**EXPERIMENT NO: 11**

**Title**: Implementation of set operations like union, intersections & set difference.

**Aim:**Implementation of set operations like union, intersections & set difference.

**Theory:**

**Union**

Union combines two different results obtained by a query into a single result in the form of a table. However, the results should be similar if union is to be applied on them. Union removes all duplicates, if any from the data and only displays distinct values. If duplicate values are required in the resultant data, then UNION ALL is used.

An **example** of union is −

Select Student\_Name from Art\_Students

UNION

Select Student\_Name from Dance\_Students

This will display the names of all the students in the table Art\_Students and Dance\_Students i.e John, Mary, Damon and Matt.

**Intersection**

The intersection operator gives the common data values between the two data sets that are intersected. The two data sets that are intersected should be similar for the intersection operator to work. Intersection also removes all duplicates before displaying the result.

An **example**of intersection is −

Select Student\_Name from Art\_Students

INTERSECT

Select Student\_Name from Dance\_Students

This will display the names of the students in the table Art\_Students and in the table Dance\_Students i.e all the students that have taken both art and dance classes .Those are Mary and Damon in this example.

**Set difference**

The set difference operators takes the two sets and returns the values that are in the first set but not the second set.

An **example**of set difference is −

Select Student\_Name from Art\_Students

**MINUS**

Select Student\_Name from Dance\_Students

This will display the names of all the students in table Art\_Students but not in table Dance\_Students i.e the students who are taking art classes but not dance classes.

**Conclusion :-**

Thus we have Implementset operations like union, intersections & set difference.

**Sample Questions**

1. What is the difference between union of set and intersection of set?
2. Is set difference the same as intersection?
3. What are the different operations of a set?
4. What are the 4 operations of sets?
5. What is the difference between union and universal set?
6. What is union and intersection examples?

**EXPERIMENT NO: 12**

**Title**: Implementation of Join Operations like cross join, self join, inner join, natural join, left outer join, right outer join and full outer join.

**Aim:**Implementation of Join Operations like cross join, self join, inner join, natural join, left outer join, right outer join and full outer join.

## Theory:

**SQL join types**

**SQL inner join**

The simplest and most common form of a join is the SQL inner join the default of the SQL join types used in most database management systems. It’s the default SQL join you get when you use the join keyword by itself.

The result of the SQL inner join includes rows from both the tables where the join conditions are met.

Syntax:

|  |  |
| --- | --- |
| 1  2  3 | SELECT ColumnList from LeftTable L  INNER join  RightTable R  ON L.Column=R.Column |

**Note:** It is very easy to visualize a join query as a Venn diagram, where each of the tables is represented by intersecting shapes. The intersection of the shapes, where the tables overlap, are the rows where a condition is met. Unique columns (ID) are often used for this purpose, where the condition to be met is matching the ids of rows.

Equi join:

An equi join is the most common form of SQL inner join used in practice. If the join contains an equality operator e.g. =, then it’s an equi-join.

The following example returns all matching state names and stateProvinceIDs.

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT DISTINCT A.StateProvinceID,S.Name  FROM Person.Address A  inner join Person.StateProvince S  On A.StateProvinceID=S.StateProvinceID |

Theta join (Non-equi join):

In general, this a Theta join used to specify operators or conditions (the ON clause in SQL). In practice, this is a rarely used SQL join types. In most cases, the join will use a non-equality condition e.g. >

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT p1.FirstName, p2. FirstName  FROM PErson.Person p1  INNER join PErson.Person p2  ON len(p1.FirstName) > len(p2.FirstName); |

SQL self join

A SQL Self join is a mechanism of joining a table to itself. You would use a self join when you wanted to create a result set joining records in the table with some other records from the same table.

For a SQL self join example, consider an Employee table where managers are listed because they are also employees, and we would like to take a look at a result set that returns all of the employees and indicating who their managers are

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT e.ename, e.empno, m.ename as manager, e.mgr  FROM      emp e, emp m  WHERE e.mgr = m.empno |

SQL cross join

A CROSS join returns all rows for all possible combinations of two tables. It generates all the rows from the left table which is then combined with all the rows from the right table. This type of join is also known as a Cartesian product(A\*B).

