

SQL

Structured Query Language

Lets do practical on DATABASE...

SQL – Structured Query Language

- *Is a language that enables you to create and operate on relational databases*
- *It is the standard language used by almost all the database s/w vendors.*
- *Pronounced as SEQUEL*
- *Original version was developed by IBM's Almanden Research Center*
- *Latest ISO standard of SQL was released in 2008 and named as SQL:2008*

SQL – features

- *Allows creating/modifying a database's structure*
- *Changing security settings for system*
- *Permitting users for working on databases or tables*
- *Querying database*
- *Inserting/modifying/deleting the database contents*

MYSQL Elements

- *Literals*
- *Data types*
- *Nulls*
- *Comments*

Literals

- *It means the fixed value or constant value. It may be of character, numeric or date time type.*
- *Character and date/time literals are always in single quotation marks whereas numeric literals must be without single quotation marks*
- *For example – ‘Virat’, 12, 12.56, ‘04-20-2018’*
- *Date and time values are always in the format
YYYY-MM-DD HH:MI:SS*
- *Special character like quotes are always written be preceding it back-slash(\). For example if we want to store value as Tom’s Cat then it should be written as Tom\'s Cat*

Data Type

- *Means the type of value and type of operation we can perform on data. For example on numeric value we can store numbers and perform all arithmetic operations and so on.*
- *MySQL support three categories of data types:*
 - *Numeric*
 - *Date and time*
 - *String types*

Numeric Data Types

Data type	Description
INT	Numbers without decimal. Store up to 11 digits. -2147483648 to 2147483647
TINYINT	Small integer value between 0 – 255 (4 digits)
SMALLINT	More than TINYINT between -32768 to 32767 (5 digit)
MEDIUMINT	Integer values up to 9 digits
BIGINT	Very large integer value up to 11 digits
FLOAT(M,D)	Real numbers i.e. number with decimal. M specify length of numeric value including decimal place D and decimal symbol. For example if it is given as FLOAT(8,2) then 5 integer value 1 decimal symbol and 2 digit after decimal TOTAL – 8. it can work on 24 digits after decimal.
DOUBLE(M,D)	Real numbers with more precision up to 53 place after decimal.
DECIMAL	It is used to store exact numeric value that preserve exact precision for e.g. money data in accounting system. DECIMAL(P,D) means P no. of significant digits (1-65), D represent no. of digit after decimal(0-30), for e.g DECIMAL(6,2) means 4 digit before decimal and 2 digit after decimal. Max will be 9999.99

Date and Time Types

Data type	Description
DATE	A date in YYYY-MM-DD format between 1000-01-01 to 9999-12-31 In oracle data format is DD-MON-YYYY for e.g 10-SEP-2019
DATETIME	Combination of date and time. For example to store 4 th December 2018 and time is afternoon 3:30 then it should be written as – 2018-12-04 15:30:00
TIMESTAMP	Similar to DATETIME but it is written without hyphen for example the above date time is stored as 20181204153000
TIME	To store time in the format HH:MM:SS
YEAR(M)	To store only year part of data where M may be 2 or 4 i.e. year in 2 digit like 18 or 4 digit like 2018

String Types

Data type	Description
CHAR(M)	<p>Fixed length string between 1 and 255. it always occupy M size for each data for example if size is CHAR(20) and we store value 'MOBILE' , although the size of MOBILE is 6 but in a table it will occupy 20 size with space padded at right side for remaining place.</p> <p>Mostly use in the case where the data to be insert is of fixed size like Grade (A,B,C,..) or Employee code as E001, E002, etc. In this case CHAR will give better performance than varchar</p>
VARCHAR(M)	<p>Variable length string between 1 and 65535 (from MySQL 5.0.3) , earlier it was 255. it takes size as per the data entered for example with VARCHAR(20) if the data entered is MOBILE then it will take only 6 byte. It is useful for the data like name, address where the number of character to be enter is not fixed.</p>
VARCHAR2	<p>It is supported in ORACLE, both are almost same with minor difference. The difference is in the way they are handling Empty String and NULL, for VARCHAR these two are different where as VARCHAR2 treats both same.</p>

Difference between CHAR & VARCHAR

CHAR	VARCHAR
Fixed length string	Variable length string
Used where number of character to enter is fixed like Grade, EmpCode, etc	Used where number of character to be enter is not fixed like name, address etc.
Fast, no memory allocation every time	Slow, as it take size according to data so every time memory allocation is done
It takes more memory	It takes less space

