JLC Java Learning Center

MySQL Study Guide

Author
Srinivas Dande

JLC My JAVA LEARNING CENTER

1. Introduction to Data Stores

- Data Store is a place where you can store your enterprise Data.
- There are different types of Data Stores.

1. **File Systems**
    - Text Files
    - XML Files
    - JSON Files

    etc

2. **SQL Databases**
    - MySQL
    - Oracle
    - PostgreSQL
    - MS SQL Server
    - DB2
    - Sybase
    - MS SQL Server

    etc

3. **NoSQL databases**
    - MongoDB
    - Cassandra
    - Hbase
    - BigTable,
    - DynamoDB

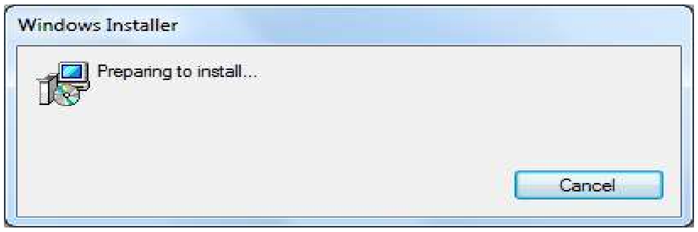
    etc

4. **Massive Storages**
    - HDFS
    - BigQuery
    - SnowFlake

    etc

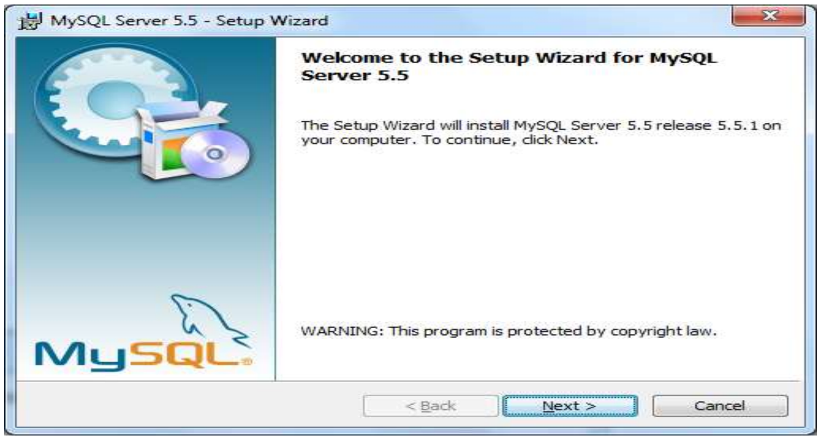
2A. MySQL Server 5.5 Installation

1) Double click on the MySQL installer "MySQL-5.5.41-win64.exe".



Preparing to install...

2) Click on next button.

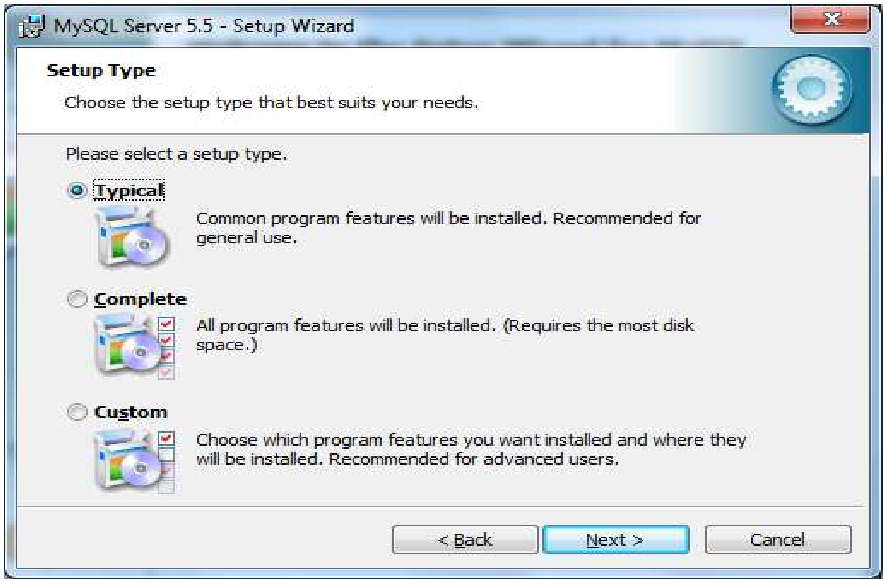


Welcome to the Setup Wizard for MySQL Server 5.5

The Setup Wizard will install MySQL Server 5.5 release 5.5.1 on your computer. To continue, click Next.

WARNING: This program is protected by copyright law.

3) Use the default selection and click on next button.



Setup Type

Choose the setup type that best suits your needs.

Please select a setup type.

[✓ Typical]

Common program features will be installed. Recommended for general use.

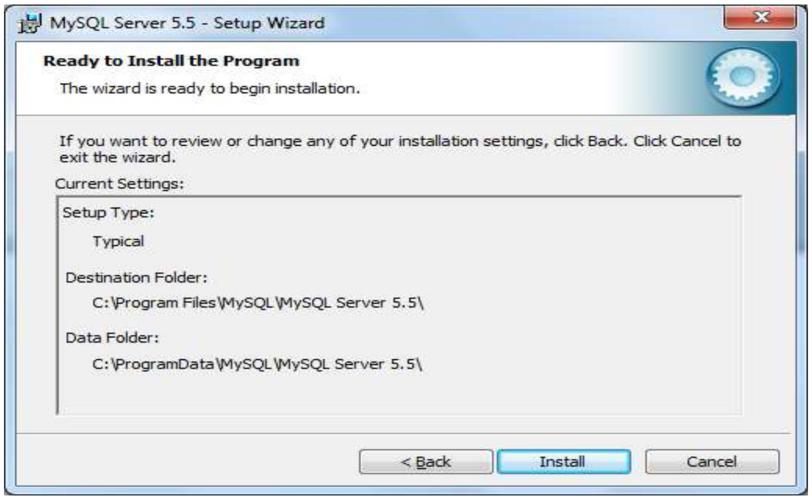
Complete

All program features will be installed. (Requires the most disk space.)

Custom

Choose which program features you want installed and where they will be installed. Recommended for advanced users.

4) **Click on install button and wait...**



MySQL Server 5.5 - Setup Wizard

Ready to Install the Program

The wizard is ready to begin installation.

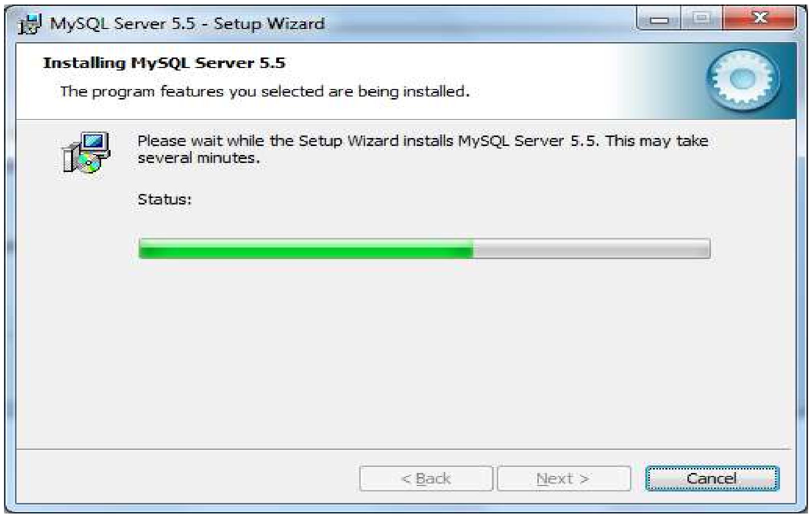
If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Current Settings:

Setup Type:
Typical

Destination Folder:
C:\Program Files\MySQL\MySQL Server 5.5\

Data Folder:
C:\ProgramData\MySQL\MySQL Server 5.5\



MySQL Server 5.5 - Setup Wizard

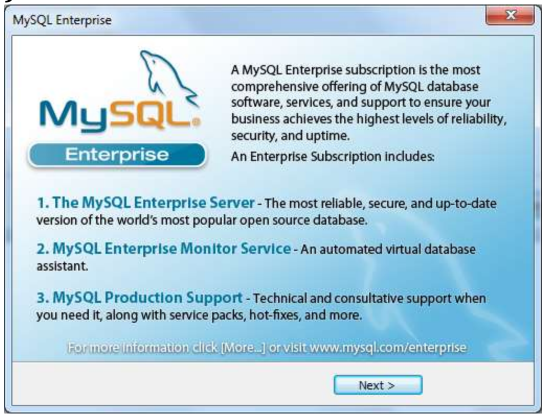
Installing MySQL Server 5.5

The program features you selected are being installed.

Please wait while the Setup Wizard installs MySQL Server 5.5. This may take several minutes.

Status:

5) **Click on next button.**



MySQL Enterprise

A MySQL Enterprise subscription is the most comprehensive offering of MySQL database software, services, and support to ensure your business achieves the highest levels of reliability, security, and uptime.

An Enterprise Subscription includes:

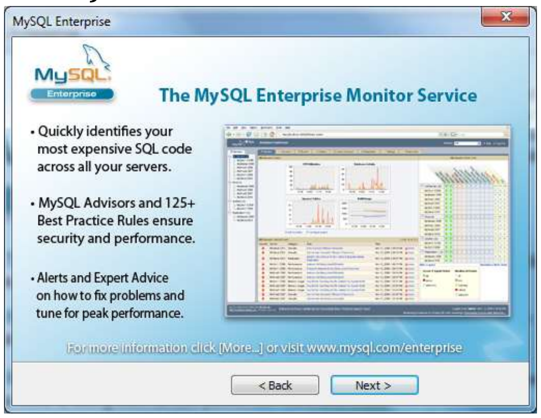
1. The MySQL Enterprise Server - The most reliable, secure, and up-to-date version of the world's most popular open source database.

2. MySQL Enterprise Monitor Service - An automated virtual database assistant.

3. MySQL Production Support - Technical and consultative support when you need it, along with service packs, hot-fixes, and more.

For more information click [More...] or visit www.mysql.com/enterprise

6) **Click on next button.**



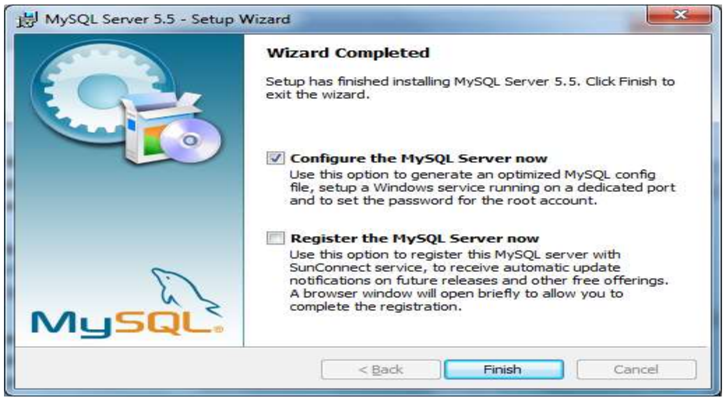
MySQL Enterprise

The MySQL Enterprise Monitor Service

- Quickly identifies your most expensive SQL code across all your servers.
- MySQL Advisors and 125+ Best Practice Rules ensure security and performance.
- Alerts and Expert Advice on how to fix problems and tune for peak performance.

For more information click [More...] or visit www.mysql.com/enterprise

7) Uncheck then radio button "Register the MySQL…" and Click on Finish button.


MySQL Server 5.5 - Setup Wizard

Wizard Completed

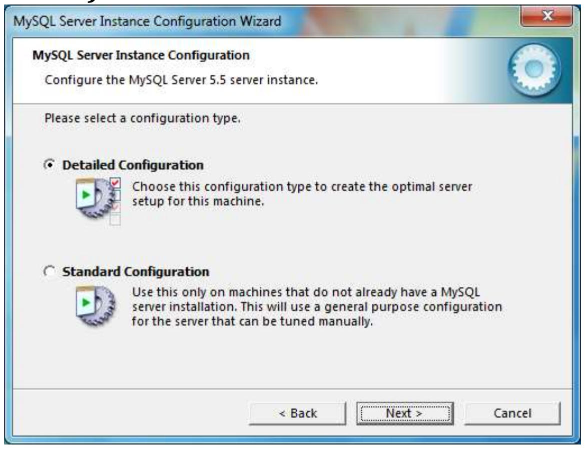
Setup has finished installing MySQL Server 5.5. Click Finish to exit the wizard.

[✓] Configure the MySQL Server now
Use this option to generate an optimized MySQL config file, setup a Windows service running on a dedicated port and to set the password for the root account.

[ ] Register the MySQL Server now
Use this option to register this MySQL server with SunConnect service, to receive automatic update notifications on future releases and other free offerings. A browser window will open briefly to allow you to complete the registration.

2B. MySQL Server 5.5 Instance Configuration Steps

1) Click on next button.


MySQL Server Instance Configuration Wizard

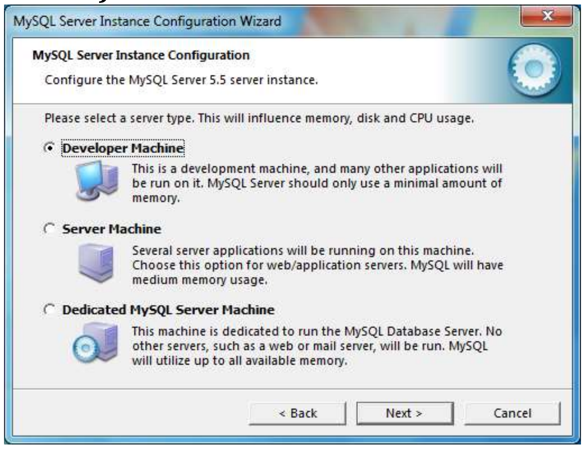
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

Please select a configuration type.

(●) Detailed Configuration
Choose this configuration type to create the optimal server setup for this machine.

( ) Standard Configuration
Use this only on machines that do not already have a MySQL server installation. This will use a general purpose configuration for the server that can be tuned manually.

2) Click on next button.


