**Experiment 4. Practicing DDL commands**

practiced the following DDL commands.

SQL DDL Commands:

The DDL commands are

1. Create
2. Alter
3. Drop
4. Truncate
5. Rename

Enter password: \*\*\*\*

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 6

Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

Syntax: Create database database\_name;

Ex:create databse cse;

mysql> use cse;

Database changed

* 1. **Creation of Table:**

This command is used for creating tables.

Syntax:

create table tablename(column-name data-type constraints….);

Egl:

SQL>create table passenger(passport\_id integer primary key, name varchar(50) not null, age integer not null, sex char(1), address varchar(50) not null);

Table created

* 1. **Altering theTable:**

It is used formodifying the table structure.

Input:

SQL> alter table passenger add phone no(10); Table altered

SQL>desc passenger ;

|  |  |  |
| --- | --- | --- |
| Name | Null | Type |
| Passport\_id | Not null | Number(30) |
| Name | Not null | Varchar950) |
| Age | Not null | Number(36) |
| Sex  Address Ph.no | Not null | Char(1)  Varchar(50) Number(10) |

* 1. **Dropping of theTable;**

It is used for deleting the table structure and data permanently. SQL>drop table passenger;

Table dropped

* 1. **Truncate theTable:**

It is used for deleting the data in the table but the table structure exists. SQL>truncate table passenger;

Table truncated

* 1. **Renaming of theTable:**

It is used for changing the existing table names.

SQL>rename passenger to passenger1; Table renamed

SQL>desc passenger;

Error

ORA-04043: Object passenger does not exists SQL> desc passenger1;

|  |  |  |
| --- | --- | --- |
| Name | Null | Type |
| Passport\_id | Not null | Number(30) |
| Name | Not null | Varchar950) |
| Age | Not null | Number(36) |
| Sex  Address Ph.no | Not null | Char(1)  Varchar(50) Number(10) |

**Experiment 5: Practicing DML commands**

DML commands are used for managing data with in schema objects. Few DML commands are

1. select
2. insert
3. update
4. delete
5. **select:**

This command is used to retrieve data from the table.

Syntax:

Select \* from table name; SQL> select \* from passenger;

Bus\_number source destination

|  |  |  |
| --- | --- | --- |
| 1234 | hyd | tirupathi |
| 2345 | hyd | banglore |
| 23 | hyd | kolkata |

1. **Insert:**

This command is used to insert the data into database.

**Creation of bus table:**

Create table bus(bus-number varchar(20),source varchar(20), destination varchar(20)); Table created

**Inserting values into bus table:**

SQL> insert into bus values (1234,’hyd’, ‘tirupathi’); Datainserted

SQL> insert into bus values (2345,’hyd’, ‘banglore’); Datainserted

SQL> insert into bus values (23,’hyd’,‘kolkata’); Datainserted

**Inserting values into passenger table:**

SQL> insert into passenger values (1,45, ‘ramesh’,45,’M’,’abc123’); Datainserted

SQL> insert into passenger values (2,78,’geetha’,36,’F’,’abc124’); Datainserted

SQL> insert into passenger values(3,90,’ram’,36,’M’,’abc12’); Datainserted

1. **Update:**

This command is used to update existing data with in a table.

Command:

SQL>update bus destination ‘banglore’ where bus no=23; Data updated

1. **Delete:**

Delete command is used to delete the records (or) rows (or) complete table from the database

Command:

SQL>delete from bus where busno=’1234’; 1 row deleted

SQL>delete from bus where busno=’23’; 1 row deleted

**Experiment 6:Querying**

CONDITIONAL SELECTIONS AND OPERATORS

We have two clauses used in this

* **Where**
* **Orderby**

USING WHERE

**syntax: select\*from<*table\_name*>where<*condition*>;**

the following are the different types of operators used in where clause.

* **Arithmeticoperators**
* **Comparisonoperators**
* **Logicaloperators**

**Arithmetic operators -- highest precedence**

+, -, \*, /

**Comparison operators**

 =, !=, >, <, >=, <=,<>

* **between, notbetween**
* **in, notin**
* **null, not null**
* **like**

#### Logical operators

* **And**
* **Or -- lowestprecedence**
* **not**

a) USING =, >, <, >=, <=, !=,<>

Ex: SQL>select\*fromstudentwhereno=2;

NONAME MARKS

|  |  |
| --- | --- |
| 2 Saketh | 200 |
| 2 Naren  2 rows selected. | 400 |

SQL>select \* from student where no < 2;

NONAME MARKS

1 Sudha 100

1. Jagan 300
2. rowsselected.

SQL>select \* from student where no > 2;

NONAME MARKS

1. Ramesh
2. Madhu
3. Visu
4. Rattu

4 rows selected.