NULL VALUE

- NULL means missing information
- NULL can appear in any type of column if it is not restricted by NOT NULL or PRIMARY KEY
- Always remember NULL is neither equal to 0 nor space. NULL means nothing
- Used in situation like if email id is not available with students then we will insert NULL

COMMENTS

- It is a text that is not executed, only for documentation purpose. Comments in MySQL can be written as
 - Begin the comment with `/*` and `*/`
 - Begin the comment with `–` (followed by space)
 - Begin then comment with `#`
- For example
 - `/* Select * from emp where empno=4 */`
 - `Select * from emp; --` it will fetch all details

SQL COMMAND SYNTAX

Commands	Description
Keywords	That have special meaning in SQL. They are the commands in mysql
Clause	They are used to support mysql commands like FROM, WHERE etc.
Arguments	Values passed to clause like table name to FROM clause conditions to WHERE clause for e.g. <code>SELECT * FROM EMP WHERE SALARY>12000;</code> In the above command SELECT is keyword FROM AND WHERE is clause EMP is an argument to FROM SALARY>12000 is argument to WHERE

CREATING and USING DATABASE

CREATE DATABASE <DATABASE NAME>

CREATE DATABASE MYDB;

TO SEE LIST OF DATABASES:

SHOW DATABASES;

TO OPEN ANY DATABASE TO WORK

USE DATABASENAME

USE MYDB

CREATING TABLE

Syntax:-

*Create Table TableName(ColumnName datatype(size),
ColumnName datatype(size),.....);*

Example:-

*Create Table Employee(empno int, name varchar(20), dept
varchar(20), salary int);*

*Create table Student(roll int, name varchar(20), stream
varchar(20), per int);*

INSERTING RECORDS IN TABLE

Syntax:-

Insert into tablename values(value1,value2,...)

Note:-

- 1) char, varchar and date value must be in single quotes*
- 2) Values must be passed in the order of their column*
- 3) Date values are passed in the format
dd-mon-yyyy i.e. 20-Sep-2015 (in oracle)
yyyy-mm-dd (in mysql)*

INSERTING RECORDS IN TABLE

Syntax:-

Insert into emp values(1, 'Rakesh','Sales',34000)

Insert into student values(1,'Mahi','Science',89);

Inserting in selected columns

**Insert into emp (empno, name, dept) values
(2,'dipanker','IT')**

SELECTING RECORD

Select statement allows to send queries to table and fetch the desired record. Select can be used to select both horizontal and vertical subset.

Syntax:-

```
Select * / columnnames FROM tablename [ where  
condition ]
```

SELECTING RECORD

Selecting all record and all columns

```
Select * from emp;
```

Selecting desired columns

```
select empno, name from emp;
```

Changing the order of columns

```
select dept, name from emp;
```

DISTINCT keyword

DISTINCT keyword is used to eliminate the duplicate records from output. For e.g. if we select dept from employee table it will display all the department from the table including duplicate rows.

Select dept from emp;

Output will be:-

Dept

Sales

Sales

IT

IT

HR

Empno	Name	Dept	Salary
1	Ravi	Sales	24000
2	Sunny	Sales	35000
3	Shobit	IT	30000
4	Vikram	IT	27000
5	nitin	HR	45000

DISTINCT keyword

If we don't want to see the duplicate rows in output we have to use DISTINCT keyword.

Select DISTINCT dept from emp;

Output will be:-

Dept

Sales

IT

HR

Empno	Name	Dept	Salary
1	Ravi	Sales	24000
2	Sunny	Sales	35000
3	Shobit	IT	30000
4	Vikram	IT	27000
5	nitin	HR	45000

PERFORMING SIMPLE CALCULATION

While performing SQL operations sometimes simple calculations are required, SQL provides facility to perform simple arithmetic operations in query. In MySQL we can give these queries without FROM clause i.e. table name is not required for these queries,

For Example

```
Select 10*2;
```

```
Select 10*3/6;
```

PERFORMING SIMPLE CALCULATION

MySQL also provides DUAL table to provide compatibility with other DBMS. It is dummy table used for these type queries where table name is not required. It contains one row and one column. For example:

```
Select 100+200 from DUAL;
```

```
Select curdate() from dual;
```

