ଓଡ଼ିଶା ରାଜ୍ୟ ମୁକ୍ତ ବିଶ୍ୱବିଦ୍ୟାଳୟ, ସମ୍ବଲପୁର, ଓଡ଼ିଶା
Odisha State Open University, Sambalpur, Odisha
Established by an Act of Government of Odisha.

DIPLOMA IN COMPUTER APPLICATION

DCA-05 DATABASE SYSTEMS

BLOCK

4 LABORATORY MANUAL

ଓଡ଼ିଶା ରାଜ୍ୟ ମୁକ୍ତ ବିଶ୍ୱବିଦ୍ୟାଳୟ, ସମ୍ବଲପୁର, ଓଡ଼ିଶା
Odisha State Open University, Sambalpur, Odisha
Established by an Act of Government of Odisha.

EXPERT COMMITTEE

Dr. P.K Behera(Chairman)
Reader in Computer Science
Utkal University, Bhubaneswar, Odisha

Dr.J.R Mohanty(Member)
Professor and HOD
KIIT University. Bhubaneswar, Odisha

Sri Pabitrananda Pattnaik(Member)
Scientist-E, NIC
Bhubaneswar, Odisha

Sri Malaya Kumar Das (Member)
Scientist-E, NIC
Bhubaneswar, Odisha

Dr. Bhagirathi Nayak (Member)
Professor and Head (IT & System)
Sri Sri University
Bhubaneswar,Odisha

Dr.Manoranjan Pradhan(Member)
Professor and Head (IT & System)
G.I.T.A, Bhubaneswar, Odisha

Sri Chandrakant Mallick(Convener)
Consultant (Academic)
School of Computer and Information Science., Odisha State Open University
Sambalpur, Odisha

DIPLOMA IN COMPUTER APPLICATION

Course Writer

Chandrakant Mallick
Odisha State Open University, Sambalpur, Odisha

DCA-05 DATABASE SYSTEMS LABORATORY

LIST OF EXPERIMENTS

SL. No.	Experiment
1	Installing MySQL on Ms- Windows and Linux
2	Implementation of DDL commands of SQL with suitable examples. • Create table • Alter table • Drop Table
3	Implementation of DML commands of SQL with suitable examples. • Insert • Update • Delete
4	Implementation of different types of functions with suitable examples. • Number function • Aggregate Function • Character Function • Conversion Function • Date Function
5	Implementation of different types of operators in SQL. • Arithmetic Operators • Logical Operators • Comparison Operator • Special Operator • Set Operation
6	Implementation of different types of Joins • Simple Join • Self Join • Outer Join

EXPERIMENT-1

Aim: To install MySQL on Windows and Linux Operating System.

1.0 Learning Objective

At the end of the session you will be able to

- Install MySQL on windows operating system
- Install MySQL in Linux operating system

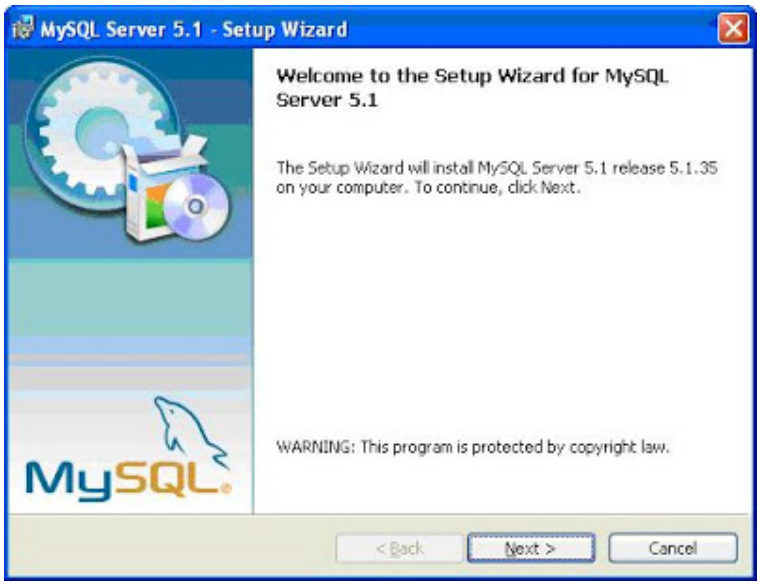
1.1 Introduction

MySQL is an open-source database management system, commonly installed as part of the popular LAMP (Linux, Apache, MySQL, PHP /Python/Perl) stack. It uses a relational database and SQL (Structured Query Language) to manage its data.

You can download the MySQL database from the MySQL website http://www.mysql.com by clicking on the **downloads** tab. Scroll down to the MySQL database server & standard clients section and select the latest production release of MySQL, 5.1.35 at the time of writing.

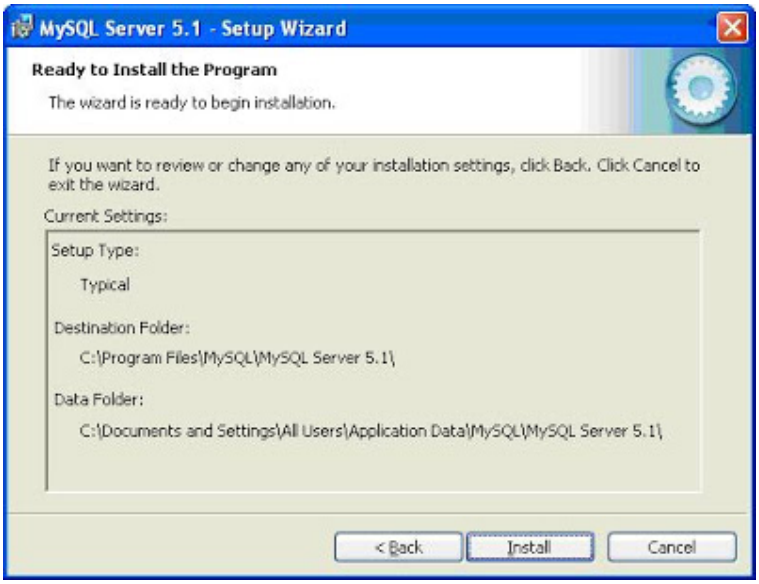
1.3 Installation of MySQL Server in Windows Operating System

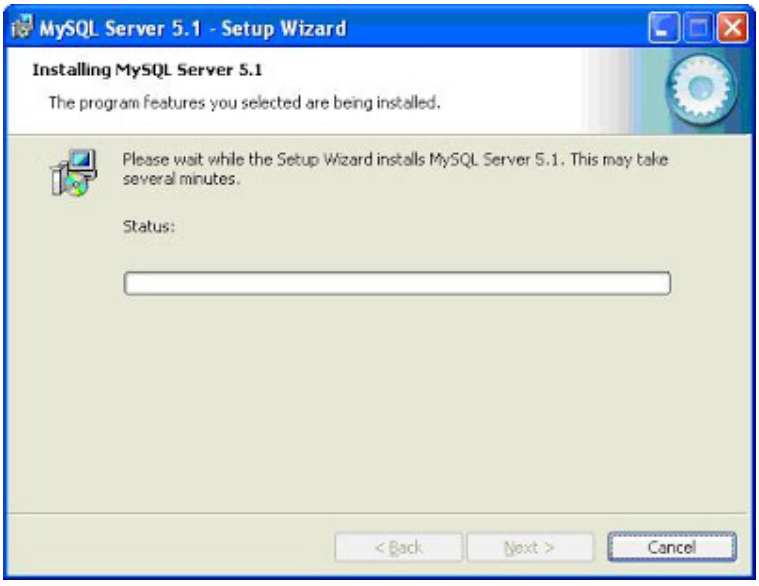
Unzip the setup file and execute the downloaded MSI file. Follow the instructions below exactly when installing MySQL Server:

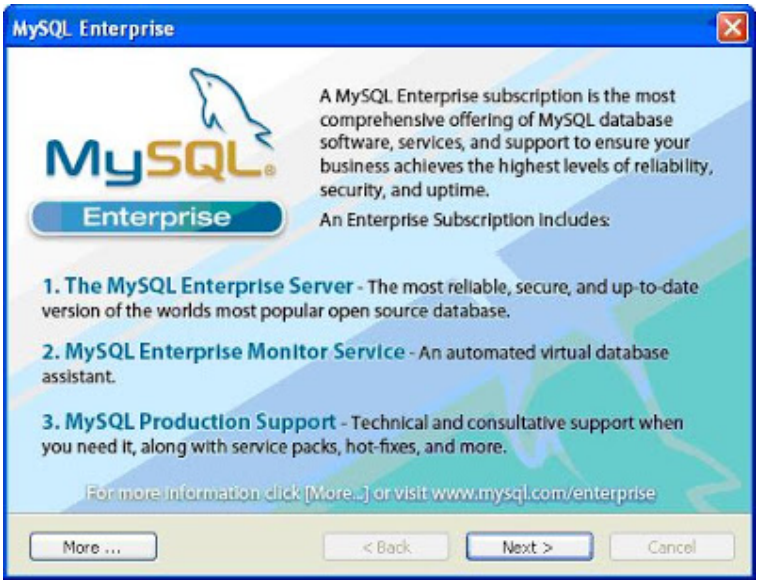


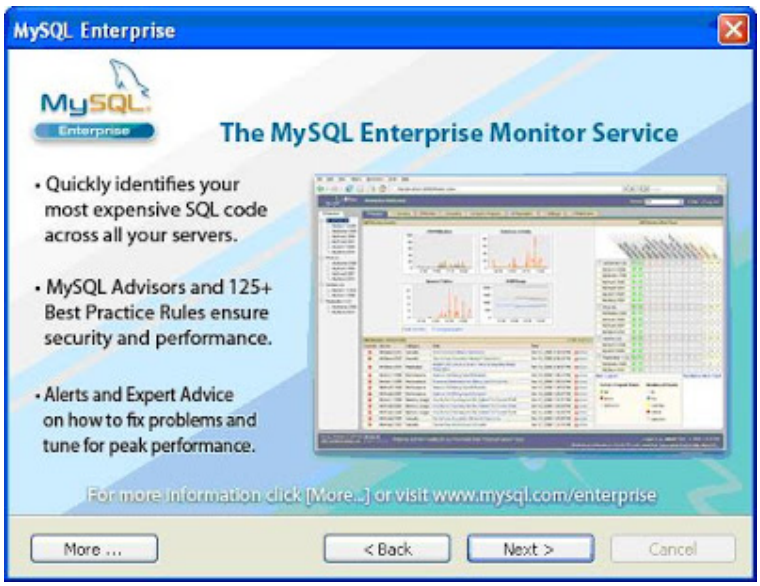
Click on the "setup"

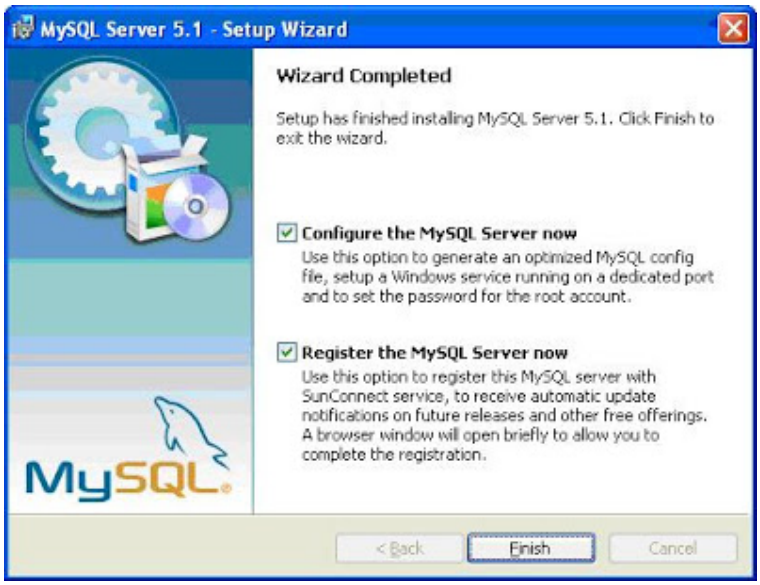
Perform a typical installation


MySQL Server 5.1 - 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 5.1\
Data Folder:
C:\Documents and Settings\All Users\Application Data\MySQL\MySQL Server 5.1\
< Back
Install
Cancel

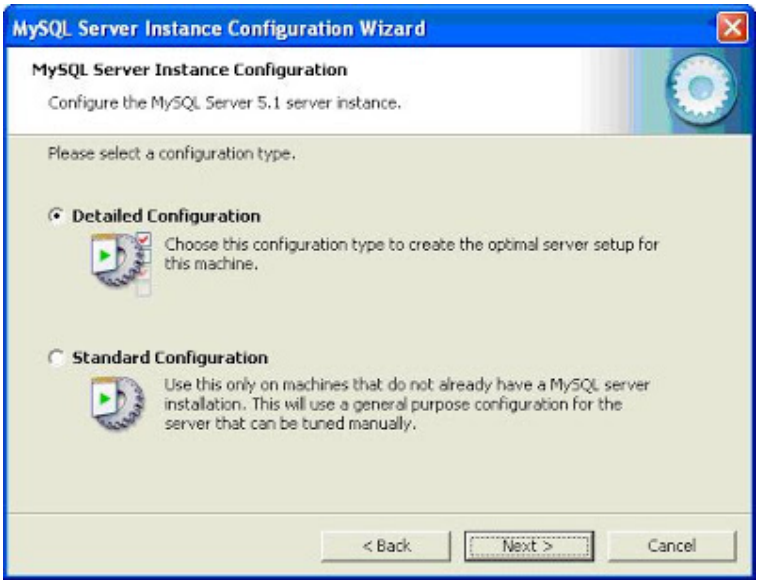

MySQL Server 5.1 - Setup Wizard
Installing MySQL Server 5.1
The program features you selected are being installed.
Please wait while the Setup Wizard installs MySQL Server 5.1. This may take several minutes.
Status:
< Back
Next >
Cancel


MySQL Enterprise
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 worlds 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
More ...
< Back
Next >
Cancel

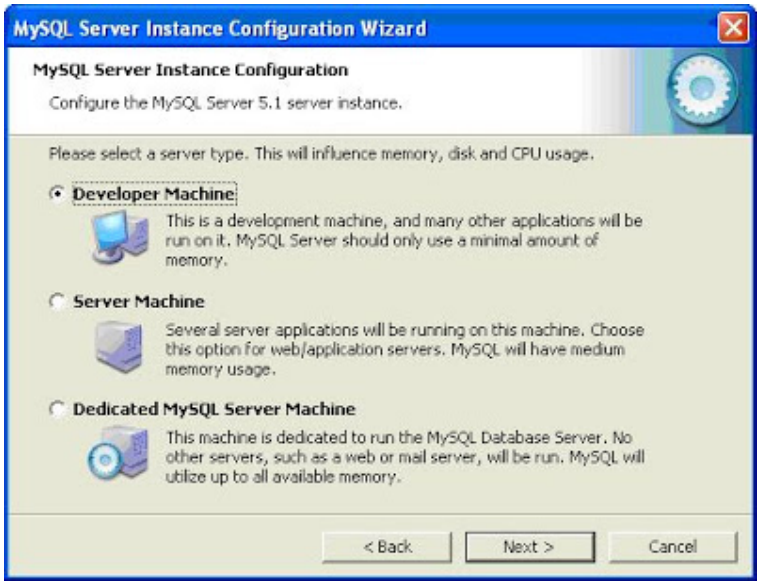

MySQL Enterprise
MySQL
Enterprise
The MySQL Enterprise Monitor Service
• Quickly identifies your most expensive SQL code across all your servers.
• MySQL Advisors and 125+ Best Practice Rules ensure security and performance.
• Alerts and Expert Advice on how to fix problems and tune for peak performance.
For more information click [More...] or visit www.mysql.com/enterprise
More ...
< Back
Next >
Cancel



- Check box to configure MySQL Server
- If you checked the Configure the MySQL Server now check box on the final dialog of the MySQL Server installation, then the MySQL Server Instance Configuration Wizard will automatically start.
- Follow the instructions below carefully to configure your MySQL Server to run correctly with EventSentry.