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

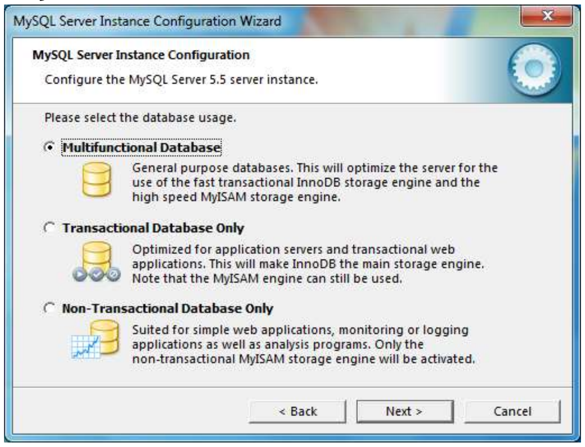
Please select a server type. This will influence memory, disk and CPU usage.

(●) Developer Machine
This is a development machine, and many other applications will be run on it. MySQL Server should only use a minimal amount of memory.

( ) Server Machine
Several server applications will be running on this machine. Choose this option for web/application servers. MySQL will have medium memory usage.

( ) Dedicated MySQL Server Machine
This machine is dedicated to run the MySQL Database Server. No other servers, such as a web or mail server, will be run. MySQL will utilize up to all available memory.

3) Click on next button.


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

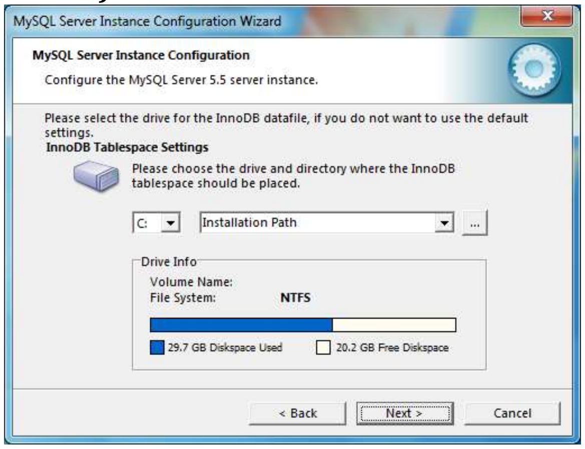
Please select the database usage.

(●) Multifunctional Database
General purpose databases. This will optimize the server for the use of the fast transactional InnoDB storage engine and the high speed MyISAM storage engine.

( ) Transactional Database Only
Optimized for application servers and transactional web applications. This will make InnoDB the main storage engine. Note that the MyISAM engine can still be used.

( ) Non-Transactional Database Only
Suited for simple web applications, monitoring or logging applications as well as analysis programs. Only the non-transactional MyISAM storage engine will be activated.

4) Click on next button.


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

Please select the drive for the InnoDB datafile, if you do not want to use the default settings.

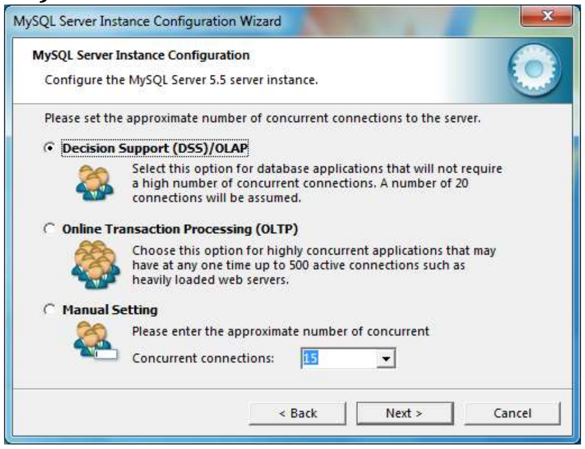
InnoDB Tablespace Settings
Please choose the drive and directory where the InnoDB tablespace should be placed.

C: | Installation Path

Drive Info
Volume Name:
File System: NTFS

🟦 29.7 GB Diskspace Used
☐ 20.2 GB Free Diskspace

5) Click on next button.



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

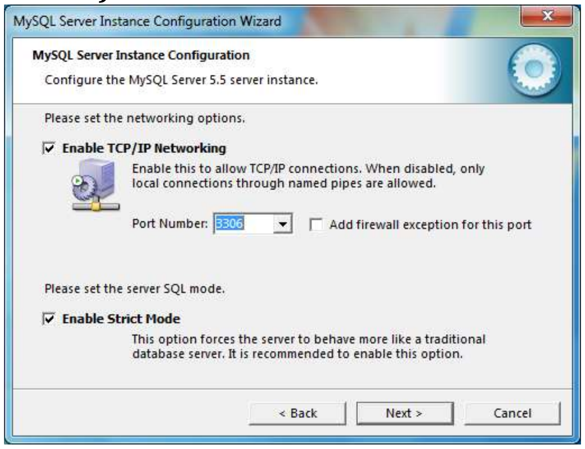
Please set the approximate number of concurrent connections to the server.

[✓ Decision Support (DSS)/OLAP]
Select this option for database applications that will not require a high number of concurrent connections. A number of 20 connections will be assumed.

Online Transaction Processing (OLTP)
Choose this option for highly concurrent applications that may have at any one time up to 500 active connections such as heavily loaded web servers.

Manual Setting
Please enter the approximate number of concurrent
Concurrent connections: 15

6) Click on next button.



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

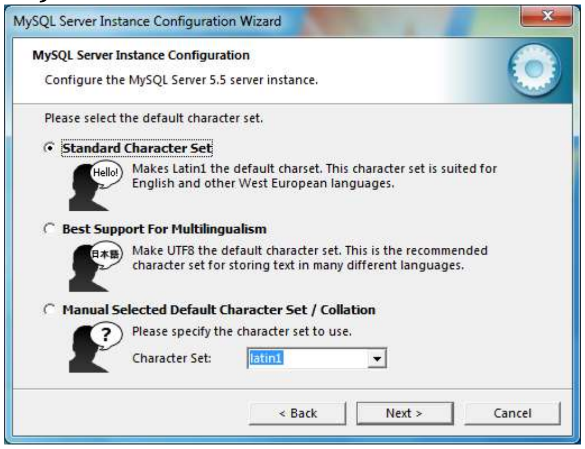
Please set the networking options.

[✓] Enable TCP/IP Networking
Enable this to allow TCP/IP connections. When disabled, only local connections through named pipes are allowed.
Port Number: 3306 [ ] Add firewall exception for this port

Please set the server SQL mode.

[✓] Enable Strict Mode
This option forces the server to behave more like a traditional database server. It is recommended to enable this option.

7) Click on next button.



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

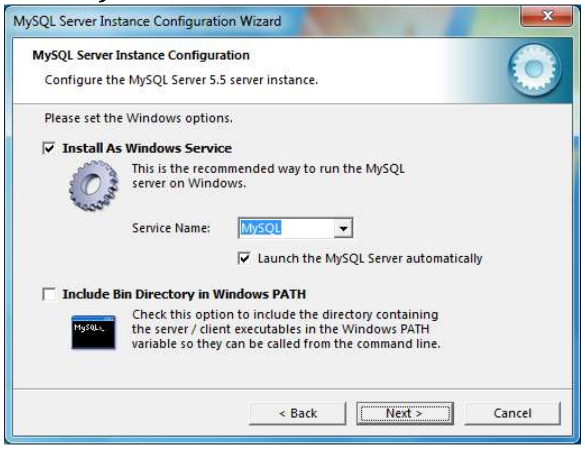
Please select the default character set.

[✓ Standard Character Set]
Makes Latin1 the default charset. This character set is suited for English and other West European languages.

Best Support For Multilingualism
Make UTF8 the default character set. This is the recommended character set for storing text in many different languages.

Manual Selected Default Character Set / Collation
Please specify the character set to use.
Character Set: latin1

8) Click on next button.



MySQL Server Instance Configuration Wizard

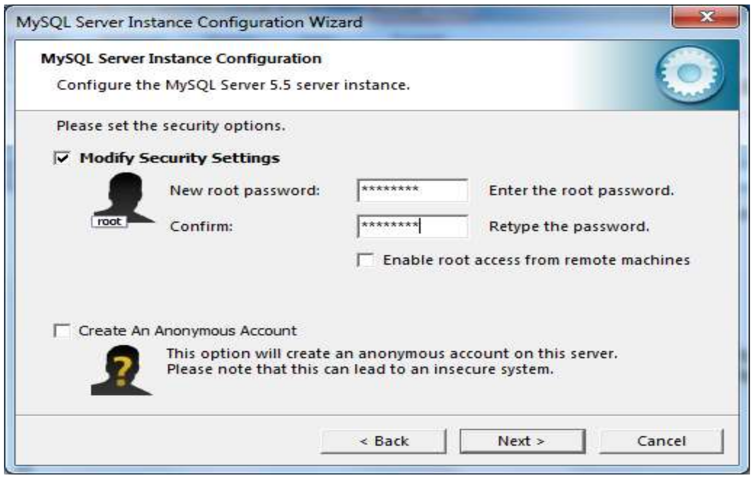
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

Please set the Windows options.

[✓] Install As Windows Service
This is the recommended way to run the MySQL server on Windows.
Service Name: MySQL
[✓] Launch the MySQL Server automatically

[ ] Include Bin Directory in Windows PATH
Check this option to include the directory containing the server / client executables in the Windows PATH variable so they can be called from the command line.

9) Provide the password for ROOT user.



MySQL Server Instance Configuration Wizard

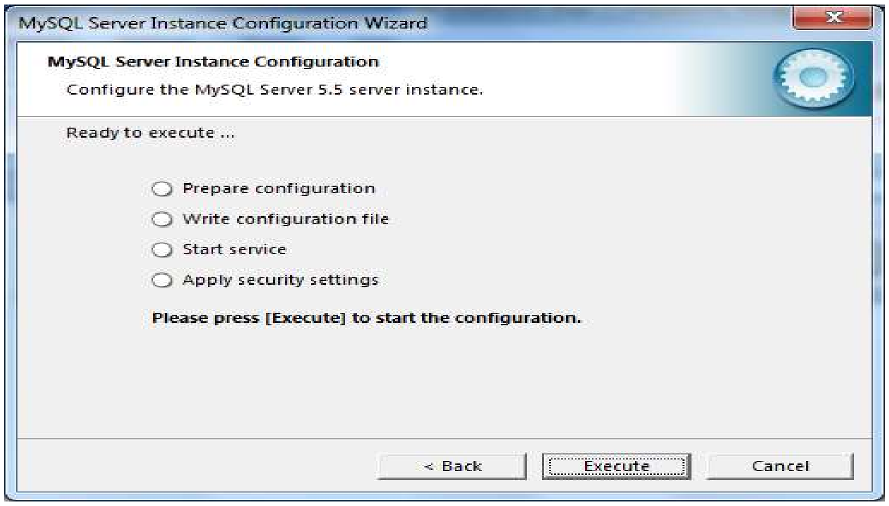
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

Please set the security options.

[✓] Modify Security Settings
New root password: ******** Enter the root password.
Confirm: ******** Retype the password.
[ ] Enable root access from remote machines

[ ] Create An Anonymous Account
This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

10) Click on Execute button.



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

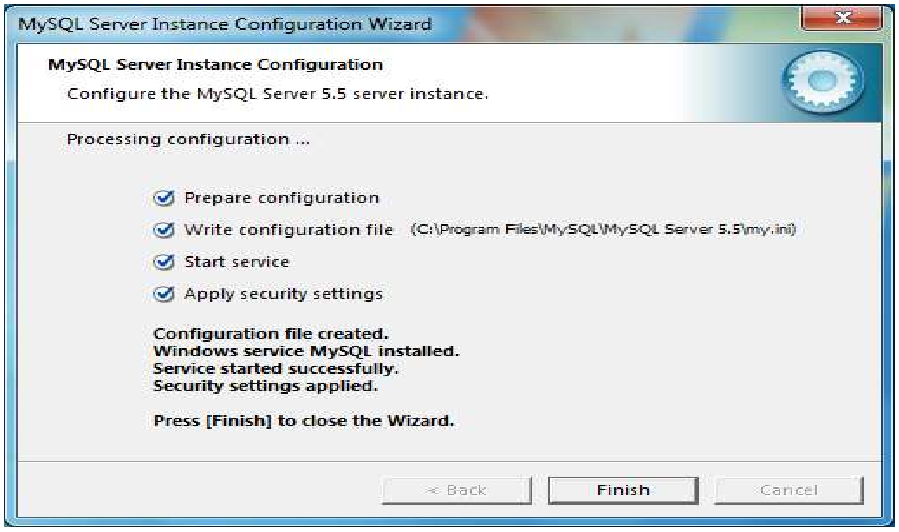
Configure the MySQL Server 5.5 server instance.

Ready to execute ...

- Prepare configuration
- Write configuration file
- Start service
- Apply security settings

Please press [Execute] to start the configuration.

11) Click on Finish button.



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.5 server instance.

Processing configuration ...

- ✓ Prepare configuration
- ✓ Write configuration file (C:\Program Files\MySQL\MySQL Server 5.5\my.ini)
- ✓ Start service
- ✓ Apply security settings

Configuration file created.
Windows service MySQL installed.
Service started successfully.
Security settings applied.

Press [Finish] to close the Wizard.

2C. Working with MySQL Server 5.5

1)Open MySQL Client

- Select Start Menu ->MySQL -> MySQL Server 5.5->MySQL Command Line Client.
- Provide password for root user
- When your password is correct, you can see the following SQL Prompt:

  MySQL>
- Now You can start Playing with MySQL.

2)Basic Operations

1) To See All the Databases

```
show   databases;
```

2) To Create the New Database

Syntax:	**Ex:**
create database <dbName>;	create database myjlcdb;

3) To Drop the Database

Syntax:	**Ex:**
drop database <dbName>;	drop database myjlcdb;

4) To select Required Database

Syntax:	**Ex:**
use <dbName>;	use myjlcdb;

5) To See the List of Tables in the Database

```
show   tables;
```

3. SQL Datatypes

About SQL

- SQL - Structured Query Language
- SQL is the standard language provided by ANSI(American National Standards Institute).
- All the database vendors has to follow the Syntax of ANSI while implementing their databases.
- Using SQL you can prepare simple SQL Statements to perform operations with a database like insert, update, delete, read etc.
- Datatypes are database vendor dependent.