CALCULATION WITH TABLE DATA

```
Select name, salary, salary * 12 Annual_Salary from emp;
```

```
Select empno, salary+1000 from emp
```

```
Update student set total=phy+chem+maths+cs+eng;
```

COLUMN ALIAS

It is a temporary name/label given to column that will appear in output. For example if column name is dept and you want Department to appear as column heading then we have to give Column Alias. If we want alias name of multiple words then it should be enclosed in double quotes. Its format is :

ColumnName [AS] ColumnAlias

Example

(i) Select empno Employee_Number, name, dept Department, Salary Income from emp;

(ii) Select name, Salary*12 as "Annual Income" from emp;

HANDLING NULL

```
mysql> select * from emp;
+-----+-----+-----+-----+
| empno | name  | dept | salary |
+-----+-----+-----+-----+
| 1     | Amit  | IT   | 8000   |
| 2     | Sumit | SALES | 9000   |
| 3     | Ajit  | HR   | 8500   |
| 4     | Vikram | SALES | 10000  |
| 5     | Shaban | HR   | NULL   |
+-----+-----+-----+-----+
5 rows in set (0.15 sec)
```

```
mysql> select empno,name,ifnull(salary,"not assigned") from emp;
+-----+-----+-----+-----+
| empno | name  | ifnull(salary,"not assigned") |
+-----+-----+-----+-----+
| 1     | Amit  | 8000                           |
| 2     | Sumit | 9000                           |
| 3     | Ajit  | 8500                           |
| 4     | Vikram | 10000                          |
| 5     | Shaban | not assigned                    |
+-----+-----+-----+-----+
5 rows in set (0.05 sec)
```

From the above table we can observe that salary of Shaban is NULL i.e. not assigned, Now if we want 0 or “not assigned” for the salary information of shaban, we have to use **IFNULL()**

Select empno,name,IFNULL(Salary,"not assigned") from emp;

Column

value to substitute if NULL found

PUTTING TEXT IN QUERY OUTPUT

*SQL allows to put user defined symbols or text with table output.
Like 'Rs' with Salary or '%' symbol with commission*

For e.g.

Select name, dept, 'Rs.', salary from emp;

**Select name, ' works in department', dept, ' and getting salary rs.
, salary from emp;**

Select name, concat('Rs. ', salary) from emp;

WHERE clause

WHERE clause is used to select specified rows. It allows to select only desired rows by applying condition. We can use all comparison(>, <, >=, <=, =, <>) and logical operator (AND, OR, NOT).

AND (&&), OR (||), NOT (!)

For example

Select * from emp **where salary>4000;**

Select * from emp **where empno=1;**

Select name,dept from emp **where dept='HR';**

WHERE clause

AND(&&) means both conditions must be true, OR(||) means any condition must be true to produce output. NOT(!) will do the reverse checking.

```
Select * from emp where salary>4000 and salary<8000;
```

```
Select * from emp where dept='Sales' and salary<30000;
```

```
Select name,dept from emp where dept='HR' and salary>=20000 and salary<=40000;
```

```
Select * from emp where dept='HR' or dept='IT';
```

```
Select * from emp where NOT empno=4;
```

SQL operators

- 1) BETWEEN
- 2) IN
- 3) LIKE
- 4) IS NULL

BETWEEN

BETWEEN allows to specify range of values to search in any column. It is used with AND clause and it will include the specified values during the searching. For e.g.

*Select * from emp where salary **between 18000 and 30000**;*

*Select name from emp where empno **between 2 and 5**;*

*Select * from emp where salary **NOT between 25000 and 35000***

IN

IN allows to specify LIST of values in which searching will be performed. It will return all those record that matches any value in a given list of values. It can be thought as an alternative of multiple ORs

*Select * from emp where dept **IN**('sales','it');*

*Select name from emp where empno **IN** (2,4,5);*

*Select * from emp where dept **NOT IN**('sales','it')*

LIKE

LIKE allows to search based on pattern. It is used when we don't want to search an exact value or we don't know that exact value, and we know only the pattern of value like name starting from any particular letter, or ending with and containing any particular letter or word.

