RDBMS

by - Sameer Dehadrai Sir

(022-22021984)

#computer

computaire ---------------------------------------> to compute / calculate

Data (**raw data**) --------> computer ---------> meaningful Information (**meaningful data , processed data**)

#Database

- collection of large amount of data

#DBMS (database management system)

- It is a collection of program (readymade software that allow to **insert update delete and process data**)
- e. g. Ms-Excel
- processing means convert the data into **meaningful information**.

## Various DBMS

- MS-Excel
- dBase
- FoxBASE
- FoxPro
- Data flex
- DB-vista

---

sr. no	DBMS	RDBMS
1	MS-Excel	Oracle, MySQL
2	Field	Column, Attribute
3	Record	Row, Tuple, Entity, Opportunity
4	File	Table, Relation, entity, class, Applet, matrix
5	naming conventions (Nomenclatures)	naming conventions (Nomenclatures)
6	Relationship between two files is maintained programmatically.	Relationship between two tables can be specified at the time of table creation (e. g. Foreign Key constraint).
7	More Programming	Less Programming
8	More time require for development	Less time require for development
9	high n/w traffic	low n/w traffic
10	processing is on client machine	processing is on server machine (**known as client server architecture**)
11	slow and expensive	Faster And cheaper (**in terms of n/w traffic, hardware cost, internet cost, infrastructure cost**)
12	client-server architecture is not supported	client-server architecture is supported
13	File level locking	Row level locking (**table is not a file but internally every row is file**)
14	Not suitable for multi-user	Suitable for multi-user
15	Distributed database are not supported	most of Databases are Supports Distributed database exceptional cases : MS-Access, Vatcom SQL, Paradox, DB2, etc.
16	No security of data	Multilevel of security 1) Logging in security (**username and password**) . 2) Command level security (Create **table, create user, create trigger**). 3) object level security (**access the table of another user**).

Various RDBMS Available

1) Informix

- Fastest in terms of processing speed

2) Oracle

- most popular RDBMS
- best s/w development tools
- make programming very easy.

Product of Oracle Corporation (1977), largest DB s/w Development Company in the world, overall #2 largest s/w company in the world.

63% of world commercial database market (in client-server environment).

86% of world commercial database market (in the internet environment).

works with 113 OS.

3) Sybase

- going down
- ASP acquired Sybase

4) MS SQL Server

- good RDBMS
- competition for oracle
- very popular with .net programmers
- works only with windows OS
- 16% of world commercial DB market

DB Server has been a mainframe (super computer):-

5) DB2

- good RDBMS from IBM

6) CICS

7) TELON

8) IDMS

Single-user RDBMS :-

- Client server architecture not supported.
- Distributed database not supported.

9) MS Access

- RDBMS from Microsoft

10) Paradox

11) Vatcom SQL

Personal Oracle:-

- single user edition
- free RDBMS

12) MySQL

- MySQL was launched by a Swedish company in 1995 (C and C++ source code).
- its name is a combination of "My", the name of co-founder Michael Widenius ' daughter, and "SQL"
- MySQL is an open-source RDBMS
- most widely used open-source RDBMS
- free RDBMS (42% of world free database market)
- part of the widely used ***LAMP(Linux Apache MySQL Perl/python/PHP)*** open-source web application s/w stack (and other "AMP" stacks)
- **e. g. LAMP for LINUX ,**
- **e. g. WAMP for WINDOWS**,
- **e. g. MAMP for MAC.**

Following Companies using MySQL

- Facebook
- WhatsApp
- Twitter
- Flicker
- YouTube
- WordPress
- Google (though for not searches), etc.

S/w Development tools for MySQL

SQL

- structure Query Language
- confirms to ANSI standards
- confirms to ISO standards (for QA)
- common for all RDBMS
- initially founded by IBM (1975-77)

- earlier known as RQBE (Relational Query By Example)
- IBM gave RQBE free of cost to ANSI
- now controlled by ANSI (same for all RDBMS)

MySQL Command Line Client

- MySQL client software
- used for running SQL, MySQL commands and MySQL/PL programs
- character based

MySQL Workbench

- MySQL client software
- used for running SQL, MySQL commands and MySQL/PL programs
- GUI based

MySQL/PL

- MySQL programming language
- programming language from MySQL

MySQL for Excel

MySQL Notifier

MySQL Enterprise Backup

MySQL Enterprise High Availability

MySQL Enterprise Encryption

MySQL Enterprise Monitor

MySQL:-

SQL

- Structure Query Language
- commonly pronounced as "sequel"
- common for all RDBMS
- confirm to ANSI (e. g. 1 char = 1 byte)
- and ISO standards (for QA)
-

Sub-divisions of SQL:-

1. **DDL (Data Definition language)**

Create

Drop

Alter

2. **DML (Data Manipulation Language)**

   Insert

   Update

   Delete

3. **DCL (Data Control Language)**

   Grant

   Revoke

4. **DQL (Data Query Language)**

   Select

Not ANSI standards:-

5th component of SQL:-

Extra in MySQL and Oracle RDBMS:-

DTL/TCL (Data Transaction Language) / (Transaction Control Langauge)

Commit

Rollback

Savepoint

Extra in Oracle RDBMS only:-

DML (Data Manipulation Language)

Upsert

Merge

Rules for tableNames, columnNames, variableNames :-

- max 30 characters
- A - Z, a - z, 0 - 9, allowed
- has to begin with alphabet
- Special characters $, _ allowed
- In oracle, you want to special characters # in tablename, enclose it in back-quotes (' ') ;
- 134 Reserved worlds not allowed in tableaname.

Data Types:-

char:-

- allows any character
- could be alpha-numeric also
- max upto 255 characters
- fixed lenght character data
- wastage of HD space
- e. g. Roll_No, PAN_No, etc.

Benifit of char, the searching and retrieval will be very fast

Varchar:-

- variable lenght character
- allows any character
- could be alpha-numeric also
- max upto 65,535 character
- conserve on HD space
- e. g. ADDRESS, CITY, etc.

Longtext :-

- allows any character
- max 4,294,967,295 characters
- 4 Gb - 1
- e. g. RESUME, EXPERIENCE, REMARK, COMMENTS, etc.

Longblob:-

- long binary large object
- allows binary data
- max 4,294,967,295 bytes of binary data
- e. g. PHOTOS, GRAPHS, SOUND, MUSIC, VIDEO, CHARTS, etc.
- longblob is a multimedia datatype

Date

- **YYYY-MM-DD** ('1000-01-01' to '9999-12- 31')
- Year values in the range 70-99 are converted to 1970-1999
- Year values in the range 00-69 are converted to 2000-2069

Time

- hh:mm:ss or 'HHH:MM:SS'
- TIME values may range from '-838:59:59' to '838:59:59'

Datetime

- (YYYY-MM-DD hh:mm:ss)
- '1000-01- 01 00:00:00' to '9999-12-31 23:59:59'

Year

- (YYYY)(1901 to 2155)
- Max 4096 columns per table provided the row size <= 65,535 Bytes.

Create query:-

```
create database < database name >;

create table Emp (
Empno Varchar(4),
Ename Varchar(25),
Sal float(7,2),
City Varchar(15),
Dob date
);
```

' ; ' is known as terminator (denotes end of command).

Sal float (7,2)

7 --> precision (total number of digits)

2 --> scale (reserve for decinmal)

Insert Query :-

```
insert into emp values ( '1' , 'King', 'Mumbai', '1990-10-15' );
```

note :- (for char, varchar, date use ' ' only).

year vlaues it the range 70-99 are automatically converted to 1970-1999

year values in the range 00-69 are automatically converted to 2000-2069

Flexible

```
insert into emp values ('1',' King ',' Mumbai ',' 1990-10-15 ');
```

Readable

```
insert into emp (empno,sa,ename,city,dob) values ( '2',6000,'jack','Delhi','1985-11-16');
```

- **NULL means nothing**
- **NULL values is having ASCII value 0**
- **special treatment given to NULL value in all RDBMS**
- **NULL value is independent of datatype**
- **NULL value occupies only 1 Byte**
- **if row is ending with NULL's. thos column will not occupy any space**
- **those column that are likely to have large number of NULL's, they should preferably be spacified at end of the table structure, to conserve on HD space**

insert into emp value ('4','Atul'); -----> this will throw error

insert into emp value ('4','Atul',null,null,null); ----- > this is right

---

```
insert into SALESPEOPLE (SNUM,SNAME,CITY,COMM)
values (1001,'Peel','London',.12),
(1002,'Serres','SanJose',.13),
(1004,'Motika','London',.11));
```

The above insert statements is supported by MySQL RDBMS, but not supported by Oracle RDBMS.

in oracle RDBMS, if you want to INSERT multiple rows, then you will require a separate INSERT statement for each row.

a int(2)

set a = 10000000;

this is the bug in MySQL RDBMS, this is not a problem in oracle RDBMS;

## This table we will need to perform next operations :

```
create table emp ( DEPTNO INT(2),JOB VARCHAR(20), ENAME VARCHAR(20), SAL INT(7),   HIREDATE DATE);
DESC EMP;
INSERT INTO emp
values(10 ,'PRESIDENT','KING', 5000, '1981-11-17'),
(30 ,'MANAGER','BLAKE', 2850, '1981-05-01'),
(10 ,'MANAGER','CLARK', 2450, '1981-06-09'),
(20 ,'MANAGER','JONES', 2975, '1981-04-02'),
(30 ,'SALESMAN','MARTIN', 1250, '1981-09-28'),
(30 ,'SALESMAN','ALLEN', 1600, '1981-02-20'),
(30 ,'SALESMAN','TURNER', 1500, '1981-09-08'),
(30 ,'CLERK','JAMES', 950, '1981-12-03'),
(30 ,'SALESMAN','WARD', 1250, '1981-02-22'),
(20 ,'ANALYST','FORD', 3000, '1981-12-03'),
(20 ,'CLERK','SMITH', 800, '1980-12-17'),
(20 ,'ANALYST','SCOTT', 3000, '1982-12-09'),
(20 ,'CLERK','ADAMS', 1100, '1983-01-12'),
(10 ,'CLERK','MILLER', 1300, '1982-01-23');

+--------+-----------+--------+------+------------+
| DEPTNO | JOB       | ENAME  | SAL  | HIREDATE   |
+--------+-----------+--------+------+------------+
|     10 | PRESIDENT | KING   | 5000 | 1981-11-17 |
|     30 | MANAGER   | BLAKE  | 2850 | 1981-05-01 |
|     10 | MANAGER   | CLARK  | 2450 | 1981-06-09 |
|     20 | MANAGER   | JONES  | 2975 | 1981-04-02 |
|     30 | SALESMAN  | MARTIN | 1250 | 1981-09-28 |
|     30 | SALESMAN  | ALLEN  | 1600 | 1981-02-20 |
|     30 | SALESMAN  | TURNER | 1500 | 1981-09-08 |
|     30 | CLERK     | JAMES  |  950 | 1981-12-03 |
|     30 | SALESMAN  | WARD   | 1250 | 1981-02-22 |
|     20 | ANALYST   | FORD   | 3000 | 1981-12-03 |
|     20 | CLERK     | SMITH  |  800 | 1980-12-17 |
|     20 | ANALYST   | SCOTT  | 3000 | 1982-12-09 |
|     20 | CLERK     | ADAMS  | 1100 | 1983-01-12 |
|     10 | CLERK     | MILLER | 1300 | 1982-01-23 |
+--------+-----------+--------+------+------------+
```

