may need to join more than two tables. For example, to display the last name, the department name, and the city for each employee, you have to join the EMPLOYEES, DEPARTMENTS, and LOCATIONS tables.

SELECT e.last\_name, d.department\_name, l.city FROM employees e, departments d, locations l WHERE e.department\_id = d.department\_id AND d.location\_id = l.location\_id;

# Non-equi-joins

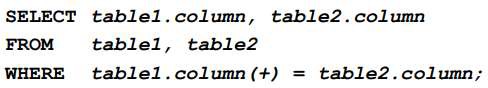
A nonequijoin is a join condition containing something other than an equality operator. The relationship between the EMPLOYEES table and the JOB\_GRADES table has an example of a nonequijoin. A relationship between the two tables is that the SALARY column in the EMPLOYEES table must be between the values in the LOWEST\_SALARY and HIGHEST\_SALARY columns of the JOB\_GRADES table. The relationship is obtained using an operator other than equals (=).



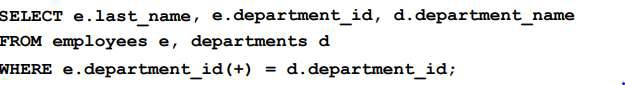
# Outer Join:

* You use an outer join to also see rows that do not meet the join condition.
* The outer join operator is the plus sign (+).

**Syntax:**



**Example:**



**Self-Join:**

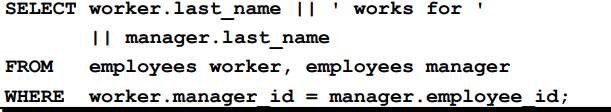
Joining a Table to Itself:

Sometimes you need to join a table to itself. To find the name of each employee’s manager, you need to join the EMPLOYEES table to itself, or perform a self-join. For example, to find the name of Whalen’s manager, you need to:

* Find Whalen in the EMPLOYEES table by looking at the LAST\_NAME column.
* Find the manager number for Whalen by looking at the MANAGER\_ID column. Whalen’s manager number is 101.
* Find the name of the manager with EMPLOYEE\_ID 101 by looking at the LAST\_NAME column. Kochhar’s employee number is 101, so Kochhar is Whalen’s manager.

In this process, you look in the table twice. The first time you look in the table to find Whalen in the LAST\_NAME column and MANAGER\_ID value of 101. The second time you look in the EMPLOYEE\_ID column to find 101 and the LAST\_NAME column to find Kochhar.

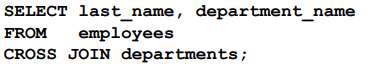
# Example



**Cross Join:**

* The CROSS JOIN clause produces the cross product of two tables.
* This is the same as a Cartesian product between the two tables.

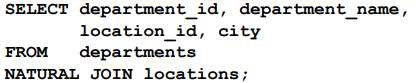
# Example:



**Natural Join:**

* The NATURAL JOIN clause is based on all columns in the two tables that have the same name.
* It selects rows from the two tables that have equal values in all matched columns.
* If the columns having the same names have different data types, then an error is returned.

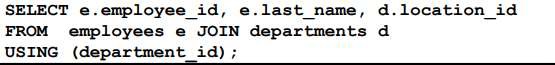
# Example:



**Creating Joins with the USING Clause:**

* If several columns have the same names but the data types do not match, the NATURAL JOIN clause can be modified with the USING clause to specify the columns that should be used for an equijoin. Note: Use the USING clause to match only one column when more than one column matches.
* Do not use a table name or alias in the referenced columns.
* The NATURAL JOIN and USING clauses are mutually exclusive.

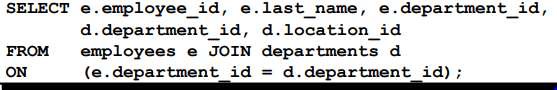
# Example:



**Creating Joins with the ON Clause**

* The join condition for the natural join is basically an equijoin of all columns with the same name.
* To specify arbitrary conditions or specify columns to join, the ON clause is used.
* Separates the join condition from other search conditions.
* The ON clause makes code easy to understand.

# Example:

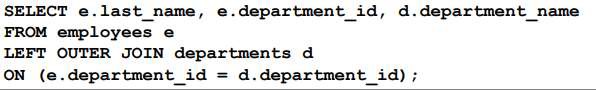


**INNER versus OUTER Joins**

* In SQL: 1999, the join of two tables returning only matched rows is an inner join.
* A join between two tables that returns the results of the inner join as well as unmatched rows left (or right) tables is a left (or right) outer join.
* A join between two tables that returns the results of an inner join as well as the results of a left and right join is a full outer join.

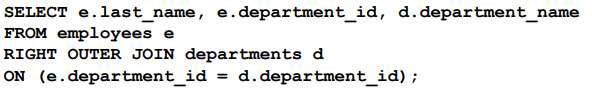
# Left Outer Join:

* This query retrieves all rows in the EMPLOYEES table, which is the left table even if there is no match in the DEPARTMENTS table.



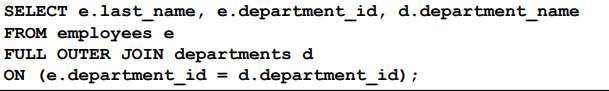
# Right Outer Join:

* This query retrieves all rows in the DEPARTMENTS table, which is the right table even if there is no match in the EMPLOYEES table.



# Full Outer Join:

* This query retrieves all rows in the EMPLOYEES table, even if there is no match in the DEPARTMENTS table. It also retrieves all rows in the DEPARTMENTS table, even if there is no match in the EMPLOYEES table.



