MYSQL – REVISION TOUR

What is the Database

- A database is an organized collection of interrelated data stored together to serve applications. It work like a container which may contains the various database objects.
- Most of the databases stores data in the form of Relations (also called Tables). Such Database are known as Relational Database.
- A Software used to manage Relational database is called RDBMS (Relational Database Management System
- Some Commonly used RDBMS software are- Oracle, MySQL, MS SQL Server, SyBase and Ingress etc.

Database Advantages

Databases reduces Redundancy

It removes duplication of data because data are kept at one place and all the application refers to the centrally maintained database.

Database controls Inconsistency

When two copies of the same data do not agree to each other, then it is called Inconsistency. By controlling redundancy, the inconsistency is also controlled.

Database facilitate Sharing of Data

Data stored in the database can be shared among several users.

Database ensures Security

Data are protected against accidental or intentional disclosure to unauthorized person or unauthorized modification.

Database maintains Integrity

It enforces certain integrity rules to insure the validity or correctness of data. For ex. A date can't be like 25/25/2000.

Data Model

Data model describes '**How data is organized or stored**' in the database. It may be-

Relational Data Model

In this model data is organized into Relations or Tables (i.e. Rows and Columns). A row in a table represents a relationship of data to each other and also called a **Tuple** or Record. A column is called Attribute or Field.

Network Data Model

In this model, data is represented by collection of records and relationship among data is shown by Links.

Hierarchical Data Model

In this model, Records are organized as Trees. Records at top level is called Root record and this may contains multiple directly linked children records.

Object Oriented Data Model

In this model, records are represented as a objects. The collection of similar types of object is called class.

RDBMS TERMINOLOGY

Relation (Table)

A Relation or Table is Two-Dimensional (Matrix) like structure arranged in Rows and Columns. It has the following properties-

- Column homogeneous - All items in a column are of same data type.
- Each column assigned a unique name and must have atomic (indivisible) value.
- All rows of a relation are distinct i.e. no two identical rows (records) can exist in the Relation.
- Ordering or Rows (Records) or Columns (fields) are immaterial.

Domain

It is collection (set) of possible values from which the value for a column is derived.

Tuple/ Entity/ Record

A Row of a table is called Tuple or Record.

Attribute/ Field:
Column of a table is called Attribute or Field.
Degree : Number of columns (attributes) in a table.
Cardinality : Number of Records in a table.

CONCEPT OF KEYS

In a Relation, each record must be unique i.e. no two identical records are allowed in the Database. A column or combination of column which identifies a record called Key of the Table. A key attribute must have unique (non-repeatable) value.

Primary Key

A set of one or more column that can uniquely identify a record in the relation is called Primary Key.

Candidate Key

A Column or group of columns which can be used as primary key are called Candidate keys, as they are candidate to become as Primary key.

Alternate Key

A Candidate Key that is not a Primary key is called Alternate key.

Foreign Key

A non-key column whose values are derived from the primary key of some other table is called Foreign key.

INTRODUCTION TO SQL

MySQL is an Open Source, Fast and Reliable Relational Database Management System (RDBMS) . It is alternative to many of the commercial RDBMS. The main features of MySQL are-

Open Source & Free of Cost:

It is Open Source and available free of cost. It is part of LAMP (Linux, Apache, MySQL, PHP/ Perl/ Python) Open Source group.

Portability:

It can be installed and run on any types of Hardware and OS like Linux, MS Windows or Mac etc.

Security :

It offers privilege and password system for authorization.

Connectivity

It may connect various types of client using different protocols and Programming Languages .

Query Language

It uses SQL (Structured Query Language) as query language, which is standardized by ANSI.

TYPES OF SQL COMMANDS

MySQL follows SQL specifications for its commands . These SQL commands can be categorized as -

Data Definition Language (DDL)

These SQL commands are used to create, alter and delete database objects like table, views, index etc.
Example : CREATE , ALTER , DROP etc.

Data Manipulation Language (DML)

These commands are used to insert, delete, update and retrieve the stored records from the table.
Ex. SELECT…., INSERT…, DELETE…, UPDATE…. etc.