SQL>select \* from student where no <= 2;

NONAME MARKS

|  |  |  |
| --- | --- | --- |
| 1 | Sudha | 100 |
| 2 | Saketh | 200 |
| 1 | Jagan | 300 |
| 2 | Naren | 400 |

4 rows selected.

SQL>select \* from student where no >= 2;

NONAME MARKS

|  |  |  |
| --- | --- | --- |
| 2 | Saketh | 200 |
| 2 | Naren | 400 |
| 3 | Ramesh |  |
| 4 | Madhu |  |
| 5 | Visu |  |
| 6 | Rattu |  |

6 rows selected.

SQL>select \* from student where no != 2;

NONAME MARKS

|  |  |  |
| --- | --- | --- |
| 1 | Sudha | 100 |
| 1 | Jagan | 300 |
| 3 | Ramesh |  |
| 4 | Madhu |  |
| 5 | Visu |  |
| 6 | Rattu |  |

6 rows selected.

SQL>select \* from student where no <> 2;

NONAME MARKS

|  |  |  |
| --- | --- | --- |
| 1 | Sudha | 100 |
| 1 | Jagan | 300 |
| 3 | Ramesh |  |
| 4 | Madhu |  |
| 5 | Visu |  |
| 6 | Rattu |  |

6 rows selected.

1. **USINGAND**

Thiswillgivestheoutputwhenalltheconditionsbecometrue. syntax:

**select \* from <*table\_name*> where <*condition1*> and <*condition2*> and ..**

**<*conditionn*>;**

Ex: SQL>select\*fromstudentwhereno=2andmarks>=200;

NONAME MARKS

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

2 Saketh 200

2 Naren 400

2 rows selected.

1. **USINGOR**

Thiswillgivestheoutputwheneitheroftheconditionsbecometrue. syntax:

**select \* from <*table\_name*> where <*condition1*> and <*condition2*> or ..**

**<*conditionn*>; Ex:**

SQL>select \* from student where no = 2 or marks >= 200;

NONAME MARKS

2 Saketh 200

1. Jagan 300
2. Naren 400
3. rowsselected.
4. **USINGBETWEEN**

Thiswillgivestheoutputbasedonthecolumnanditslowerbound, upperbound.

syntax:

**select\*from<*table\_name*>where<*col*>between<*lowerbound*>and<*upper bound*>;**

Ex:

SQL>select \* from student where marks between 200 and 400;

NONAME MARKS

2 Saketh 200

1. Jagan 300
2. Naren 400
3. rowsselected.
4. **USING NOTBETWEEN**

Thiswillgivestheoutputbasedonthecolumnwhichvaluesarenotinitslower bound,upperbound.

syntax:

**select \* from <*table\_name*> where <*col*> not between <*lower bound*> and**

**<*upper bound*>; Ex:**

SQL>select \* from student where marks not between 200 and 400;

NONAME MARKS

1 Sudha 100

1 row selected.

1. **USINGIN**

Thiswillgivestheoutputbasedonthecolumnanditslistofvaluesspecified. syntax:

**select \* from <*table\_name*> where <*col*> in ( *value1, value2, value3 …***

***valuen*); Ex:**

SQL>select \* from student where no in (1, 2, 3);

NONAME MARKS

|  |  |  |
| --- | --- | --- |
| 1 | Sudha | 100 |
| 2 | Saketh | 200 |
| 1 | Jagan | 300 |
| 2 | Naren | 400 |
| 3 | Ramesh |  |

5 rows selected.

1. **USING NOTIN**

Thiswillgivestheoutputbasedonthecolumnwhichvaluesarenotinthelist of valuesspecified.

syntax:

**select \* from <*table\_name*> where <*col*> not in ( *value1, value2, value3 …***

***valuen*); Ex:**

SQL>select \* from student where no not in (1, 2, 3);

NONAME MARKS

1. Madhu
2. Visu
3. Rattu

3 rows selected.

1. **USINGNULL**

Thiswillgivestheoutputbasedonthenullvaluesinthespecifiedcolumn. syntax:

**select\*from<*table\_name*>where<*col*>isnull; Ex:**

SQL>select \* from student where marks is null;

NONAME MARKS

1. Ramesh
2. Madhu
3. Visu
4. Rattu

4 rows selected.