SELECT :-

Metacharacter (all columns)

```
select * from < table_name > ;
```

To restrict columns

```
select deptno, ename from emp ;
+--------+--------+
| deptno | ename  |
+--------+--------+
|     10 | KING   |
|     30 | BLAKE  |
|     10 | CLARK  |
|     20 | JONES  |
|     30 | MARTIN |
|     30 | ALLEN |
|     30 | TURNER |
|     30 | JAMES |
|     30 | WARD   |
|     20 | FORD   |
|     20 | SMITH |
|     20 | SCOTT |
|     20 | ADAMS |
|     10 | MILLER |
+--------+--------+
```

- the position of column in select statement will determine the position of column in the output
- you will write the select statement as per user requirements

TO RESTRICT ROWS :-

WHERE CLAUSE :

- where clause is used for searching
- searching takes place in DB server HD
- where clause is used to restrict the rows
- where clause is used to retrive the rows from DB server HD to server RAM
- ex :

```
mysql> select * from emp where deptno = 10;

+--------+-----------+--------+------+------------+
| DEPTNO | JOB       | ENAME | SAL | HIREDATE   |
+--------+-----------+--------+------+------------+
|     10 | PRESIDENT | KING   | 5000 | 1981-11-17 |
|     10 | MANAGER   | CLARK | 2450 | 1981-06-09 |
|     10 | CLERK     | MILLER | 1300 | 1982-01-23 |
+--------+-----------+--------+------+------------+
```

OPERATORS IN SQL

Relational operators :-

5. greater than (>)
6. greater than or equal to (>=)
7. less than (<)
8. less than or equal to(<=)
9. Not equal to (!= / <>)
10. equal to (=)

example :

```
select * from emp where sal > 2000 and sal < 3000;

+--------+---------+-------+------+------------+
| DEPTNO | JOB     | ENAME | SAL  | HIREDATE   |
+--------+---------+-------+------+------------+
|     30 | MANAGER | BLAKE | 2850 | 1981-05-01 |
|     10 | MANAGER | CLARK | 2450 | 1981-06-09 |
|     20 | MANAGER | JONES | 2975 | 1981-04-02 |
+--------+---------+-------+------+------------+
```

Logical operators :-

1. NOT
2. AND
3. OR

example :

```
select * from emp where deptno = 10 and deptno = 20;

Empty set (0.00 sec)

select * from emp where deptno = 10 or deptno = 20;
+--------+-----------+--------+------+------------+
| DEPTNO | JOB       | ENAME  | SAL  | HIREDATE   |
+--------+-----------+--------+------+------------+
|     10 | PRESIDENT | KING   | 5000 | 1981-11-17 |
|     10 | MANAGER   | CLARK  | 2450 | 1981-06-09 |
|     20 | MANAGER   | JONES  | 2975 | 1981-04-02 |
|     20 | ANALYST   | FORD   | 3000 | 1981-12-03 |
|     20 | CLERK     | SMITH  |  800 | 1980-12-17 |
|     20 | ANALYST   | SCOTT  | 3000 | 1982-12-09 |
|     20 | CLERK     | ADAMS  | 1100 | 1983-01-12 |
|     10 | CLERK     | MILLER | 1300 | 1982-01-23 |
+--------+-----------+--------+------+------------+
```

Arithmetic Operators :-

1. **( )**
2. ****** (used for exponentiation)
3. ex sal**3   ----> this operator supported by Oracle RDBMS.
4. (sal ^ 3). --- > this operator supported by MySQL RDBMS.
5. **+**
6. **-**

Note :-

- data is case-sensitive in MySQL RDBMS and Oracle RDBMS
- queries are case-insensitive in MySQL RDBMS
- queries are case-sensitive in Oracle RDBMS (secure about naming)

example :-

```
select * from emp where job = 'manager';
```

```
+--------+---------+-------+------+------------+
| DEPTNO | JOB     | ENAME | SAL  | HIREDATE   |
+--------+---------+-------+------+------------+
|     30 | MANAGER | BLAKE | 2850 | 1981-05-01 |
|     10 | MANAGER | CLARK | 2450 | 1981-06-09 |
|     20 | MANAGER | JONES | 2975 | 1981-04-02 |
+--------+---------+-------+------+------------+


select * from emp where job = 'MANAGER';

+--------+---------+-------+------+------------+
| DEPTNO | JOB     | ENAME | SAL  | HIREDATE   |
+--------+---------+-------+------+------------+
|     30 | MANAGER | BLAKE | 2850 | 1981-05-01 |
|     10 | MANAGER | CLARK | 2450 | 1981-06-09 |
|     20 | MANAGER | JONES | 2975 | 1981-04-02 |
+--------+---------+-------+------+------------+
```

sal*12

- not a column of emp table
- is known as a computed column (derived column)
- also known as a fake column (Pseudo column).

example :-

```
select ename , sal, sal*12 from emp;

+--------+------+--------+
| ename  | sal  | sal*12 |
+--------+------+--------+
| KING   | 5000 | 60000  |
| BLAKE  | 2850 | 34200  |
| CLARK  | 2450 | 29400  |
| JONES  | 2975 | 35700  |
| MARTIN | 1250 | 15000  |
| ALLEN  | 1600 | 19200  |
| TURNER | 1500 | 18000  |
| JAMES  | 950  | 11400  |
| WARD   | 1250 | 15000  |
| FORD   | 3000 | 36000  |
| SMITH  | 800  |   9600 |
| SCOTT  | 3000 | 36000  |
| ADAMS  | 1100 | 13200  |
| MILLER | 1300 | 15600  |
+--------+------+--------+
```

Alias :-

- alias means give the different name of existing column.
- ' as ' keyword its optional in MySQL RDBMS and Oracle RDBMS.

- Example with ' as '

```
select ename,sal,sal*12 as "ANNUAL" from emp;

+--------+------+--------+
| ename | sal | ANNUAL |
+--------+------+--------+
| KING   | 5000 | 60000 |
| BLAKE | 2850 | 34200 |
| CLARK | 2450 | 29400 |
| JONES | 2975 | 35700 |
| MARTIN | 1250 | 15000 |
| ALLEN | 1600 | 19200 |
| TURNER | 1500 | 18000 |
| JAMES | 950 | 11400 |
| WARD   | 1250 | 15000 |
| FORD   | 3000 | 36000 |
| SMITH | 800 |   9600 |
| SCOTT | 3000 | 36000 |
| ADAMS | 1100 | 13200 |
| MILLER | 1300 | 15600 |
+--------+------+--------+
```

- Example without ' as ' :-

```
select ename,sal,sal*12 "ANNUAL" from emp;

+--------+------+--------+
| ename | sal | ANNUAL |
+--------+------+--------+
| KING   | 5000 | 60000 |
| BLAKE | 2850 | 34200 |
| CLARK | 2450 | 29400 |
| JONES | 2975 | 35700 |
| MARTIN | 1250 | 15000 |
| ALLEN | 1600 | 19200 |
| TURNER | 1500 | 18000 |
| JAMES | 950 | 11400 |
| WARD   | 1250 | 15000 |
| FORD   | 3000 | 36000 |
| SMITH | 800 |   9600 |
| SCOTT | 3000 | 36000 |
| ADAMS | 1100 | 13200 |
| MILLER | 1300 | 15600 |
+--------+------+--------+
```

you cannot use alias in the WHERE clause

```
select ename,sal,sal*12 as "ANNUAL" from emp WHERE sal > 5000;   ----->  this will throw you error
```

## DISTINCT :

- distinct is use to filter duplicate data
- whenever you use DISTINCT, sorting takes place in server RAM

Example :

select distinct job from emp;

```
mysql> select distinct job from emp;


+-----------+
| job       |
+-----------+
| PRESIDENT |
| MANAGER   |
| SALESMAN  |
| CLERK     |
| ANALYST   |
+-----------+
```

UNIQUE is a synonym for distinct

distinct is common for all RDBMS but **UNIQUE** is works only in Oracle RDBMS

avoid using unique, use distinct, so that the same SELECT statement can be used in MySQL and others RDBMS.