Transaction Control Language (TCL)

These commands are used to control the transaction. Ex. COMMIT, ROLLBACK, SAVEPOINT etc.

Data Control Language (DCL)

These commands are used to manipulate permissions or access rights to the tables etc.
Ex. GRANT , REVOKE etc.

DATABASE HANDLING COMMANDS IN MYSQL

Creating a Database.
The following command will create School database in MySQL. mysql> CREATE DATABASE college;
Opening a database
To open an existing database, following command is used. mysql> USE college ;

Getting listings of database and tables
mysql> SHOW DATABASES;
mysql> SHOW TABLES;
Deleting a Database and Table
mysql> DROP DATABASE college;
mysql> DROP TABLE Student;
Viewing Table Structure
mysql> DESCRIBE Student;

DATA TYPE IN MYSQL

NUMERIC DATA TYPES:
INTEGER or INT – up to 11 digit number without decimal.
SMALLINT – up to 5 digit number without decimal.
FLOAT (M,D) or DECIMAL(M,D) or NUMERIC(M,D) Stores Real numbers upto M digit length (including .) with D decimal places.
e.g. Float (10,2) can store 1234567.89

DATE & TIME DATA TYPES:
DATE - Stores date in YYYY-MM-DD format.
TIME - Stores time in HH:MM:SS format.

STRING OR TEXT DATA TYPE:
CHAR(Size)
A fixed length string up to 255 characters. (default is 1)
VARCHAR(Size)
A variable length string up to 255 characters.

CREATING TABLES

CREATING SIMPLE TABLES:
CREATE TABLE < Table Name>
(<Col name1><data type>[(size)][Constraints],….);
Data types- INTEGER, NUMERIC(P,D), CHAR(n), VARCHAR(n), DATE etc.
mysql> CREATE TABLE Emp (empID integer, ename char(30), city char(25), pay decimal(10,2));

Creating Table from Existing Table:
CREATE TABLE <Table name> [AS] (<Select Query>);
CREATE TABLE Staff (Select empID, ename, pay From Emp);

CREATE TABLE Staff AS (Select * From Emp);

MAKING SIMPLE QUERIES

Selecting all columns
If you want to view all columns of the student table, then you should give the following command-
mysql> SELECT * FROM Student ;

MySQL will display the all records with all columns in the Student table is used to represent all columns.

StID	Name	Fname	DOB	City	Class
S1	Amitabh	Harivansh Rai	1948-11-10	Allahabad	12
S2	Sharukh	Firoz	1970-05-10	Delhi	11
S3	Irphan	Akbar	1970-10-05	Jaipur	11
S4	Salman	Salim Javed	1972-04-10	Mumbai	10

S5	Abhishek	Amitabh	1975-03-12	Mumbai	10

MAKING SIMPLE QUERIES

Selecting columns

If you want to view only Name and City columns of the student table mysql> SELECT Name, City FROM Student ;

Name	City
Amitabh	Allahabad
Sharukh	Delhi
Irphan	Jaipur
Salman	Mumbai
Abhishek	Mumbai

mysql> SELECT City, Name FROM Student ;

City	Name
Allahabad	Amitabh
Delhi	Sharukh
Jaipur	Irphan
Mumbai	Salman
Mumbai	Abhishek

Eliminating Duplicate values in a column - DISTINCT

mysql> SELECT City FROM Student ;

City
Allahabad
Delhi
Jaipur
Mumbai
Mumbai

MYSQL>SELECT DISTINCT city from student;

City
Allahabad
Delhi
Jaipur
Mumbai

Doing simple calculations

We can also perform simple calculations with SQL Select command. SQL provide a dummy table named DUAL, which can be used for this purpose.

mysql> SELECT 4*3 ;

We can also extend this idea with a columns of the existing table. mysql> SELECT Name, Sal *12 FROM EMP ;

Using Column Aliases

We can give a different name to a column or expression (Alias) in the output of a query.

mysql> SELECT Name, Sal*12 AS 'Annual Salary' FROM EMP;

mysql> SELECT Name, DOB AS 'Date of Birth' FROM Student;

mysql> SELECT 22/7 AS PI FROM Dual;

When Alias name is a single word then ' ' is not required.