1. **USING NOTNULL**

Thiswillgivestheoutputbasedonthenotnullvaluesinthespecifiedcolumn. syntax:

**select\*from<*table\_name*>where<*col*>isnotnull; Ex:**

SQL>select \* from student where marks is not null;

NONAME MARKS

|  |  |  |
| --- | --- | --- |
| 1 | Sudha | 100 |
| 2 | Saketh | 200 |
| 1 | Jagan | 300 |
| 2 | Naren | 400 |

4 rows selected.

1. **USINGLIKE**

This will beusedtosearchthroughtherowsofdatabasecolumnbasedonthe pattern youspecify.

syntax: select\*from<*table\_name*>where<*col*>like<*pattern*>; Ex: i)Thiswillgive therowswhosemarksare100.

SQL>select \* from student where marks like 100;

NONAME MARKS

1 Sudha 100

1. rowselected.
2. **Thiswillgivetherowswhosenamestartwith‘S’. SQL>select \* from student where name like 'S%';** NONAME MARKS

|  |  |
| --- | --- |
| 1 Sudha | 100 |
| 2 Saketh | 200 |
| 2 rowsselected. |  |

1. **Thiswillgivetherowswhosenameendswith‘h’. SQL>select \* from student where name like '%h';** NONAME MARKS

1. Saketh 200
2. Ramesh

2 rows selected.

iV) This will give the rows whose name’s second letter start with ‘a’.

SQL>select \* from student where name like '\_a%';

NONAME MARKS

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

|  |  |  |
| --- | --- | --- |
| 2 | Saketh | 200 |
| 1 | Jagan | 300 |
| 2 | Naren | 400 |
| 3 | Ramesh |  |
| 4 | Madhu |  |
| 6 | Rattu |  |

6 rows selected.

V) This will give the rows whose name’s third letter start with ‘d’.

SQL>select\*fromstudentwherenamelike' d%';

NONAME MARKS

1 Sudha 100

4 Madhu

2 rows selected.

Vi) This will give the rows whose name’s second letter start with ‘t’ from

ending.

SQL>select \* from student where name like '%t\_';

NONAME MARKS

2 Saketh 200

6 Rattu

2 rows selected.

Vii) This will give the rows whose name’s third letter start with ‘e’ from

ending.

SQL>select\*fromstudentwherenamelike'%e ';

NONAME MARKS

1. Saketh 200
2. Ramesh

2 rows selected.

Viii) This will give the rows whose name cotains 2 a’s.

SQL>select \* from student where name like '%a% a %';

NONAME MARKS

1 Jagan 300

1 row selected.

\* You have to specify the patterns in *like* using underscore ( \_ ).

#### USING ORDER BY

Thiswillbeusedtoorderingthecolumnsdata(ascendingordescending). Syntax:

Select\*from<*table\_name*>orderby<*col*>desc; Bydefaultoraclewilluseascendingorder.

Ifyouwantoutputindescendingorderyouhavetouse*desc*keywordafterthe column.

Ex: SQL>select\*fromstudentorderbyno;

NONAME MARKS

|  |  |  |
| --- | --- | --- |
| 1 | Sudha | 100 |
| 1 | Jagan | 300 |
| 2 | Saketh | 200 |
| 2 | Naren | 400 |
| 3 | Ramesh |  |
| 4 | Madhu |  |
| 5 | Visu |  |
| 6 | Rattu |  |

8 rows selected.

SQL>select \* from student order by no desc;

NONAME MARKS

6 Rattu

5 Visu

|  |  |
| --- | --- |
| 4 Madhu  3 Ramesh |  |
| 2 Saketh | 200 |
| 2 Naren | 400 |
| 1 Sudha | 100 |
| 1 Jagan | 300 |
| 8rowsselected. |  |

#### SUBQUERIES AND EXISTS

SUBQUERIES

Nestingofqueries,onewithintheotheristermedasasubquery. Astatementcontainingasubqueryiscalledaparentquery.

Subqueries are used to retrieve data from tables that depend on the values in the table itself.