DBMS

- In a RDBMS, data is stored in a file
- Data is stored sequentially in a DBMS (inside the file data is stored sequentially)

RDBMS

- In RDBMS, table is not a file
- every row is a file
- Rows of a table are not stored sequentially
- Rows of a table are scattered (fragmented) all over the DB server HD (this is common for all RDBMS).
- when you INSERT into a table, whenever the system finds the free space in the DB server HD, it will store the row there the reason that RDBMS does this, is to speed up the INSERT statement
- when you select a table, the order of rows in the output, depends on the row address

- it will always be in ascending order of row address when you UPDATE a row lenght is increasing ,then the entire may be moved to some other address later table, when you select from the you will find the row at some other position in the output
- it's only in case of VARCHAR that the row lenght may increase or decrease
- hence it is not possible to view the first or last ' N ' rows of tables

---

## ORDER BY CLAUSE

- order by clause is purposely use for sorting the column in ascending or descending order
- asc
- desc
- by default order by clause is in ascending order.

example :

```
select deptno, job,ename,sal,hiredate from emp order by ename asc;

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

| deptno | job       | ename  | sal  | hiredate   |

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

|     20 | CLERK     | ADAMS  | 1100 | 1983-01-12 |

|     30 | SALESMAN  | ALLEN  | 1600 | 1981-02-20 |

|     30 | MANAGER   | BLAKE  | 2850 | 1981-05-01 |

|     10 | MANAGER   | CLARK  | 2450 | 1981-06-09 |

|     20 | ANALYST   | FORD   | 3000 | 1981-12-03 |

|     30 | CLERK     | JAMES  |  950 | 1981-12-03 |

|     20 | MANAGER   | JONES  | 2975 | 1981-04-02 |

|     10 | PRESIDENT | KING   | 5000 | 1981-11-17 |

|     30 | SALESMAN  | MARTIN | 1250 | 1981-09-28 |

|     10 | CLERK     | MILLER | 1300 | 1982-01-23 |

|     20 | ANALYST   | SCOTT  | 3000 | 1982-12-09 |

|     20 | CLERK     | SMITH  |  800 | 1980-12-17 |

|     30 | SALESMAN  | TURNER | 1500 | 1981-09-08 |

|     30 | SALESMAN  | WARD   | 1250 | 1981-02-22 |

+--------+-----------+--------+------+------------+
```

- no upper limit on the number of columns that you can use in ORDER BY clause
- sorting is one operation that always slows down the SELECT statements
- sorting takes place in server RAM

```
select ename , sal*12  from emp order by sa*12

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

| ename  | sal*12 |

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

| SMITH  |   9600 |

| JAMES  |  11400 |

| ADAMS  |  13200 |

| MARTIN |  15000 |

| WARD   |  15000 |

| MILLER |  15600 |

| TURNER |  18000 |

| ALLEN  |  19200 |

| CLARK  |  29400 |

| BLAKE  |  34200 |

| JONES  |  35700 |

| FORD   |  36000 |

| SCOTT  |  36000 |

| KING   |  60000 |


mysql> select *  from emp order by 2;

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

| DEPTNO | JOB       | ENAME  | SAL  | HIREDATE   |

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

|     20 | ANALYST   | FORD   | 3000 | 1981-12-03 |

|     20 | ANALYST   | SCOTT  | 3000 | 1982-12-09 |

|     30 | CLERK     | JAMES  |  950 | 1981-12-03 |

|     20 | CLERK     | SMITH  |  800 | 1980-12-17 |

|     20 | CLERK     | ADAMS  | 1100 | 1983-01-12 |

|     10 | CLERK     | MILLER | 1300 | 1982-01-23 |

|     30 | MANAGER   | BLAKE  | 2850 | 1981-05-01 |

|     10 | MANAGER   | CLARK  | 2450 | 1981-06-09 |

|     20 | MANAGER   | JONES  | 2975 | 1981-04-02 |
|     10 | PRESIDENT | KING   | 5000 | 1981-11-17 |
|     30 | SALESMAN  | MARTIN | 1250 | 1981-09-28 |
|     30 | SALESMAN  | ALLEN  | 1600 | 1981-02-20 |
|     30 | SALESMAN  | TURNER | 1500 | 1981-09-08 |
|     30 | SALESMAN  | WARD   | 1250 | 1981-02-22 |
+--------+-----------+--------+------+------------+
```

- where clause is specified before the order by clause
- order by clause is the LAST clause in SELECT statement **(V. IMP)**.

MYSQL WORKBENCH

- GUI based
- when you install MySQL database s/w the ' root ' user is automatically created
- ' root ' user is having DBA privileges
- e. g creates new users, drop users, assign permissions, take backup,etc.

To Connect to MySQL databases :-

steps :

1. open MySQL workbench
2. MySQL connections (click on plus (+) sign to create new connection)
3. Connection Name :- Test connection for root user
4. method :- standard (TCP/IP)
5. Hostname :- Server I/p address or machine name local host
6. port :- 3306 is the standard port number Oracle (1521)
7. Username :- root
8. password :- (Store in vault ...push button) --> click on it ->> < password >
9. click on ok
10. Default schema :- (schema is synonym for Database)
11. Click on Test connection (push button)
12. close
13. after that you will see an object navigation on LHS
14. you will see an query window at top
15. you will see an output window at below

ctrl+enter --------> Execute program

use mySQL ;   ------ > use for select mysql

To see the List of user

```
select * from user
```

- user --> is a system table (63 system tables on MySQL)

- set of all system tables --> known as DATA DICTIONARY
- STORE all user name

HOW TO CREATE NEW USER

- create user < username > identified by < password >;
- create user Atul@'%' identified by 'abdavane';

Create Database or schema for atul

```
create database atul;

or

create  schema atul;
```

TO GRANT PERMISSIONS :-

1. click on server (menu at the top)
2. User and privileges click on it
3. select the user name that you created
4. click on administrative role from tab menu
5. select DBA role
6. click on Apply (push button)
7. select Atul schema
8. click on ok
9. click on ' select all '
10. click on ' Apply '
11. Exit from MySQL workbench

BLANK-PADDED COMPARISION SEMANTICS :-

- when you compare two strings of different light, the shorter of the two strings are temporarily padded with blank space so on RHS such that their lengths become equal
- then MySQL will do the comparison, character by character,based on ASCII value

Special operator :-

1) Like

wildcards

% ------> any character and any number of character

Example :-

```
------------------------------ This is from front----------------------------------------

select * from emp where ename like 's%';

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

| DEPTNO | JOB     | ENAME | SAL  | HIREDATE   |

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

|     20 | CLERK   | SMITH |  800 | 1980-12-17 |

|     20 | ANALYST | SCOTT | 3000 | 1982-12-09 |

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


------------------------------ This is from end----------------------------------------


select * from emp where ename like  '%s';

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

| DEPTNO | JOB     | ENAME | SAL  | HIREDATE   |

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

|     20 | MANAGER | JONES | 2975 | 1981-04-02 |

|     30 | CLERK   | JAMES |  950 | 1981-12-03 |

|     20 | CLERK   | ADAMS | 1100 | 1983-01-12 |


------------------------------ This is from amongs---------------------------------------

select * from emp where ename like  '%s%';

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

| DEPTNO | JOB     | ENAME | SAL  | HIREDATE   |

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

|     20 | MANAGER | JONES | 2975 | 1981-04-02 |

|     30 | CLERK   | JAMES |  950 | 1981-12-03 |

|     20 | CLERK   | SMITH |  800 | 1980-12-17 |

|     20 | ANALYST | SCOTT | 3000 | 1982-12-09 |

|     20 | CLERK   | ADAMS | 1100 | 1983-01-12 |
+--------+---------+-------+------+------------+
```

' _A% ' -------> second position of character

' _ _A% ' -------> third position of character

```
select * from emp where ename  not like  '%s%';

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

| DEPTNO | JOB       | ENAME  | SAL  | HIREDATE   |

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

|     10 | PRESIDENT | KING   | 5000 | 1981-11-17 |

|     30 | MANAGER   | BLAKE  | 2850 | 1981-05-01 |

|     10 | MANAGER   | CLARK  | 2450 | 1981-06-09 |

|     30 | SALESMAN  | MARTIN | 1250 | 1981-09-28 |

|     30 | SALESMAN  | ALLEN  | 1600 | 1981-02-20 |

|     30 | SALESMAN  | TURNER | 1500 | 1981-09-08 |

|     30 | SALESMAN  | WARD   | 1250 | 1981-02-22 |

|     20 | ANALYST   | FORD   | 3000 | 1981-12-03 |

|     10 | CLERK     | MILLER | 1300 | 1982-01-23 |

+--------+-----------+--------+------+------------+
```

2) Between

```
select * from  emp where sal between 1250 and 2850 order by deptno;

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

| DEPTNO | JOB      | ENAME  | SAL  | HIREDATE   |

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

|     10 | MANAGER  | CLARK  | 2450 | 1981-06-09 |

|     10 | CLERK    | MILLER | 1300 | 1982-01-23 |

|     30 | MANAGER  | BLAKE  | 2850 | 1981-05-01 |

|     30 | SALESMAN | MARTIN | 1250 | 1981-09-28 |

|     30 | SALESMAN | ALLEN  | 1600 | 1981-02-20 |

|     30 | SALESMAN | TURNER | 1500 | 1981-09-08 |

|     30 | SALESMAN | WARD   | 1250 | 1981-02-22 |

+--------+----------+--------+------+------------+
```

3) IN

IN ------> Logical OR

```
select * from emp  where deptno = 10 or deptno = 20 or deptno = 30;

or

select * from emp  where deptno in(10,20,30);

+--------+-----------+--------+------+------------+
| DEPTNO | JOB       | ENAME  | SAL  | HIREDATE   |
+--------+-----------+--------+------+------------+
|     10 | PRESIDENT | KING   | 5000 | 1981-11-17 |
|     30 | MANAGER   | BLAKE  | 2850 | 1981-05-01 |
|     10 | MANAGER   | CLARK  | 2450 | 1981-06-09 |
|     20 | MANAGER   | JONES  | 2975 | 1981-04-02 |
|     10 | CLERK     | MILLER | 1300 | 1982-01-23 |
+--------+-----------+--------+------+------------+


----------------------------------This is for not in-----------------------------------


select * from emp  where deptno not in(10,20);

+--------+----------+--------+------+------------+
| DEPTNO | JOB      | ENAME  | SAL  | HIREDATE   |
+--------+----------+--------+------+------------+
|     30 | MANAGER  | BLAKE  | 2850 | 1981-05-01 |
|     30 | SALESMAN | MARTIN | 1250 | 1981-09-28 |
|     30 | SALESMAN | ALLEN  | 1600 | 1981-02-20 |
|     30 | SALESMAN | TURNER | 1500 | 1981-09-08 |
|     30 | CLERK    | JAMES  |  950 | 1981-12-03 |
|     30 | SALESMAN | WARD   | 1250 | 1981-02-22 |
+--------+----------+--------+------+------------+
```

BETWEEN AND IN TOGETHER

example :

```
select *  from emp where sal between 900 and 3500 and deptno Not In(10,20);

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

| DEPTNO | JOB      | ENAME  | SAL  | HIREDATE   |

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

|     30 | MANAGER  | BLAKE  | 2850 | 1981-05-01 |

|     30 | SALESMAN | MARTIN | 1250 | 1981-09-28 |

|     30 | SALESMAN | ALLEN  | 1600 | 1981-02-20 |

|     30 | SALESMAN | TURNER | 1500 | 1981-09-08 |

|     30 | CLERK    | JAMES  |  950 | 1981-12-03 |

|     30 | SALESMAN | WARD   | 1250 | 1981-02-22 |

+--------+----------+--------+------+------------+
```

# UPDATE

- update is a DML command
- it is used to update the data in table

examples :

```
update emp set sal = 10000, city = 'pune' where empno = 1 ;


update emp set sal = 10000 where city = 'pune' ;


update emp set sal = sal + sal*0.4  where city = ' pune ';


update emp set sal = 10000 where sal = 1000;


update emp set sal = 10000;
```