Selecting Specific Rows – WHERE clause

- **WHERE <Condition>**

We can select specific records by specifying condition with WHERE clause.

mysql> SELECT * FROM Student WHERE City='Mumbai';

StID	Name	Fname	DOB	City	Class
S4	Salman	Salim Javed	1972-04-10	Mumbai	10
S5	Abhishek	Amitabh	1975-03-12	Mumbai	10

mysql> SELECT Name, Fname, City from Student WHERE Class >10;

Name	Fname	City	Class
Amitabh	Harivansh Rai	Allahabad	12
Sharukh	Firoz	Delhi	11
Irphan	Akbar	Jaipur	11

WHERE clause(Operator)

Relational Operators

We can use the following Relational operators in condition.

=, > , < , >=, <=, <>, IS , LIKE, IN, BETWEEN

Logical Operators

We can use the following Logical Operators to connect two conditions.

OR , AND , NOT (!)

mysql> SELECT Name, City from Student WHERE City <> 'Mumbai' AND Class>10;

mysql> SELECT * FROM Emp WHERE Sal >10000 OR Job ='Manager';

mysql> SELECT * FROM Student WHERE NOT Grade='A';

Selecting Specific Rows – WHERE clause

Specifying Range of Values – BETWEEN Operator

mysql> SELECT * FROM Emp WHERE Sal BETWEEN 5000 AND 10000 ;

The same query can also be written as -

mysql> SELECT * FROM Emp WHERE Sal >= 5000 AND Sal<=10000 ;

Other Logical operators also can be applied-

mysql> SELECT * FROM Emp WHERE NOT Sal BETWEEN 5000 AND 10000 ;

Specifying List – IN Operator

mysql> SELECT * FROM EmpWHERE Sal IN (5000, 10000) ;

The same query can also be written as -

mysql> SELECT * FROM Emp
WHERE Sal = 5000 OR Sal =10000 ;

mysql> SELECT * FROM Student
WHERE City IN ('Mumbai', 'Delhi','Kanpur') ;
Selecting Specific Rows – WHERE clause

Pattern Matching – LIKE Operator

A string pattern can be used in SQL using the following wild card

- **% Represents a substring in any length**
- **_ Represents a single character**

Example.

'A%' represents any string starting with 'A' character. '_ _A' represents any 3 character string ending with 'A'. '_B%' represents any string having second character 'B' '_ _ _' represents any 3 letter string.

A pattern is case sensitive and can be used with LIKE operator.

mysql> SELECT * FROM Student WHERE Name LIKE 'A%';

mysql> SELECT * FROM Student WHERE Name LIKE '%Singh%'; mysql> SELECT Name, City

FROM Student WHERE Class>=9 AND Name LIKE '%Kumar%' ;

Searching NULL Values – IS Operator

mysql> SELECT * FROM Student WHERE City IS NULL ;

The NOT Operator can also be applied -

mysql> SELECT * FROM Student WHERE City IS NOT NULL;

Ordering Query Result – ORDER BY Clause

A query result can be orders in ascending (A-Z) or descending (Z-A) order as per any column. Default is Ascending order.

mysql> SELECT * FROM Student ORDER BY City;

To get descending order use DESC key word.

mysql> SELECT * FROM Student ORDER BY City DESC;

mysql> SELECT Name, Fname, City FROM Student Where Name LIKE 'R%' ORDER BY Class;

Inserting Records in a Table

You can insert record in the table by using by using the following DML command.
INSERT INTO <Table Name> [<Column list>] VALUES <list of values>
Suppose a table named STUDENT has been created with the following structure.