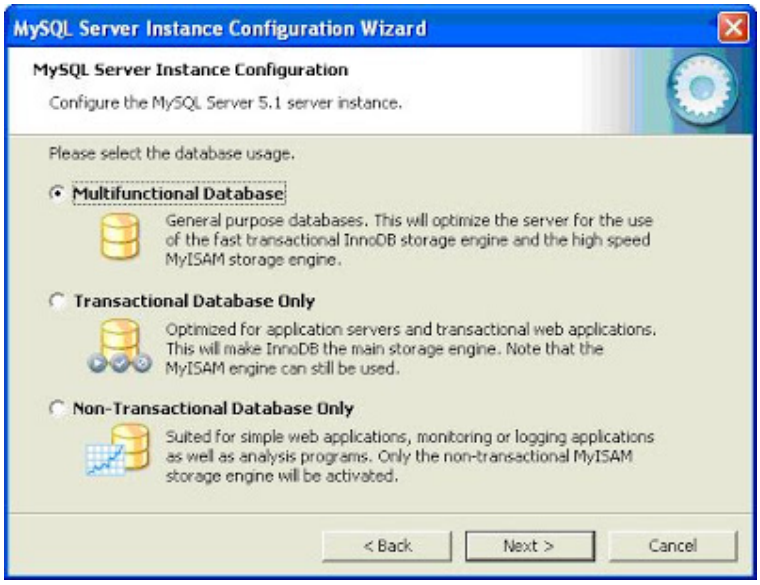


Select Detailed Configuration

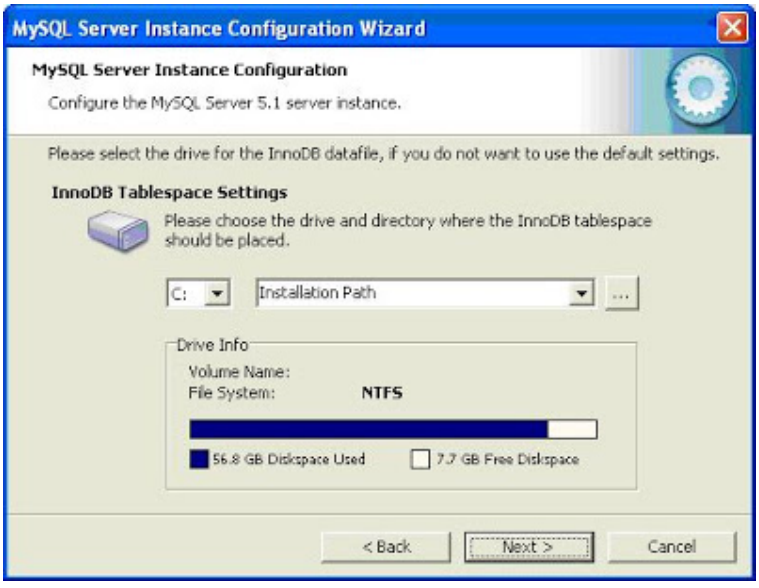


I was installing it on my local machine where other applications & tools are running I decided to opt "developer machine" but it is recommended that you use a Dedicated MySQL Server Machine for your MySQL database, if this is not an option then select "Server Machine".

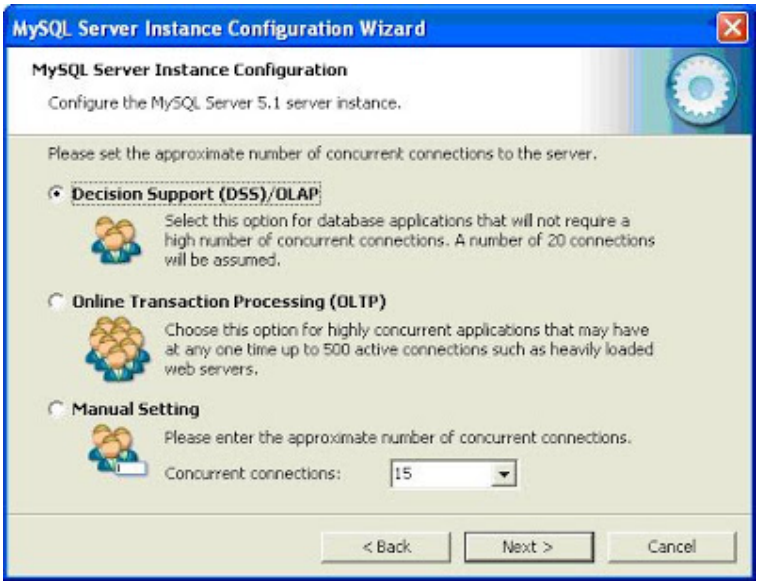
If you selected Dedicated MySQL Server Machine and your MySQL service does not start after the wizard completes, then try to re-run the wizard (or re-install) MySQL, but this time select the Server Machine option.

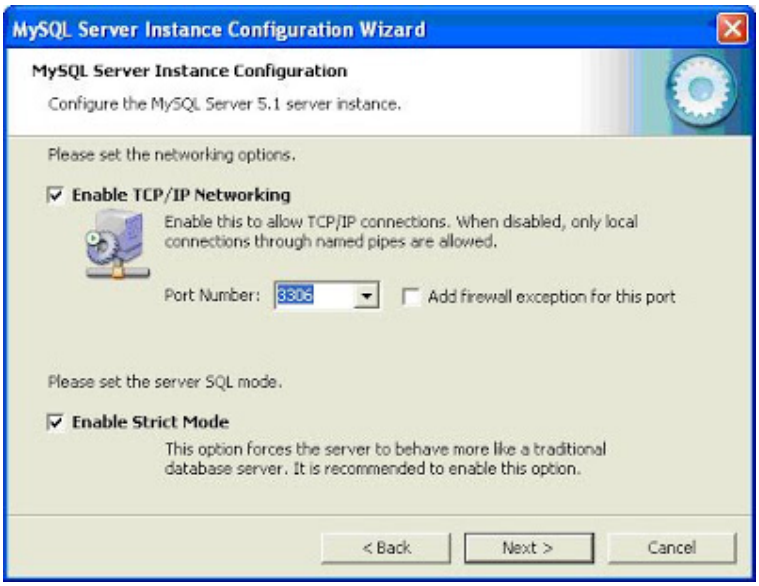


I have checked "Multifunctional databases" as I wanted MyISAM as default storage engine but if you want you can select "Transactional Database Only", this will make sure that InnoDB is the main storage engine. If you have checked 3rd option then only myISAM engine would be available