For example, if the left table has 100 rows and the right table has 100 then the cross join result will yield 10,000 rows.

|  |  |
| --- | --- |
| 1  2  3 | SELECT e.BusinessEntityID, d.Name AS Department  FROM HumanResources.Employee AS e  CROSS join HumanResources.Department AS d |

SQL outer join

On joining tables with a SQL inner join, the output returns only matching rows from both the tables. When using a SQL outer join, not only it will list the matching rows, it will also list the unmatched rows from the other tables.

A **SQL left outer join** will return all the records from the left table in the join clause, regardless of matching records in the right table. The left SQL outer join includes rows where the condition is met plus all the rows from the table on the left where the condition is not met. Fields from the right table with no match will be displayed as null values.

Syntax:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT ColumnList from LeftTable L  LEFT join  RightTable R  ON L.Column=R.Column  Where R.Column is NULL |

The following example joins two tablesProduct and SalesOrderDetail on ProductID and preserves the unmatched rows from the left table. The Product table is matched with the SalesOrderDetail table on the ProductID columns in each table. All products, ordered and not ordered, appear in the result set.

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT p.Name, so.SalesOrderID  FROM Production.Product  p  LEFT OUTER join Sales.SalesOrderDetail so  ON p.ProductID = so.ProductID  ORDER BY p.Name ; |

A**right outer join** will return all the records in the right table in the join clause, regardless of matching records in the left table. Using the right SQL outer join includes all the rows from the table on the right. The right SQL outer join is considered a special case and many databases don’t support right joins. Generally, a SQL right join can be rewritten as a SQL left join by simply changing the order of the tables in the query. In this instance, fields from the left table with no match will display null values

Syntax:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT ColumnList from LeftTable L  RIGHT join  RightTable R  ON L.Column=R.Column  Where L.Column is NULL |

The following example joins two tables on TerritoryID(SalesTerritory) and preserves the unmatched rows from the right table(SalesPerson). The SalesTerritory table is matched with the SalesPerson table on the TerritoryID column in each table. All salespersons appear in the result set, whether or not they are assigned a territory.

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT s.Name AS Territory, p.BusinessEntityID  FROM Sales.SalesTerritory  s  RIGHT OUTER join Sales.SalesPerson p  ON s.TerritoryID = p.TerritoryID ; |

A **SQL outer join**, as you might expect by now, will return all the rows in both tables. When rows don’t have a match in one of the tables, the field will display a null value. A full SQL outer join combines the effects of the SQL left joins and SQL right joins. Many databases do not support the implementation of full SQL outer joins

Syntax:

|  |  |
| --- | --- |
| 1  2  3 | SELECT ColumnList from LeftTable L  FULL OUTER join  RightTable R  ON L.Column=R.Column |

zSQL inner join

The simplest and most common form of a join is the SQL inner join the default of the SQL join types used in most database management systems. It’s the default SQL join you get when you use the join keyword by itself.

The result of the SQL inner join includes rows from both the tables where the join conditions are met.

Syntax:

|  |  |
| --- | --- |
| 1  2  3 | SELECT ColumnList from LeftTable L  INNER join  RightTable R  ON L.Column=R.Column |

**Note:** It is very easy to visualize a join query as a Venn diagram, where each of the tables is represented by intersecting shapes. The intersection of the shapes, where the tables overlap, are the rows where a condition is met. Unique columns (ID) are often used for this purpose, where the condition to be met is matching the ids of rows.

Equi join:

An equi join is the most common form of SQL inner join used in practice. If the join contains an equality operator e.g. =, then it’s an equi-join.

The following example returns all matching state names and stateProvinceIDs.