StID	NAME	FNAME	DOB	CITY	CLASS

We can insert a record as follows-
mysql> INSERT INTO Student VALUES
('s1','Amitabh', 'Harivansh','1955-10-25', 'Mumbai', 12);
mysql> INSERT INTO Student VALUES

('s2','Sharukh Khan', NULL,'1972-5-25', 'Delhi', 10);
mysql> INSERT INTO Student (StID, FName, Name, Class) VALUES ('s3','Amitabh', 'Abhishek', 10);

Inserting Records from Other Table

You can insert all or selected record(s) in the table from another table by using Select … command in place of Values.
Suppose a table named NEWSTUDENT has been created and records to be inserted from OLDSTUDENT table having the same structure of columns.

mysql> INSERT INTO Newstudent VALUES (SELECET * FROM Oldstudent);
mysql>INSERT INTO Newstudent VALUES
(SELECT * FROM Oldstudent WHERE City='Mumbai'); mysql> INSERT INTO Newstudent (StID, Name, Class)
VALUES (Select StID, Name,Class FROM Oldstudent
WHERE Class>=11);

Deleting Records from the Table

You can delete all or selected record(s) from the table by using the following DML command.
DELETE FROM <Table Name> [WHERE <Condition>]
This command will
mysql> DELETE FROM Student ;
delete all records…
mysql> DELETE FROM Student WHERE City='Mumbai' ; mysql> DELETE FROM Student WHERE Class >=11 ;

- **You can recall (Undelete) records by giving ROLLBACK command.**
  mysql> ROLLBACK ;
- **You can issue COMMIT command to record the changes permanently.**

mysql> COMMIT;

Modifying Records in the Table

You can modify the values of columns of all or selected records in the table by using the following DML command.

UPDATE <Table Name> SET <Column> = <Expression> [WHERE <Condition>]

mysql> UPDATE Student SET Class =10 ;

mysql> UPDATE Student SET FName= CONACT('Mr.', FName') ; mysql> UPDATE Emp SET Sal = Sal+(Sal*10/100);

mysql> UPDATE Emp SET Sal = Sal+(Sal*10/100) WHERE Sal <=10000;

mysql> UPDATE Emp SET City = 'Dehradun' WHERE CITY IS NULL;

WORKING WITH FUNCTIONS

What is Function?

- **A function is a special types of command that performs some operation and returns a single value as a result.**
- **It is similar to method or function in JAVA, which can be called by giving some argument.**

Types of Functions:

- **Numeric Functions**
- **String Functions**
- **Date & Time Function**
- **Aggregate Functions**

Numeric Functions

These functions may accept some numeric values and performing required operation, returns numeric values as result

Name	Purpose	Example
MOD (M, N)	Returns remainder of M divide by N	Select MOD(11,4) ; 3
POWER (M, N) POW (M, N)	Returns M^N	Select POWER(3,2); 9
ROUND (N [,M])	Returns a number rounded off up to M place. If M is -1, it rounds nearest 10. If M is not given, the N is rounded to the nearest Integer.	Select ROUND(15.193,1); 15.2 Select ROUND(15.193); 15
SQRT (N)	Returns square root of N	Select SQRT(25); □ 5
TRUNCATE(N,M)	Returns number after truncating M decimal place.	Select TRUNCATE(15.79,1) □ 15.7

String functions

Name	Purpose	Example
CONCAT(str1,str2)	Returns concatenated string i.e. str1+str2.	Select CONCAT(Name, City) from Student;
LOWER(str) / LCASE(str)	Returns the given string in lower case.	Select LOWER('ABC'); □ abc
UPPER(str) / UCASE(str)	Returns the given String in upper case.	Select UPPER('abc'); □ ABC
LTRIM(str) RTRIM(str) TRIM(str)	Removes Leading/Trailing/both spaces from given string.	Select TRIM(' ABC '); □ 'ABC'
LEFT(str, N) RIGHT(str,N)	Returns the (N) characters from left/right from the given string.	Select LEFT('Computer',4); □ Comp
SUBSTR(str,P,[N]) / MID	Returns the substring for given position(P) and	Select SUBSTR('Computer',3,2);

(str,P,N)	length (N). If M is (- ve) then backward position counted.	mp
INSTR(str1,str2)	Returns the index of first occurrence of str2 in str1.	Select INSTR('Common', 'm'); 3
LENGTH(str)	Returns the length of given string	Select LENGTH('Common'); 6