Usage	**MySQL**
Integer Values	INT BIGINT LONG
Floating Values	FLOAT DOUBLE

Alphanumeric	CHAR(N)
	VARCHAR(N)
	TEXT

Date	DATE (YYYY-MM-DD)
Large Documents	CLOB
Binary Files (mp3, images, video)	BLOB and LONG BLOB

4. Creating Tables

- **Syntax:**

```
create table <table_name>(
<col1> <datatype>[size] [<constraint>],
<col2> <datatype>[size] [<constraint>],
  ..    ..        ..     ..,
<coln> <datatype>[size] [<constraint>]  );
```

- **Ex:**

```
create table mystudents(
sid int(3) primary key,
sname char(15),
email char(15),
phone long,
city char(15),
course char(15),
bal double
);
```

5. Inserting Data into Tables

- When you are inserting record in the table, you can provide the values for all the columns or for some columns.
- Use the following syntax when you provide the values for all the columns

  Syntax:

  ```
  insert into <table_name> values(val1,val2,...);
  ```

- Use the following syntax when you provide the values for some columns only

  Syntax:

  ```
  insert into <table_name>(col1,col2,...) values(val1,val2,...);
  ```

- **Examples:**

```
insert into mystudents values(101,'Sri','sri@jlc',123456,'Blore','Java FSD',9000);
insert into mystudents values(102,'sd','sd@jlc',654321,'Blore','Java FSD',5000);

insert into mystudents(sid,sname,email,city,course)
            values(103,'hello','hello@jlc','Blore','Java FSD');

insert into mystudents(sid,sname,email,city)
            values(104,'hello','hello@jlc','Blore');

insert into mystudents(sname,email,sid,city)
             values('hai','hai@jlc',105,'Blore');
```

6. Updating Records of Table

- **Syntax:**

  update <table_name> set col1=val1, col2=val2 ..... where <condition>;

- **Examples:**

  1) Update the bal of student whose whose sid is 101
  Ans :
  ```
  update mystudents set bal=0 where sid=101;
  ```

  2) Update the name,email and phone of student whose whose sid is 104.
  Ans:
  ```
  update mystudents set sname='srinivas',email='ss@myjlc',phone=99999
  where sid=104;
  ```

  3) Update the course name to Java FullStack whose sid >= 103
  Ans:
  ```
  update mystudents set course='Java FullStack' where sid>=103;
  ```

  4) Update the course name to Java FullStack
  Ans:
  ```
  update mystudents set course='Java FullStack';
  ```

7. Deleting Records of Table

- **Syntax:**

  delete from <table_name> where <condition>;

- **Examples:**

  1) Delete the student whose sid is 105.
  Ans:
  ```
  delete from mystudents where  sid=105;
  ```

  2) Delete the student whose course is Java FSD.
  Ans:
  ```
  delete from mystudents where course='Java FSD';
  ```

  3) Delete all the students
  Ans:
  ```
  delete from mystudents;
  ```

8. SQL Operators

a) Arithemetic Operators
    + , - , *, / , mod

b) Relational Operators
    <,<=,>,>=,
    =
    !=
    <>

c) Logical Operators
    AND, OR, NOT

d) IN Opeator

e) BETWEEN-AND Operator

f) LIKE Operator

g) IS NULL Operator

h) NOT Operator

9. Reading Records from Table

- **Syntax:**

```
select * (or) col1, col2....
from <table_name>
where <condition>
order by <col_name> [asc/desc]
group by <col_name>
having < condition >;
```

Practice Lab - 1

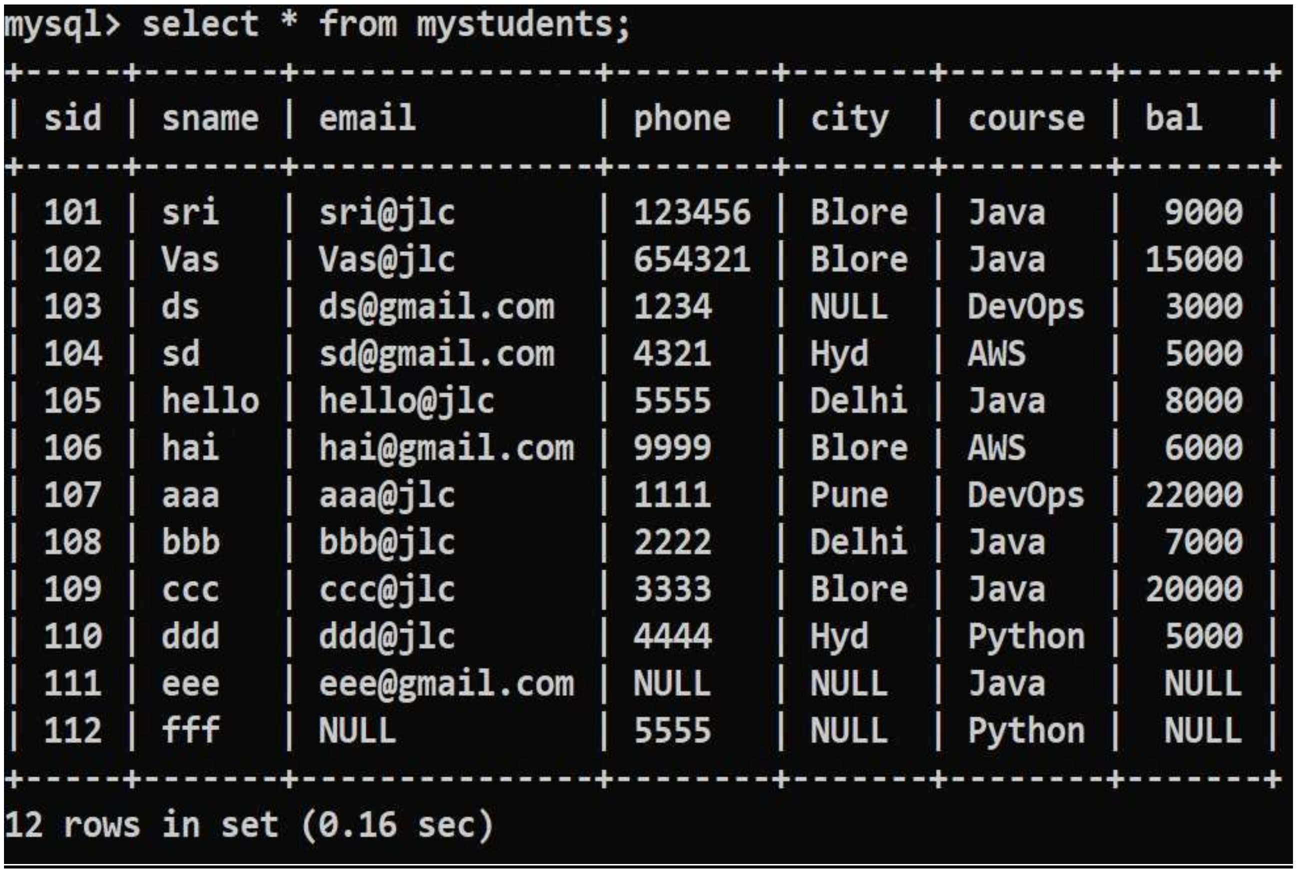
Step 1: Create the Table called mystudents as follows

```
create table mystudents(
sid int(3) primary key,
sname char(15),
email char(15),
phone long,
city char(15),
course char(15),
bal double
);
```

Step 2: Insert the Following Records

```
insert into mystudents values(101,'Sri','sri@jlc',123456,'Blore','Java',9000);
insert into mystudents values(102,'Vas','Vas@jlc',654321,'Blore','Java',15000);
insert into mystudents values(103,'ds','ds@gmail.com',1234,'Blore','DevOps',3000);
insert into mystudents values(104,'sd','sd@gmail.com',4321,'Hyd','AWS',5000);
insert into mystudents values(105,'hello','hello@jlc',5555,'Delhi','Java',8000);
insert into mystudents values(106,'hai','hai@gmail.com',9999,'Blore','AWS',6000);
insert into mystudents values(107,'aaa','aaa@jlc',1111,'Pune','DevOps',22000);
insert into mystudents values(108,'bbb','bbb@jlc',2222,'Delhi','Java',7000);
insert into mystudents values(109,'ccc','ccc@jlc',3333,'Blore','Java',20000);
insert into mystudents values(110,'ddd','ddd@jlc',4444,'Hyd','Python',5000);
insert into mystudents(sid,sname,email,course)
          values(111,'eee','eee@gmail.com','Java');
insert into mystudents(sid,sname,phone,course) values(112,'fff',5555,'Python');
```

Note : After inserting the records, table looks as follows



```
mysql> select * from mystudents;
+-----+-------+---------------+--------+-------+--------+-------+
| sid | sname | email         | phone  | city  | course | bal   |
+-----+-------+---------------+--------+-------+--------+-------+
| 101 | sri   | sri@jlc       | 123456 | Blore | Java   |  9000 |
| 102 | Vas   | Vas@jlc       | 654321 | Blore | Java   | 15000 |
| 103 | ds    | ds@gmail.com  | 1234   | NULL  | DevOps |  3000 |
| 104 | sd    | sd@gmail.com  | 4321   | Hyd   | AWS    |  5000 |
| 105 | hello | hello@jlc     | 5555   | Delhi | Java   |  8000 |
| 106 | hai   | hai@gmail.com | 9999   | Blore | AWS    |  6000 |
| 107 | aaa   | aaa@jlc       | 1111   | Pune  | DevOps | 22000 |
| 108 | bbb   | bbb@jlc       | 2222   | Delhi | Java   |  7000 |
| 109 | ccc   | ccc@jlc       | 3333   | Blore | Java   | 20000 |
| 110 | ddd   | ddd@jlc       | 4444   | Hyd   | Python |  5000 |
| 111 | eee   | eee@gmail.com | NULL   | NULL  | Java   |  NULL |
| 112 | fff   | NULL          | 5555   | NULL  | Python |  NULL |
+-----+-------+---------------+--------+-------+--------+-------+
12 rows in set (0.16 sec)
```

<u>Step 3: Practice the Following Queries</u>

Q1) Display full info of all the students.
Ans:

```
select * from mystudents;
```

2) Display name,phone and bal of all the students
Ans:

```
select sname,phone,bal from mystudents;
```

3) Display name,course of all the students
Ans:

```
select sname,course from mystudents;
```

4) Display the student whose sid is 105
Ans:

```
select * from mystudents
where sid=105;
```

5) Display the students who are staying in Bangalore.
Ans:

```
select * from mystudents
where city = 'Blore';

select * from mystudents
where city IN (Blore) ;
```

6) Display the students who are staying in Hyd and Delhi
Ans:

```
select * from mystudents
where city = 'Hyd' OR city = 'Delhi' ;

select * from mystudents
where city IN ('Hyd','Delhi') ;
```

7) Display the students who are staying in Hyd ,Delhi,Pune
Ans:

```
select * from mystudents
where city = 'Hyd' OR city = 'Delhi' OR city = 'Pune' ;

select * from mystudents
where city IN ('Hyd','Delhi','Pune') ;
```

8) Display the students who are Not staying in Hyd ,Blore
Ans:

```
select * from mystudents
where city != 'Hyd' AND city != 'Blore' ;

select * from mystudents
where city NOT IN ('Hyd','Blore');
```

9) Display the students whose bal is > 5000
Ans:

```
select * from mystudents
where bal>5000;
```

10) Display the students whose bal is <= 5000
Ans:

```
select * from mystudents
where bal<=5000;
```

11) Display the students whose bal is >=8000 and <=20000
Ans:

```
select * from mystudents
where bal>=8000 AND bal<=20000;

select * from mystudents
where bal BETWEEN 8000 AND 20000;
```

12) Display the students whose bal is >8000 and <20000
Ans:

```
select * from mystudents
where bal> 8000 AND bal<20000;
```

13) Display the students whose bal is <=8000 and >=20000
Ans:

```
select * from mystudents
where bal<=8000 OR bal>=20000;
```

14) Display the students whose bal is <8000 and >20000
Ans:

```
select * from mystudents
where bal<8000 OR bal>20000;

select * from mystudents
where bal NOT BETWEEN 8000 AND 20000;
```

15) Display the students whose bal is >=15000 and Staying in Blore
Ans:

```
select * from mystudents
where bal>=15000  AND city='Blore';
```

16) Display the students whose bal is >=5000 and Staying in Blore and Joined for AWS
Ans:

```
select * from mystudents
where bal>=5000  AND city='Blore' and course='AWS';
```

17) Display the students who joined for Java,AWS and Python
Ans:

```
select * from mystudents
where course IN ('Java','AWS','Python');
```

18) Display the Bangalore students who joined for Java,AWS and Python
Ans:

```
select * from mystudents
where city='Blore' AND course IN ('Java','AWS','Python');
```

19) Display the students who joined for Java,AWS and who are from Blore and Delhi abd whose bal is between 8000 and 20000
Ans:

```
select * from mystudents
where  bal BETWEEN 8000 AND 20000
        AND course IN ('Java','AWS')
        AND city in ('Blore','Delhi');
```

20) Display the students whose name starts with 's'
Ans:

```
select * from mystudents
where sname LIKE 's%';
```

21) Display the students whose name ends with 's'
Ans:

```
select * from mystudents
where sname LIKE '%s';
```

22) Display the students whose name's 2nd letter is 'a'
Ans:

```
select * from mystudents
where sname LIKE '_a%';
```

23) Display the students whose namecontains 'd'.
Ans:

```
select * from mystudents
where sname LIKE '%d%';
```

24) Display the students whose name's 2nd letter is 'a' from last.
Ans:

```
select * from mystudents
where sname LIKE '%a_';
```

25) Display the students whose are having gmail ID.
Ans:

```
select * from mystudents
where email LIKE '%gmail.com';
```

25) Display the students whose are having gmail ID.
Ans:

```
select * from mystudents
where email LIKE '%gmail.com';
```

26) Display the students who did not provide city.
Ans:

```
select * from mystudents
where city IS NULL;
```

27) Display the students who provided phone numbers.
Ans:

```
select * from mystudents
where phone IS NOT NULL;
```

28) Display the students whose name's 2nd letter is 'a' and staying Blore and bal is greater than 10000
Ans:

```
select * from mystudents
where sname LIKE '_a%'
        AND city='Blore'
        AND bal>=10000;
```

29) Display the students who are Not having gmail Id and Not Staying Blore and bal is <=10000 and joined Java Course
Ans:

```
select * from mystudents
where email NOT LIKE '%gmail.com'
        AND city!='Blore'
        AND bal <=10000
        AND course='Java';
```

30) Display the students by sorting the Bal in ASC
Ans:

```
select * from mystudents
order by bal;

select * from mystudents
order by bal ASC;
```

31) Display the students by sorting the Bal in DESC
Ans:

```
select * from mystudents
order by bal DESC;
```