- you can update multiple rows and column simultaneously, but you can update only 1 table at time if you want update multiple tables, you will have to write a separate UPDATE command for each table

---

# DELETE

- delete is also DML command
- used for delete rows in table
- it will only delete data not structure of table

example :

```
delete from emp where empno = 1;



delete from emp where city = 'pune';



delete from emp where city = 'pune' and .......;



delete form emp;
```

- **FROM ---> ANSI SQL**
- **FROM is compulsory in MySQL but optional in Oracle**

UPDATE DELETE commands without WHERE CLAUSE will not allowed in MySQL Workbench If you want to DELETE or UPDATE multiple rows :-

1. click on edit
2. preferences
3. SQL Editor
4. "safe updates " (ckechbox at the bottom)
5. uncheck it
6. click ok

# DROP

- drop is DDL command
- drop is used to delete table with structure
- you canot used where condition with drop table
- if you want to drop multiple tables, you will have to drop each table separately

Example :-

```
drop table < tablename >;
```

# TRANSACTION PROCESSING

COMMIT

- commit will save all the DML changes since the last committed state
- when the user issues a commit, it is known as End of Transaction
- commit wil make the Transaction permanent
- when oyu issue the commit, depends on the logical scope of work
- TOTAL WORK DONE = T1 + T2 + T3 + ...Tn;

syntax :

```
commit;

or

commit work;
```

WORK --> ANSI SQL

WORK --> optional in MySQL and Oracle

ROLLBACK

- Rollback will undo all the changes since the last committed state
- what has been commited, that cannot be rolledback
- only DML commands are affected by Rollback and commit
- when you EXIT from SQL , the system automatically commits
- any kind of power failure , network failure, system failure, PC reboot,window close , imporoper exit from SQL, etc. your last uncommited transaction is automatically Rolled back.
- any DDL command, the system automatically commits
- you can rollback to a Savepoint
- you cannot to a Savepoint
- commit will save all the DML changes since the last committed state
- you can only Rollback sequentially

syntax :

```
ROLLBACK;
```

savepoint syntax :

```
savepoint   < save point name >;
```

savepoint abc;

rollback to abc;

- when you commit or Rollback, the intermediate Savepoints are automatically cleared
- if you want to use them again, you will have to reissue them in some new work

Example for all TC :

```
mysql> insert into demo values ( 1,'prachi'),( 2,'pallu');


mysql> commit;


mysql> select * from  demo;

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

| id   | Name   |

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

|    1 | prachi |

|    2 | pallu  |

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


mysql> insert into demo values ( 3,'prachi');


mysql> select * from  demo;

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

| id   | Name   |

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

|    1 | prachi |

|    2 | pallu  |

|    3 | prachi |

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


mysql> rollback;
```

```
mysql> select * from  demo;

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

| id   | Name   |
+------+--------+
|    1 | prachi |
|    2 | pallu  |
+------+--------+



----------------------------------- Save-point -----------------------------------------


insert into demo values ( 1,'prachi');


 savepoint prachi;


insert into demo values ( 1,'pallu');


savepoint pallu;


insert into demo values ( 1,'sakshi');


savepoint sakshi;


insert into demo values ( 1,'pooja');


select * from  demo;
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    1 | pallu  |
|    1 | sakshi |
|    1 | pooja  |
+------+--------+
```

```
rollback to sakshi;


mysql> select * from  demo;
```

```
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    1 | pallu  |
|    1 | sakshi |
+------+--------+

rollback to pallu;

mysql> select * from  demo;
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    1 | pallu  |
+------+--------+

 select *,job as job1 from emp;
+--------+-----------+--------+-------+------------+-----------+
| DEPTNO | JOB       | ENAME  | SAL   | HIREDATE   | job1      |
+--------+-----------+--------+-------+------------+-----------+
|     10 | PRESIDENT | KING   |  5000 | 1981-11-17 | PRESIDENT |
|     30 | MANAGER   | BLAKE  |  2850 | 1981-05-01 | MANAGER   |
|     10 | MANAGER   | CLARK  |  2450 | 1981-06-09 | MANAGER   |
|     20 | MANAGER   | JONES  |  2975 | 1981-04-02 | MANAGER   |
+--------+-----------+--------+-------+------------+-----------+
```

DROP	**DELETE**	**TRUNCATE**
use to delete database table with structure.	used to delete specific row.	used to delete all data from table.
Data Can not Rollback.	Data Can Rollback.	Data Can not Rollback.
DDL	DML	DDL
Slower	slower	fast

NOTE :- if we take int or not provide size the it will take by default range of int and 10 bit long length

ZEROFILL :

- when you select a column with type zerofill it pads the display value of the filed with zeros up to the display width specified in the column defination.
- using zerofill and a display width has no effect on how the data is sorted, it affect only how it is display

```
create table demo2(id int zerofill ,name char);

insert into demo2 values( 1,'a');

select * from demo2;

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

| id         | name |

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

| 0000000001 | a    |

+------------+------+
```

TOP,LIMIT,ROWNUM :-

- this all statements we used for select the data from top.
- only Limit allow in MySQL

Limit

```
mysql> select * from emp limit 5;

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

| DEPTNO | JOB       | ENAME  | SAL  | HIREDATE   |

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

|     10 | PRESIDENT | KING   | 5000 | 1981-11-17 |

|     30 | MANAGER   | BLAKE  | 2850 | 1981-05-01 |

|     10 | MANAGER   | CLARK  | 2450 | 1981-06-09 |

|     20 | MANAGER   | JONES  | 2975 | 1981-04-02 |
|     30 | SALESMAN  | MARTIN | 1250 | 1981-09-28 |

+--------+-----------+--------+------+------------+
```

READ WRITE CONSISTENCY

- when you SELECT from a table, you can view ONLY the committed data of other users plus
- changes made by you

ROW LOCKING

- when you UPDATE or DELETE a row, that row is automtically locked for other users
- when you UPDATE or DELETE a row, that row becomes READ_ONLY for other users
- Row LOCKING IN MySQL and ORACLE is AUTOMATIC
- other users can SELECT from that table (they wil view the old data before your changes)
- other users insert into the table
- other users can UPDATE or DELETE ' others ' rows
- no other user can UPDATE or DELETE your locked row, till you have issued a rollback or commit
- LOCKS AUTOMATICALLY RELEASED WHEN YOU ROLLBACK OR COMMIT
- there is alsways have one request queue for waiting commit requests, this is follow FIFO principle.

example :

```
###### USER-1 #####

select * from demo;

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

| id   | Name   |

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

|    1 | prachi |

|    2 | pallu  |

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



mysql> set autocommit = 0;
```

```
mysql> insert into demo values (3,'sakshi');



mysql> select * from demo;

+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    2 | pallu  |
|    3 | sakshi |
+------+--------+

mysql> commit;

mysql> update demo set name = 'dimple' where id = 2;

mysql> select * from demo;
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    2 | dimple |
|    3 | sakshi |
+------+--------+

mysql> commit;

mysql> commit;

mysql> select * from demo;
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    2 | pallu  |
|    3 | sakshi |
+------+--------+

mysql> update demo set name = 'dimple' where id = 2;
mysql> commit;

####### USER-2 #######

mysql> select * from demo;
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    2 | pallu  |
+------+--------+

mysql> set autocommit = 0;

mysql> select * from demo;
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    2 | pallu  |
+------+--------+

mysql> commit;

mysql> select * from demo;
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    2 | pallu  |
|    3 | sakshi |
+------+--------+

mysql> select * from demo;
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    2 | pallu  |
|    3 | sakshi |
+------+--------+

mysql> commit;

mysql> select * from demo;
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    2 | dimple |
|    3 | sakshi |
+------+--------+

mysql> update demo set name = 'pallu' where id = 2;

mysql> select * from demo;
+------+--------+
| id   | Name   |
+------+--------+
|    1 | prachi |
|    2 | pallu  |
|    3 | sakshi |
+------+--------+
```

```
mysql> commit;
```

```
mysql> update demo set name = 'pallu' where id = 2;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
mysql> update demo set name = 'pallu' where id = 2;
mysql> select * from demo;

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

| id   | Name   |

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

|    1 | prachi |

|    2 | pallu  |

|    3 | sakshi |

+------+--------+
```

OPTIMISTIC ROW LOCKING

- automatic row locking mechanism in MySQL and Oracle

PESSIMISTIC ROW LOCKING

- manually lock the rows the table BEFORE issuing UPDATE or DELETE
- To lock the rows manually , you will have to use SELECT statement with the FOR UPDATE clause

ex :

select * from emp FOR UPDATE;

```
select * from emp

where deptno = 10

FOR UPDATE wait 60 ;   <-- by default



60 ---- seconds ( time for wait)



select * from emp

where deptno = 10

FOR UPDATE wait;
```

- LOCKS are AUTOMATICALLY released when you ROLLBACK or COMMIT
- WAIT/NOWAIT options not available in MySQL