Date & TimeFunctions

Name	Purpose	Example
CURDATE() CURRENT_DATE()	Returns the current date in YYYY-MM-DD format.	Select CURDATE(); 2013-10-02
NOW()	Returns the current date & Time as YYYY-MM-DD HH:MM:SS	Select NOW(); 2013-10-02 11:30:02
SYSDATE()	Returns the current date & Time as YYYY-MM-DD HH:MM:SS	Select SYSDATE(); 2013-10-02 11:30:10
DATE()	Returns the date part of a date- time expression.	Select DATE(SYSDATE()); 2013-10-02
MONTH() YEAR()	Returns the Month/Year from given date argument.	Select MONTH('2012-10-02'); 10
DAYNAME()	Returns the name of the weekday	Select DAYNAME(CURDATE()); SUNDAY
DAYOFMONTH()	Returns the day of month (1-31).	Select DAYOFMONTH(CURDATE());
DAYOFWEEK()	Returns the day of week (1-7).	Select DAYOFWEEK(CURDATE());
DAYOFYEAR()	Returns the day of year(1-366).	Select DAYOFYEAR(CURDATE());

Aggregate function:

Name	Purpose	Example
SUM()	Returns the sum of given column.	Select SUM(Pay) from Emp; Select Sum(Pay), Sum(Net) from Emp;
MIN()	Returns the minimum value in the given column.	Select MIN(Pay) from Emp;
MAX()	Returns the maximum value in the given column.	Select MAX(Pay) from Emp;
AVG()	Returns the Average value of the given column.	Select AVG(Pay) from Emp;
COUNT()	Returns the total number of values/ records in given column.	Select COUNT(Name) from Emp; Select COUNT(*) from Emp;

Modifying Table Structure

You can alter (modify) the structure of existing table by the using ALTER TABLE…. Command of MySQL.
You can do the following with the help of ALTER TABLE.. Command.

- Add a new Column or Constraints
- Modifying existing column (name, data type, size etc.)
- Delete an existing column or Constraints

Changing Column Name ALTER TABLE <TableName>
ADD|MODIFY|DROP|CHANGE <Column Definition(s)>
Modifying Table Structure

Adding new column

ALTER TABLE <Table Name>
ADD <Column>[<data type> <size>][<Constraints>]

mysql> ALTER TABLE Student ADD (TelNo Integer);

mysql> ALTER TABLE Student ADD (Age Integer DEFAUL 10);

Modifying Existing Column

ALTER TABLE <Table Name>

MODIFY <Column>[<data type> <size>] [<Constraints>] mysql> ALTER TABLE Student MODIFY Name VARCHAR(40); mysql> ALTER TABLE Emp MODIFY (Sal DECIMAL (10,2));

- **Removing Column & Constraints**

ALTER TABLE <Table Name>
DROP <Column name> |<Constraints>
mysql> ALTER TABLE Student DROP TelNo; mysql> ALTER TABLE Emp DROP JOB, DROP Pay;

More On Database & SQL– Advanced Concepts

Integrity Constraints

One of the major responsibility of a DBMS is to maintain the Integrity of the data i.e. Data being stored in the Database must be correct and valid.

An Integrity Constraints or Constraints are the rules, condition or checks applicable to a column or table which ensures the integrity or validity of data.

The following constraints are commonly used in MySQL.

- **NOT NULL**
- **PRIMARY KEY**
- **UNIQUE ***
- **DEFAULT ***
- **CHECK ***
- **FOREIGN KEY ***
- **Type of Constraints**

S.N	Constraints	Description
1	NOT NULL	Ensures that a column cannot have NULL value.
2	DEFAULT	Provides a default value for a column, when nothing is given.
3	UNIQUE	Ensures that all values in a column are different.
4	CHECK	Ensures that all values in a column satisfy certain condition.
5	PRIMARY KEY	Used to identify a row uniquely.
6	FOREIGN KEY	Used to ensure Referential Integrity of the data.