32) Display the students who are staying in Balore by sorting the name in Alphabetical Order
Ans:

```
select * from mystudents
where city='Blore'
order by sname;
```

10. Aggregate Functions

- Aggregate functions perform the Calculations on multiple values of a column and returns one value after Calculation.
- Aggregate functions ignore NULL values when it performs the calculation.
- We use aggregate functions with the **GROUP BY** and **HAVING** clauses of the **SELECT** statement.
- There are 5 aggregate functions in SQL:

  Count()

  Sum()

  Avg()

  Min()

  Max()

Practice Lab - 2

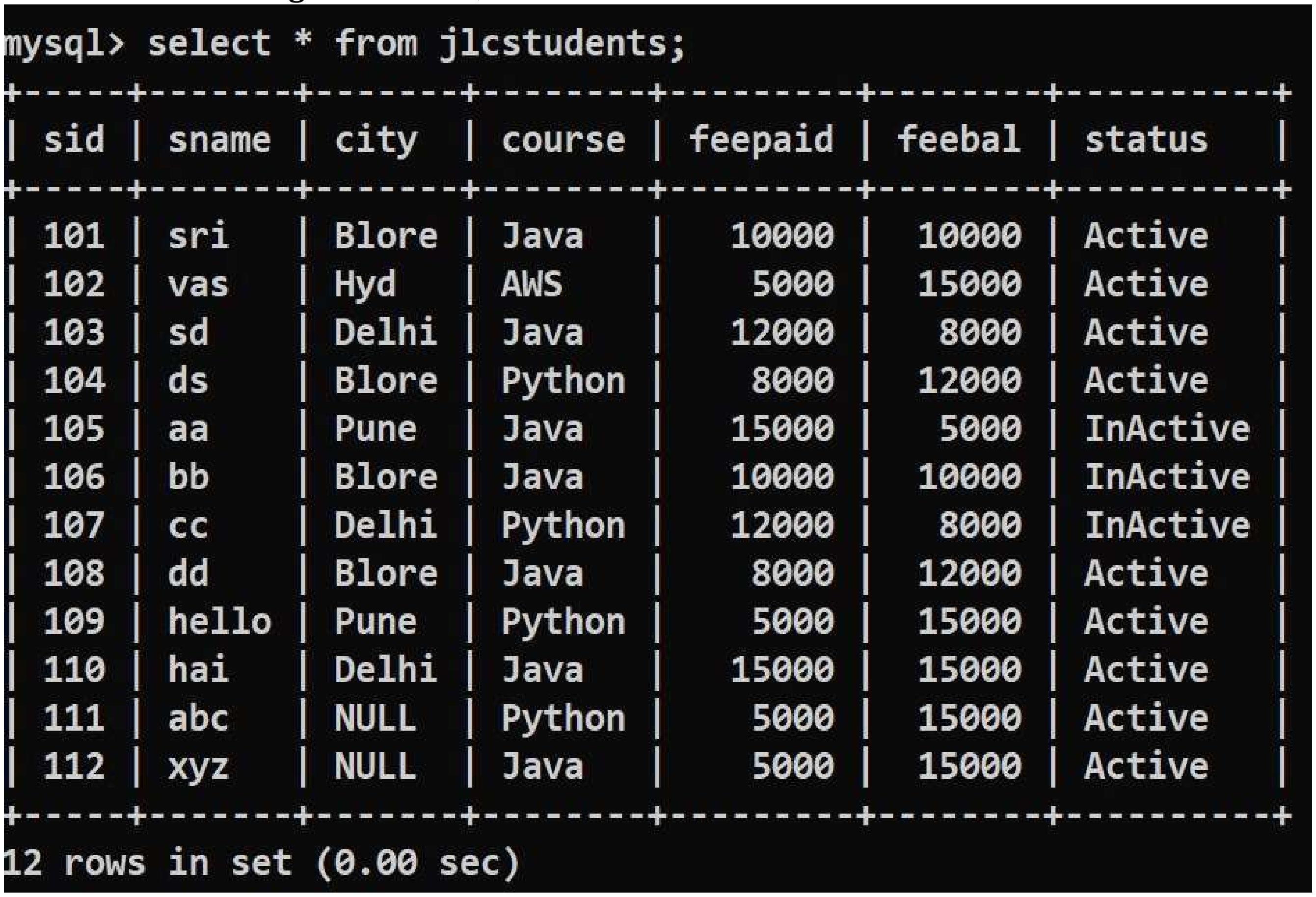
Step 1: Create the Table called jlcstudents as follows

```
create table jlcstudents(
sid int(3) primary key,
sname char(15),
city char(15),
course char(15),
feepaid double,
feebal double,
status char(10)
);
```

Step 2: Insert the Following Records

```
insert into jlcstudents values(101,'sri','Blore','Java',10000,10000,'Active');
insert into jlcstudents values(102,'vas','Hyd','AWS',5000,15000,'Active');
insert into jlcstudents values(103,'sd','Delhi','Java',12000,8000,'Active');
insert into jlcstudents values(104,'ds','Blore','Python',8000,12000,'Active');
insert into jlcstudents values(105,'aa','Pune','Java',15000,5000,'InActive');
insert into jlcstudents values(106,'bb','Blore','Java',10000,10000,'InActive');
insert into jlcstudents values(107,'cc','Delhi','Python',12000,8000,'InActive');
insert into jlcstudents values(108,'dd','Blore','Java',8000,12000,'Active');
insert into jlcstudents values(109,'hello','Pune','Python',5000,15000,'Active');
insert into jlcstudents values(110,'hai','Delhi','Java',15000,15000,'Active');

insert into jlcstudents(sid,sname,course,feepaid,feebal)
values(111,'abc','Python',5000,15000);

insert into jlcstudents(sid,sname,course,feepaid,feebal)
values(112,'xyz','Java',5000,15000);
```

Note : After inserting the records, table looks as follows



```
mysql> select * from jlcstudents;
+-----+-------+-------+--------+---------+--------+----------+
| sid | sname | city  | course | feepaid | feebal | status   |
+-----+-------+-------+--------+---------+--------+----------+
| 101 | sri   | Blore | Java   |   10000 |  10000 | Active   |
| 102 | vas   | Hyd   | AWS    |    5000 |  15000 | Active   |
| 103 | sd    | Delhi | Java   |   12000 |   8000 | Active   |
| 104 | ds    | Blore | Python |    8000 |  12000 | Active   |
| 105 | aa    | Pune  | Java   |   15000 |   5000 | InActive |
| 106 | bb    | Blore | Java   |   10000 |  10000 | InActive |
| 107 | cc    | Delhi | Python |   12000 |   8000 | InActive |
| 108 | dd    | Blore | Java   |    8000 |  12000 | Active   |
| 109 | hello | Pune  | Python |    5000 |  15000 | Active   |
| 110 | hai   | Delhi | Java   |   15000 |  15000 | Active   |
| 111 | abc   | NULL  | Python |    5000 |  15000 | Active   |
| 112 | xyz   | NULL  | Java   |    5000 |  15000 | Active   |
+-----+-------+-------+--------+---------+--------+----------+
12 rows in set (0.00 sec)
```

Step 3: Practice the Following Queries

1) Display info of all the students
Ans:

```
select * from jlcstudents;
```

2) How many students are joined till date.
Ans:

```
select count(*) from jlcstudents;
select count(sid) from jlcstudents;

select count(*) as "Total Students" from jlcstudents;
```

3) How many students are joined for Java?
Ans:

```
select count(*) from jlcstudents
where course = 'Java';
```

4) How many students are joined for Java from Blore?
Ans:

```
select count(*) from jlcstudents
where course = 'Java' AND city = 'Blore';
```

5) How many students have provided City?
Ans:

```
select count(*) from jlcstudents
where city is NOT NULL;
```

5) How many Course are there in JLC?
Ans:

```
select count(course) from jlcstudents;

select count(DISTINCT course) from jlcstudents;
```

6) How many City related students joined in JLC?
Ans:

```
select count(city) from jlcstudents;

select count(DISTINCT city) from jlcstudents;
```

7) What is the Total Fee Collected?
Ans:

```
select sum(feepaid) from jlcstudents;

select sum(feepaid) AS "Total Fee Collected" from jlcstudents;
```

8) What is the Total Fee Collected for Java Course?
Ans:

```
select sum(feepaid) from jlcstudents
where course='Java';
```

9) What is the Total Fee Collected for Java Course from Blore?
Ans:

```
select sum(feepaid) from jlcstudents
where course='Java' and city='Blore';
```

10) What is the Minimum Fee Paid?
Ans:

```
select min(feepaid) from jlcstudents;
```

11) What is the Maximum Fee Paid?
Ans:

```
select max(feepaid) from jlcstudents;
```

12) What is the Minimum Fee Bal?
Ans:

```
select min(feebal) from jlcstudents;
```

13) What is the Maximum Fee Bal?
Ans:

```
select max(feebal) from jlcstudents;
```

14) What is the average fee Paid?
Ans:

```
select avg(feepaid) from jlcstudents;
select sum(feepaid)/count(feepaid) from jlcstudents;
```

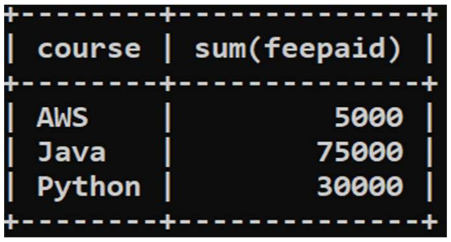
15) What is the Next Student Id to provide?
Ans:

```
select max(sid)+1 from jlcstudents;
```

11. Group By and Having Clauses

- **GROUP BY** clause is used to divide the table into multiple groups based on specified column.
- **HAVING** clause is used to specify the condition on groups.

16) Display the course wise fee Collection in the following format

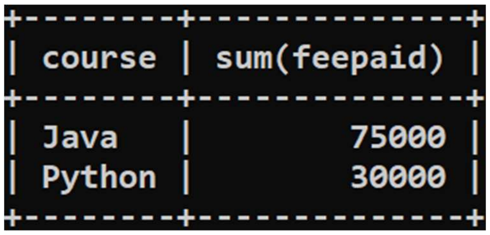


```
+--------+--------------+
| course | sum(feepaid) |
+--------+--------------+
| AWS    |         5000 |
| Java   |        75000 |
| Python |        30000 |
+--------+--------------+
```

Ans:

select course,sum(feepaid) from jlcstudents
group by course;

17) Display the course wise fee Collection of Java and Python in the following format

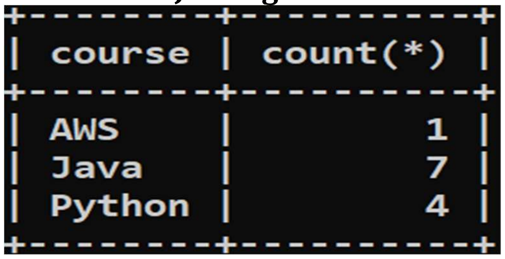


```
+--------+--------------+
| course | sum(feepaid) |
+--------+--------------+
| Java   |        75000 |
| Python |        30000 |
+--------+--------------+
```

Ans:

select course,sum(feepaid) from jlcstudents
group by course
having course IN ('Java','Python');

18) Display the course wise Joinings in the following format

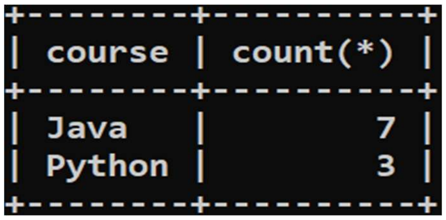


```
+--------+----------+
| course | count(*) |
+--------+----------+
| AWS    |        1 |
| Java   |        7 |
| Python |        4 |
+--------+----------+
```

Ans:

select course, count(*) from jlcstudents
group by course;

19) Display the course wise Joinings of Java and Python in the following format

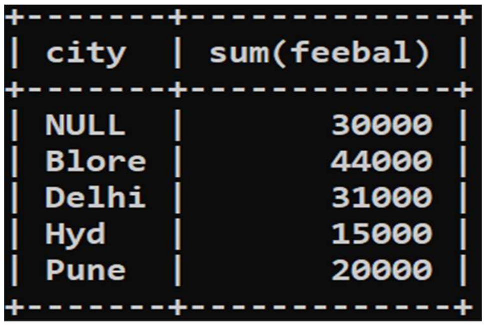


```
+--------+----------+
| course | count(*) |
+--------+----------+
| Java   |        7 |
| Python |        3 |
+--------+----------+
```

Ans:

select course, count(*) from jlcstudents
group by course
having course IN ('Java','Python');

20) Display the City wise fee Balances in the following format

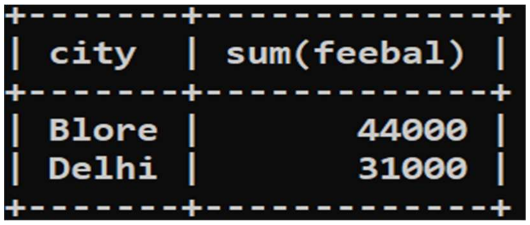


```
+-------+-------------+
| city  | sum(feebal) |
+-------+-------------+
| NULL  |       30000 |
| Blore |       44000 |
| Delhi |       31000 |
| Hyd   |       15000 |
| Pune  |       20000 |
+-------+-------------+
```

Ans:

select city,sum(feebal) from jlcstudents
group by city;

21) Display the City wise fee Balances from Blore and Delhi in the following format

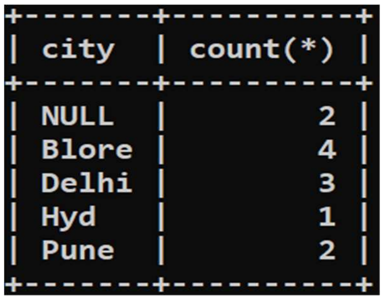


```
+-------+-------------+
| city  | sum(feebal) |
+-------+-------------+
| Blore |       44000 |
| Delhi |       31000 |
+-------+-------------+
```

Ans:

select city,sum(feebal) from jlcstudents
group by city
having city IN ('Blore','Delhi');

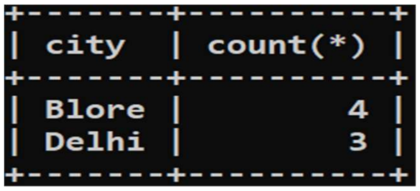
22) Display the City wise Joinings in the following format


```
+-------+----------+
| city  | count(*) |
+-------+----------+
| NULL  |        2 |
| Blore |        4 |
| Delhi |        3 |
| Hyd   |        1 |
| Pune  |        2 |
+-------+----------+
```

