Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

1. Implement DDL Statement.

• Create table , Modify table, Drop table

The DDL(Data Definition Language) Commands in Structured Query Language are used to create and modify the schema of the database and its objects. The syntax of DDL commands is predefined for describing the data. The commands of DDL deal with how the data should exist in the database.

➢ **CREATE TABLE Statement:**

To create a new table in the database, use the SQL CREATE TABLE statement. A table's structure, including column names, data types, and constraints are defined when it is created in SQL.

Syntax:

CREATE table table_name
(
Column1 datatype (size),
column2 datatype (size),
.
.
columnN datatype(size)
);

➢ **ALTER TABLE STATEMENT:**

The ALTER TABLE statement in SQL is used to modify an existing table structure in a database without losing any data. It allows you to add, remove, or modify columns, change data types, or apply constraints to improve data integrity and ensure that the table meets evolving **business requirements**. It allows for structural changes like **adding new columns**, **modifying existing ones**, deleting columns, and renaming columns within a table.

Syntax:

ALTER TABLE table_name

[ADD | DROP | MODIFY] column_name datatype;

- **ADD** is used to add a new column.
- **DROP** is used to remove an existing column.
- **MODIFY** is used to change the datatype or definition of an existing column.

➢ **TRUNCATE**

This command is used to remove all rows from the table, but the structure of the table still exists.

Syntax –

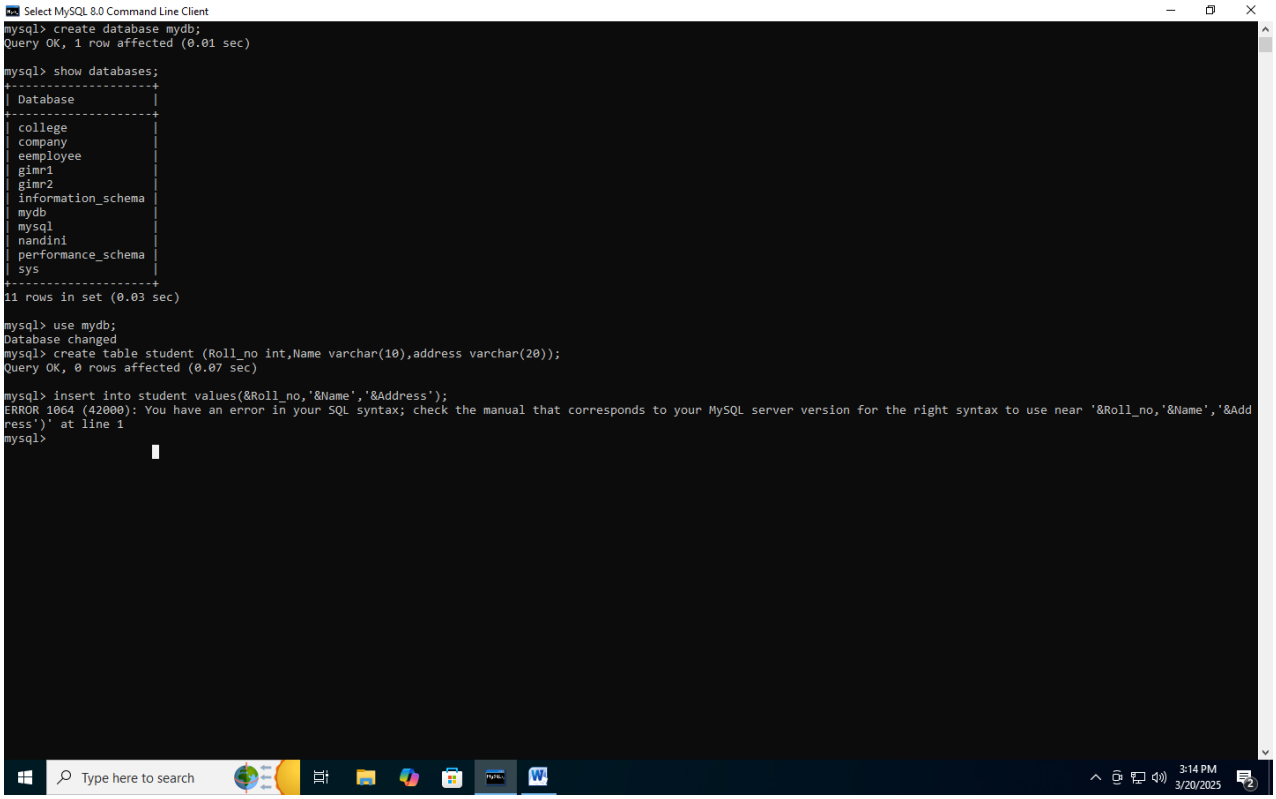
TRUNCATE TABLE table_name;