UNIQUE v/s PRIMARY KEY

- **UNIQUE allows NULL values but PRIMERY KEY does not.**
- **Multiple column may have UNIQUE constraints, but there is only one PRIMERY KEY constraints in a table.**

Implementing Primary Key Constraints

Defining Primary Key at Column Level:

```
mysql> CREATE TABLE Student
      ( StCode char(3) NOT NULL PRIMARY KEY,
        Stname char(20) NOT NULL,
          ……...
      );
```

Defining Primary Key at Table Level:

```
mysql> CREATE TABLE Student
      ( StCode char(3) NOT NULL, Stname char(20) NOT NULL,
        ……..
PRIMARY KEY (StCode) );
```

A Composite (multi-column) Primary key can be defined as only a Table level whereas Single-column Primary key can be defined in both way i.e. Column level or Table level

Implementing Constraints in the Table

```
mysql> CREATE TABLE Student
      (StCode char(3) NOT NULL PRIMARY KEY,
       Stname char(20) NOT NULL, StAdd  varchar(40),
       AdmNo  char(5)  UNIQUE, StSex char(1) DEFAULT  'M',
       StAge integer CHECK (StAge>=5) );
CREATE TABLE EMP ( Code char(3) NOT NULL,
                   Name char(20) NOT NULL, City varchar(40),
                   Pay Decimal(10,2),
                   PRIMARY KEY (Code) );
```

Implementing Foreign Key Constraints

- **A Foreign key is non-key column in a table whose value is derived from the Primary key of some other table.**
- **Each time when record is inserted or updated in the table, the other table is referenced. This constraints is also called Referential Integrity Constraints.**
- **This constraints requires two tables in which Reference table (having Primary key) called Parent table and table having Foreign key is called Child table.**

Implementing Foreign Key

- **CREATE TABLE Department**
- **(DeptNo char(2) NOT NULL PRIMARY KEY,**
- **DeptName char(10) NOT NULL, Head char(30));**
- **CREATE TABLE Employee**
- **(EmpNo char(3) NOT NULL PRIMARY KEY,**
- **Name char(30) NOT NULL,**
- **City char(20),**
- **Sal decimal(8,2), DeptNo char(2),**

FOREGIN KEY (DeptNo) REFERENCES Departmet (DeptNo));

A Table may have multiple Foreign keys.
Foregn key may have repeated values i.e. Non-Key Column
Modifying Table Constraints

- **Adding new column and Constraints**
- **ALTER TABLE <Table Name>**
- **ADD <Column>[<data type> <size>][<Constraints>]**
- **mysql> ALTER TABLE Student ADD (TelNo Integer);**

mysql> ALTER TABLE Student ADD (Age Integer CHECK (Age>=5)); mysql> ALTER TABLE Emp ADD Sal Number(8,2) DEFAULT 5000 ; mysql> ALTER TABLE Emp ADD PRIMARY KEY (EmpID);
mysql> ALTER TABLE Emp ADD PRIMARY KEY (Name,DOB);

- **Modifying Existing Column and Constraints**
- **ALTER TABLE <Table Name>**
- **MODIFY <Column>[<data type> <size>] [<Constraints>]**
- **mysql> ALTER TABLE Student MODIFY Name VARCHAR(40); mysql> ALTER TABLE Emp MODIFY (Sal DEFAULT 4000); mysql> ALTER TABLE Emp MODIFY (EmpName NOT NULL);**
- **Modifying Table Constrains cont..**
- **Removing Column & Constraints**
- **ALTER TABLE <Table Name>**
- **DROP <Column name> |<Constraints>**
- **mysql> ALTER TABLE Student DROP TelNo; mysql> ALTER TABLE Emp DROP JOB, DROP Pay;**
- **mysql> ALTER TABLE Student DROP PRIMARY KEY;**
- **Changing Column Name of Existing Column**
- **ALTER TABLE <Table Name> CHANGE <Old name><New Definition> mysql> ALTER TABLE Student**

CHANGE Name Stname Char(40);

Viewing & Disabling Constraints

- **To View the Constraints**

The following command will show all the details like columns definitions and constraints of EMP table. mysql> SHOW CREATE TABLE EMP;
Alternatively you can use DESCribe command:
mysql> DESC EMP;

- **Enabling / Disabling Foreign Key Constraint**
  - ✓ **You may enable or disable Foreign key constraints by setting the value of FOREIGN_KEY_CHECKS variable.**
  - ✓ **You can't disable Primary key, however it can be dropped (deleted) by Alter Table… command.**