ex :

```
##### USER-1 ######

mysql> select * from emp;
```

```
+--------+-----------+--------+-------+------------+
| DEPTNO | JOB       | ENAME  | SAL   | HIREDATE   |
+--------+-----------+--------+-------+------------+
|     10 | PRESIDENT | KING   |  5000 | 1981-11-17 |
|     30 | MANAGER   | BLAKE  |  2850 | 1981-05-01 |
|     10 | MANAGER   | CLARK  |  2450 | 1981-06-09 |
|     20 | MANAGER   | JONES  |  2975 | 1981-04-02 |
|     30 | SALESMAN  | MARTIN |  1250 | 1981-09-28 |
|     30 | SALESMAN  | ALLEN  |  1600 | 1981-02-20 |
|     30 | SALESMAN  | TURNER |  1500 | 1981-09-08 |
|     30 | CLERK     | JAMES  |   950 | 1981-12-03 |
|     30 | SALESMAN  | WARD   |  1250 | 1981-02-22 |
|     20 | ANALYST   | FORD   |  3000 | 1981-12-03 |
|     20 | CLERK     | SMITH  |   800 | 1980-12-17 |
|     20 | ANALYST   | SCOTT  |  3000 | 1982-12-09 |
|     20 | CLERK     | ADAMS  |  1100 | 1983-01-12 |
|     10 | CLERK     | MILLER |  1300 | 1982-01-23 |
|     10 | manager   | prachi | 34222 | 2020-02-11 |
|     10 | manager   | sakshi | 34222 | 2020-02-11 |
|     10 | manager   | pallu  | 34222 | 2020-02-11 |
+--------+-----------+--------+-------+------------+
17 rows in set (0.00 sec)


mysql> select * from emp where deptno = 10 For UPDATE;
+--------+-----------+--------+-------+------------+
| DEPTNO | JOB       | ENAME  | SAL   | HIREDATE   |
+--------+-----------+--------+-------+------------+
|     10 | PRESIDENT | KING   |  5000 | 1981-11-17 |
|     10 | MANAGER   | CLARK  |  2450 | 1981-06-09 |
|     10 | CLERK     | MILLER |  1300 | 1982-01-23 |
|     10 | manager   | prachi | 34222 | 2020-02-11 |
|     10 | manager   | sakshi | 34222 | 2020-02-11 |
|     10 | manager   | pallu  | 34222 | 2020-02-11 |
+--------+-----------+--------+-------+------------+
6 rows in set (0.01 sec)
```

```
mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from emp where deptno = 20 For UPDATE;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
mysql> select * from emp where deptno = 20 For UPDATE;
+--------+---------+-------+------+------------+
| DEPTNO | JOB     | ENAME | SAL  | HIREDATE   |
+--------+---------+-------+------+------------+
|     20 | MANAGER | JONES | 2975 | 1981-04-02 |
|     20 | ANALYST | FORD  | 3000 | 1981-12-03 |
|     20 | CLERK   | SMITH |  800 | 1980-12-17 |
|     20 | ANALYST | SCOTT | 3000 | 1982-12-09 |
|     20 | CLERK   | ADAMS | 1100 | 1983-01-12 |
+--------+---------+-------+------+------------+
5 rows in set (0.00 sec)

##### USER-2 ######
mysql> select * from emp;
+--------+-----------+--------+-------+------------+
| DEPTNO | JOB       | ENAME  | SAL   | HIREDATE   |
+--------+-----------+--------+-------+------------+
|     10 | PRESIDENT | KING   |  5000 | 1981-11-17 |
|     30 | MANAGER   | BLAKE  |  2850 | 1981-05-01 |
|     10 | MANAGER   | CLARK  |  2450 | 1981-06-09 |
|     20 | MANAGER   | JONES  |  2975 | 1981-04-02 |
|     30 | SALESMAN  | MARTIN |  1250 | 1981-09-28 |
|     30 | SALESMAN  | ALLEN  |  1600 | 1981-02-20 |
|     30 | SALESMAN  | TURNER |  1500 | 1981-09-08 |
|     30 | CLERK     | JAMES  |   950 | 1981-12-03 |
|     30 | SALESMAN  | WARD   |  1250 | 1981-02-22 |
|     20 | ANALYST   | FORD   |  3000 | 1981-12-03 |
|     20 | CLERK     | SMITH  |   800 | 1980-12-17 |
|     20 | ANALYST   | SCOTT  |  3000 | 1982-12-09 |
|     20 | CLERK     | ADAMS  |  1100 | 1983-01-12 |
|     10 | CLERK     | MILLER |  1300 | 1982-01-23 |
|     10 | manager   | prachi | 34222 | 2020-02-11 |
|     10 | manager   | sakshi | 34222 | 2020-02-11 |
|     10 | manager   | pallu  | 34222 | 2020-02-11 |
+--------+-----------+--------+-------+------------+
17 rows in set (0.00 sec)


mysql> select * from emp where deptno = 10 For UPDATE NOWAIT;
ERROR 3572 (HY000): Statement aborted because lock(s) could not be acquired immediately and NOWAIT is set.
mysql> select * from emp where deptno = 10 For UPDATE NOWAIT;
+--------+-----------+--------+-------+------------+
| DEPTNO | JOB       | ENAME  | SAL   | HIREDATE   |
+--------+-----------+--------+-------+------------+
|     10 | PRESIDENT | KING   |  5000 | 1981-11-17 |
|     10 | MANAGER   | CLARK  |  2450 | 1981-06-09 |
|     10 | CLERK     | MILLER |  1300 | 1982-01-23 |
|     10 | manager   | prachi | 34222 | 2020-02-11 |
|     10 | manager   | sakshi | 34222 | 2020-02-11 |
|     10 | manager   | pallu  | 34222 | 2020-02-11 |
+--------+-----------+--------+-------+------------+
6 rows in set (0.00 sec)


mysql> commit
```

for workbench open new thab for currentb server

click on ' **Query** ' at menu ---> click on ' **New tab to current server** '

FUNCTIONS

SINGLE- ROW FUNCTIONS

concat

```
mysql> select concat(fname,lname) as fullname from emp1;

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

| fullname    |

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

| Atulpatil   |

| sakshipatil |

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


### take space between  ( this is support to All RDBMS )


mysql> select concat(concat(fname,' '),lname) as fullname from emp1;

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

| fullname     |

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

| Atul patil   |

| sakshi patil |
```

```
 ## ( but this is not support to Oracle) ##
```

```
mysql> select concat(fname,' ',lname) as fullname from emp1;

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

| fullname     |

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

| Atul patil   |
| sakshi patil |
+--------------+


mysql> select concat(fname,null,lname) as fullname from emp1;
+----------+
| fullname |
+----------+
| NULL     |
| NULL     |
+----------+
```

for character function max upto 255 levels for function within function

UPPER

- it is make the character in UPPERCASE

```
mysql> select upper(fname) from emp1;


+--------------+
| upper(fname) |
+--------------+
| ATUL         |
| SAKSHI       |
+--------------+
```

INITCAP

- select initcap(fname) from emp1;
- *INITCAP is not supported in MySQL (support by Oracle)*

LPAD

- it will used to right justification.
- left side of our character print start except the length of our data

ex :

```
mysql> select lpad(fname, 25,'*')from emp1;

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

| lpad(fname, 25,'*')       |

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

| *********************Atul |

| *******************sakshi |

+---------------------------+
```

RPAD

- it will used to left justification.
- left side of our character print start except the length of our data
- to covert varchar to char

```
mysql> select rpad(fname, 10,'*')from emp1;



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

| rpad(fname, 10,'*') |

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

| Atul******          |

| sakshi****          |

+---------------------+
```

FIELD

ELT

- it will give us column whichever you want in below example you will got first column

```
mysql> select elt(1,fname,lname) from emp1;

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

| elt(1,fname,lname) |

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

| Atul               |

| sakshi             |

+--------------------+
```

LTRIM

- o/p is left justified

```
+--------------+
| rtrim(fname) |
+--------------+
| Atul         |
| sakshi       |
|     prachi   |
+--------------+
3 rows in set (0.00 sec)

mysql> select ltrim(fname) from emp1;
+--------------+
| ltrim(fname) |
+--------------+
| Atul         |
| sakshi       |
| prachi       |
+--------------+
```

RTRIM

- o/p is right justified

TRIM

- o/p is center justified

LOCATE

- it is return the position of character from particular tuple .

```
mysql> select  locate('a',fname) from emp1 where fname = 'Atul' ;
```

```
+-------------------+
| locate('a',fname) |
+-------------------+
|                 1 |
+-------------------+
```

SUBSTR

- use to extract the part of the string

```
mysql> select substr(name,2) from demo;

+----------------+
| substr(name,2) |
+----------------+
| rachi          |
| allu           |
| akshi          |
+----------------+

3 ---->  starting position

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

mysql> select substr(name,3,2) from demo;
+------------------+
| substr(name,3,2) |
+------------------+
| ac               |
| ll               |
| ks               |
+------------------+
3 ---->  starting position
2 ---->  number of character

mysql> select substr(name,-3) from demo;
```

```
+-----------------+
| substr(name,-3) |
+-----------------+
| chi             |
| llu             |
| shi             |
+-----------------+
mysql> select substr(name,-3,2) from demo;
+-------------------+
| substr(name,-3,2) |
+-------------------+
| ch                |
| ll                |
| sh                |
+-------------------+
```

REPLACE

- replace is use for replace the particular character or string.

```
mysql> select replace(name,'p','s') from demo;
+-----------------------+
| replace(name,'p','s') |
+-----------------------+
| srachi                |
| sallu                 |
| sakshi                |
+-----------------------+
```

```
select replace(replace(name,'p','s'),'s','p') from demo;

mysql> select replace(replace(name,'p','s'),'s','p') from demo;
+----------------------------------------+
| replace(replace(name,'p','s'),'s','p') |
+----------------------------------------+
| prachi                              |

| pallu                               |

| pakphi                              |

+----------------------------------------+
```

TRANSLATE

- Translate is not supported by MySQL (available in oracle).

```
select translate(name,'pa','sr') from demo;


p  ---- >   s


a  ----- >  r
```

ENSTR

- it is return the starting position of the string

```
mysql> select instr(name,'k') from demo;


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

| instr(name,'k') |

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

|               0 |

|               0 |

|               3 |

+-----------------+
```

LENGTH

- find the length of string

```
mysql> select length(name) from demo;
```

```
+--------------+
| length(name) |
+--------------+
|            6 |
|            5 |
|            6 |
+--------------+
```

ASCII

- find the ascii value of particular character

```
mysql>  select ASCII(name) from demo;
+-------------+
| ASCII(name) |
+-------------+
|         112 |
|         112 |
|         115 |
+-------------+

mysql> select ASCII('z') from demo;
+------------+
| ASCII('z') |
+------------+
|        122 |
|        122 |
|        122 |
+------------+
3 rows in set (0.00 sec)

mysql> select distinct ASCII('z') from demo;
+------------+
| ASCII('z') |
+------------+
|        122 |

+------------+
```

DUAL

- dual is a system table
- contain only one row and one column
- is a dummy table
- is present in all RDBMS
- set of system table ----- > known as DATA DICTIONARY, known as DATABASE CATALOG
- 63 system tables in mysql
- 2000 in Oracle

select char(65 using utf8) from dual;

where utf8 is the given character set for U.S ENGLISH else default is binary character set

insert

```
mysql> select insert('atul',1,2,'sakshi');

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

| insert('atul',1,2,'sakshi') |

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

| sakshiul                    |

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



1 ------- > starting position of character



2 ------- > count of character
```

```
mysql> select insert('atulpatil',1,3,'prachi');
```

```
+----------------------------------+

| insert('atulpatil',1,3,'prachi') |

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

| prachilpatil                     |

+----------------------------------+
```

how to print name of emp except start with vowels

```
SELECT DISTINCT ename FROM emp WHERE ename NOT= RLIKE '^[aeiouAEIOU].*$';


+--------+
| ename  |
+--------+
| KING   |
| BLAKE  |
| CLARK  |
| JONES  |
| MARTIN |
| TURNER |
| JAMES  |
| WARD   |
| FORD   |
| SMITH  |
| SCOTT  |
| MILLER |
| prachi |
| sakshi |
| pallu  |
+--------+
```