LIKE is used with two wildcard characters:

- a) **%** : used when we want to substitute multiple characters. With % length is not fixed
- b) **_** (underscore) : used when we want to substitute Single character

LIKE

Search for employee whose name begins from 's'

Select * from emp where name like 's%';

Search for employee whose name ends with 'r'

Select * from emp where name like '%r';

Search for employee whose name contains 'a' anywhere

Select * from emp where name like '%a%';

Search for employee whose dob is in feb

Select * from emp where dob like '%-02-%';

Search employee whose name is of 5 letters begins from 's'

Select * from emp where name like 's_____';

IS NULL


IS NULL is used to compare NULL values present in any column. Because NULL is not considered as value so we cannot compare with = sign, so to compare with NULL SQL provides IS NULL.

```
Select * from emp where salary is null;
```

```
Select * from emp where salary is not null;
```

OPERATOR PRECEDENCE

When multiple operators are used in expression, then evaluation of expression takes place in the order of precedence. Higher precedence operator will execute first.

!	
*, /, DIV, %, MOD	
- +	
<, >	
==, >=, <=, !=, IS, LIKE, IN, BETWEEN	
NOT	
AND	
OR	

SORTING OUTPUT

By default records will come in the output in the same order in which it was entered. To see the output rows in sorted or arranged in ascending or descending order SQL provide **ORDER BY** clause. By default output will be ascending order(ASC) to see output in descending order we use DESC clause with ORDER BY.

Select * from emp **order by name;** (ascending order)

Select * from emp **order by salary desc;**

Select * from emp **order by dept asc, salary desc;**

MYSQL FUNCTIONS

A function is built – in code for specific purpose that takes value and returns a single value. Values passed to functions are known as arguments/parameters.

There are various categories of function in MySQL:-

- 1) String Function
- 2) Mathematical function
- 3) Date and time function

String Function

Function	Description	Example
CHAR()	Return character for given ASCII Code	Select Char(65); Output- A
CONCAT()	Return concatenated string	Select concat(name, ' works in ', dept, department);
LOWER()/LCASE()	Return string in small letters	Select lower('INDIA'); Output- india Select lower(name) from emp;
SUBSTRING(S,P,N) / MID(S,P,N)	Return N character of string S, beginning from P	Select SUBSTRING('LAPTOP',3,3); Output – PTO Select SUBSTR('COMPUTER',4,3); Output – PUT
UPPER()/UCASE()	Return string in capital letters	Select Upper('india'); Output- INDIA
LTRIM()	Removes leading space	Select LTRIM(' Apple'); Output- 'Apple'
RTRIM	Remove trailing space	Select RTRIM('Apple '); Output- 'Apple'

String Function

Function	Description	Example
TRIM()	Remove spaces from beginning and ending	Select TRIM(' Apple '); Output-'Apple' Select * from emp where trim(name) = 'Suyash';
INSTR()	It search one string in another string and returns position, if not found 0	Select INSTR('COMPUTER','PUT'); Output-4 Select INSTR('PYTHON','C++'); Output – 0
LENGTH()	Returns number of character in string	Select length('python'); Output- 7 Select name, length(name) from emp
LEFT(S,N)	Return N characters of S from beginning	Select LEFT('KV OEF',2); Output- KV
RIGHT(S,N)	Return N characters of S from ending	Select RIGHT('KV OEF',3); Output- OEF

Numeric Function

Function	Description	Example
MOD(M,N)	Return remainder M/N	Select MOD(11,5); Output- 1
POWER(B,P)	Return B to power P	Select POWER(2,5); Output-32
ROUND(N,D)	Return number rounded to D place after decimal	Select ROUND(11.589,2); Output- 11.59 Select ROUND(12.999,2); Output- 13.00
SIGN(N)	Return -1 for -ve number 1 for +ve number	Select sign(-10); Output : -1 Select sign(10); Output : 1
SQRT(N)	Returns square root of N	Select SQRT(144); Output: 12
TRUNCATE(M,N)	Return number upto N place after decimal without rounding it	Select Truncate(15.789,2); Output: 15.79

Date and Time Function

Function	Description	Example
CURDATE()/ CURRENT_DATE() / CURRENT_DATE	Return the current date	Select curdate(); Select current_date();
DATE()	Return date part from date-time expression	Select date('2018-08-15 12:30'); Output: 2018-08-15
MONTH()	Return month from date	Select month('2018-08-15'); Output: 08
YEAR()	Return year from date	Select year('2018-08-15'); Output: 2018
DAYNAME()	Return weekday name	Select dayname('2018-12-04'); Output: Tuesday
DAYOFMONTH()	Return value from 1-31	Select dayofmonth('2018-08-15') Output: 15
DAYOFWEEK()	Return weekday index, for Sunday-1, Monday-2, ..	Select dayofweek('2018-12-04'); Output: 3
DAYOFYEAR()	Return value from 1-366	Select dayofyear('2018-02-10') Output: 41