- **To Disabling Foreign Key Constraint**
  mysql> SET FOREIGN_KEY_CHECKS = 0;
- **To Enable Foreign Key Constraint**

mysql> SET FOREIGN_KEY_CHECKS = 1

Grouping Records in a Query

- **Some time it is required to apply a Select query in a group of records instead of whole table.**
- **You can group records by using GROUP BY <column> clause with Select command. A group column is chosen which have non-distinct (repeating) values like City, Job etc.**
- **Generally, the following Aggregate Functions [MIN(), MAX(), SUM(), AVG(), COUNT()] etc. are applied on groups.**

Name	Purpose
SUM()	Returns the sum of given column.
MIN()	Returns the minimum value in the given column.
MAX()	Returns the maximum value in the given column.
AVG()	Returns the Average value of the given column.
COUNT()	Returns the total number of values/ records as per given column.

Aggregate Functions & NULL Values

Consider a table Emp having following records as-

Emp		
Code	Name	Sal
E1	Ram Kumar	NULL
E2	Suchitra	4500
E3	Yogendra	NULL
E4	Sushil Kr	3500
E5	Lovely	4000

mysql> Select Sum(Sal) from EMP; □ 12000

mysql> Select Min(Sal) from EMP; □ 3500

mysql> Select Max(Sal) from EMP; □ 4500

mysql> Select Count(Sal) from EMP; □ 3

mysql> Select Avg(Sal) from EMP; □ 4000

mysql> Select Count(*) from EMP; □ 5

Aggregate Functions & Group

An Aggregate function may applied on a column with DISTINCT or ALL keyword. If nothing is given ALL is assumed.

- **Using SUM (<Column>)**
  This function returns the sum of values in given column or expression.
  mysql> Select Sum(Sal) from EMP;
  mysql> Select Sum(DISTINCT Sal) from EMP;
  mysql> Select Sum (Sal) from EMP where City='Kanpur'; mysql> Select Sum (Sal) from EMP Group By City; mysql> Select Job, Sum(Sal) from EMP Group By Job;

Using MIN (<column>)

This functions returns the Minimum value in the given column.

mysql> Select Min(Sal) from EMP;

mysql> Select Min(Sal) from EMP Group By City; mysql> Select Job, Min(Sal) from EMP Group By Job;

Aggregate Functions & Group

- **Using MAX (<Column>)**
  This function returns the Maximum value in given column.
  mysql> Select Max(Sal) from EMP;
  mysql> Select Max(Sal) from EMP where City='Kanpur'; mysql> Select Max(Sal) from EMP Group By City;
  Using AVG (<column>)

This functions returns the Average value in the given column.

mysql> Select AVG(Sal) from EMP;

mysql> Select AVG(Sal) from EMP Group By City;

- **Using COUNT (<*|column>)**

This functions returns the number of rows in the given column.

mysql> Select Count (*) from EMP;

mysql> Select Count(Sal) from EMP Group By City; mysql> Select Count(*), Sum(Sal) from EMP Group By Job;

Aggregate Functions & Conditions

You may use any condition on group, if required.

HAVING

<condition> clause is used to apply a condition on a group.

mysql> Select Job, Sum(Pay) from EMP
Group By Job HAVING Sum(Pay)>=8000; mysql> Select Job, Sum(Pay) from EMP
Group By Job HAVING Avg(Pay)>=7000; mysql> Select Job, Sum(Pay) from EMP
Group By Job HAVING Count(*)>=5;

mysql> Select Job, Min(Pay),Max(Pay), Avg(Pay) from EMP Group By Job HAVING Sum(Pay)>=8000;

mysql> Select Job, Sum(Pay) from EMP Where City='Dehradun'
Group By Job HAVING Count(*)>=5;

Displaying Data from Multiple Tables - Join Query

Some times it is required to access the information from two or more tables, which requires the Joining of two or more tables. Such query is called Join Query.