DATE FUNCTION

SYSDATE

- its return the current server date and time

```
 mysql> select sysdate();
+---------------------+
| sysdate()           |
+---------------------+
| 2020-07-22 15:32:24 |
+---------------------+
```

NOW

- its return the date and time when server start the process

```
mysql> select now();
+---------------------+
| now()               |
+---------------------+
| 2020-07-22 15:33:54 |
+---------------------+
```

ADDDATE

- we can add the days in given date

```
mysql> select adddate(hiredate,20)from emp;
+----------------------+
| adddate(hiredate,20) |
+----------------------+
| 1981-12-07           |
| 1981-05-21           |
+----------------------+
```

ADDTIME

- it will add seconds

```
select addtime( '2020-01-15 10:00:00 ',' 1 ') from dual;
+----------------------------------------+
| addtime( '2020-01-15 10:00:00 ',' 1 ') |
+----------------------------------------+
| 2020-01-15 10:00:01                    |
+----------------------------------------+
```

## LIST FUNCTIONS (independent of datatype)

- char,Number,Date functions
- any comparision done with null, return null.
- pessimistic quering --> searching for null values
- select * from emp where comm = null ;
- select * from emp where comm is null ;
- select * from emp where comm not null ;
- select * from emp where comm != null ;

select sal+comm form emp;

IFNULL

select sal + ifnull(comm,0) from emp;

```
if comm is null then
            return 0;
else
            return comm;
endif;
```

ifnull(comm,100)

if null(comm,mumbai)

GREATEST , LEAST

find the greatest sal as compare to 3000.

```
select  greatest(sal,3000) from emp;
```

find the least sal as compare to 3000.

```
select  greatest(sal,3000) from emp;
```

CASE STATEMENTS

```
when deptno = 10 then  ' Training'
when deptno = 20 then  ' Exports'
when deptno = 30 then  ' Sales'
else ' others'
end
from emp;
```

## ENVIRONMENT FUNCTION

USER

- user is use to select DB user

```
select user( ) from dual;
```

MULTI - ROW FUNCTIONS (group by function / aggrigate function)

SUM

use to add the full all salary

```
mysql> select  sum(sal) from emp;
+----------+
| sum(sal) |
+----------+
|   131691 |
+----------+
```

AVERAGE

is return the average of salary

```
mysql> select  avg(sal) from emp;

| avg(sal)  |

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

| 7746.5294 |

+-----------+
```

MIN, MAX

```
mysql> select min( sal ) from emp;

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

| min( sal ) |

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

|        800 |

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

mysql> select max( sal ) from emp;

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

| max( sal ) |

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

|      34222 |

+------------+
```

NOTE - the null value is not counted by group function

```
it will return the rows where sal is not null


mysql> select count(sal )from emp;

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

| count(sal ) |

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

|          17 |

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

mysql> select max(Sal)/min(sal) from emp;

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

| max(Sal)/min(sal) |

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

|           42.7775 |
+-------------------+
```

You can not select regular column with group function

```
this will work in MySQL but error in oracle

mysql> select ename,min(sal) from emp;

+-------+----------+
| ename | min(sal) |
+-------+----------+
| KING  |      800 |
+-------+----------+
```

the below query will give error in oracle and MySQL ass well

```
mysql> select *  from emp  where sal > avg(sal);
```

GROUP BY CLUASE

Group by and Having clause The GROUP BY clause is a SQL command that is used to group rows that have the same values. The GROUP BY clause is used in the SELECT statement .Optionally it is used in conjunction with aggregate functions to produce summary reports from the database.

SELECT CLUASE ---> select the perticular column

FROM CLUSE ---> tells the int which you have to table perform the operation

GROUP BY CLUASE ---> group by cluase is grouping the data of column

IN FOUR STEPS GROUP BY CLAUSE IS WORK

1. Rows retrive from DB server HD to server RAM
2. sorting with column wise whichever you passed in group by clause
3. grouping of rows wise whichever you passed in group by clause
4. summation/calculation
5. having clause

6. order by

RULES

#1 Besides the group function, whichever column is present in SELECT cluase,it HAS to be present in GROUP BY cluase.

```
select deptno , sum(sal) from emp;          <----- error in  oracle ( works in mysql but it is meaningless  )

select deptno , sum(sal) from emp group by deptno;     <-------- right way

select  sum(sal) from emp group by deptno;
```

2D Query :- any select statement with group by cluase is known as a 2D Query .because you can plot the graph form the input.(oracle is beat for graphing).

```
mysql> select  sum(sal) from emp where sal > 2000 group by deptno ;
```

NOTE :- WHERE CLUASE has to specified before the GROUP BY clause

There is not upper limit to the number of columns in group by clause

```
mysql> select job,deptno, sum(sal) from emp  group by deptno,job ;
```

but if you use large number of rows in the table and if you have large number of column in GROUP BY clause, then the execution will be very slow

```
+--------+-----------+--------+-------+------------+
| DEPTNO | JOB       | ENAME  | SAL   | HIREDATE   |
+--------+-----------+--------+-------+------------+
|     10 | PRESIDENT | KING   |  5000 | 1981-11-17 |
|     30 | MANAGER   | BLAKE  |  2850 | 1981-05-01 |
|     10 | MANAGER   | CLARK  |  2450 | 1981-06-09 |
|     20 | MANAGER   | JONES  |  2975 | 1981-04-02 |
|     30 | SALESMAN  | MARTIN |  1250 | 1981-09-28 |
|     30 | SALESMAN  | ALLEN  |  1600 | 1981-02-20 |
|     30 | SALESMAN  | TURNER |  1500 | 1981-09-08 |
|     30 | CLERK     | JAMES  |   950 | 1981-12-03 |
|     30 | SALESMAN  | WARD   |  1250 | 1981-02-22 |
|     20 | ANALYST   | FORD   |  3000 | 1981-12-03 |
|     20 | CLERK     | SMITH  |   800 | 1980-12-17 |
|     20 | ANALYST   | SCOTT  |  3000 | 1982-12-09 |
|     20 | CLERK     | ADAMS  |  1100 | 1983-01-12 |
|     10 | CLERK     | MILLER |  1300 | 1982-01-23 |
|     10 | manager   | prachi | 34222 | 2020-02-11 |
|     10 | manager   | sakshi | 34222 | 2020-02-11 |
|     10 | manager   | pallu  | 34222 | 2020-02-11 |
+--------+-----------+--------+-------+------------+
```

FASTER AS FOLLOWED BY ABOVE TABLE (because deptno is less as compare to jobs).

```
mysql> select deptno,job, sum(sal) from emp  group by deptno , job ;
```

SLOWER AS FOLLOWED BY ABOVE TABLE (because deptno is less as compare to jobs).

```
mysql> select deptno,job, sum(sal) from emp  group by deptno , job ;
```

- position of column in SELECT cluase and the position of column in GROUP BY cluase, need not be the same
- position of the column in select cluase will determine the position of column in the output (you shall determine the position of column in select cluase as per user requirements)
- position of the column in GROUP BY cluase will determine the sorting order grouping order summation order,and hence the speed processing.

GROUP BY city, country, district, state <-------------- SLOW

GROUP BY country, state, district, city <-------------- FAST

HAVING CLUASE

- main objective to introduce having cluase is we could not be use where function with aggrigate functions or group functions .
- having clause works AFTER the summation is done
- its recommmended that only group functions should be used in HAVING clause.

```
mysql> select deptno, sum(sal) from emp  group by deptno having sum(sal) > 950 ;
```

ORDER BY

then last you can take order by clause

```
mysql> select deptno, sum(sal) from emp  group by deptno having sum(sal) < 11000 order by deptno ;
```

follow this order

GROUP BY , HAVING, ORDER BY

Nesting of GROUP functions are supported in oracle (but not in MySQL).

```
mysql>  select max(sum(sal)) from emp;  ------------->   error in MySQL      -------> FAST
```

Solution For This But Difficult And Slow

```
mysql>  select max(sum_sal) from (select sum(sal) sum_sal from  emp group by deptno) as tempp; ----------------------> SLOW
```

JOINS

- to view/combine the column of 2 or more tables

Redundancy :- unneccessary duplication of data (leads to wastage of time).

```
*select dname,ename from emp2, dept where dept.deptno = emp2. deptno;*     ----------->  FAST
```

in above Query

from emp2,dept (**dept ----> Driving table , emp2 ---> Driven table**)

dept.deptno (**dept -----> tablename**, **deptno-----> columnname**)

```
select dname,ename from  dept,emp2 where dept.deptno = emp2. deptno;          -----------> SLOW



select dname,ename from emp2, dept where emp2.deptno = dept. deptno;          ------------> SLOW
```

IN ORDER FOR THE JOIN TO WORK FASTER, PREFERABLY THE DRIVING TABLE SHOULD BE TABLE WITH " LESSER " NUMBER OF ROWS

```
mysql> select dname,ename,location,empno,job,sal, deptfrom  dept,emp2 where dept.deptno = emp2. deptno ORDER BY 1;



this will throw ambiguaty error because of dept
```

this is very good practice for write query

```
mysql> select dept.dname,emp2.ename,dept.location,emp2.empno,emp2.job,emp2.sal from emp2,dept where dept.deptno = emp2. deptno ORDER BY 1;
```

TYPES OF JOINS

EQUI-JOIN (also known as Natural Join) :-

- join based on equality condition
- shows matching rows of both the tables
- most frequently used join (> 90%)

ex :-

```
select dname, ename from emp, dept where dept.deptno = emp.deptno;
```

INEQUI-JOIN (also known as Non-Equi-join) :-

- join based on inequality condition

ex :-

```
select dname, ename from emp, dept where dept.deptno != emp.deptno;
```

OUTER-JOIN

1) Half Outer-join :-

- show matching rows of both the tables plus non matching rows of " outer " table .
- "Outer " table -- > table which is Outer side of (+) sign.