Ans:

select city,count(*) from jlcstudents
group by city;

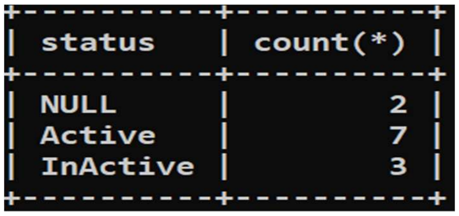
23) Display the City wise Joinings from Blore and Delhi in the following format


```
+-------+----------+
| city  | count(*) |
+-------+----------+
| Blore |        4 |
| Delhi |        3 |
+-------+----------+
```

Ans:

select city,count(*) from jlcstudents
group by city
having city IN ('Blore','Delhi');

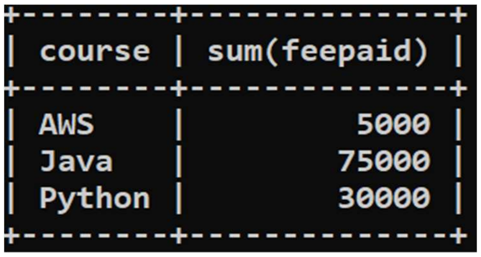
24) Display the students count status-wise in the following format


```
+----------+----------+
| status   | count(*) |
+----------+----------+
| NULL     |        2 |
| Active   |        7 |
| InActive |        3 |
+----------+----------+
```

Ans:

select status,count(*) from jlcstudents
group by status;

25) Display the course wise fee Collection in the following format by Sorting the Course in ASC

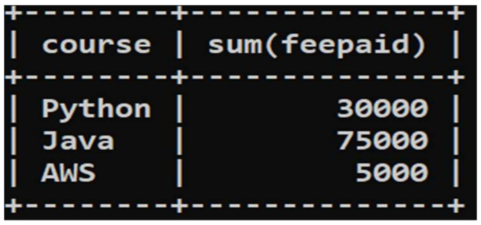


```
+--------+--------------+
| course | sum(feepaid) |
+--------+--------------+
| AWS    |         5000 |
| Java   |        75000 |
| Python |        30000 |
+--------+--------------+
```

Ans:

```
select course,sum(feepaid) from jlcstudents
group by course
order by course ASC;
```

26) Display the course wise fee Collection in the following format By Sorting the Course in DESC



```
+--------+--------------+
| course | sum(feepaid) |
+--------+--------------+
| Python |        30000 |
| Java   |        75000 |
| AWS    |         5000 |
+--------+--------------+
```

Ans:

```
select course,sum(feepaid) from jlcstudents
group by course
order by course DESC;
```

12. Constraints

- Constraints are the rules that will be applied on the column of a table.
- It allows you to restrict only specific data that meets the regulations to go to a table

Types of Constraints

1) NOT NULL Constraint
2) UNIQUE Constraint
3) PRIMARY KEY Constraint
4) FOREIGN KEY Constraint
5) DEFAULT Constraint
6) CHECK Constraint

1) NOT NULL Constraint

- When you are not supplying any value for the column then NULL will be inserted by default.
- NULL means value is not available
  - NULL is not equivalent to NULL or null or space or \0 or any other character.
- If you don't want NULL value for any column or if you want to force the user to supply value for given column then you can use NOT NULL constraint for that column.
- NOT NULL constraint can be used for multiple column of the same table

Practice Lab - 3

Step 1: Create the following table

```
create table mycustomers1(
cid int NOT NULL,
cname char(15) NOT NULL,
email char(15) NOT NULL,
phone int(10)
city char(15)
);
```

Step 2: Describe the table to see the table structure

```
desc mycustomers1;
```

Step 3: Insert the Following Records

```
insert into mycustomers1(cid) values(101); //Error

insert into mycustomers1(cid,cname,email) values(101,'sri','sri@myjlc.com');  //OK

insert into mycustomers1 values(102,'sd','sd@myjlc.com',12345,'Blore'); //OK

insert into mycustomers1 values(102,'sd','sd@myjlc.com',12345,'Blore'); //OK
```

2) UNIQUE Constraint

- By default you can insert duplicate value in any column.
- If you want unique values for any column then you can use UNIQUE Constraint.
- Column specified with UNIQUE Constraint can not have duplicate values but can many nulls in that coloumn.
- UNIQUE constraint can be used for multiple column of the same table.
- UNIQUE constraint can be used in combination with NOT NULL also.

Practice Lab - 4

Step 1: Create the following table

```
create table mycustomers2(
cid int NOT NULL UNIQUE,
cname char(15) NOT NULL,
email char(15) NOT NULL UNIQUE,
phone int(10) UNIQUE,
city char(15)
);
```

Step 2: Describe the table to see the table structure

```
desc mycustomers2;
```

Step 3: Insert the Following Records

```
insert into mycustomers2(cid,cname,email) values(101,'sri','sri@myjlc.com');   //OK

insert into mycustomers2(cid,cname,email) values(102,'sri','sri@myjlc.com');  //Error

insert into mycustomers2 values(102,'sd','sd@myjlc.com',12345,'Blore'); //OK

insert into mycustomers2 values(102,'sd','sd@jlc.com',12345,'Blore'); //Error

insert into mycustomers2(cid,cname,email) values(103,'ds','ds@myjlc.com');   //OK

insert into mycustomers2(cid,cname,email) values(104,'ds','ds@jlc.com'); //OK

insert into mycustomers2 values(105,'hello','hello@myjlc.com',NULL,'Blore'); //OK

insert into mycustomers2 values(106,'hai',NULL,NULL,'Blore'); // Error

insert into mycustomers2 values(106,'hai','NULL',NULL,'Blore'); //OK
```

3) PRIMARY KEY Constraint

- When you apply Primary Key constraint then both unique and not null constraints will be applied on the column.
- Table should contain only one Primary Key.
- Primary key is mainly used to identify the records uniquely in a table.

Types Of Primary Key

a) Simple Primary Key
b) Composite Primary Key

a) Simple Primary Key:

- When you specify the Primary Key for single column of a table then it is called as Simple Primary Key.
- Table must have only one Primary Key.

Practice Lab - 5

Step 1: Create the following table

```
create table mycustomers3(
cid int Primary Key,
cname char(15) NOT NULL,
email char(15) NOT NULL UNIQUE,
phone int(10) NOT NULL UNIQUE,
city char(15)
);
```

Step 2: Describe the table to see the table structure

```
desc mycustomers3;
```

Step 3: Insert the Following Records

```
insert into mycustomers3(cid,cname,email) values(101,'sri','sri@myjlc.com');   //Error

insert into mycustomers3(cid,cname,email,phone) values(101,'sri','sri@myjlc.com',123);

insert into mycustomers3 values(102,'sd','sd@myjlc.com',12345,'Blore'); //OK

insert into mycustomers3 values(102,'ds','ds@jlc.com',54321,'Blore'); //Error
```

Q) Display mycustomers details based on P.K
Ans:

```
select * from mycustomers3 where cid=101;
```

b) Composite Primary Key:

- When you specify Primary Key for combination of two or more columns of a table then it is called as Composite Primary Key.

Practice Lab - 6

Step 1: Create the following table

```
create table students(
bid char(3),
sid int,
sname char(15) NOT NULL,
email char(15) NOT NULL UNIQUE,
phone int(10) NOT NULL UNIQUE,
course char(15),
Primary Key(bid,sid)
);
```

Step 2: Describe the table to see the table structure

```
desc students;
```

Step 3: Insert the Following Records

```
insert into students values('B1',101,'sri','sri@myjlc.com',111,'Java');
insert into students values('B1',102,'sd','sd@myjlc.com',222,'Java');
insert into students values('B2',101,'sri','ss@myjlc.com',333,'Java');
insert into students values('B2',102,'sri','dd@myjlc.com',444,'Java');
insert into students values('B2',101,'hello','hello@myjlc.com',555,'Java');  //Error
insert into students values('B2',103,'hello','hello@myjlc.com',555,'Java');
```

Q) Display mycustomers details based on P.K
Ans:

```
select * from students where bid='B2' and sid=103;
```

4) FOREIGN KEY Constraint

- Foreign Key Constraint is used to establish the relationship among two or more tables.
- The table which contains main information is called as master table.
- The table which contains related information is called as child table.
- When you are inserting the information in child table then related information will be verified in parent table.
- When you are deleting the information from parent table then related information from child table also should be verified.

Practice Lab - 7

Step 1: Create the following 3 tables

```
create table mycustomers(
cid int Primary Key,
cname char(15) NOT NULL,
email char(15) NOT NULL UNIQUE,
phone int(10) UNIQUE,
city char(15)
);

create table myaccounts(
mycid int ,
accno int Primary Key,
atype char(15) NOT NULL UNIQUE,
branch char(15) NOT NULL,
bal double,
foreign key(mycid) references mycustomers(cid)
);

create table mytransactions(
myaccno int,
txNumber int primary key,
txDate date NOT NULL,
amount double,
txType char(2),
foreign key(myaccno) references myaccounts(accno)
);
```

Step 2: Describe the tables to see the table structure

```
desc mycustomers;
desc myaccounts;
desc mytransactions;
```

Step 3: Insert the Following Records

```
insert into mycustomers values(101,'sri','sri@jlc',12345,'Blore');

insert into myaccounts values(101,555,'SA','BTM',25000);
insert into myaccounts values(101,999,'CA','BTM',55000);

insert into mytransactions values(999,1,sysdate(),5000,'Dr');
insert into mytransactions values(555,2,sysdate(),2000,'Cr');
```

5) DEFAULT Constraint

- By default, NULL is the default value for all the columns.
- Default Constraint can be used to supply value other than NULL value as default value.
- If you supply value for all the columns then your value will be inserted otherwise default value will be inserted.
- Syntax:

```
<col_name> default(value);
```

Practice Lab - 8

Step 1: Create the following 3 tables

```
create table students1(
sid int primary key,
sname char(15) NOT NULL,
email char(15) NOT NULL UNIQUE,
phone int(10) NOT NULL UNIQUE,
course char(15) DEFAULT 'Java',
city char(15) DEFAULT 'Blore'
);
```

Step 2: Describe the tables to see the table structure

```
desc students1;
```

Step 3: Insert the Following Records

```
insert into students1(sid,sname,email,phone) values(101,'sri','sri@jlc',123);
```

6) CHECK Constraint

- Check Constraint is used to design user defined rules on any Column.
- Syntax:

```
<col_name> check(<condition>);
```

Practice Lab - 9

Step 1: Create the following 3 tables

```
create table students2(
sid int primary key,
sname char(15) NOT NULL,
email char(15) NOT NULL UNIQUE,
phone int(10) UNIQUE,
course char(15) DEFAULT 'Java',
totalfee double CHECK(totalfee >=25000)
);
```

Step 2: Describe the tables to see the table structure

```
desc students2;
```

Step 3: Insert the Following Records

```
insert into students2 values(101,'sri','sri@jlc',123,'Java',20000);
```

13. Joins

- Joins can be used fetch the data from two or more tables.
- When you are joining tables, there should be some common columns available to specify join condition.
- SQL Joins are mostly used when a user is trying to extricate data from multiple tables at one time.

Types of Joins

- Inner Joins
- Outer Joins
    - Left Outer Joins
    - Right Outer Joins
    - Full Outer Joins
- Self joins
- Cross Joins

Practice Lab - 10

Step 1: Create the following 3 tables

```
create table customers(
cid int(3) Primary Key,
cname char(15) NOT NULL,
email char(15) NOT NULL UNIQUE,
phone int(10) NOT NULL UNIQUE
);

create table accounts(
mycid int(3),
accno int(5) Primary Key,
atype char(2) NOT NULL,
bal double NOT NULL
);

create table address(
mycid int(3),
addid int(3) Primary Key,
street char(15) NOT NULL,
city char(15) NOT NULL,
state char(15) NOT NULL
);
```

Step 2: Describe the table to see the table structure

```
desc customers;
desc accounts;
desc address;
```

Step 3: Insert the Following Records

```
insert into customers values(101,'sri','sri@jlc',111);
insert into customers values(102,'vas','vas@jlc',222);
insert into customers values(103,'sd','sd@jlc',333);
insert into customers values(104,'ds','ds@jlc',444);
insert into customers values(105,'hello','hello@jlc',555);
insert into customers values(106,'hai','hai@jlc',666);
insert into customers values(107,'aaa','aaa@jlc',777);
insert into customers values(108,'bbb','bbb@jlc',888);
insert into customers values(109,'ccc','ccc@jlc',999);

insert into accounts values(101,12345,'SA',5000);
insert into accounts values(102,12346,'SA',15000);
insert into accounts values(103,12347,'SA',25000);
insert into accounts values(107,12348,'SA',3000);
insert into accounts values(108,12349,'SA',13000);
insert into accounts values(109,12350,'SA',18000);

insert into address values(101,1,'BTM','Blore','KA');
insert into address values(102,2,'MHA','Blore','KA');
insert into address values(103,3,'P1','Pune','MH');
insert into address values(104,4,'D1','Delhi','Delhi');
insert into address values(109,5,'D2','Delhi','Delhi');
insert into address values(110,6,'P2','Pune','MH');
insert into address values(111,7,'H1','Hyd','TG');
insert into address values(112,8,'pp','Patna','BR');
```

Step 4: Inner Joins Questions

- Inner join is also called as Equi Join.
- Inner Joins gives the Matching Records of the Joined tables

Q1) Display customers personal and accounts info?
Ans:

```
select *
from customers cust,accounts acc
where cust.cid=acc.mycid;

select *
from customers cust
inner join accounts acc  on cust.cid=acc.mycid;
```

Q2) Display cid,cname,phone,accno,bal of customers
Ans:

```
select cid,cname,phone,accno,bal
from customers cust,accounts acc
where cust.cid=acc.mycid;

select cid,cname,phone,accno,bal
from customers cust
inner join accounts acc on cust.cid=acc.mycid;
```

Q3) Display customers personal and address info
Ans:

```
select *
from customers cust,address addr
where cust.cid=addr.mycid;

select *
from customers cust
inner join address addr on cust.cid=addr.mycid;
```

Q4) Display cid,cname,phone,city,state of customers
Ans:

```
select cid,cname,phone,city,state
from customers cust,address addr
where cust.cid=addr.mycid;

select cid,cname,phone,city,state
from customers cust
inner join address addr on cust.cid=addr.mycid;
```