TYPES

* **Single rowsubqueries**
* **Multi rowsubqueries**
* **Multiplesubqueries**
* **Correlatedsubqueries**

SINGLE ROW SUBQUERIES

Insinglerowsubquery,itwillreturnonevalue. Ex:

SQL>select\*fromempwheresal>(selectsalfromempwhereempno=7566);

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7788 | SCOTT | ANALYST | 7566 | 19-APR-87 3000 | 20 |
| 7839 | KING | PRESIDENT |  | 17-NOV-81 5000 | 10 |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 3000 | 20 |

MULTI ROW SUBQUERIES

In multi row sub query, it will return more than one value. In such cases we should include operatorslike**any,all,in,notin,between**,thecomparisionoperatorandthesubquery. **Any:**

Ex:

SQL>select\*fromempwheresal>any(selectsalfromempwheresal between2500and4000);

|  |  |  |  |
| --- | --- | --- | --- |
| EMPNO | ENAME | JOB MGR HIREDATESALCOMM | DEPTNO |
| 7566 | JONES | MANAGER 7839 02-APR-81 2975 | 20 |
| 7788 | SCOTT | ANALYST 756619-APR-87 3000 | 20 |
| 7839 | KING | PRESIDENT 17-NOV-815000 | 10 |
| 7902 | FORD | ANALYST 756603-DEC-81 3000 | 20 |
| **All:** |  |  |  |

SQL>select\*fromempwheresal>all(selectsalfromempwheresalbetween 2500 and4000);

EMPNO ENAME JOB MGR HIREDATE SAL COMMDEPTNO

7839 KING PRESIDENT 17-NOV-81 5000 10

MULTIPLE SUBQUERIES

Thereisnolimitonthenumberofsubqueriesincludedinawhereclause.Itallowsnesting ofaquerywithinasubquery.

Ex:

SQL>select\*fromempwheresal=(selectmax(sal)fromempwheresal< (selectmax(sal)fromemp));

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

|  |  |  |  |
| --- | --- | --- | --- |
| 7788 | SCOTT ANALYST 7566 | 19-APR-87 3000 | 20 |
| 7902 | FORD ANALYST 7566 | 03-DEC-81 3000 | 20 |

CORRELATED SUBQUERIES

A subquery is evaluated once for the entire parent statement where as a correlated subquery is evaluated once for every row processed by the parent statement.

Ex:

SQL>selectdistinctdeptnofromempewhere5<=(selectcount(ename)from empwheree.deptno=deptno);

DEPTNO

20

30

EXISTS

Existsfunctionisatestforexistence.Thisisalogicaltestforthereturnofrowsfroma query.

Ex:

Supposewewanttodisplaythedepartmentnumberswhichhasmorethan4employees.

SQL>select deptno,count(\*) from emp group by deptno having count(\*) > 4;

DEPTNO COUNT(\*)

20 5

30 6

From the above query can you want to display the names of employees?

SQL>selectdeptno,ename,count(\*)fromempgroupbydeptno,enamehaving count(\*) >4;

no rows selected

Theabovequeryreturnsnothingbecausecombinationofdeptnoandenameneverreturn morethanonecount.

The solution is to use exists which follows.

SQL>selectdeptno,enamefromempe1whereexists(select\*fromempe2 wheree1.deptno=e2.deptnogroupbye2.deptnohavingcount(e2.ename)>4) order bydeptno,ename;

DEPTNO ENAME

20 ADAMS

20 FORD

20 JONES

20 SCOTT

20 SMITH

30 ALLEN

30 BLAKE

30 JAMES

30 MARTIN

30 TURNER

30 WARD

11 rows selected.

NOT EXISTS

SQL>selectdeptno,enamefromempe1wherenotexists(select\*fromempe2 wheree1.deptno=e2.deptnogroupbye2.deptnohavingcount(e2.ename)>4) order bydeptno,ename;

DEPTNO ENAME

10 CLARK

10 KING

10 MILLER

#### SET OPERATORS

TYPES

* **Union**
* **Unionall**
* **Intersect**
* **Minus**

UNION

Thiswillcombinetherecordsofmultipletableshavingthesamestructure. Ex: SQL>select\*fromstudent1unionselect\*fromstudent2;

UNION ALL

Thiswillcombinetherecordsofmultipletableshavingthesamestructurebut includingduplicates.

Ex: SQL>select\*fromstudent1unionallselect\*fromstudent2;

INTERSECT

Thiswillgivethecommonrecordsofmultipletableshavingthesamestructure. Ex: SQL>select\*fromstudent1intersectselect\*fromstudent2;

MINUS

Thiswillgivetherecordsofatablewhoserecordsarenotinothertableshaving the samestructure.

Ex: SQL>select\*fromstudent1minusselect\*fromstudent2;