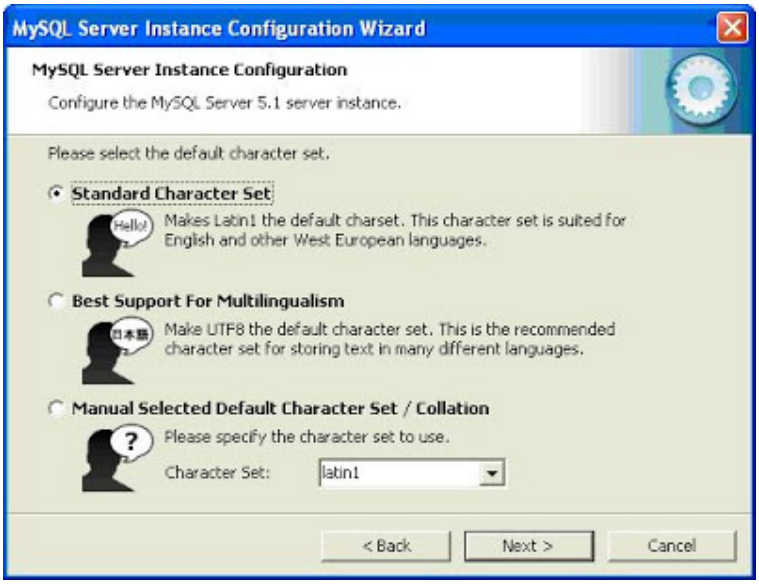


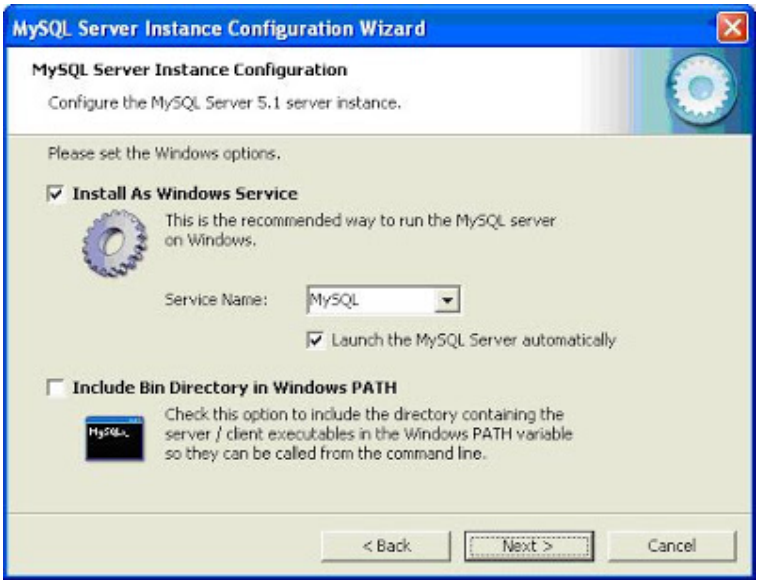
Select the drive where the database files will be stored. Select the drive on the fastest drive(s) on your server.





It is recommended that you leave the default port 3306 in place; however EventSentry will also work with non-standard ports if necessary.

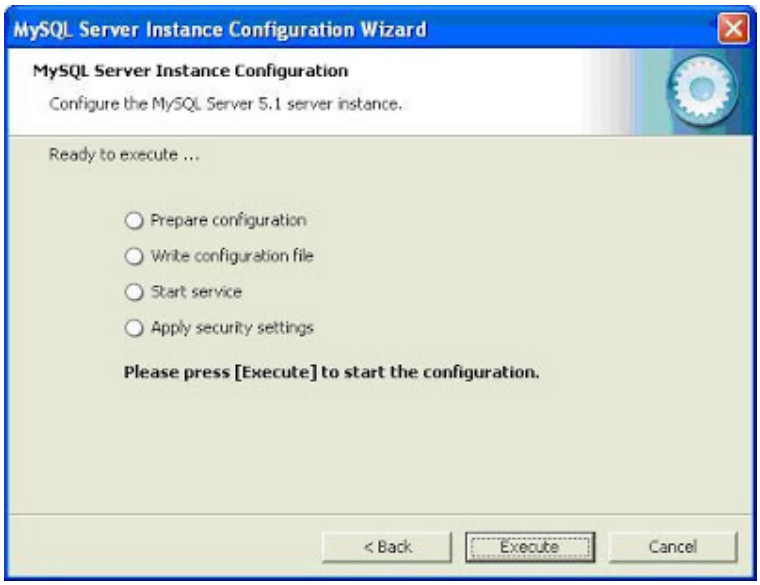


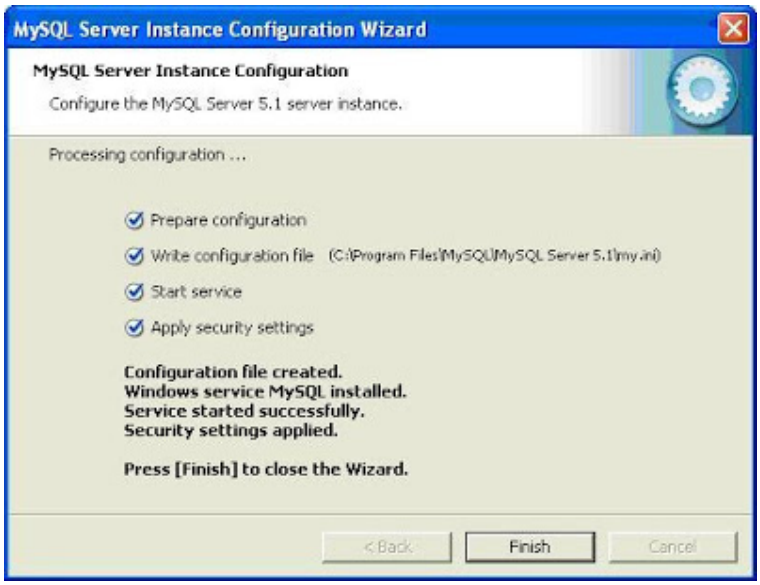


It is highly recommended that you run the MySQL Server as a Windows service (you can disable this if you want to start it manually whenever required) and include the binary directory in the search path.

Specify a secure root password; you may want to check the box Enable root access from remote machines if you plan on administering your MySQL server from your workstation or other servers.

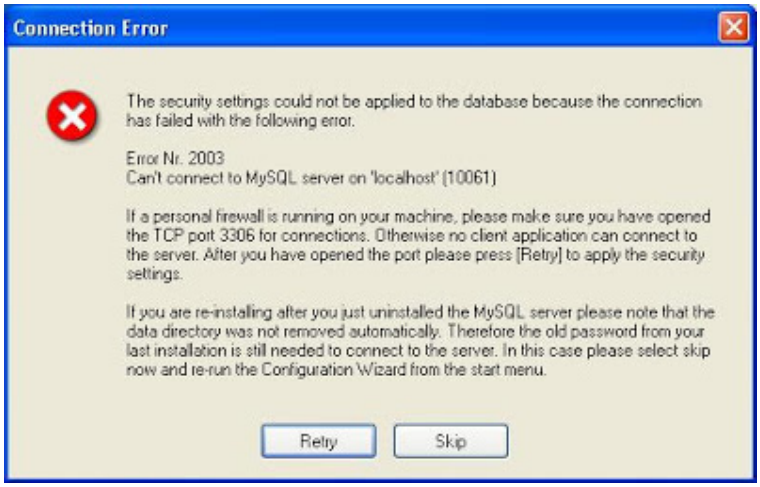
If you are getting an error message after clicking the Next button, then please enable port 3306 in the Windows XP Firewall Settings.





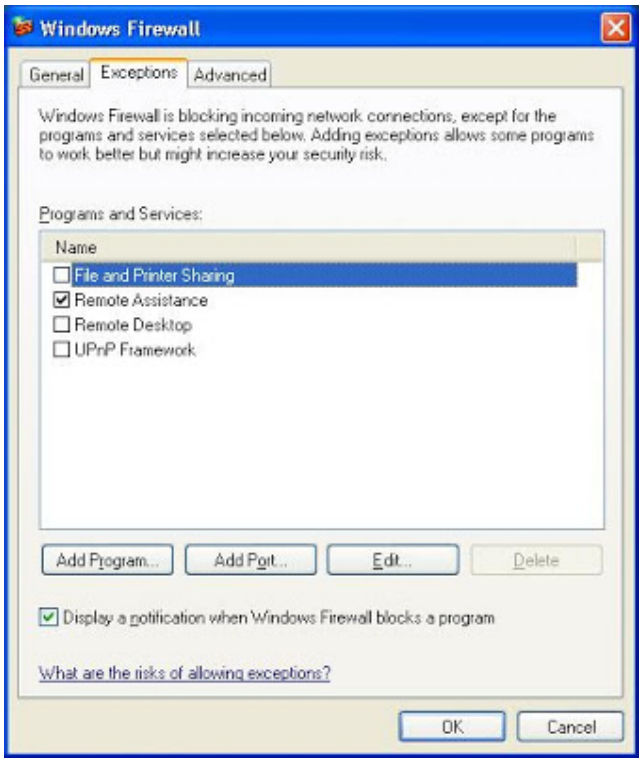
Done!!!