Date and Time Function

Function	Description	Example
NOW()	Return both current date and time at which the function executes	Select now();
SYSDATE()	Return both current date and time	Select sysdate();

Difference Between NOW() and SYSDATE() :

NOW() function return the date and time at which function was executed even if we execute multiple NOW() function with select. whereas SYSDATE() will always return date and time at which each SYDATE() function started execution. For example.

```
mysql> Select now(), sleep(2), now();
```

```
Output: 2018-12-04 10:26:20, 0, 2018-12-04 10:26:20
```

```
mysql> Select sysdate(), sleep(2), sysdate();
```

```
Output: 2018-12-04 10:27:08, 0, 2018-12-04 10:27:10
```

AGGREGATE functions

Aggregate function is used to perform calculation on group of rows and return the calculated summary like sum of salary, average of salary etc.

Available aggregate functions are –

1. *SUM()*
2. *AVG()*
3. *COUNT()*
4. *MAX()*
5. *MIN()*
6. *COUNT(*)*

AGGREGATE functions

Empno	Name	Dept	Salary
1	Ravi	Sales	24000
2	Sunny	Sales	35000
3	Shobit	IT	30000
4	Vikram	IT	27000
5	nitin	HR	45000

Select SUM(salary) from emp;

Output – 161000

Select SUM(salary) from emp where dept='sales';

Output - 59000

AGGREGATE functions

Empno	Name	Dept	Salary
1	Ravi	Sales	24000
2	Sunny	Sales	35000
3	Shobit	IT	30000
4	Vikram	IT	27000
5	nitin	HR	45000

Select AVG(salary) from emp;

Output – 32200

Select AVG(salary) from emp where dept='sales';

Output - 29500

AGGREGATE functions

Empno	Name	Dept	Salary
1	Ravi	Sales	24000
2	Sunny	Sales	35000
3	Shobit	IT	30000
4	Vikram	IT	27000
5	nitin	HR	45000

Select COUNT(name) from emp;

Output – 5

Select COUNT(salary) from emp where dept='HR';

Output - 1

Select COUNT(DISTINCT dept) from emp;

Output - 3

AGGREGATE functions

Empno	Name	Dept	Salary
1	Ravi	Sales	24000
2	Sunny	Sales	35000
3	Shobit	IT	30000
4	Vikram	IT	27000
5	nitin	HR	45000

Select MAX(Salary) from emp;

Output – 45000

Select MAX(salary) from emp where dept='Sales';

Output - 35000

AGGREGATE functions

Empno	Name	Dept	Salary
1	Ravi	Sales	24000
2	Sunny	Sales	35000
3	Shobit	IT	30000
4	Vikram	IT	27000
5	nitin	HR	45000

Select MIN(Salary) from emp;

Output – 24000

Select MIN(salary) from emp where dept='IT';

Output - 27000

AGGREGATE functions

Empno	Name	Dept	Salary
1	Ravi	Sales	24000
2	Sunny	Sales	35000
3	Shobit	IT	30000
4	Vikram	IT	27000
5	nitin	HR	45000
6	Krish	HR	

Select COUNT() from emp;*

Output – 6

Select COUNT(salary) from emp;

Output - 5

count(*) Vs count()

Count() function is used to count the number of rows in query output whereas count() is used to count values present in any column excluding NULL values.*

Note:

All aggregate function ignores the NULL values.

GROUP BY

GROUP BY clause is used to divide the table into logical groups and we can perform aggregate functions in those groups. In this case aggregate function will return output for each group. For example if we want sum of salary of each department we have to divide table records

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Aggregate functions by default takes the entire table as a single group that's why we are getting the sum(), avg(), etc output for the entire table. Now suppose organization wants the sum() of all the job separately, or wants to find the average salary of every job. In this case we have to logically divide our table into groups based on job, so that every group will be passed to aggregate function for calculation and aggregate function will return the result for every group.

Group by clause helps up to divide the table into logical groups based on any column value. In those logically divided records we can apply aggregate functions.