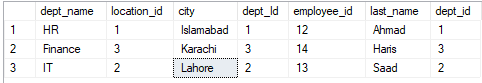
**Lab Tasks:**

* 1. Write a query to combine all rows of table EMP and table DEPT. Sort the columns in ascending order by employee names.

Code:

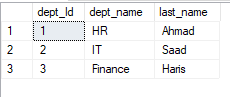
SELECT \* FROM Departments FULL OUTER JOIN Employee ON Departments.dept\_Id=Employee.dept\_id

ORDER BY Employee.last\_name



* 1. Write a query to display the name, department number, and department name for all employees.

SELECT Departments.dept\_Id,Departments.dept\_name,Employee.last\_name FROM Departments RIGHT JOIN Employee ON Departments.dept\_Id=Employee.dept\_id



* 1. Write a query for table EMP and DEPT to retrieve enames , sal, job, deptno, loc where length of job is greater than length of loc.

SELECT Employee.last\_name,Employee.salary,Employee.job,Departments.city,Departments.dept\_Id FROM Employee LEFT JOIN Departments ON Departments.dept\_Id=Employee.dept\_id WHERE LEN(job)>LEN(Departments.city)



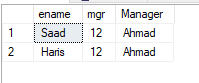
* 1. Write a query to retrieve ename, length of enames, loc , deptno and sal by join table EMP with table DEPT where deptno is 20 and sal is greater than 10000. Sort the sal in descending order

SELECT Employee.last\_name,LEN(Employee.last\_name) AS 'Length',Employee.salary,Employee.job,Departments.city,Departments.dept\_Id FROM Employee LEFT JOIN Departments ON Departments.dept\_Id=Employee.dept\_id WHERE Departments.dept\_Id = 3 AND Employee.salary > 10000 ORDER BY Employee.salary DESC



* 1. Write a query to join table EMP with itself where mgr exists for any employee.

SELECT A.last\_name AS ename,A.mgr,B.last\_name as Manager FROM Employee A Join Employee B ON A.mgr = B.employee\_id



* 1. Create a query that displays ename, deptno of those employees who work in the same department as a given employee. Give each column an appropriate Label.

SELECT e1.last\_name AS employee\_last\_name,

e1.dept\_id AS employee\_department

FROM employee e1

JOIN employee e2 ON e1.dept\_id = e2.dept\_id

WHERE e2.last\_name = 'Saaid';

A blue rectangular box with black text

Description automatically generated

* 1. Write a query to display the last name, department number, and department name for all employees.

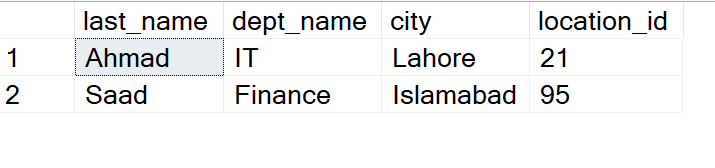
SELECT last\_name ,employee.dept\_id, dept\_name FROM employee JOIN departments ON employee.dept\_id = departments.dept\_id;

A white grid with black text

Description automatically generated

* 1. Write a query to display the employee last name, department name, location ID, and city of all employees who earn a commission.

SELECT last\_name , dept\_name,city,location\_id FROM employee JOIN departments ON employee.dept\_id = departments.dept\_id WHERE comission IS NOT NULL;



* 1. Display the employee last name and department name for all employees who have an a (lowercase) in their last names.

SELECT last\_name , dept\_name FROM employee JOIN departments ON employee.dept\_id = departments.dept\_id WHERE last\_name LIKE '%a%' ;

A screenshot of a computer

Description automatically generated

* 1. Write a query to display the last name, job, department number, and department name for all employees who work in Toronto.

SELECT last\_name , dept\_name,job,employee.dept\_id FROM employee JOIN departments ON employee.dept\_id = departments.dept\_id WHERE city = 'Lahore' ;

A screenshot of a computer

Description automatically generated

* 1. Display the employee last name and employee number along with their manager’s last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively.

SELECT e1.last\_name AS Employee,e1.emp\_id AS Emp#,e2.last\_name as Manager,e1.manager AS Mgr# FROM employee e1 JOIN employee e2 on e1.manager = e2.emp\_id;

A white grid with black text

Description automatically generated

* 1. Create a query that displays employee last names, department numbers, and all the employees who work in the same department as a given employee. Give each column an appropriate label.

SELECT e1.last\_name AS employee\_last\_name,

e1.dept\_id AS employee\_department

FROM employee e1

JOIN employee e2 ON e1.dept\_id = e2.dept\_id

WHERE e2.last\_name = 'Saaid';

A blue rectangular box with black text

Description automatically generated

* 1. Create a query to display the name and hire date of any employee hired after employee Davies.

SELECT e1.last\_name, e1.hire\_date FROM employee e1 JOIN employee e2 ON e1.hire\_date > e2.hire\_date WHERE e2.last\_name = 'Ahmad';

A screenshot of a computer

Description automatically generated

* 1. Display the names and hire dates for all employees who were hired before their managers, along with their manager’s names and hire dates. Label the columns Employee, Emp Hired, Manager, and Mgr Hired, respectively.

SELECT e1.last\_name AS Employee, e1.hire\_date AS Emp\_Hired,e2.last\_name AS Manager, e2.hire\_date AS Mgr\_Hired FROM employee e1 JOIN employee e2 ON e1.manager = e2.emp\_id WHERE e1.hire\_date < e2.hire\_date ;

A screenshot of a computer

Description automatically generated