But if you are installing MySQL on a Windows XP workstation, or any other computer that has a firewall enabled, and the wizard fails with an error message similar to the one shown below (Can't connect to MySQL server on 'localhost'), then you will have to exclude the MySQL daemon from your firewall configuration

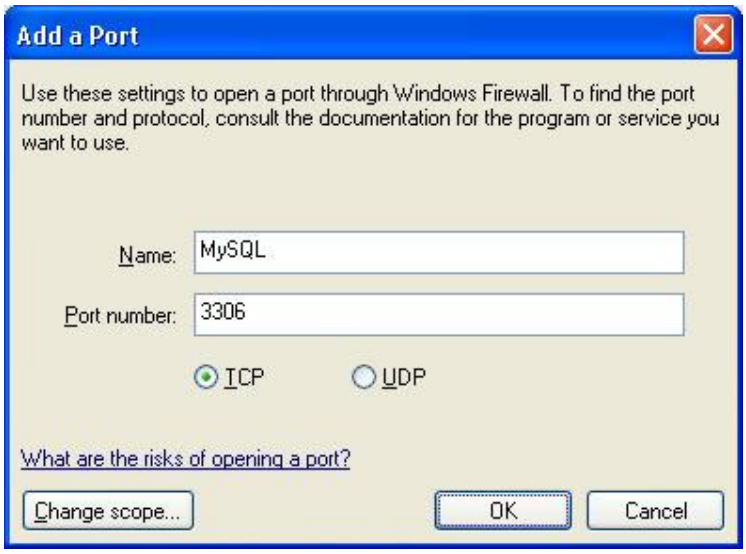


On Windows XP, you can exclude MySQL from the firewall by following the steps below:

1. Navigate to Start -> Settings -> Control Panel -> Windows Firewall



2. In the resulting dialog, enter the information as shown in the screenshot



After clicking OK twice, return to the MySQL error message and select retry. MySQL should now be able to create the instance correctly.

1.4 Installation of MySQL in Linux Operating System

All downloads for MySQL are located at MySQL Downloads. Pick the version number for *MySQL Community Server* you want and, as exactly as possible, the platform you want.

The recommended way to install MySQL on a Linux system is via RPM. MySQL AB makes the following RPMs available for download on its web site:

- **MySQL -** The MySQL database server, which manages databases and tables, controls user access, and processes SQL queries.
- **MySQL-client -** MySQL client programs, which make it possible to connect to and interact with the server.
- **MySQL-devel -** Libraries and header files that come in handy when compiling other programs that use MySQL.
- **MySQL-shared -** Shared libraries for the MySQL client.
- **MySQL-bench -** Benchmark and performance testing tools for the MySQL database server.

The MySQL RPMs listed here are all built on a SuSE Linux system, but they'll usually work on other Linux variants with no difficulty.

How to install MySQL version 5.5 on a Ubuntu 14.04 server.

1.5 Installing MySQL

To install MySQL this way, update the package index on your server and install the package with apt-get.

$ sudo apt-get update

$ sudo apt-get install mysql-server

You'll be prompted to create a root password during the installation. Choose a secure one and make sure you remember it, because you'll need it later. Move on to step two from here.

1.6 Installing MySQL 5.5

If you want to install MySQL 5.5 specifically, the process is still very straightforward. First, update the package index on your server.

$ sudo apt-get update

Then, to install MySQL 5.5, install the mysql-server-5.5 package.

$ sudo apt-get install mysql-server-5.5

You'll be prompted to create a root password during the installation. Choose a secure one and make sure you remember it, because you'll need it later.

1.7 Post-installation Steps

MySQL ships with a blank password for the root MySQL user. As soon as you have successfully installed the database and client, you need to set a root password as follows:

```
$ mysqladmin -u root password "new_password";
```

Now to make a connection to your MySQL server, you would have to use the following command:

```
$ mysql -u root -p
$ Enter password:*******
```

UNIX users will also want to put your MySQL directory in your PATH, so you won't have to keep typing out the full path every time you want to use the command-line client. For bash, it would be something like:

```
$ export PATH=$PATH:/usr/bin:/usr/sbin
```

1.8 Running MySQL at boot time

If you want to run MySQL server at boot time, and then make sure you have following entry in /etc/rc.local file.

```
/etc/init.d/mysqld start
```

Also,you should have mysql binary in /etc/init.d/ directory.

1.9 Conclusion

Upon completion of this practical session you should be able to independently, install MySQL on windows and in Linux operating system as well.

EXPERIMENT-2

AIM: To implement the following DDL commands of SQL with examples.

- **Create table**
- **Alter table**
- **Drop Table.**

2.0 Learning Objective

At the end of the session you will be able to

- Identify the commands and its use in data definition
- Create a database table using an example schema in a database.
- Alter a database table using an example schema in a database.
- Drop an existing database table using an example schema in a database.

2.1 Oracle Tools

Oracle has many tools such as SQL * PLUS, Oracle Forms, Oracle Report Writer, OracleGraphics etc.

- **SQL * PLUS:** The SQL * PLUS tool is made up of two distinct parts. These are
- **Interactive SQL:** Interactive SQL is designed for create, access and manipulate data structures like tables and indexes.
- **PL/SQL:** PL/SQL can be used to developed programs for different applications.
- **Oracle Forms:** This tool allows you to create a data entry screen along with the suitable menu objects. Thus it is the oracle forms tool that handles data gathering and data validation in a commercial application.
- **Report Writer:** Report writer allows programmers to prepare innovative reports using data from the oracle structures like tables, views etc. It is the report writer tool that handles the reporting section of commercial application.
- **Oracle Graphics:** Some of the data can be better represented in the form of pictures. The oracle graphics tool allows programmers to prepare graphs using data from oracle structures like tables, views etc.

2.2 SQL (Structured Query Language)

Structured Query Language is a database computer language designed for managing data in relational database management systems (RDBMS), and

originally based upon Relational Algebra. Its scope includes data query and update, schema creation and modification, and data access control.

SQL was one of the first languages for Edgar F. Codd's relational model and became the most widely used language for relational databases.

- IBM developed SQL in mid of 1970's.
- Oracle incorporated in the year 1979.
- SQL used by IBM/DB2 and DS Database Systems.
- SQL adopted as standard language for RDBS by ASNI in 1989.

2.3 Data Types

1. **CHAR (Size):** This data type is used to store character strings values of fixed length. The size in brackets determines the number of characters the cell can hold. The maximum number of character is 255 characters.
2. **VARCHAR (Size) / VARCHAR2 (Size):** This data type is used to store variable length alphanumeric data. The maximum character can hold is 2000 character.
3. **NUMBER (P, S):**The NUMBER data type is used to store number (fixed or floating point). Number of virtually any magnitude may be stored up to 38 digits of precision.Number as large as $9.99 * 10^{124}$. The precision (p) determines the number of places to the right of the decimal. If scale is omitted then the default is zero. If precision is omitted, values are stored with their original precision up to the maximum of 38 digits.
4. **DATE:** This data type is used to represent date and time. The standard format is DD-MM-YYas in 17-SEP-2009. To enter dates other than the standard format, use the appropriate functions. Date time stores date in the 24-Hours format. By default the time in a date field is 12:00:00 am, if no time portion is specified. The default date for a date field is the first day the current month.
5. **LONG:** This data type is used to store variable length character strings containing up to 2GB. Long data can be used to store arrays of binary data in ASCII format. LONG values cannot be indexed, and the normal character functions such as SUBSTR cannot be applied.
6. **RAW:** The RAW data type is used to store binary data, such as digitized picture or image. Data loaded into columns of these data types are stored without any further conversion. RAW data type can have a maximum length of 255 bytes. LONG RAW data type can contain up to 2GB.