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT DISTINCT A.StateProvinceID,S.Name  FROM Person.Address A  inner join Person.StateProvince S  On A.StateProvinceID=S.StateProvinceID |

Theta join (Non-equi join):

In general, this a Theta join used to specify operators or conditions (the ON clause in SQL). In practice, this is a rarely used SQL join types. In most cases, the join will use a non-equality condition e.g. >

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT p1.FirstName, p2. FirstName  FROM PErson.Person p1  INNER join PErson.Person p2  ON len(p1.FirstName) > len(p2.FirstName); |

SQL self join

A SQL Self join is a mechanism of joining a table to itself. You would use a self join when you wanted to create a result set joining records in the table with some other records from the same table.

For a SQL self join example, consider an Employee table where managers are listed because they are also employees, and we would like to take a look at a result set that returns all of the employees and indicating who their managers are

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT e.ename, e.empno, m.ename as manager, e.mgr  FROM      emp e, emp m  WHERE e.mgr = m.empno |

SQL cross join

A CROSS join returns all rows for all possible combinations of two tables. It generates all the rows from the left table which is then combined with all the rows from the right table. This type of join is also known as a Cartesian product(A\*B).

For example, if the left table has 100 rows and the right table has 100 then the cross join result will yield 10,000 rows.

|  |  |
| --- | --- |
| 1  2  3 | SELECT e.BusinessEntityID, d.Name AS Department  FROM HumanResources.Employee AS e  CROSS join HumanResources.Department AS d |

SQL outer join

On joining tables with a SQL inner join, the output returns only matching rows from both the tables. When using a SQL outer join, not only it will list the matching rows, it will also list the unmatched rows from the other tables.

A **SQL left outer join** will return all the records from the left table in the join clause, regardless of matching records in the right table. The left SQL outer join includes rows where the condition is met plus all the rows from the table on the left where the condition is not met. Fields from the right table with no match will be displayed as null values.

Syntax:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT ColumnList from LeftTable L  LEFT join  RightTable R  ON L.Column=R.Column  Where R.Column is NULL |

The following example joins two tablesProduct and SalesOrderDetail on ProductID and preserves the unmatched rows from the left table. The Product table is matched with the SalesOrderDetail table on the ProductID columns in each table. All products, ordered and not ordered, appear in the result set.

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT p.Name, so.SalesOrderID  FROM Production.Product  p  LEFT OUTER join Sales.SalesOrderDetail so  ON p.ProductID = so.ProductID  ORDER BY p.Name ; |

A**right outer join** will return all the records in the right table in the join clause, regardless of matching records in the left table. Using the right SQL outer join includes all the rows from the table on the right. The right SQL outer join is considered a special case and many databases don’t support right joins. Generally, a SQL right join can be rewritten as a SQL left join by simply changing the order of the tables in the query. In this instance, fields from the left table with no match will display null values

Syntax:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT ColumnList from LeftTable L  RIGHT join  RightTable R  ON L.Column=R.Column  Where L.Column is NULL |

The following example joins two tables on TerritoryID(SalesTerritory) and preserves the unmatched rows from the right table(SalesPerson). The SalesTerritory table is matched with the SalesPerson table on the TerritoryID column in each table. All salespersons appear in the result set, whether or not they are assigned a territory.

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT s.Name AS Territory, p.BusinessEntityID  FROM Sales.SalesTerritory  s  RIGHT OUTER join Sales.SalesPerson p  ON s.TerritoryID = p.TerritoryID ; |

A **SQL outer join**, as you might expect by now, will return all the rows in both tables. When rows don’t have a match in one of the tables, the field will display a null value. A full SQL outer join combines the effects of the SQL left joins and SQL right joins. Many databases do not support the implementation of full SQL outer joins

Syntax:

|  |  |
| --- | --- |
| 1  2  3 | SELECT ColumnList from LeftTable L  FULL OUTER join  RightTable R  ON L.Column=R.Column |

**Conclusion :-**

Thus we have ImplementJoin Operations like cross join, self join, inner join, natural join, left outer join, right outer join and full outer join.

**Sample Questions**

1. What is the difference between inner and outer join explain with example?
2. What is the difference between inner join left join and right join?
3. What is the difference between inner join and outer join operation?
4. What's the difference between left join and inner join?
5. What is cross join?
6. What is the difference between join and inner join in SQL?