Q5) Display customers personal , accounts and address info
Ans:

```
select *
from customers cust, accounts acc,address addr
where cust.cid=acc.mycid and cust.cid=addr.mycid;

select *
from customers cust
inner join accounts acc on  cust.cid=acc.mycid
inner join address addr on cust.cid=addr.mycid;
```

Q6) Display cid,cname,phone,accno,bal,city,state of customers.
Ans:

```
select cid,cname,phone,accno,bal,city,state
from customers cust, accounts acc,address addr
where cust.cid=acc.mycid and cust.cid=addr.mycid;

select cid,cname,phone,accno,bal,city,state
from customers cust
inner join accounts acc on  cust.cid=acc.mycid
inner join address addr on cust.cid=addr.mycid;
```

Step 5: Left Outer Joins Questions

- Left Outer Joins gives the Matching Records of the Joined tables + Records remaining in the Left side Table

Q1) Display customers personal and accounts info?
Ans:
```
select *
from customers cust
left join accounts acc  on cust.cid=acc.mycid;
```

Q2) Display cid,cname,phone,accno,bal of customers
Ans:
```
select cid,cname,phone,accno,bal
from customers cust
left join accounts acc  on cust.cid=acc.mycid ;
```

Q3) Display customers personal and address info
Ans:
```
select *
from customers cust
left join address addr on cust.cid=addr.mycid;
```

Q4) Display cid,cname,phone,city,state of customers
Ans:
```
select cid,cname,phone,city,state
from customers cust
left join address addr on cust.cid=addr.mycid;
```

Q5) Display customers personal , accounts and address info
Ans:
```
select *
from customers cust
left  join accounts acc on  cust.cid=acc.mycid
left  join address addr on cust.cid=addr.mycid;
```

Q6) Display cid,cname,phone,accno,bal,city,state of customers.
Ans:

```
select cid,cname,phone,accno,bal,city,state
from customers cust
left  join accounts acc on  cust.cid=acc.mycid
left  join address addr on cust.cid=addr.mycid;
```

Step 6: Right Outer Joins Questions

- Right Outer Joins gives the Matching Records of the Joined tables + Records remaining in the Right side Table

Q1) Display customers personal and accounts info?
Ans:

```
select *
from customers cust
right join accounts acc  on cust.cid=acc.mycid;
```

Q2) Display cid,cname,phone,accno,bal of customers
Ans:

```
select cid,cname,phone,accno,bal
from customers cust
right join accounts acc  on cust.cid=acc.mycid ;
```

Q3) Display customers personal and address info
Ans:

```
select *
from customers cust
right join address addr on cust.cid=addr.mycid;
```

Q4) Display cid,cname,phone,city,state of customers
Ans:

```
select cid,cname,phone,city,state
from customers cust
right join address addr on cust.cid=addr.mycid;
```

Q5) Display customers personal , accounts and address info

Ans:

```
select *
from customers cust
right join accounts acc on  cust.cid=acc.mycid
right join address addr on cust.cid=addr.mycid;
```

Q6) Display cid,cname,phone,accno,bal,city,state of customers.

Ans:

```
select cid,cname,phone,accno,bal,city,state
from customers cust
right join accounts acc on  cust.cid=acc.mycid
right join address addr on cust.cid=addr.mycid;
```

Step 7: Full Outer Joins Questions

- Full Outer Joins gives the Matching Records of the Joined tables + Records remaining in the Left side Table + + Records remaining in the Right side Table
- **Use Union to implement the Full Outer Joins**

Q1) Display customers personal and accounts info?

Ans:

```
select *  from customers cust
left join accounts acc  on cust.cid=acc.mycid
union
select * from customers cust
right join accounts acc  on cust.cid=acc.mycid;
```

Q2) Display customers personal and address info

Ans:

```
select *  from customers cust
left join address addr on cust.cid=addr.mycid
union
select * from customers cust
right join address addr on cust.cid=addr.mycid;
```

Q3) Display customers personal , accounts and address info
Ans:

```
select *  from customers cust
left join accounts acc on  cust.cid=acc.mycid
left join address addr on cust.cid=addr.mycid
union
select *  from customers cust
right join accounts acc on  cust.cid=acc.mycid
right join address addr on cust.cid=addr.mycid;
```

Step 8: Joins with Extra Conditions - Questions

Q1) Display customers personal and accounts info whose bal>=15000
Ans:

```
select * from customers cust
inner join accounts acc  on cust.cid=acc.mycid and bal>=15000;
```

Q2) Display customers personal and address info who are staying in Blore
Ans:

```
select * from customers cust
inner join address addr on cust.cid=addr.mycid and city='Blore';
```

Q3) Display customers personal and address info who are staying in Blore and Pune
Ans:

```
select * from customers cust
inner join address addr on cust.cid=addr.mycid and city in ('Blore','Pune');
```

Q4) Display customers personal , accounts and address info who are staying in Blore and Delhi
Ans:

```
select * from customers cust
inner join accounts acc on  cust.cid=acc.mycid
inner join address addr on cust.cid=addr.mycid and city in ('Blore','Delhi');
```

Q5) Display customers personal , accounts and address info whose bal>=15000
Ans:

```
select * from customers cust
inner join accounts acc on  cust.cid=acc.mycid and bal>=15000
inner join address addr on cust.cid=addr.mycid;
```

Q6) Display customers personal , accounts and address info who are staying in Blore and Delhi and whose bal>=15000
Ans:

```
select * from customers cust
inner join accounts acc on  cust.cid=acc.mycid and bal>=15000
inner join address addr on cust.cid=addr.mycid and city in ('Blore','Delhi');
```

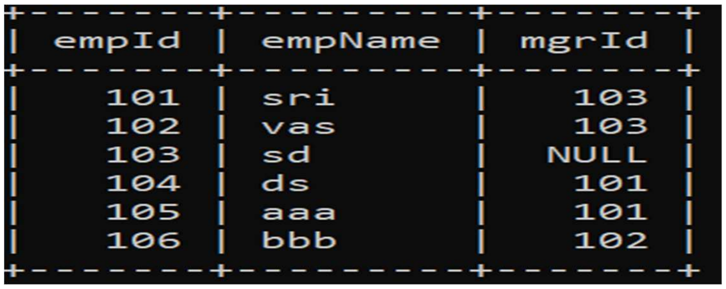
Step 9: Self Joins - Questions

- Joining the table to itself is called as self join.
- Self join is same as any other join, but in this join multiple instance of same table will participate the join query.

Ex:

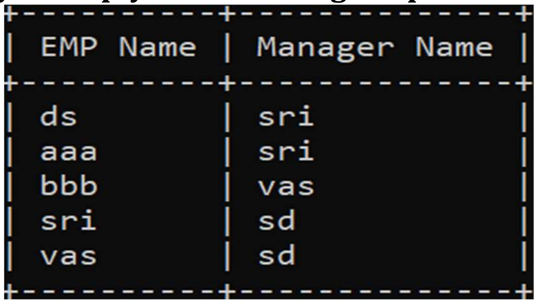
```
create table myemployees(
empId int(3),
empName char(15),
mgrId int(3)
);

insert into myemployees values(101,'sri',103);
insert into myemployees values(102,'vas',103);
insert into myemployees values(103,'sd',NULL);
insert into myemployees values(104,'ds',101);
insert into myemployees values(105,'aaa',101);
insert into myemployees values(106,'bbb',102);
```

After Inserting the Records, Table looks as follows.



```
+-------+---------+-------+
| empId | empName | mgrId |
+-------+---------+-------+
|   101 | sri     |   103 |
|   102 | vas     |   103 |
|   103 | sd      |  NULL |
|   104 | ds      |   101 |
|   105 | aaa     |   101 |
|   106 | bbb     |   102 |
+-------+---------+-------+
```

Q1) Write the Query to disply the following Output?

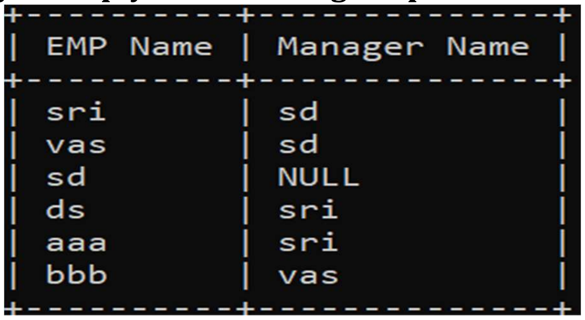


```
+----------+--------------+
| EMP Name | Manager Name |
+----------+--------------+
| ds       | sri          |
| aaa      | sri          |
| bbb      | vas          |
| sri      | sd           |
| vas      | sd           |
+----------+--------------+
```

Ans:

```
select emp.empName as "EMP Name", mgr.empName as "Manager Name"
from  myemployees emp, myemployees mgr
where emp.mgrId = mgr.empId;

select emp.empName as "EMP Name", mgr.empName as "Manager Name"
from  myemployees emp
inner join myemployees mgr on emp.mgrId = mgr.empId;
```

Q2) Write the Query to disply the following Output?



```
+----------+--------------+
| EMP Name | Manager Name |
+----------+--------------+
| sri      | sd           |
| vas      | sd           |
| sd       | NULL         |
| ds       | sri          |
| aaa      | sri          |
| bbb      | vas          |
+----------+--------------+
```

Ans:

```
select emp.empName as "EMP Name", mgr.empName as "Manager Name"
from  myemployees emp
left join myemployees mgr on emp.mgrId = mgr.empId;
```

Step 10: Cross Joins - Questions

- When you are not providing condition in join query then it will give cartiasian product of joined table called as cross join.

Ex:

```
select cid,cname,accno,bal
from customers cust, accounts acc;
```

14. Sub Queries

- Subquery is a Query that is nested inside Other statement.
- Subquery can be nested inside another Subquery also.
- Subquery is also called an inner query or inner select, while the statement containing a subquery is also called an outer query or outer select or main query.
- First sub query will be evaluated and results returned by subquery will be used main query
- Sub Query returns zero or more results depending on the condition provided in subquery.
- When sub query returns exactly one record then you use = operator to assign the result of sub query to main query.
- When sub query returns more than one records then you use IN operator to assign the result of sub query to main query.

Practice Lab - 11

Step 1: Create the following 3 tables

- **Tables same as Practice Lab10**

Step 2: Describe the table to see the table structure

```
desc customers;
desc accounts;
desc address;
```

Step 3: Insert the Following Records

- **Records same as Practice Lab10**

Step 4: SubQuery Questions

Q1) Display the Bal of the Customer whose Phone is 333 (Phone is Unique)
Ans:

Without Subquery

```
select cid from customers where phone=333; ( sub query)
select bal from accounts where mycid=103;  ( main query)
```

With Subquery

```
select bal from accounts
where mycid = (select cid from customers where phone=333) ;
```

Q2) Display the Accno and Bal of the Customers who are staying in Blore.
Ans:

<u>**Without Subquery**</u>

```
select mycid from address where city='Blore'; ( sub query)
select accno,bal from accounts where mycid = (101,102);  ( main query)
```

<u>**With Subquery**</u>

```
select accno,bal from accounts
where mycid IN (select mycid from address where city='Blore') ;
```

Q3) Display the City of Customer whose email is ccc@jlc (Email is Unique)
Ans:

```
select city from address
where mycid = (select cid from customers where email='ccc@jlc') ;
```

Q4) Display the cname,email, phone of customer whose accno is 12345.
Ans:

```
select cname,email,phone from customers
where cid=(select mycid from accounts where accno=12345);
```

Q5) Display the Customers who are not maintaining the Min Balance in the Account.(min = 20000)
Ans:

```
select cname,email,phone from customers
where cid IN (select mycid from accounts where bal<20000);
```

Q6) Display the cname,email,street,city who are not maintaining the Min Balance in the Account.(min = 20000)
Ans:

```
select cname,email,street,city
from customers cust inner join address addr on cust.cid = addr.mycid and cid IN (select mycid from accounts where bal<20000);
```

Q7) Display cname,email,accno,bal,street,city of customers who are in Blore.
Ans:

```
select cname,email,accno,bal,street,city
from customers cust
inner join accounts acc on cust.cid=acc.mycid
inner join address addr on cust.cid = addr.mycid and  addr.mycid in (select mycid from address where city='Blore');
```

Creating new table from existing table

Q8) Create table from customers with all columns and data
Ans:

```
create table mycust1 as
select * from customers;
```

Q9) Create table from customers with cid,cname,phone columns and data
Ans:

```
create table mycust2 as
select cid,cname,phone from customers;
```

Q10) Create table from customers with all columns and without data
Ans:

```
create table mycust3 as
select * from customers where 1=2;
```

Q11) Create table from customers with cid,cname,phone columns and without data
Ans:

```
create table mycust4 as
select cid,cname,phone from customers where 1=2;
```

Q12) Create table from address with all columns and having city as Blore.
Ans:

```
create table myadd1 as
select * from address where city='Blore';
```

Q13) Create table from customers,accounts,address with required columns and data
Ans:

```
create table hellocustomers as
select cname,email,accno,bal,street,city
from customers cust
inner join accounts acc on cust.cid=acc.mycid
inner join address addr on cust.cid = addr.mycid;
```

15.Generating Values for Primary Keys

- MySQL supports Auto-increment feature.
- Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.
- You can specify the **auto-increment** for generating the values for Primary Key Columns.
- Mostly, Primary Key Column type can be Integer type or Char type
- If Primary Key Column type is char type then Your Application logic has to generate the Unique value for Primary Key
- If Primary Key Column type is Integer type then You can use MySQL **auto-increment** to generate the Unique value for Primary Key

Practice Lab - 12

Step 1: Create the following 2 tables

```
create table students1(
sid int(3) Primary Key,
sname char(15) NOT NULL,
phone int(10) UNIQUE NOT NULL
);

create table students2(
sid int(3) Primary Key AUTO_INCREMENT,
sname char(15) NOT NULL,
phone int(10) UNIQUE NOT NULL
);
```

2) Insert the following Records

```
insert into students1 values(101,'sri',123); //OK

insert into students1 values('sri',123); // Syntax Error

insert into students1(sname,phone) values('sri',123); //Voilating P.K Contraint

insert into students2(sname,phone) values('sri',123); //OK

insert into students2(sname,phone) values('vas',321); //OK

insert into students2(sname,phone) values('sd',555); //OK
```

Practice Lab - 13

Step 1: Create the following table

```
create table students(
sid int(3) Primary Key AUTO_INCREMENT,
sname char(15) NOT NULL,
phone int(10) UNIQUE NOT NULL
)  AUTO_INCREMENT=101   ;
```

2) Insert the following Records

```
insert into students(sname,phone) values('sri',123);
insert into students(sname,phone) values('vas',321);
insert into students(sname,phone) values('sd',555);
```

16. Indexes

- Indexes are used to read the records with specific column values quickly.