MySQL facilitates you to handle Join Queries. The major types of Join is as follows-

- **Cross Join (Cartesian Product)**
- **Equi Join**
- **Non-Equi Join**
- **Natural Join**
- **Cross Join – Mathematical Principle**
- **Consider the two set A= {a,b} and B={1,2}**
- **The Cartesian Product i.e. AxB = {(a,1) (a,2) (b,1) (b,2)} Similarly, we may compute Cross Join of two tables by joining each Record of first table with each record of second table.**
- **Equi Join – Mathematical Principle**
- **In Equvi Join, records are joined on the equality condition of Joining Column. Generally, the Join column is a column which is common in both tables.**
- **Consider the following table R and S having C as Join column.**

Natural Join – Mathematical Principle

The Natural Join is much similar to Equi Join i.e. records are joined on the equality condition of Joining Column except that the common column appears one time.

Consider the following table R and S having C as Join column.

How to Join ?

MySQL offers different ways by which you may join two or more tables.

- **Method 1 : Using Multiple table with FROM clause**

  The simplest way to implement JOIN operation, is the use of multiple table with FROM clause followed with Joining condition in WHERE clause.

Select * From EMP, DEPT

Where Emp.DeptNo = Dept.DeptNo ;

If common column are differently spelled then no need to use Qualified name.

- **Method 2: Using JOIN keyword**

  MySQL offers JOIN keyword, which can be used to implement all type of Join operation.

Select * From EMP JOIN DEPT ON Emp.DeptNo=Dept.DeptNo ;

Using Multiple Table with FROM clause

The General Syntax of Joining table is-

SELECT < List of Columns> FROM <Table1, Table 2, …> WHERE <Joining Condition> [Order By ..] [Group By ..]

- **You may add more conditions using AND/OR NOT operators, if required.**
- **All types of Join (Equi, No-Equi, Natural etc. are implemented by changing the Operators in Joining Condition and selection of columns with SELECT clause.**

Ex. Find out the name of Employees working in Production Deptt.

```
Select Ename From EMP, DEPT
Where Emp.DeptNo=Dept.DeptNo AND Dname='Production';
```

Ex. Find out the name of Employees working in same city from where they belongs (hometown).

```
Select Ename From EMP, DEPT
Where Emp.DeptNo=Dept.DeptNo And City=Location;
```

Using JOIN keyword with FROM clause

MySQL 's JOIN Keyword may be used with From clause.

SELECT < List of Columns>
FROM <Table1> JOIN <Table2> ON <Joining Condition> [WHERE <Condition>] [Order By ..] [Group By ..]

Ex. Find out the name of Employees working in Production Deptt.

```
Select Ename From EMP JOIN DEPT ON Emp.DeptNo=Dept.DeptNo Where Dname='Production';
```

Ex. Find out the name of Employees working in same city from where they belongs (hometown) .

```
Select Ename From EMP JOIN DEPT ON Emp.DeptNo = Dept.DeptNo WHERE City=Location;
```

Nested Query (A query within another query)

Sometimes it is required to join two sub-queries to solve a problem related to the single or multiple table. Nested query contains multiple query in which inner query evaluated first.

The general form to write Nested query is-

Select …. From <Table>
Where <Column1> <Operator>
(Select Column1 From <Table> [Where <Condition>])

Ex. Find out the name of Employees working in Production Deptt.

```
Select Ename From EMP
 Where DeptNo = (Select DeptNo From DEPT Where
DName='Production');
```

Ex. Find out the name of Employees who are getting more pay than 'Ankit'.

```
Select Ename From EMP
Where Pay >= (Select Pay From EMP Where Ename='Ankit' );
```

Union of Tables

Sometimes it is required to combine all records of two tables without having duplicate records. The combining records of two tables is called UNION of tables.

UNION Operation is similar to UNION of Set Theory.

E.g. If set A= {a,c,m,p,q} and Set B= {b,m,q,t,s} Then AUB= {a,c,m,p,q,b,t,s}

[All members of Set A and Set B are taken without repeating] Select …. From <Table1>[Where <Condition>]
UNION [ALL]
Select …. From <Table2> [Where <Condition>];

Ex. Select Ename From PROJECT1 UNION
Select Ename From PROJECT2 ;

)