2.4 SQL language is sub-divided into several language elements, including

- *Clauses,* which are in some cases optional, constituent components of statements and queries.
- *Expressions,*which can produce either scalar values or tables consisting of columns and rows of data.
- *Predicates*which specify conditions that can be evaluated to SQL three-valued logic (3VL) Boolean truth values and which are used to limit the effects of statements and queries, or to change program flow.
- *Queries*which retrieve data based on specific criteria.
- *Statements*which may have a persistent effect on schemas and data, or which may control transactions, program flow, connections, sessions, or diagnostics.
- SQL statements also include the semicolon (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar.
- Insignificant white spaceis generally ignored in SQL statements and queries, making it easier to format SQL code for readability.

There are five types of SQL statements. They are:

1. Data Definition Language (DDL)
2. Data Manipulation Language (DML)
3. Data Retrieval Language (DRL)
4. Transitional Control Language (TCL)
5. Data Control Language (DCL)

2.5 Data Definition Language (DDL)

The Data Definition Language (DDL) is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project. Let's take a look at the structure and usage of four basic DDL commands:

1. CREATE 2. ALTER 3. DROP 4. RENAME

1. CREATE:

(a)CREATE TABLE: This is used to create a new relation (table)

Syntax:

CREATE TABLE <relation_name/table_name> (field_1 data_type(size),field_2 data_type(size), .. .);

Example:

SQL> CREATE TABLE Student (sno NUMBER (3), sname CHAR (10), class CHAR (5));

2. ALTER:

(a) ALTER TABLE ...ADD...: This is used to add some extra fields into existing relation.

Syntax:

ALTER TABLE relation_name ADD (new field_1 data_type (size), new field_2 data_type(size),..);

Example:

SQL>ALTER TABLE std ADD (Address CHAR (10));

(b) ALTER TABLE...MODIFY...: This is used to change the width as well as data type of fields of existing relations.

Syntax:

ALTER TABLE relation_name MODIFY (field_1 newdata_type(Size), field_2
newdata_type(Size),....field_newdata_type(Size));

Example:

SQL>ALTER TABLE student MODIFY (sname VARCHAR (10),class VARCHAR(5));

(c) ALTER TABLE..DROP...: This is used to remove any field of existing relations.

Syntax:

ALTER TABLE relation_name DROP COLUMN (field_name);

Example:

SQL>ALTER TABLE student DROP column (sname);

(d) ALTER TABLE..RENAME...: This is used to change the name of fields in existing relations.

Syntax:

ALTER TABLE relation_name RENAME COLUMN (OLD field_name) to
(NEW field_name);

Example:

SQL>ALTER TABLE student RENAME COLUMN sname tostu_name;

3. DROP TABLE: This is used to delete the structure of a relation. It permanently deletes the records in the table.

Syntax:

DROP TABLE relation_name;

Example:

SQL>DROP TABLE std;

4. RENAME: It is used to modify the name of the existing database object.

Syntax:

RENAME TABLE old_relation_name TO new_relation_name;

Example:

SQL>RENAME TABLE std TO std1;

2.6 LAB PRACTICE ASSIGNMENTS

1. Create a table EMPLOYEE with following schema:
2. (Emp_no, E_name, E_address, E_ph_no, Dept_no, Dept_name,Job_id , Salary)
3. Add a new column; HIREDATE to the existing relation.
4. Change the datatype of JOB_ID from char to varchar2.
5. Change the name of column/field Emp_no to E_no.
6. Modify the column width of the job field of emp table

EXPERIMENT-3

AIM: To implement the following DML commands of SQL with suitable examples.

- **Insert values in table**
- **Update table**
- **Delete Table**
- **Select rows**
- **Truncate**

3.0 Learning Objective

At the end of the session you will be able to be familiar with

- To understand different issues involved in the design and implementation of a database system
- To understand and use data manipulation language to query, update and manage a database

3.1 Data Manipulation Language (DML)

The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

1. INSERT 2. UPDATE 3. DELETE

1. INSERT INTO: This is used to add records into a relation. These are three type of INSERT INTO queries which are as

a) Inserting a single record

Syntax:

INSERT INTO < relation/table name> (field_1, field_2.....field_n)VALUES (data_1,data_2,.....data_n);

Example:

SQL>INSERT INTO student (sno, sname, class, address) VALUES (1,'Ravi','M.Tech','Palakol');

b) Inserting a single record

Syntax:

INSERT INTO < relation/table name>VALUES (data_1, data_2,.....data_n);

Example:

SQL>INSERT INTO student VALUES (1,'Ravi','M.Tech','Palakol');

c) Inserting all records from another relation

Syntax:

INSERT INTO relation_name_1 SELECT Field_1, field_2, field_n FROM relation_name_2 WHERE field_x=data;

Example:

SQL>INSERT INTO std SELECT sno, sname FROM student WHERE name = 'Ramu';

d) Inserting multiple records

Syntax:

INSERT INTO relation_name field_1,field_2,.....field_n) VALUES(&data_1,&data_2,.....&data_n);

Example:

SQL>INSERT INTO student (sno, sname, class,address) VALUES (&sno,'&sname','&class','&address');

Enter value for sno: 101

Enter value for name: Ravi

Enter value for class: M.Tech

Enter value for name: Palakol

2. UPDATE-SET-WHERE: This is used to update the content of a record in a relation.

Syntax:

SQL>UPDATE relation name SET Field_name1=data,field_name2=data, WHERE field_name=data;

Example:

SQL>UPDATE student SET sname = 'kumar' WHERE sno=1;

3. DELETE-FROM: This is used to delete all the records of a relation but it will retain the structure of that relation.

a) DELETE-FROM: This is used to delete all the records of relation.

Syntax:

SQL>DELETE FROM relation_name;

Example:

SQL>DELETE FROM std;

b) DELETE -FROM-WHERE: This is used to delete a selected record from a relation.

Syntax:

SQL>DELETE FROM relation_name WHERE condition;

Example:

SQL>DELETE FROM student WHERE sno = 2;

5. TRUNCATE: This command will remove the data permanently. But structure will not be removed.

Difference between Truncate & Delete

- By using truncate command data will be removed permanently & will not get back where as by using delete command data will be removed temporally & get back by using roll back command.
- By using delete command data will be removed based on the condition where as by using truncate command there is no condition.
- Truncate is a DDL command & delete is a DML command.

Syntax:

TRUNCATE TABLE <Table name>

Example

TRUNCATE TABLE student;

3.3 To retrieve data from one or more tables

1. SELECT FROM: To display all fields for all records.

Syntax:

SELECT * FROM relation_name;

Example:

SQL> select * from dept;

```
DEPTNO   DNAME        LOC
--------    -----------     ----------
10   ACCOUNTING    NEW YORK
20   RESEARCH        DALLAS
30   SALES              CHICAGO
40   OPERATIONS     BOSTON
```

2. SELECT FROM: To display a set of fields for all records of relation.

Syntax:

SELECT a set of fields FROM relation_name;

Example:

SQL> select deptno, dname from dept;

```
DEPTNO  DNAME
---------      ----------
10   ACCOUNTING
20   RESEARCH
30   SALES
```

3. SELECT - FROM -WHERE: This query is used to display a selected set of fields for a selected set of records of a relation.

Syntax:

SELECT a set of fields FROM relation_name WHERE condition;

Example:

SQL> select * FROM dept WHERE deptno<=20;

```
DEPTNO   DNAME        LOC
-----------    -----------      ------------
10       ACCOUNTING   NEW YORK
20         RESEARCH     DALLAS
```

3.4 LAB PRACTICE ASSIGNMENT

Create a table EMPLOYEE with following schema:

(Emp_no, E_name, E_address, E_ph_no, Dept_no, Dept_name,Job_id , Salary)

Write SQL queries for following question:

1. Insert a least 5 rows in the table.
2. Display all the information of EMP table.
3. Display the record of each employee who works in department D10.
4. Update the city of Emp_no-12 with current city as Nagpur.
5. Display the details of Employee who works in department MECH.
6. Delete the email_id of employee James.