- **Without an index,**
    - ✓ SQL Engine must begin with the first record and then read through the entire table to find the relevant records .
    - ✓ When table is very large, then operation takes lots of time to retrive the records from table.

- **Without an index on column,**
    - ✓ SQL Engine can quickly determine the position to seek to in the middle of the data file without having to look at all the data.
    - ✓ This is much faster than reading every row sequentially.

Syntax:

```
create index <index_name> on  <table_name>(col_name);
```

Ex:

create index **myindex** on **mycustomers**(**city**);

- Index will be created for Primary Key column automatically.
- You need to create the indexes for other columns if required

Practice Lab - 14

Step 1: Create the following table

```
create table students(
sid int(3) Primary Key AUTO_INCREMENT,
sname char(15) NOT NULL,
phone int(10) UNIQUE NOT NULL
)  AUTO_INCREMENT=101   ;
```

2) Insert the following Records

```
insert into students(sname,phone) values('sri',123);
insert into students(sname,phone) values('vas',321);
insert into students(sname,phone) values('ds',999);
insert into students(sname,phone) values('aa',111);
insert into students(sname,phone) values('bb',555);
insert into students(sname,phone) values('cc',333);
insert into students(sname,phone) values('dd',222);
insert into students(sname,phone) values('ee',444);
insert into students(sname,phone) values('ff',777);
insert into students(sname,phone) values('gg',888);
```

3) Select the Records as follows

```
select * from students where phone=777;
select * from students where sname='dd';
```

- SQL Engine search the student sequencally because No indexes created as of now for phone and sname columns.

4) Create the Index as follows

```
create index phone_index on students(phone);
create index sname_index on students(sname);
```

5) Select the Records as follows

```
select * from students where phone=777;
select * from students where sname='dd';
```

- SQL Engine search the student very fastly because indexes created now for phone and sname columns.

17. VIEWS

- Views are database objects like Tables.
- Views are also called as logical tables.
- View contains only the schema part it does not contain the data physically.
- View is mainly used to provide the restrictions on columns of a table to particular
- Views will be creatd based on one or more existing tables.

Syntax:

```
create view <view_name> as select statement;
```

Practice Lab - 15

Step 1: Create the following table

```
create table customers(
cid int(3) Primary Key,
cname char(15) NOT NULL,
email char(15) UNIQUE NOT NULL,
phone int(10) UNIQUE NOT NULL,
city char(15) NOT NULL,
status char(15) NOT NULL,
accno int(6) UNIQUE NOT NULL,
atype char(2) NOT NULL,
branch char(10) NOT NULL,
bal double
);
```

2) Insert the following Records

```
insert into customers
values(101,'sri','sri@myjlc',123,'Blore','Active',5001,'SA','B01',25000);

insert into customers
values(102,'vas','vas@myjlc',321,'Blore','Active',5002,'SA','B01',25000);
```

```
insert into customers
values(103,'sd','sd@myjlc',234,'Hyd','Active',5003,'SA','B01',25000);

insert into customers
values(104,'ds','ds@myjlc',345,'Hyd','Active',5004,'SA','B01',25000);

insert into customers
values(105,'aa','aa@myjlc',111,'Delhi','Active',5005,'SA','B01',25000);

insert into customers
values(106,'bb','bb@myjlc',222,'Delhi','Active',5006,'SA','B01',25000);

insert into customers
values(107,'cc','cc@myjlc',333,'Blore','Active',5007,'SA','B01',25000);

insert into customers
values(108,'dd','dd@myjlc',444,'Blore','Active',5008,'SA','B01',25000);

insert into customers
values(109,'ee','ee@myjlc',555,'Hyd','Active',5009,'SA','B01',25000);

insert into customers
values(110,'ff','ff@myjlc',666,'Hyd','Active',5010,'SA','B01',25000);

insert into customers
values(111,'gg','gg@myjlc',777,'Delhi','Active',5011,'SA','B01',25000);

insert into customers
values(112,'hh','hh@myjlc',888,'Delhi','Active',5012,'SA','B01',25000);
```

3) See the Records

```
select * from customers;
```

4) Create the View for bank teller and check it

```
create view teller_view as
select cid,cname,city,accno,atype,bal from customers;

select * from teller_view; //OK

select email from teller_view; //Error
```

5) Create the View for bank teller city-wise and check it

```
create view blore_teller_view as
select cid,cname,city,accno,atype,bal from customers where city='Blore';

select * from blore_teller_view;

create view hyd_teller_view as
select cid,cname,city,accno,atype,bal from customers where city='Hyd';

select * from hyd_teller_view;

create view delhi_teller_view as
select cid,cname,city,accno,atype,bal from customers where city='Delhi';

select * from delhi_teller_view;
```

6) Create the View for support team and check it

```
create view support_view as
select cid,cname,email,phone,city,status from customers;

select * from support_view; //OK

select bal from support_view; //Error
```

Types of Views

- Depending on the updation of data with view you can divide the view into two types.
  - Static Views
  - Dynamic Views

Static Views

- Static View is also called as read only view.
- If view is Static View then you can only read the data from view, you cannot perform any insert, delete or update operation on view.
- When view creation statement meets any of the following terms then it becomes Static view.
  - Select Statement is having order by clause
  - Select Statement is having group by clause
  - Select Statement is not having Primary Key.

Dynamic Views

- Dynamic View is also called as updatable view.
- If view is Dynamic View then you can read the data from view as well as you can perform insert, delete or update operation on view also.
- When view creation statement meets any of the following terms then it becomes Static view.
  - Select Statement is not having order by clause
  - Select Statement is not having group by clause
  - Select Statement is having Primary Key.

7) Do the Updatable Operations on myview as follows

a) Create the myview as follows

```
create view myview as
select cid,cname,email,phone,city,status from customers;
```

b) See the Data in table and myview

```
select * from myview;
select * from customers;
```

c) Update the data in myview

```
update myview set email='sd@jlc.com', phone=999 where cid=103;
```

d) See the Data in table and myview

```
select * from myview;
select * from customers;
```

e) Update the data in myview

```
delete from myview where cid>=110;
```

f) See the Data in table and myview

```
select * from myview;
select * from customers;
```

18. Transaction management

- Transaction is set of Database Operations Performing as Atomic Unit.
- When all the Operations in Transaction are success then Transaction is Success and it has to commited.
- When anyone Operation in Transaction is failed then Transaction is failed and it has to rolled back.
- All DB vendors have to provide the facility of transaction management.

Ex:

```
Begin Tx
    Op1: Insert
    Op2: Update
    Op3: Update
    Op4: Insert
    Op5: Update
Commit Tx/Rollback Tx
```

- To manage the transaction you need to use following keywords:
  - autocommit
  - commit
  - rollback
  - savepoint

- **commit:**
  - The Query will be committed or saved and can't be rolled back.

- **rollback:**
  - Uncommitted or unsaved query will be rolled back. (Cancelled)

- **savepoint:**
  - Marking from where the uncommitted or unsaved query will be rolled back.

- **Autocommit:**
  - Auto Commit is enabled by default in MySQL.
  - MySQL engine issues the commit automatically at the end of every DB Operation.
  - When you are implementing Txs you have to disable Auto Commit

a) See the current status of autocommit

```
select @@autocommit;
```

b) b) diable autocommit

```
set autocommit=0;
```

c) c) See the current status of autocommit

```
select @@autocommit;
```

Practice Lab - 16

Step 1: Create the following table

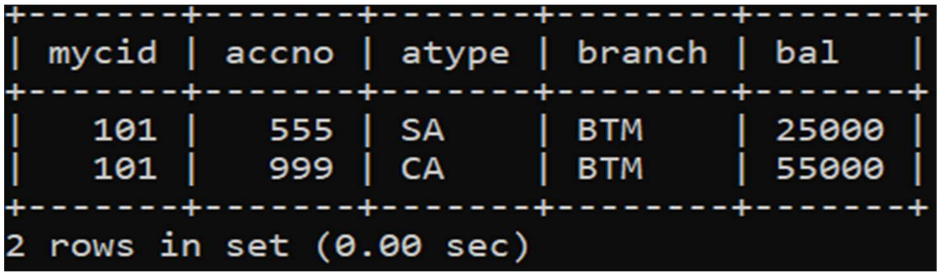
```
create table myaccounts(
mycid int ,
accno int primary key,
atype char(2),
branch char(10),
bal double
);
```

2) Insert the following Records

```
insert into myaccounts values(101,555,'SA','BTM',25000);
insert into myaccounts values(101,999,'CA','BTM',55000);
```

3) See the Records

select * from myaccounts;



```
+-------+-------+-------+--------+-------+
| mycid | accno | atype | branch | bal   |
+-------+-------+-------+--------+-------+
|   101 |   555 | SA    | BTM    | 25000 |
|   101 |   999 | CA    | BTM    | 55000 |
+-------+-------+-------+--------+-------+
2 rows in set (0.00 sec)
```

4) Do the Following Tasks

select * from myaccounts;

update myaccounts set bal=45000 where accno=555;

select * from myaccounts;

rollback;

select * from myaccounts;

Note : It is Not Rolling back right?

5) Do the Following Tasks

insert into myaccounts values(102,666,'SA','BTM',5000);

update myaccounts set bal=5500 where accno=666;

select * from myaccounts;

rollback;

select * from myaccounts;

Note : It is Not Rolling back right?

6) Disable Auto Commit

```
set autocommit=0;
```

4) Do the Following Tasks

```
select * from myaccounts;
update myaccounts set bal=45000 where accno=555;
select * from myaccounts;
rollback;
select * from myaccounts;
```

Note : It is Rolling back right?

5) Do the Following Tasks

```
insert into myaccounts values(105,777,'SA','BTM',5000);
update myaccounts set bal=5500 where accno=666;
select * from myaccounts;
rollback;
select * from myaccounts;
```

Note : It is Rolling back right?

Exploring SavePoint

This is to specify the begining of Tx.

```
set autocommit=0;
SQL Stmt1
SQL Stmt2
SQL Stmt3
commit;
SQL Stmt4
SQL Stmt5
SQL Stmt 6
savepoint jlc1
SQL Stmt 7
SQL Stmt 8
SQL Stmt 9
savepoint jlc2
SQL Stmt 10
SQL Stmt 11

rollback jlc2;  //10,11
rollback jlc1;  //7,8,9,10,11
rollback;  //4,5,6,7,8,9,10,11
```

19. SQL Built-in Functions

1) Arithmetic Functions
2) String Functions
3) Date Functions
4) Conversion Functions
5) Aggregate Functions

1) <u>Arithmetic Functions</u>

sqrt()
power()/pow()
mod()
abs()
ceil()
floor()
round()

<u>Ex:</u>

```
select sqrt(25);
select power(25,2);
select power(25,.5);
select mod(10,2);
select mod(10,3);
select abs(9);
select abs(-9);

select ceil(5);
select ceil(5.2);
select ceil(5.9);

select floor(5);
select floor(5.2);
select floor(5.9);
select round(5);
select round(5.2);
select round(5.9);
```

2) <u>String Functions</u>

length()
trim()
rtrim()
ltrim()
rpad()
lpad()
substr()/substring()
lower()
lcase()
upper()
ucase()
ascii()
char()
concat()
replace()

<u>Ex:</u>

```
select length('Srinivas');
select length('  Sri  nivas   ');
select trim('  Sri  nivas   ');
select length(trim('  Sri  nivas   '));
select length(ltrim('  Sri  nivas   '));
select length(rtrim('  Sri  nivas   '));

select rpad('JLC',5,'*');
select rpad('JLC',6,'*');
select rpad('JLC',3,'*');
select rpad('JLC',2,'*');
select lpad('JLC',5,'*');
select lpad('JLC',6,'*');
select lpad('JLC',3,'*');
select lpad('JLC',2,'*');

select substr('Srinivas',4);
select substring('Srinivas',3);

select substr('Srinivas',3,5);
select substr('Srinivas',3,4);
select substr('Srinivas',3,3);

select lower('SriNiVas');
select lcase('SriNiVas');
select upper('SriNiVas');
select ucase('SriNiVas');

select ascii('A');
select ascii('a');

select char(97);
select char(65);


select concat('hello','guys');
select length(concat('hello','guys'));
select substr(concat('hello','guys'),3,5);
select length(substr(concat('hello','guys'),3,5));

select replace('helloguys','hello','JLC');
select replace('helloguys','hello','JLC ');
```

Date functions

MySQL - YYYY-MM-DD => 2024-02-20

```
sysdate()
now()
date()
time()
day()
month()
year()
```

Ex:

```
select sysdate();
select now();

select date(sysdate());
select date(now());

select time(sysdate());
select time(now());

select day(sysdate());
select day(now());

select month(sysdate());
select month(now());

select year(sysdate());
select year(now());
```

FORMAT SPECIFIERS(MYSQL)

%a	:	Thu
%b	:	Sep
%c	:	Month in digit
%d	:	Date (00 - 31)
%D	:	Date with suffix st,th,nd,rd
%e	:	Date (1 - 31)
%r	:	Time in 12 Hours
%T	:	Time in 24 Hour
%W	:	Thursday
%Y	:	YYYY
%y	:	YY
%M	:	September
%m	:	Month in two digit

Date Formats

a) select date_format(sysdate(),'%D %M %Y');

b) select date_format(sysdate(),'%d-%m-%y');
c) select date_format(sysdate(),'%d-%m-%Y');

d) select date_format(sysdate(),'%W %d-%m-%Y');
e) select date_format(sysdate(),'%a %d-%m-%Y');

f) select date_format(sysdate(),'%d-%b-%Y');

g) select time_format(sysdate(),'%r');

g) select time_format(sysdate(),'%T');

4) Conversion Functions

- **str_to_date()**

```
select str_to_date('August 10 2024','%D-%M-%Y');

create table hello(
id int,
hello date
);

insert into hello values(101, str_to_date('August 10 2024','%D-%M-%Y') );
```

20. Alter statements

Practice Lab - 17

Step 1: Create the following table

```
create table mycustomers (
    cid int(3) primary key,
    cname char(15),
    email char(15),
    phone long,
    city char(15)
);
```

