|  |  |  |
| --- | --- | --- |
| **Ex. No. 6** | **JOINING TABLES** | **Date :** |

**Alias:**

* **SELECT *column\_name(s)*  
  FROM *table\_name*AS *alias\_name;***

**Joins**

* Used to combine the data spread across tables

SELECT *table1.column, table2.column*

FROM *table1, table2*

WHERE *table1.column1* = *table2.column2*;

* A JOIN Basically involves more than one Table to interact with.
* Where clause specifies the JOIN Condition.
* Ambiguous Column names are identified by the Table name.
* If join condition is omitted, then a **Cartesian product** is formed. That is all rows in the first table are joined to all rows in the second table

# Types of Joins

* Inner Join (Simple Join) : It retrieves rows from 2 tables having a common column.

**SELECT *column\_name(s)*  
FROM table1  
INNER JOIN table2  
ON table1.column\_name = table2.column\_name;**

* Equi Join : A join condition with relationship = .
* Non Equi Join : A join condition with relationship other than = .
* Self Join : Joining of a table to itself

A self join is a regular join, but the table is joined with itself.

**SELECT column\_name(s)  
FROM table1 T1, table1 T2  
WHERE condition;**

**T1 and T2 are different table aliases for the same table.**

* Outer Join : Returns all the rows returned by simple join as well as

those rows from one table that do not match any row from the other table. The symbol (+) represents outer joins.

* Left join

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

**SELECT *column\_name(s)*  
FROM *table1*  
LEFT JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;**

* Right join

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

**SELECT *column\_name(s)*  
FROM *table1*  
RIGHT JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;**

* Full join/Full outer join

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

**SELECT *column\_name(s)*  
FROM *table1*  
FULL OUTER JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*  
WHERE *condition*;**

Customer

customer\_id | cust\_name | city | grade | salesman\_id

-------------+----------------+------------+-------+-------------

3002 | Nick Rimando | New York | 100 | 5001

3007 | Brad Davis | New York | 200 | 5001

3005 | Graham Zusi | California | 200 | 5002

3008 | Julian Green | London | 300 | 5002

3004 | Fabian Johnson | Paris | 300 | 5006

3009 | Geoff Cameron | Berlin | 100 | 5003

3003 | Jozy Altidor | Moscow | 200 | 5007

3001 | Brad Guzan | London | | 5005

Salesman

salesman\_id | name | city | commission

-------------+------------+----------+------------

5001 | James Hoog | New York | 0.15

5002 | Nail Knite | Paris | 0.13

5005 | Pit Alex | London | 0.11

5006 | Mc Lyon | Paris | 0.14

5007 | Paul Adam | Rome | 0.13

5003 | Lauson Hen | San Jose | 0.12

Order

ord\_no purch\_amt ord\_date customer\_id salesman\_id

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

70001 150.5 2012-10-05 3005 5002

70009 270.65 2012-09-10 3001 5005

70002 65.26 2012-10-05 3002 5001

70004 110.5 2012-08-17 3009 5003

70007 948.5 2012-09-10 3005 5002

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

70010 1983.43 2012-10-10 3004 5006

70003 2480.4 2012-10-10 3009 5003

70012 250.45 2012-06-27 3008 5002

70011 75.29 2012-08-17 3003 5007

70013 3045.6 2012-04-25 3002 5001

**Apply all the types of join between the tables and write the output.**

1. Equijoin
2. Non- equijoin
3. Outer join – Left and right
4. Self join
5. Cross join
6. Natural join

**Exercise Queries**

1. write a SQL query to find the salesperson and customer who reside in the same city. Return Salesman, cust\_name and city.
2. write a SQL query to find the salesperson(s) and the customer(s) he represents. Return Customer Name, city, Salesman, commission.
3. Write a SQL statement to make a Cartesian product between salesman and customer
4. Write a SQL statement to generate a list in ascending order of salespersons who work either for one or more customers or have not yet joined any of the customers.
5. Write a SQL query to find salespeople who received commissions of more than 10 percent from the company. Return Customer Name, customer city, Salesman, commission.