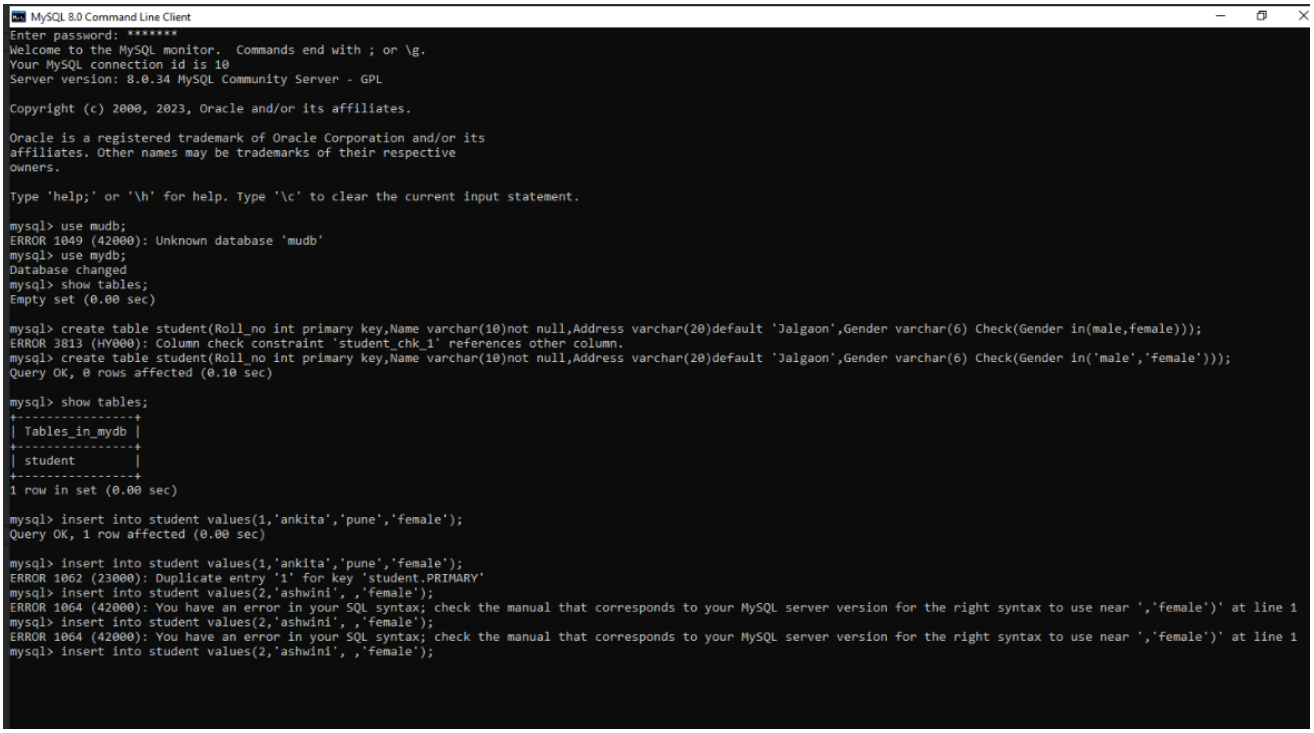
➢ **DROP**

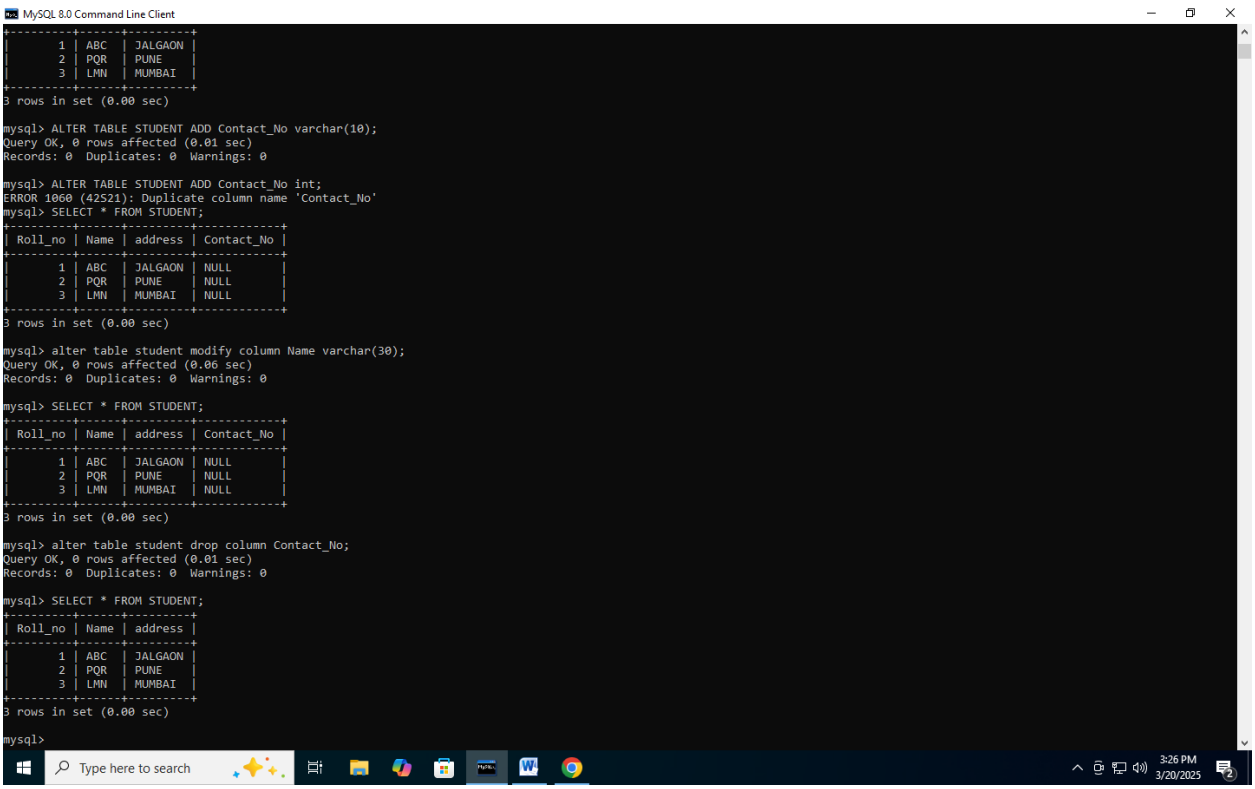
This command is used to remove an existing table along with its structure from the Database.

Syntax –

DROP TABLE table_name;






MySQL 8.0 Command Line Client
+---------+------+---------+
1	ABC	JALGAON
2	PQR	PUNE
3	LMN	MUMBAI
+---------+------+---------+
3 rows in set (0.00 sec)
mysql> ALTER TABLE STUDENT ADD Contact_No varchar(10);
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> ALTER TABLE STUDENT ADD Contact_No int;
ERROR 1060 (42521): Duplicate column name 'Contact_No'
mysql> SELECT * FROM STUDENT;
+---------+------+---------+------------+
| Roll_no | Name | address | Contact_No |
+---------+------+---------+------------+
1	ABC	JALGAON	NULL
2	PQR	PUNE	NULL
3	LMN	MUMBAI	NULL
+---------+------+---------+------------+
3 rows in set (0.00 sec)
mysql> alter table student modify column Name varchar(30);
Query OK, 0 rows affected (0.06 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> SELECT * FROM STUDENT;
+---------+------+---------+------------+
| Roll_no | Name | address | Contact_No |
+---------+------+---------+------------+
1	ABC	JALGAON	NULL
2	PQR	PUNE	NULL
3	LMN	MUMBAI	NULL
+---------+------+---------+------------+
3 rows in set (0.00 sec)
mysql> alter table student drop column Contact_No;
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> SELECT * FROM STUDENT;
+---------+------+---------+
| Roll_no | Name | address |
+---------+------+---------+
1	ABC	JALGAON
2	PQR	PUNE
3	LMN	MUMBAI
+---------+------+---------+
3 rows in set (0.00 sec)
mysql>
Type here to search
3:26 PM
3/20/2025

Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

---

2. Implement DML Statement.

• Adding/Modify/Delete data using Insert/ Update/ Delete

The **SQL commands** that deal with the **manipulation of data** present in the database belong to **DML** or Data Manipulation Language and this includes most of the **SQL statements.** It is the component of the SQL statement that controls access to data and to the database.