LeftOuterJoin

```
################# This Query is not work in MySQL #######################



select dname, ename from emp, dept where dept.deptno = emp2.deptno ( + );
```

RightOuterJoin

```
################# This Query is not work in MySQL #######################



select dname, ename from emp, dept where dept.deptno ( + ) = emp2.deptno ;
```

We have two tables **DEPT** and **EMP2**

- **DEPT** is mastre table or parent table or independent table
- **EMP2** is details table or child table or dependent table

```
#### EMP2 #####



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

| EMPNO | ENAME  | SAL  | DEPTNO | job  | MGR  |

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

|     1 | Arun   | 8000 |      1 | M    |    4 |

|     2 | Ali    | 7000 |      1 | C    |    1 |

|     3 | kiran  | 3000 |      1 | C    |    1 |

|     4 | jack   | 9000 |      2 | M    | NULL |

|      5| Thomas | 8000 |      1 | C    |    4 |
|     6 | Atul   | 10000|      5 | D    |    3 |
+-------+--------+------+--------+------+------+
##### DEPT #####
+--------+-----------+----------+
| deptno | dname     | location |
+--------+-----------+----------+
|      1 | training  | mumbai   |
|      2 | Exp       | Abad     |
|      3 | marketing | nashik   |
+--------+-----------+----------+
```

2) FullOuterJoin

- based on nested do-while loop
- shows matching rows of both the tables plus non-matching rows of both the tables

```
######### This is only for Oracle ###########


select dname, ename from emp, dept where dept.deptno = emp2.deptno ( + );
                                                                  union
select dname, ename from emp, dept where dept.deptno ( + ) = emp2.deptno;
```

ANSI Syntax for LEFT OuterJoin :-

```
select dname, ename from emp2 left outer join dept on ( dept.deptno = emp2.deptno );
```

ANSI Syntax for RIGHT OuterJoin :-

```
select dname, ename from emp2 right outer join dept on ( dept.deptno = emp2.deptno );
```

ANSI Syntax for FULL OuterJoin :- (this is supported by all RDBMS except MySQL)

```
select dname, ename from emp2 full outer join dept on ( dept.deptno = emp2.deptno );
```

To achive Full Outer Join in MySQL then you will have to use UNION of ANSI syntax of RIGHT and LEFT Outer Joins

```
select dname, ename from emp2 left outer join dept on ( dept.deptno = emp2.deptno )

                                                                                union

select dname, ename from emp2 right outer join dept on ( dept.deptno = emp2.deptno );
```

INNER-JOIN

do not mention in interviews, unless asked

by default, every join is inner join putting a (+) sign is what makes it an Outer join

CARTESIAN-JOIN

- join without a WHERE condition
- every row of driving table is combined with each and every row of driven table
- cartesian join used to printing purposes :- e. g (Student and marksheet)

```
                                      < -----  ( it is always work like that outer loop to inner loop

select dname,ename from emp,dept;     ------ > FAST( lesser I/O between server HD and q



select dname,ename from emp,dept;     ------ > SLOW( more I/O between server HD and q
```

SELF-JOIN

- joining a table to itself
- used when parent-column and child column both are present in the same table
- self-join is the slowest join

```
select dname , ename from emp E , dept D  WHERE  D.deptno =  E.deptno;       --> NOT



select a.ename , b.ename from emp2 b , emp2 a  WHERE  a.mgr =  b.empno;
```

Which one is fastest join ?

Cartesian join is the fastest join; there is no WHERE cluase involved, hence there is no serching involved.

MULTIPLE TABLES

```
select dname, ename,headname  from  emp2 , dept ,depthead  where  (depthead.deptno = dept.deptno  and  dept.deptno=
emp2.deptno) ;
```

- no upper limit on the number tables that you can join

Types of Relationships :-

one : one

- (Dept : DeptHead) or (DeptHead : Dept)

one : many

- (Dept : Emp) and (Depthead : Emp)

many : one

- (Emp : Dept) and (Emp : Depthead)

many : many

- (projects : Emp) or (Emp : Projects)

Intersaction Table : - required for many : many relationship

```
select pname ,cname,ename from emp,projects,projects_emp where projects.projno = projects_emp.projno and emp.empno =
proejcts_emp.empno;
```

Sub Queries

- Nested Query / Select within select / Query within Query.
- max upto 255 levels for sub-queries
- try to reduse the number of sub select statements because it become execution slow
- join faster that query

Problem Statement

1) How to Display Ename Who is receiving the min(sal) :-

```
select ename from emp where sal =          --------- > Main Query
( select min(Sal) from emp);                                    --------- > Sub Query
```

2) Display the rows that belong to the same DEPTNO as ' Thomas ' :

```
select * from emp where deptno >
( select  deptno from emp where ename = ' Adams' );
#### This following queries usnig sub query with DML is  work only in Oracle NOt supported by MySQL ###


delete from emp where deptno =
( select  deptno from emp where ename = ' Adams' );


update emp set where sal = 20000  where deptno =
( select  deptno from emp where ename = ' Adams' );


#### Solution ####


delete from emp where deptno =
( select  tempp.deptno from (select deptno from emp where ename = 'pullu' ) as tempp );




UPDATE emp set sal = 20000 where deptno =
( select  tempp.deptno from (select deptno from emp where ename = 'Adams' ) as tempp );
```

Using Sub-Query in Having clause

In Oracle :-

Display the dname that is HAVING the max(sum(sal)) :

```
select deptno, sum(sal) from dept.deptno= emp.deptno group by dname having sum(sal) = (select max(sum(sal)) from emp
group by deptno );
```

In MySQL :-

Display the dname that is HAVING the max(sum(sal)) :

```
select max(SUM_SAL) from ( select sum(sal)  SUM_SAL  from emp group by deptno ) as tempp;
select dname, sum(sal) from emp2, dept where dept.deptno = emp2.deptno group by dname having sum(sal) = (select max(SUM_SAL) from (select sum(sal) SUM_SAL from emp2 group by deptno ) as tempp );

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

| dname    | sum(sal) |

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

| training |    26000 |

+----------+----------+
```

QUERY RETURNS MULTIPLE ROWS

ANY

```
######



select * from emp2 where sal = ANY ( select sal from emp2 where job = 'M' );



select * from emp2 where sal >= ANY ( select sal from emp2 where job = 'M' );



select * from emp2 where sal <= ANY ( select sal from emp2 where job = 'M' );



select * from emp2 where sal != ANY ( select sal from emp2 where job = 'M' );
```

IN

```
select * from emp2 where sal IN ( select sal from emp2 where job = 'M' );
```

NOTE :-

- IN is faster than ANY
- ANY is more powerful than IN operator
- with IN operator, we can only check for IN, or NOT IN with ANY operator, we can check (=,ANY !=ANY, <=ANY, >=ANY, >ANY, <ANY)
- if you want to check fro equality or inequality, use the IN operator
- if you want to check for >,>=,<,<= the use the ANY operator

ALL

- all operator return trueif all subquery values meet the condition

```
select * from emp2 where sal > ALL  ( select sal from emp2 where job = 'M' );
```

To Speed Up Sub-Query

- join is fater than sub-query
- try to reduce the number of level for sub-query
- try to reduce the number of rows returned by sub-query

Correlated Sub-Query

- If you have a join, along with Distinct , to make it work faster, use Correlated Sub-Query (use with EXISTS operator).

```
select dname from dept where exists ( select deptno from emp2 where dept.deptno = emp2.deptno);
```

select dname from dept where exists

- >TRUE / FALSE

(select deptno from emp2 where dept.deptno = emp2.deptno);

Steps for Above Example

- first the main query is executed
- for every row returned by main query , it will run the sub-query once
- the sub-query returns boolean TRUE or FALSE value back to main query
- if sub-query return TRUE value then main query is eventually executed for that row
- if sub-query returns FALSE value then main query is not executed for that row
- unlike you do not use DISTINCT here , hence no sorting takes place; this speeds it up
- unlike earlier, the number of full table scans is reduced: this further speeds it up

```
select dname from dept where NOT EXISTS ( select deptno from emp2 where dept.deptno = emp2.deptno);
```

SET FUNCTUONS :-

UNION

- union return the plus of both the select statement rows and duplicate are suppressed

```
select empno,ename  "ename1"  from emp5

                                                                                 union

select empno,ename "ename2" from emp6

order by 1;


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

| empno | ename1 |

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

|     1 | A      |

|     2 | B      |

|     3 | C      |

|     4 | D      |

|     5 | E      |

+-------+--------+
```

INTERSECT

- it is return the common rows of both the select statement and duplicate are suppressed

```
select empno,ename  "ename1"  from emp5

                                          intersect

select empno, ename "ename2" from emp6

order by 1;
```

MINUS

- minus will return what is present in the first SELECT and not present in the second and the duplicate are suppressed

```
select empno,ename  "ename1"  from emp6

                                          minus

select empno, ename "ename2" from emp5

order by 1;
```

NOTE :-

- union, union supported by all RDBMS
- intersect and minus are not supported by MySQL

Pseudo Column

fake column (virtual column)

e. g.

computed column (ANNUAL = sal *12)

expression (NET_EARNINGS = sal+comm)

function-based column (COMPANY_TOTAL = sum(sal))

SYSTEM SUPPLIES PSEUDO COLUMNS :-

ROWNUM

- rownum it is the Pseudo column which is provides by server RAM to store or contain the index of rows.
- ROWNUM is not available in MySQL, it is available in oracle

```
select rownum , ename from emp;
```

Row-Id

- stands for row identifier
- Rowid is the row address
- Rowid is the address of the row in the DB server HD
- This is the actual physical memory location in the DB server HD where that row is stored
- String of 18 characters
- when you select from a table the order rows in the output will always be in ascending order of Rowid
- when you Update a row, if the row length is increasing and the free space is not available then the entire row would be moved to some other address.
- when you UPDATE a row the Rowid MAY change (this is only in the case of varchar, if the row length is increasing)

```
select rowid, ename, sal from emp;



select rowid, ename, sal from emp where rowid = ' AAAFsPAABAAALH5AAA ';
```

- you can use ROWID to UPDATE or DELETE the ROWS
- Rowid is available in MySQL and Oracle also.
- you can view the Rowid on oracle but you can not view the Rowid in MySQL.

Rowid is internally by RDBMS :-

- to distingush between 2 rows in the DB
- Rowid works as an unique identifier for every row in the DB
- to manage Row locking
- to manage indexes
- to manage cursors
- Row management
- etc.

ALTER (DDL Command)

Directly

- Rename a table :-

```
## In Oracle ##

rename emp to employee;


## In MySQL ##

rename table emp to employee;
```

- Add a column

```
Alter table emp add GST float(10);
```

- Drop a column

```
Alter table emp drop column GST;
```

- Increase width of column

```
Alter table emp modify ename varchar(30);
```

Indirectly

- Reduce the width of column

```
#### In Oracle ###


you can reduce the width provided the contents are null


Alter table emp modify x varchar(20);


update emp set x = ename , ename = null;


Alter table emp modify ename varchar(20);


### data testing and ask user and make correction ###


update emp set ename = x ;

Alter table emp drop column x ;


### MySQL ###

Alter table emp modify ename varchar(20);   <------ DATA WILL GET TRUNCATE
```

Suggestion is Please contain the extra column (Extension column) for used to extend the table

- change datatype of column

```
Alter table demo modify name char(10);
```

- copy data from one table to another table

```
## for All Table

insert into emp_k select * from demo;

## For Specific row ##

insert into emp_k select * from demo where id = 3 ;
```

- copy table

```
create table emp_k as select * from emp;
```

- copy structure of table

- change the order of column in the table structure

  change order of column in table structure (for strong considerations) (because of null values).