For. E.g.

```
SELECT SUM(SAL) FROM EMP GROUP BY DEPT;  
SELECT JOB,SUM(SAL) FROM EMP GROUP BY DEPT;  
SELECT JOB,SUM(SAL),AVG(SAL),MAX(SAL),COUNT(*)  
EMPLOYEE_COUNT FROM EMP;
```

NOTE :- when we are using GROUP BY we can use only aggregate function and the column on which we are grouping in the SELECT list because they will form a group other than any column will gives you an error because they will be not the part of the group.

For e.g.

```
SELECT ENAME,JOB,SUM(SAL) FROM EMP GROUP BY JOB;
```

Error -> because Ename is not a group expression

HAVING with GROUP BY

- If we want to filter or restrict some rows from the output produced by GROUP BY then we use HAVING clause. It is used to put condition of group of rows. With having clause we can use aggregate functions also.
- WHERE is used before the GROUP BY. With WHERE we cannot use aggregate function.
- E.g.
- **SELECT DEPT,AVG(SAL) FROM EMP GROUP BY DEPT HAVING JOB IN ('HR','SALES')**
- **SELECT DEPT,MAX(SAL),MIN(SAL),COUNT(*) FROM EMP GROUP BY DEPT HAVING COUNT(*)>2**
- **SELECT DEPT,MAX(SAL),MIN(SAL) FROM EMP WHERE SAL>=2000 GROUP BY DEPT HAVING DEPT IN('IT','HR')**

JUST A MINUTE...

- Create the following table and add the records

ItemNo	Item	Dcode	Qty	UnitPrice	StockDate
5005	Ball Pen 0.5	102	100	16	2018-03-10
5003	Ball Pen 0.25	102	150	20	2017-05-17
5002	Gel Pen Premium	101	125	14	2018-04-20
5006	Gel Pen Classic	101	200	22	2018-10-08
5001	Eraser Small	102	210	5	2018-03-11
5004	Eraser Big	102	60	10	2017-11-18
5009	Sharpener Classic	NULL	160	8	2017-06-12

JUST A MINUTE...

Write down the following queries based on the given table:

- 1) Select all record of table
- 2) Select ItemNo, name and Unitprice
- 3) Select all item record where Unitprice is more than 20
- 4) Select Item name of those items which are quantity between 100-200
- 5) Select all record of Items which contains pen word in it
- 6) Select unique dcode of all items
- 7) Display all record in the descending order of UnitPrice
- 8) Display all items which are stocked in the month of March

JUST A MINUTE...

Write down the following queries based on the given table:

- 11) Change the unitprice to 20 for itemno 5005
- 12) Delete the record of itemno 5001
- 13) Display all the item name in capital letters
- 14) Display first 4 character of every item name
- 15) Display all record whose dcode is not assigned

EXPORTING DATABASE

- Exporting means creating the text file containing all the database contents which can be imported to any other computer. We use **mysqldump** command for this purpose. To export:
- Open **cmd** from **Start** button
- Or
- Press **Window Key + R** and type **cmd** and press Enter
- At cmd prompt like **C:\....>** type **mysqldump** command

EXPORTING DATABASE

- To export all database
 - **mysqldump --all-databases > c:\mydb.sql**
 - It will create mydb.sql in C:\, we can check it by open it.
- To export single database
 - **mysqldump myworks > c:\mydb2.sql**
- To export multiple database
 - **mysqldump myworks company c:\mydb3.sql**

IMPORTING DATABASE

- **STEP – 1**

- **Open Notepad and type the following commands**

```
drop database if exists myworks;  
create database myworks;  
use myworks;  
create table emp(empno int,name varchar(20),dept varchar(20), salary int);  
create table student(roll int, name varchar(20), per int);  
insert into emp values  
(1,'Amit','IT',8000),  
(2,'Sumit','SALES',9000),  
(3,'Ajit','HR',8500),  
(4,'Vikram','SALES',10000),  
(5,'Shaban','HR',12000);  
insert into student values  
(1,'Amit',99),  
(2,'Sumit',97),  
(3,'Vikas',95),  
(4,'Jitendra',49);
```

IMPORTING DATABASE

- **STEP – 1**

- Save this file with **mydb.sql** or any name, in your desired location for e.g. C:\

- **STEP – 2**

- Open MySQL and type

- **SOURCE C:\mydb.sql**

- It will create database **myworks**, 2 tables **emp**, **students** with records. We can check it by “SHOW TABLES” or “SELECT” command