7. Display the complete record of employees working in SALES Department.

EXPERIMENT-4

AIM: To implement different types of functions in SQL with suitable examples.

- Number Function
- Aggregate Function
- Character Function
- Conversion Function
- Date Function

4.0 Learning Objective

At the end of the session you will be able to be familiar with to understand and implement various types of function in SQL.

4.1 Number Function

Abs (n) :Select abs(-15) from dual;

Exp(n): Select exp(4) from dual;

Power (m,n): Select power(4,2) from dual;

Mod (m,n): Select mod(10,3) from dual;

Round (m,n): Select round(100.256,2) from dual;

Trunc (m,n): ;Select trunc(100.256,2) from dual;

Sqrt (m,n);Select sqrt(16) from dual;

Develop aggregate plan strategies to assist with summarization of several data entries.

4.2 Aggregative operators

In addition to simply retrieving data, we often want to perform some computation or summarization. SQL allows the use of arithmetic expressions. We now consider a powerful class of constructs for computing aggregate values such as MIN and SUM.

1. Count: COUNT following by a column name returns the count of tuple in that column. If DISTINCT keyword is used then it will return only the count of unique

tuple in the column. Otherwise, it will return count of all the tuples (including duplicates) count (*) indicates all the tuples of the column.

Syntax:

COUNT (Column name)

Example:

SELECT COUNT (Sal) FROM emp;

2. SUM: SUM followed by a column name returns the sum of all the values in that column.

Syntax:

SUM (Column name)

Example:

SELECT SUM (Sal) From emp;

3. AVG: AVG followed by a column name returns the average value of that column values.

Syntax:

AVG (n1, n2...)

Example:

Select AVG (10, 15, 30) FROM DUAL;

4. MAX: MAX followed by a column name returns the maximum value of that column.

Syntax:

MAX (Column name)

Example:

SELECT MAX (Sal) FROM emp;

SQL> select deptno, max (sal) from emp group by deptno;

DEPTNOMAX (SAL)

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

10 5000

20 3000

30 2850

SQL> select deptno, max (sal) from emp group by deptno having max(sal)<3000;

DEPTNO MAX(SAL)

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

30 2850

5. MIN: MIN followed by column name returns the minimum value of that column.

Syntax:

MIN (Column name)

Example:

SELECT MIN (Sal) FROM emp;

```
SQL>select deptno,min(sal) from emp group by deptno having min(sal)>1000;
DEPTNO      MIN (SAL)
----------  -----------
10          1300
```

4.3 Character Function

initcap(char) : select initcap (“hello”) from dual;

lower (char): select lower (‘HELLO’) from dual;

upper (char) : select upper (‘hello’) from dual;

ltrim (char,[set]): selectltrim (‘cseit’, ‘cse’) from dual;

rtrim (char,[set]): select rtrim (‘cseit’, ‘it’) from dual;

replace (char,search): select replace(‘jack and jue’,‘j’,‘bl’) from dual;

4.4 Conversion Functions

To_char: TO_CHAR (number) converts n to a value of VARCHAR2 data type, using the optional number format fmt. The value n can be of type NUMBER, BINARY_FLOAT, or BINARY_DOUBLE.

```
SQL>select to_char(65,'RN')from dual;
        LXV
```

To_number: TO_NUMBER converts expr to a value of NUMBER data type.

```
SQL>Select to_number ('1234.64') from Dual;
        1234.64
```

To_date: TO_DATE converts char of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of DATE data type.

```
SQL>SELECT TO_DATE ('January 15, 1989, 11:00 A.M.')FROM DUAL;
        TO_DATE
        ---------
```

15-JAN-89

4.5 String Functions

Concat: CONCAT returns char1 concatenated with char2. Both char1 and char2 can be any of the datatypes

SQL>SELECT CONCAT (‘ORACLE’,’CORPORATION’)FROM DUAL;

ORACLECORPORATION

Lpad: LPAD returns expr1, left-padded to length n characters with the sequence of characters in expr2.

SQL>SELECT LPAD(‘ORACLE’,15,’*’)FROM DUAL;

*********ORACLE

Rpad: RPAD returns expr1, right-padded to length n characters with expr2, replicated as many times as necessary.

SQL>SELECT RPAD (‘ORACLE’,15,’*’)FROM DUAL;

ORACLE*********

Ltrim: Returns a character expression after removing leading blanks.

SQL>SELECT LTRIM(‘SSMITHSS’,’S’)FROM DUAL;

MITHSS

Rtrim: Returns a character string after truncating all trailing blanks

SQL>SELECT RTRIM (‘SSMITHSS’,’S’)FROM DUAL;

SSMITH

Lower: Returns a character expression after converting uppercase character data to lowercase.

SQL>SELECT LOWER (‘DBMS’) FROM DUAL;

dbms

Upper: Returns a character expression with lowercase character data converted to uppercase

SQL>SELECT UPPER(‘dbms’)FROM DUAL;

DBMS

Length: Returns the number of characters, rather than the number of bytes, of the given string expression, excluding trailing blanks.

SQL>SELECT LENGTH(‘DATABASE’)FROM DUAL;

8

Substr: Returns part of a character, binary, text, or image expression.

SQL>SELECT SUBSTR(‘ABCDEFGHIJ’3,4)FROM DUAL;

CDEF

Instr: The INSTR functions search string for substring. The function returns an integer indicating the position of the character in string that is the first character of this occurrence.

SQL>SELECT INSTR('CORPORATE FLOOR','OR',3,2)FROM DUAL;

14

4.6 Date Functions

Sysdate:

SQL>SELECT SYSDATE FROM DUAL;

29-DEC-08

next_day:

SQL>SELECT NEXT_DAY(SYSDATE,’WED’)FROM DUAL;

05-JAN-09

add_months:

SQL>SELECT ADD_MONTHS(SYSDATE,2)FROM DUAL;

28-FEB-09

last_day:

SQL>SELECT LAST_DAY(SYSDATE)FROM DUAL;

31-DEC-08

months_between:

SQL>SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE)FROM EMP;

4

Least:

SQL>SELECT LEAST('10-JAN-07','12-OCT-07')FROM DUAL;

10-JAN-07

Greatest:

SQL>SELECT GREATEST('10-JAN-07','12-OCT-07')FROM DUAL;

10-JAN-07

Trunc:

SQL>SELECT TRUNC(SYSDATE,'DAY')FROM DUAL;

28-DEC-08

Round:

SQL>SELECT ROUND(SYSDATE,'DAY')FROM DUAL;

28-DEC-08

to_char:

SQL> select to_char(sysdate, "dd\mm\yy") from dual;

24-mar-05.

to_date:

SQL> select to date (sysdate, "dd\mm\yy") from dual;

24-mar-o5.

4.7 LAB PRACTICE ASSIGNMENT

Create a table EMPLOYEE with following schema:

(Emp_no, E_name, E_address, E_ph_no, Dept_no, Dept_name,Job_id, Designation , Salary)

Write SQL statements for the following query.

1. List the E_no, E_name, Salary of all employees working for MANAGER.
2. Display all the details of the employee whose salary is more than the Sal of any IT PROFF..
3. List the employees in the ascending order of Designations of those joined after 1981.
4. List the employees along with their Experience and Daily Salary.
5. List the employees who are either ‘CLERK’ or ‘ANALYST’ .
6. List the employees who joined on 1-MAY-81, 3-DEC-81, 17-DEC-81,19-JAN-80 .
7. List the employees who are working for the Deptno 10 or20.
8. List the Enames those are starting with ‘S’ .
9. Dislay the name as well as the first five characters of name(s) starting with ‘H’
10. List all the emps except ‘PRESIDENT’ & ‘MGR” inasc order of Salaries.