INDEXES (v imp)

- present in all DBMS abd RDBMS, and some of the programming languages
- to speed up the serch operation (for fastest access)
- to speed up SELECT statement with a WHERE clause
- index is creates in HD server
- Indexes are automatically invoked as when required (in MySQL and Oracle).
- Indexes are automatically update indexes (in MySQL and Oracle) for all your DML Operations (insert, Update, Delete).
- Duplicate Values are stored in index
- Null values are not stored in index
- no upper limit on the number of indexes per table.
- Larger number of Indexes , the slower would be the DML operations
- you can not index text or blob column
- if you have 2 or more INDEPENDENT columns in WHERE cluase then you need to create a seperate for each column

```
select * from emp where deptno = 1  and  empno = 1 ;
```

Composite Index

- you can combine 2 or more INTER-DEPENDENT column in a single index
- In Oracle, you can combine upto 16 column in a composite index
- In MySQL, you can combine upto 32 column in a composite index

Index-Key

- column or set of column on wise basis the index has been created
- e. g. deptno

Conditions when an index should be created

- Select statement with where clause, order by clause, group by clause, DISTINCT / UNIQUE, UNION, INTERSECT, MINUS.
- if the SELECT statement retrieves < 25% of table data
- primary key and unique key column should be indexed
- common column in join operation should be always indexed

Oracle has on e product for Optimixe the indexes (Oracle Query Optimizer)

- product from oracle corporation
- works only with oracle RDBMS

Queries for Create, select, show, drop

create

```
create index emp_ind on emp( empno);


create index emp_ind on emp(enames);


create index emp_ind on emp(sal);


create index 1_ind on emp(sal desc);


create index emp_deptno_empno on emp(deptno,empno);   ---> composite  index
```

by default all indexes are in ascending orders

In MySQL :-

to see which all indexes are created for specific table

```
show indexes from demo;
```

Drop Index

```
drop index emp_id on demo;
```

if you drop the table / column then the associated indexes will be dropped automatically.

Perform one extra function i. e, it won't allow the user to INSERT duplicate values in EMPNO

at the time of creating the unique index, if you already have duplicate values in tahat particular column then MySQL will not allow you to create the unique index (this validation is performed by MySQL at the time of index creation)

Types of Indexes

1. Normal Index
2. Unique Index
3. Clustered Index (when the primary key is already created it is called as cluster index we also called as sorted table with suppressed duplication).
4. Bitmap Index ( )
5. etc.

- when you create a table using sub-query then the indexes on original table are not copied into new one
- if you want indexes for the new table then you will have to craete them manually

TIPES

1. www.Oracle.com
2. Register on oracle website
3. oracle magazine
4. MySQL magazine
5. Java magazine etc.

Constraints (Data Integrity)(V. Imp) :-

- limitations / restriction imposed on a table.

PRIMARY KEY

- column or set of columns that uniquely indentifies a row.
- duplicate value are not allowed (has to be unique).

- null values are not allowed (this is a mandatory column).
- having a primary key is not compulsory but it's recommonds that every table shoukd have a primary key.
- purpose of primary key is row uniqueness (with the help of primary key we can distinguish between 2 rows of table).
- Text and Blob cannot be Primary key .
- unique index is automatically created .
- you can have only 1 primary key per table

```
create table demp3 (

                    std_id char(4) primary key,

                    name varchar(25)

)


### Drop key ##

alter table demp3 drop primary key;


### for Alter key in existing table ###

•   alter table demp3 add primary key( std_id);

#### this command we use for show the which constraints are apply on this table  ###

 select  * from information_schema.key_column_usage where table_name = 'DEMP3';
```

COMPOSITE PRIMARY KEY

- combine 2 or more column together to the purpose of primary key
- in oracle, you can combine upto 16 column in a composite primary key
- in MySQL, you can combine upto 32 columns in a composite primary key

```
create table demp3 (

                    std_id char(4),

                    sname varchar(25),

                     primary key(std_id,sname) ;)
```

There are 2 types of constraints

1) Column level constraints (specified on 1 column)
2) Table level constraints (specified on 2 or more columns together)(composite)(has to be specified at the end of the table)

NOT NULL

- null values are not allowed
- duplication values are allowed
- you can have any number of NOT NULL constraints per table (unlike PK)
- you cannot have a composite NOT NULL constraints you will have to specify separate constraints for each column
- always a column level constraints
- In MySQL, Null ability is a part of the datatype.

```
create table emp10(

empno char(4),

ename varchar(25) not null

);


## Query for Show not null ##


desc emp10;

•

## to drop the not Null constraints afterwards ##

alter table emp10 modify ename varchar(25) null;




## if you want to add the not null constraints afterwards ##

alter table emp10 modify ename varchar(25) not null;
```

UNIQUE KEY

- duplicate values are not allowed
- null values are allowed
- Text and Blob cannot be Unique
- unique index is automatically created
- In Oracle, you can combine upto 16 columns in a composite
- In MySQL, you can combine upto 32 columns in a composite
- you can have any number of unique constraints per table

```
## Add at the time of create table ##

create table emp5( empno int(2) unique key ,ename varchar(20));
```

```
## Alter Unique key ##


Alter table emp5 add Unique key ( empno);

or

Alter table emp5 add constraint u_emp Unique( empno);

 constraint u_emp  -----> it is optional
```

Unique constraint is also an index, so to drop it

```
drop index empno on emp;
```

FOREIGN KEY

- foreign column
- column or set of column that references a column or set of column of some table
- FK constraint is specified on the child column (not the parent column)
- parent column has to be primary key or unique (this is pre-requisite for PK)
- FK column may contain duplicate values (unless specified otherwise)
- FK column may contain null value (unless specified otherwise)
- FK may reference a column of same table also (known as self-referencing)
- child table means it contains FK table and parent means it contains Primary key

```
create table emp( empno char(4) primary key, ename char(4),deptno int(3),

constraint f_emp_deptno foreign key (deptno) references dept(deptno));
```

```
## SELF -REFERENCING ##
```

```
create table emp( empno char(4) primary key,ename char(4),deptno int(3),

constraint mgr foreign key (mgr) references emp(empno));
```

column level constraint may be written at table level but a table level composite constraint cannot be written at column level (except for the NOT NULL constraint which is ALWAYS a column level constraint and terefore it must be specified at column level only)

you can’t delete the parent (master) row when child (detail) row exist

solution for that is you have to make reference as **on delete cascade**.

```
create table emp( empno char(4) primary key,ename char(4),deptno int(3),

constraint f_emp_deptno foreign key (deptno) references dept(deptno)) on delete cascade on update cascade;
```

on delete cascade --> if you delete the parent row then it will automatically delete the child rows also

WE have also one cascade **On Update cascade** it is used for update the parent row

if you update the parent row then it will automatically update the child rows also.

To disable and unable the foreign key constraint :-

for current connection :-

```
set foreign_key_checks = 0;

set foreign_key_checks = 1;
```

for all connections :-

set global foreign_key_checks = 0;

set global foreign_key_checks = 1;

CHECK

used for validations (used for checking purposes)

e. g. sal < 10000, comm <= 40% sal, etc.

example :

```
create table emp (



ename varchar(20) check ( ename = upper(ename) ),

sal float(7) check ( sal > 5000 and sal < 99000),

deptno nt(2),

status char(1) check (status in ('T','P','R'),
com,m float(7)

ppno char(15),


)

sal float(7) check default 7000;
```

To make use of default value then use the following INSERT statement :-

Default is not a constraint

Default is clause that we can with CREATE TABLE

PRIVILEGES

- Grant and invoke (DCL)

```
Atul_mysql > grant select on emp to sakshi

Atul_mysql > grant insert on emp to sakshi

Atul_mysql > grant update on emp to sakshi

Atul_mysql > grant delete on emp to sakshi

Atul_mysql > grant all on emp to sakshi
```

To see the permission received :-

```
select * from information_schema.table privileges
```

STORE OBJECTS

- objects that are stored database

- e. g. tables, indexes

---

views

- present in all RDBMS and some DBMS
- handle to a table
- stores the address of table
- view is a HD pointer
- used for indirect access to the table
- used for SECURITY purposes
- used to restrict the access of user
- used to restrict the column access also
- used to restrict the row access
- VIEW does not contain data
- only the defination is stored (data is not stored)
- view is a stored query
- Select statement on which the view is based, it is stored in system tables in the compiled format
- source code is encrypted from end-user.
- View is in executable format of SELECT statement
- execution will be faster

ex :-

```
## Create view ##
atul_mysql > create view v1 as select empno,ename from emp;

## grant view ##
atul_mysql > garant select on v1 to lalit ;

## access view ##
lalit_mysql > select * from atul.emp ;   -------- > ERROR

lalit_mysql > select * from atul.emp ;   -------- > RIGHT



Lalit_mysql > isert into atul.v1 values(6,'F');
```

- DML operations can be performed on a view
- VIEW does not contain DATA
- view is HD pointer to the table
- DML operation done on a view will affect the base table
- constraints that have been specified on the table, they will be enforced even when insert via the view

```
Update V1 SET ENAME = 'ABCD' WHERE EMPNO = 1;     ------ > WRONG

Update atul.V1 SET ENAME = 'ABCD' WHERE EMPNO = 1; ------- >RIGHT
```

DROP VIEW :-

```
drop view v1;
```

You can restrict to insert, update, delete as per the where clause in view making it view as **WITH CHECK OPTION**.

Example :-

```
atul_mysql > create view v1 as select * from emp where deptno = 1 WITH CHECK OPTION;

atul_mysql > grant select, insert on V2 to LATA;

lata_mysql > insert into akshay.v2 values(6,'F',6000, 2);   ------- > ERROR
```

DESCRIBE VIEW

```
desc view_name;
```

To see which is a table and which is a view

```
show full table;
```

To see the SELECT statement on which the view is based

```
show create view v1;
```

if you ALTER or Drop and recreate the table then the associated views will have to be recreated in MySQL.

if you drop the table then the views REMAIN