**To change/modify a column of any table:**

**Syntax:**

ALTER TABLE <tableName>

CHANGE COLUMN <colName1> <colName2> <type> <size> <constraints>;

Where, <colName1>: Old Column name or the Column Name to be modified.

<colName2>: New Column Name. Same if the change is to be done in other sections.

**Ex.**

ALTER TABLE stud

CHANGE COLUMN marks Total DECIMAL(7,2) NULL;

**To add a column in any table:**

**Syntax:**

ALTER TABLE <tableName>

ADD COLUMN <colName> <type> <size> <constraints>;

Where, <colName>: Column name or the Column Name to be added.

**Ex.**

ALTER TABLE stud

ADD COLUMN Total DECIMAL(7,2) NULL;

**To drop/delete a column of any table:**

**Syntax:**

ALTER TABLE <tableName>

DROP COLUMN <colName> ;

Where, <colName>: Column name or the Column Name to be added.

**Ex.**

ALTER TABLE stud

DROP COLUMN Total;

**Note: CHECK() constraint is a valid constraint in MySQL but it doesn’t work.**

Aggregation Functions:

MIN():

MAX():

COUNT:

AVG():

SUM():

**WHERE & HAVING Clause:**

WHERE and HAVING both filters out records based on one or more conditions. The difference is WHERE clause can only be applied on a static non-aggregated column whereas we will need to use HAVING for aggregated columns.

1. Apart from SELECT queries, we can use WHERE clause with UPDATE and DELETE clause but HAVING clause can only be used with SELECT query.

**UPDATE department SET deptName="NewSales" WHERE deptID=1 ;  // works fine**

**UPDATE department SET deptName="NewSales" HAVING deptID=1 ; ; // error**

1. WHERE clause is used for filtering rows and it applies on each and every row, while HAVING clause is used to filter groups in SQL.

**SELECT \* FROM stud GROUP BY regno WHRER sub1>AVG(sub1); // error**

**SELECT \* FROM stud GROUP BY regno HAVING sub1>AVG(sub1); // works fine**

1. When WHERE and HAVING clause are used together in a SELECT query with aggregate function,  WHERE clause is applied first on individual rows and only rows which pass the condition is included for creating groups. Once group is created, HAVING clause is used to filter groups based upon condition specified.

**SELECT \* FROM stud WHERE sub2>50 GROUP BY regno HAVING sub1>AVG(sub1); // works fine**

Here it will operate the Query with WHERE clause first with GROUP BY clause then HAVING clause will filter groups based upon the condition specified with HAVING.

**JOIN : Displaying Data from Multiple Tables**

We have faced so far that the data was extracted from a single table. There was no need to refer to more than one table in the same statement. But many times, in real applications of databases, it is required to produce reports which need data from more than one table. Extracting data from more than one table is called one or other JOIN. Joining two of more table can be done in different way in SQL:

**Equi Join:**

In the Equi Join, values in the columns of both the tables are compared for equality first then the result of both the tables will be displayed. In the output all columns of both the tables will be displayed even if they are identical.

**SELECT \* FROM empl, dept WHERE empl.deptno = dept.deptno;**

**Non-Equi Join:**

In the non-Equi Join, values in the columns of both the tables are compared without equality check between the columns of both the tables. Comparison can be done using other relationship between the tables to Join and get the result. In the output all columns of both the tables will be displayed even if they are identical.

**SELECT \* FROM empl, dept WHERE salary>12500 AND JOB = ‘Analyst’;**

**Natural Join:**

The result of Equi and non-Equi Join includes two identical columns, where as in the Natural Join eliminates one of the two identical columns from the result.

**SELECT \* FROM empl NATURAL JOIN dept WHERE empl.deptno = dept.deptno;**

Note: In NATURAL JOIN it is not necessary to define the join condition to eliminate the identical fields in the tables.

**Cross Join:**

The Cross Join is the output as Cartesian Product of two tables. It simply matches the all tuples of one table to every tuples of other table. It does not apply any condition with the tables by default but we can use the WHERE clause to restrict the result.

**SELECT \* FROM empl CROSS JOIN dept WHERE empl.deptno = dept.deptno;**

**Left Join:**

The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

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

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders  
ON Customers.CustomerID=Orders.CustomerID;

**Right Join:**

The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

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

SELECT Orders.OrderID, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees  
ON Orders.EmployeeID=Employees.EmployeeID

**Inner Join**

Inner join is the most common type of Join which is used to combine the rows from two tables and create a result set containing only such records that are present in both the tables based on the joining condition (predicate).

If none of the record matches between two tables, then INNER JOIN will return a NULL set. Below is an example of INNER JOIN and the resulting set.

SELECT dept.name DEPARTMENT, emp.name EMPLOYEE

FROM DEPT dept, EMPLOYEE emp

WHERE emp.dept\_id = dept.id;

**The SQL UNION Operator**

The UNION operator is used to combine the result-set of two or more SELECT statements. Notice that each SELECT statement within the UNION must have the same number of columns. The columns must also have similar data types. Also, the columns in each SELECT statement must be in the same order.

### SQL UNION Syntax

**SELECT *column\_name(s)* FROM *table1* UNION SELECT *column\_name(s)* FROM *table2*;**

## SQL UNION Example

The following SQL statement selects all the **different** cities (only distinct values) from the "Customers" and the "Suppliers" tables:

**SELECT City FROM Customers UNION SELECT City FROM Suppliers ORDER BY City;**

Note: UNION cannot be used to list ALL cities from the two tables. If several customers and suppliers share the same city, each city will only be listed once. UNION selects only distinct values. Use UNION ALL to also select duplicate values!

Assignment Sheet

**(Topic : SQL)**



1. Based on these tables write SQL statements for the following queries:
   1. Display the lowest and the highest classes from the table STUDENTS.
   2. Display the number of students in each class from the table STUDENTS.
   3. Display the number of students in class 10.
   4. Display details of the students of Cricket team.
   5. Display the Admission number, name, class, section, and roll number of the students whose grade in Sports table is 'A'.
   6. Display the name and phone number of the students of class 12 who are play some game.
   7. Display the Number of students with each coach.
   8. Display the names and phone numbers of the students whose grade is 'A' and whose coach is Narendra.
2. Identify the Foreign Keys (if any) of these tables. Justify your choices.
3. Predict the the output of each of the following SQL statements, and then verify the output by actually entering these statements:
   1. SELECT class, sec, count(\*) FROM students GROUP BY class, sec;
   2. SELECT Game, COUNT(\*) FROM Sports GROUP BY Game;
   3. SELECT game, name, address FROM students, Sports WHERE students.admno = sports.admno AND grade = 'A';
   4. SELECT Game FROM students, Sports WHERE students.admno = sports.admno AND Students.AdmNo = 1434;