EXPERIMENT-5

AIM: To implement different types of operators in SQL.

- Arithmetic Operator
- Logical Operator
- Comparison Operator
- Special Operator
- Set Operator

5.0 Learning Objective:

At the end of the session you will be able to be familiar with different types of operator and how to use in MySQl.

5.1 Arithmetic Operators

(+): Addition - Adds values on either side of the operator.

(-): Subtraction - Subtracts right hand operand from left hand operand.

(*): Multiplication - Multiplies values on either side of the operator.

(/): Division - Divides left hand operand by right hand operand.

(^): Power- raise to power of.

(%): Modulus - Divides left hand operand by right hand operand and returns remainder.

5.2 Logical Operators

AND: The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.

OR: The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

NOT: The NOT operator reverses the meaning of the logical operator with which it is used.

Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc.

This is a negate operator.

5.3 Comparison Operators

(=): Checks if the values of two operands are equal or not, if yes then condition becomes true.

(!=): Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.

(<>): Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.

(>): Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true

(<): Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.

(>=): Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.

(<=): Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

5.4 Special Operator

BETWEEN: The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.

IS NULL: The NULL operator is used to compare a value with a NULL attribute value.

ALL: The ALL operator is used to compare a value to all values in another value set

ANY: The ANY operator is used to compare a value to any applicable value in the list according to the condition.

LIKE: The LIKE operator is used to compare a value to similar values using wildcard operators. It allows to use percent sign (%) and underscore (_) to match a given string pattern.

IN: The IN operator is used to compare a value to a list of literal values that have been specified.

EXIST: The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.

5.5 Set Operators

The Set operator combines the result of 2 queries into a single result. The following are the operators:

- Union
- Union all
- Intersect
- Minus

Union: Returns all distinct rows selected by both the queries

Union all: Returns all rows selected by either query including the duplicates.

Intersect: Returns rows selected that are common to both queries.

Minus: Returns all distinct rows selected by the first query and are not by the second

Example 1

Unions cities from table1 and table2.

```
SELECT city FROM table1
UNION
SELECT city FROM table2;

CITY
----------
HELSINKI
RIGA
STOCKHOLM
TALLINN
VILNIUS
```

Example 2

Unions cities from table2 and table1. The query ordering is not important, result is the same, compare with Example 1.

```
SELECT city FROM table2
UNION
SELECT city FROM table1;

CITY
----------
HELSINKI
RIGA
STOCKHOLM
TALLINN
VILNIUS
```

DO NOT ASSUME that Union always return ordered row set. It is NOT TRUE. It is just because of implementation model, i.e. sort is being done to filter out duplicates. At least from version 10 Oracle has possibility to do HASH UNIQUE operation, which doesn't sort rows and you won't get them back sorted. So ALWAYS use Order by clause if you need guaranteed order of rows.

Next example just combines the rows without filtering out duplicates.

Example 3

Unions of ALL cities from table1 and table2

```
SELECT city FROM table1
UNION ALL
SELECT city FROM table2;

CITY
----------
RIGA
RIGA
```

RIGA
TALLINN
TALLINN
TALLINN
VILNIUS
HELSINKI
HELSINKI
STOCKHOLM
RIGA
RIGA
VILNIUS
VILNIUS
VILNIUS
VILNIUS
HELSINKI

17 rows selected.

Example 4

UNION [DISTINCT] even with empty set may reduce number of rows. Compare result from first two queries with third query.

SELECT city FROM table1;

CITY

RIGA
RIGA
RIGA
TALLINN
TALLINN
TALLINN
VILNIUS

HELSINKI

HELSINKI

STOCKHOLM

10 rows selected.

```
SELECT city FROM table3;
```

no rows selected

```
SELECT city FROM table1
UNION
SELECT city FROM table3;
```

CITY

HELSINKI

RIGA

STOCKHOLM

TALLINN

VILNIUS

Example 5

UNION ALL with empty set gives the same result as without it.

```
SELECT city FROM table1
UNION ALL
SELECT city FROM table3;
```

CITY

RIGA

RIGA

RIGA

TALLINN

TALLINN

TALLINN

VILNIUS
HELSINKI
HELSINKI
STOCKHOLM

10 rows selected.

Example 6

Cities in table1 intersect [distinct] cities in table2.

```
SELECT city FROM table1
INTERSECT
SELECT city FROM table2;
```

```
CITY
----------
HELSINKI
RIGA
VILNIUS
```

Example 7

Cities in table2 intersect [distinct] empty set (cities in table3). Every intersection with empty set is empty set.

```
SELECT city FROM table1
INTERSECT
SELECT city FROM table3;
```

no rows selected

Example 8

Distinct Symmetric difference of table1 and table2

```
(SELECT city FROM table1
MINUS
```

```
SELECT city FROM table2)
UNION
(SELECT city FROM table1
MINUS
SELECT city FROM table2);

CITY
----------
STOCKHOLM
TALLINN
```

5.6 LAB PRACTICE ASSIGNMENTs

1. Display all the dept numbers available with the dept and emp tables avoiding duplicates.
2. Display all the dept numbers available with the dept and emp tables.
3. Display all the dept numbers available in emp and not in dept tables and vice versa.

EXPERIMENT-6

AIM: To implement different types of Joins

1. Simple Join
2. Self Join
3. Outer Join

6.0 Learning Objective

At the end of the session you will be able to perform different types of joins

6.1 Joins

The SQL **Joins** clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each. The join is actually performed by the 'where' clause which combines specified rows of tables.

Syntax:

```
SELECT column 1, column 2, column 3...
FROM table_name1, table_name2
WHERE table_name1.column name = table_name2.columnname;
```

6.2 Types of Joins:

- Simple Join
- Self Join
- Outer Join

Simple Join:

It is the most common type of join. It retrieves the rows from 2 tables having a common column and is further classified into

Equi-join:

A join, which is based on equalities, is called equi-join.

Example:

Select * from item, cust where item.id=cust.id;

In the above statement, item-id = cust-id performs the join statement. It retrieves rows from both the tables provided they both have the same id as specified by the where clause. Since the where clause uses the comparison operator (=) to perform

a join, it is said to be equijoin. It combines the matched rows of tables. It can be used as follows:

- To insert records in the target table.
- To create tables and insert records in this table.
- To update records in the target table.
- To create views.

Non Equi-join:

It specifies the relationship between columns belonging to different tables by making use of relational operators other than'='.

Example:

Select * from item, cust where item.id<cust.id;

Table Aliases

Table aliases are used to make multiple table queries shorted and more readable. We give an alias name to the table in the 'from' clause and use it instead of the name throughout the query.

Self join:

Joining of a table to itself is known as self-join. It joins one row in a table to another. It can compare each row of the table to itself and also with other rows of the same table.

Example:

select * from emp x ,emp y where x.salary>= (select avg(salary) from x.emp where x. deptno =y.deptno);

Outer Join:

It extends the result of a simple join. An outer join returns all the rows returned by simple join as well as those rows from one table that do not match any row from the table. The symbol (+) represents outer joins.

– Left outer join

– Right outer join

– Full outer join

6.3 LAB PRACTICE ASSIGNMENT

Consider the following schema:

Sailors (sid, sname, rating, age)

Boats (bid, bname, color)

Reserves (sid, bid, day (date))

1. Find all information of sailors who have reserved boat number 101.
2. Find the name of boat reserved by Bob.
3. Find the names of sailors who have reserved a red boat, and list in the order of age.
4. Find the names of sailors who have reserved at least one boat.
5. Find the ids and names of sailors who have reserved two different boats on the same day.
6. Find the ids of sailors who have reserved a red boat or a green boat.
7. Find the name and the age of the youngest sailor.
8. Count the number of different sailor names.
9. Find the average age of sailors for each rating level.
10. Find the average age of sailors for each rating level that has at least two sailors.

****