2) Insert the following Records

```
insert into mycustomers values(101,'sri','sri@jlc',999,'Blore');
insert into mycustomers values(102,'a','a@jlc',111,'Blore');
insert into mycustomers values(103,'b','b@jlc',222,'Blore');
insert into mycustomers values(104,'c','c@jlc',333,'Blore');
insert into mycustomers values(105,'d','d@jlc',444,'Blore');
insert into mycustomers values(106,'e','e@jlc',555,'Blore');
```

3) See the Records

```
select * from mycustomers;
```

Do the following Alter Table Commands

1) Adding New Column

Syntax:

```
ALTER TABLE table_name
ADD column_name data_type(size)
AFTER column_name;
```

Ex:

```
ALTER TABLE mycustomers
ADD status char(15)
AFTER city;

ALTER TABLE mycustomers
ADD gender char(6)
AFTER phone;
```

2) Droping New Column

Syntax:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Ex:

```
ALTER TABLE mycustomers
DROP COLUMN gender;

ALTER TABLE mycustomers
DROP COLUMN status;
```

3) Modify the Column

Syntax:

```
ALTER TABLE table_name
MODIFY column_name data-type(size);
```

Ex:

```
ALTER TABLE mycustomers
MODIFY city char(25);

ALTER TABLE mycustomers
MODIFY city char(10);

ALTER TABLE mycustomers
MODIFY cid char(5);
```

4) Adding the Primary Key

Syntax:

```
ALTER TABLE table_name
ADD Primary Key(column1,column2);
```

Ex:

```
ALTER TABLE mycustomers
ADD Primary Key(cid);
```

5) Drop the Primary Key

Syntax:

```
ALTER TABLE table_name
DROP Primary Key;
```

Ex:

```
ALTER TABLE mycustomers
DROP Primary Key;
```

6) Adding Constraints***

Syntax:

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name <yours constraints here>;
```

Ex:

```
ALTER TABLE mycustomers
ADD CONSTRAINT ck1 status char(15) NOT NULL;
```

7) Drop the Constraints***

Syntax:

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```

Ex:

```
ALTER TABLE mycustomers
DROP CONSTRAINT ck1 ;
```

8) Renaming Column

Syntax:

```
ALTER TABLE table_name
RENAME TO new-table-name;
```

Ex:

```
ALTER TABLE hellocustomers
RENAME TO haicustomers;
```

21. Droping objects

Database Objects
1) tables
2) Views
3) Indexes

1) <u>Dropping Table</u>

<u>Syntax:</u>

```
drop table table_name;
```

<u>Ex:</u>

```
drop table haicustomers;
```

2) <u>Dropping View</u>

<u>Syntax:</u>

```
drop view view_name;
```

<u>Ex:</u>

```
drop view support_view;
```

3) <u>Dropping Index</u>

<u>Syntax:</u>

```
drop index Index_name;
```

<u>Ex:</u>

```
drop index phone_index;
```

22. Truncate the table

a) **delete from student;**
b) **truncate table student;**

- Above two queries will give the empty student table as a result.
- In the case of delete command, it will delete the records one by one.
- In the case of truncate command, it will drop the table and it will create the same table again.
- Delete operation can be Rolled back.
- Truncate operation cannot be Rolled back.

23. Special queries

Q1) diplay the records from mth row to nth now

```
select * from mycustomers limit 0,5;
select * from mycustomers limit 5,5;
select * from mycustomers limit 10,5;
```

Q) Display the Nth Row

```
select * from mycustomers limit 6,1;
```

Q) Display the Top N Records

```
select * from mycustomers limit 5;
```

24. Stored Procedures

- Stored Procedure is a PL/SQL Block with name.
- When you create the Stored Procedure, It will be compiled and stored in the DB memory.
- When you call the Stored Procedure , it will be executed without any compilation.

You can prefer the S.P's in the following cases.

- when you want to run the business logic inside the DB server
- when you want to reduce the number of round trips between application and DB server to improve the performance.

Example 1:

```
delimiter ##
create procedure mydisplay1()
begin
declare a int;
declare b int;
declare total int;
set a=10;
set b=20;
set total = a+b;
select total as "Sum";
end;
##
delimiter   ;
```

Call the Procedure as follows

```
call mydisplay1();
```

Example 2:

```
delimiter ##
create procedure findSum(a int,b int)
begin
declare total int;
set total = a+b;
select total as "Sum";
end;
##
delimiter  ;
```

Call the Procedure as follows

```
call findSum(10,20);
call findSum(100,200);
```

Types of Parameters

- Stored Procedure takes 3 Types of Parameters
    - a) IN Patameters
    - b) OUT Patameters
    - c) INOUT Patameters

a) IN Parameters

- IN Parameters takes the Values from Caller to the Procedure.
- By Default, Parameters are IN only.
- You can have any number of IN Parameters

Example 3:

```
delimiter ##
create procedure p1(IN a int,IN b int, c int)
begin
declare total int;
set total = a+b+c;
select total as "Sum";
end;
##
delimiter  ;
```

Call the Procedure as follows

```
call p1(10,20,30);
```

b) OUT Parameters

- OUT Parameters takes the Values from Procedure to the Caller
- You have to Specify the Parameter as OUT explicitly.
- You can have any number of OUT Parameters

Example 4:

```
delimiter ##
create procedure p2(IN a int,IN b int,OUT total int, OUT diff int)
begin
set total = a+b;
set diff  = a-b;
end;
##
delimiter  ;
```

Call the Procedure as follows

```
select @mytotal;
select @mydiff;
call p22(50,20,@mytotal,@mydiff);
select @mytotal;
select @mydiff;
```

c) INOUT Parameters

- INOUT Parameters acts as Both IN Parameter and OUT Parameter
- You have to Specify the Parameter as INOUT explicitly.
- You can have any number of INOUT Parameters

Example 5:

```
delimiter ##
create procedure p3(IN a int,INOUT b int)
begin
set b = a+b;
end;
##
delimiter  ;
```

Call the Procedure as follows

```
select @b;
set @b=20;
select @b;
call p3(10,@b);
select @b;
```

call p3(10,20); // Not OK

Conditional Statement

A) If Statement

Syntax:	Syntax:	Syntax:

if(condition) then
....
....
end if
	if(condition) then
	....
	....
	else
	....
	....
	end if;
		if(condition) then
		....
		....
		elseif (condition) then
		....
		....
		else
		--- 	
		---- 	
		end if;

Example 6:

```
delimiter ##
create procedure find2Max(IN a int,IN b int)
begin
declare c int;

if(a>b)then
set c = a;
else
set c = b;
end if;

select c as "Max";
end;
##
delimiter  ;
```

Call the Procedure as follows

```
call find2Max(10,20);
```

Example 7:

```
delimiter ##
create procedure find3Max(IN a int,IN b int,IN c int)
begin

declare d int;

if(a>b and a>c) then
set d = a;
elseif(b>c) then
set d = b;
else
set d = c;
end if;

select d as "Max";
end;
##
delimiter  ;
```

Call the Procedure as follows

```
call find3Max(10,20,30);
```

<u>While Statement:</u>

Syntax:

```
while(Condition) do
        St1
        St2
        ...
end while;
```

<u>Example 8: Print the Numbers from 1 to given number</u>

```
delimiter $$
create procedure printNums(num int)
begin
declare i int;
set i=1;

while(i<=num)do
select i;
set i= i+1;
end while;

end;

$$
delimiter  ;
```

<u>Call the Procedure as follows</u>

```
call printNums(6);
```

Example 9 : Find the Sum of the Numbers from 1 to given number

```
delimiter ##
create procedure findTotal(num int)
begin
declare i int;
declare total int;

set i=1;
set total=0;

while(i<=num)do
set total = total + i;
set i= i+1;
end while;

select total as "Sum";
end;
##
delimiter  ;
```

Call the Procedure as follows

```
call findTotal(10);
```

Example 10 : Find the Sum of Even Numbers from 1 to given number

```
delimiter ##
create procedure findEvenTotal(num int)
begin
declare i int;
declare total int;

set i=1;
set total=0;

while(i<=num)do

if(i mod 2=0)then
set total = total + i;
end if;

set i= i+1;
end while;

select total as "Sum";
end;
##
delimiter  ;
```

Call the Procedure as follows

```
call findEvenTotal(10);
```

REPEAT - UNTIL Statement

Syntax:

```
Repeat
        St1
        St2
        ...
Until Condition
end Repeat;
```

Example 11 : Print the Numbers from 1 to given number

```
delimiter $$
create procedure displayNums1(num int)
begin
declare i int;
set i=1;

Repeat
select i;
set i= i+1;
Until i>5
end Repeat;

end;
$$
delimiter  ;
```

Call the Procedure as follows

```
call displayNums1(6);
```

Example 12 : Find the Sum of the Numbers from 1 to given number

```
delimiter ##
create procedure findTotal3(num int)
begin
declare i int;
declare total int;

set i=1;
set total=0;

Repeat
set total = total + i;
set i= i+1;
Until i>num
end Repeat;

select total as "Sum";
end;
##
delimiter  ;
```

Call the Procedure as follows

```
call findTotal3(5);
```

Example 13 : Find the Sum of Even Numbers from 1 to given number

```
delimiter ##
create procedure findEvenTotal3(num int)
begin
declare i int;
declare total int;

set i=1;
set total=0;

Repeat
set total = total + i;
set i= i+1;
Until i>num
end Repeat;

select total as "Sum";
end;
##
delimiter  ;
```

Call the Procedure as follows

```
call findEvenTotal(10);
```

1) Setup the Database

```
create table jlcstudents(
sid int primary key,
sname char(10) NOT NULL,
email char(10) NOT NULL,
totalfee double DEFAULT 20000,
feepaid double  NOT NULL,
m1 int,
m2 int,
m3 int,
total int,
average double,
grade char(5)
);

select * from jlcstudents;

insert into jlcstudents(sid,sname,email,feepaid,m1,m2,m3)
values(101,'Sri','Sri@jlc',5000,60,40,90);
insert into jlcstudents(sid,sname,email,feepaid,m1,m2,m3)
values(102,'sd','sd@jlc',15000,80,80,80);
insert into jlcstudents(sid,sname,email,feepaid,m1,m2,m3)
values(103,'ds','ds@jlc',10000,20,30,40);
insert into jlcstudents(sid,sname,email,feepaid,m1,m2,m3)
values(104,'vas','vas@jlc',2000,95,96,97);

select * from jlcstudents;
```

Example 14: Write the SP to get the Fee Balance of given Student.

```
delimiter ##
create procedure getBalance(IN mysid int, OUT mybal double)
begin

declare mytotalfee  double;
declare myfeepaid double;

select totalfee, feepaid into mytotalfee,myfeepaid from jlcstudents where sid=mysid;

set mybal = mytotalfee - myfeepaid;
end;
##
delimiter  ;
```

Call the Procedure as follows

```
select @mybal;
call getBalance(101,@mybal);
select @mybal;
call getBalance(102,@mybal);
select @mybal;
```

Example 15: Write the SP to calculate and Update the Results of given student

```
delimiter ##
create procedure findGrade(IN mysid int)
begin

declare mm1 int;
declare mm2 int;
declare mm3 int;
declare mytotal int;
declare myaverage double;
declare mygrade char;

select m1,m2,m3 into mm1,mm2,mm3 from jlcstudents where sid=mysid;
set mytotal = mm1+ mm2+mm3;
set myaverage = mytotal/3;
if(myaverage >= 90) then
set mygrade ='A';
elseif(myaverage >= 80) then
set mygrade ='B';
else
set mygrade ='C';
end if;

set mygrade= trim(mygrade);
update jlcstudents set total=mytotal, average=myaverage, grade=mygrade where sid=mysid;

end;
##
delimiter  ;
```

Call the Procedure as follows

```
call findGrade(101);
```

25.Triggers

- Trigger is a PL/SQL Block with name.
- Trigger will be invoked by DBMS automatically when ever you update or delete or insert the records into the table.
- No Need to call trigger explicitly like Stored Procedure.

MySQL DB setup

```
create table mystudents(
sid int primary key,
sname char(25) NOT NULL,
email char(25) NOT NULL,
phone long
);

select * from mystudents;

insert into mystudents values(101,'Sri','Sri@jlc',111);
insert into mystudents values(102,'sd','sd@jlc',222);
insert into mystudents values(103,'ds','ds@jlc',333);
insert into mystudents values(104,'vas','vas@jlc',444);
insert into mystudents values(105,'hello','hello@jlc',555);

select * from mystudents;

create table mystudents_backup(
sid int,
sname char(25),
email char(25),
phone long,
action char(15),
actiondate date
);

select * from mystudents_backup;
```

Example #1

```
delimiter ##

create trigger t1
BEFORE UPDATE
ON mystudents
FOR EACH ROW
begin
declare mysid int;
declare mysname char(25);
declare myemail char(25);
declare myphone long;
declare myaction char(15);
declare myactiondate date;

set mysid = OLD.sid;
set mysname = OLD.sname;
set myemail = OLD.email;
set myphone = OLD.phone;

set myaction ='Update';
set myactiondate = sysdate();

insert into mystudents_backup
values(mysid,mysname,myemail,myphone,myaction,myactiondate);

end;
##
delimiter  ;
```

Do the Following Tasks

```
select * from mystudents;
select * from mystudents_backup;

update mystudents set email='sri@myjlc.com', phone=123456 where sid=101;

select * from mystudents;
select * from mystudents_backup;

update mystudents set email='aaa@myjlc.com', phone=1010101 where sid=101;

select * from mystudents;
select * from mystudents_backup;
```

Example #2

```
delimiter ##

create trigger t2
BEFORE DELETE
ON mystudents
FOR EACH ROW
begin
declare mysid int;
declare mysname char(25);
declare myemail char(25);
declare myphone long;
declare myaction char(15);
declare myactiondate date;

set mysid = OLD.sid;
set mysname = OLD.sname;
set myemail = OLD.email;
set myphone = OLD.phone;
set myaction ='Delete';
set myactiondate = sysdate();

insert into mystudents_backup
values(mysid,mysname,myemail,myphone,myaction,myactiondate);

end;

##
delimiter  ;
```

Do the Following Tasks

```
select * from mystudents;
select * from mystudents_backup;

delete from mystudents where sid=104;

select * from mystudents;
select * from mystudents_backup;

update mystudents set email='aaa@myjlc.com', phone=1010101 where sid=101;

select * from mystudents;
select * from mystudents_backup;
```