➢ Insert Statement:

The INSERT INTO statement in MySQL allows users to add new records (rows) into a specified table. It follows a concise syntax to specify the table name and the values to be inserted into the respective columns.

Syntax:

1. For single record:
   INSERT INTO table_name (column1, column2, ..., columnN)
   VALUES (value1, value2, ..., valueN);
2. For multiple records:
   INSERT INTO table_name (column1, column2, ..., columnN)
   VALUES
   (value1_1, value1_2, ..., value1_N),
   (value2_1, value2_2, ..., value2_N),
   ...,
   (valueM_1, valueM_2, ..., valueM_N);

➢ UPDATE Statement

The **UPDATE** statement in **MySQL** is used to modify existing records or data in a table. It is commonly used to correct any errors previously made or update the values of a column. It's important to remember that changes made through the UPDATE statement are permanent and cannot be undone.

Syntax:

UPDATE table_name

SET column_name = value

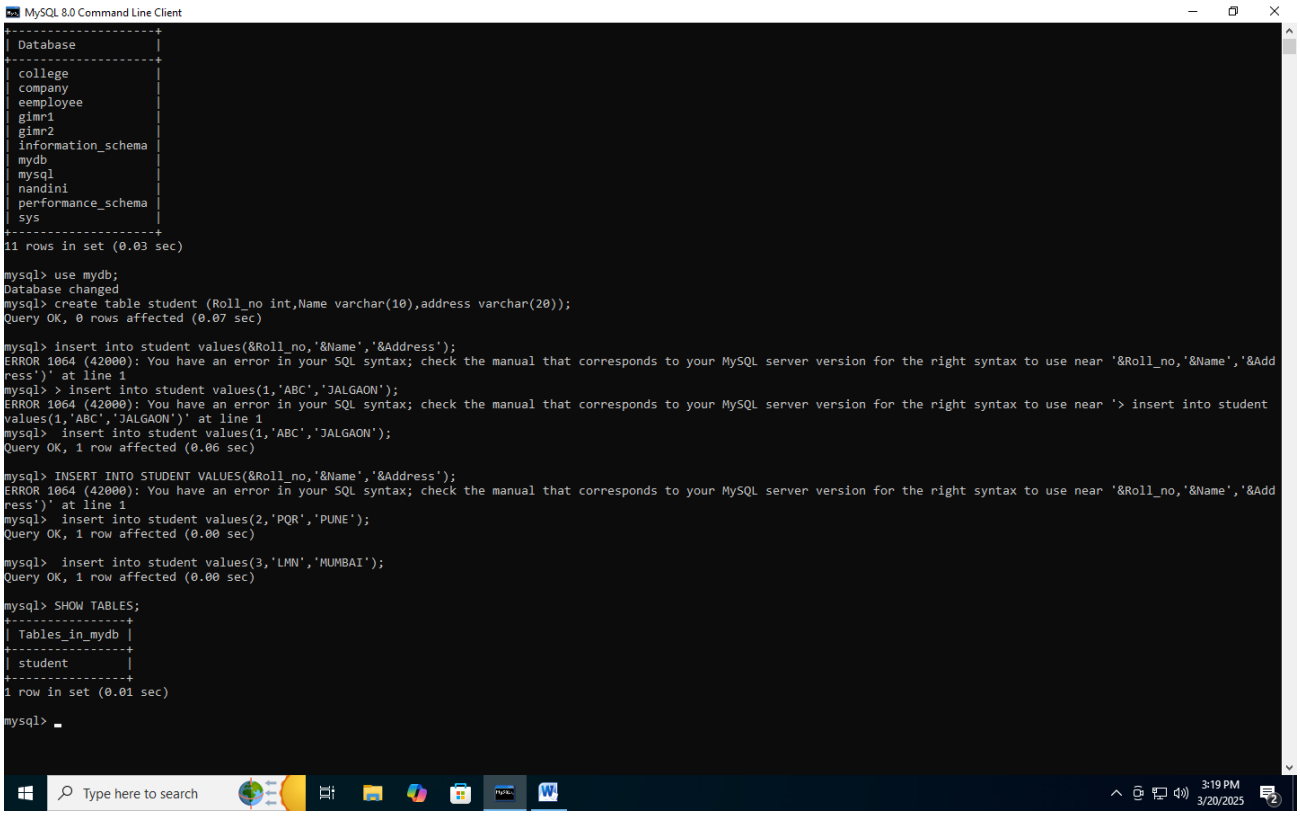
WHERE (condition);

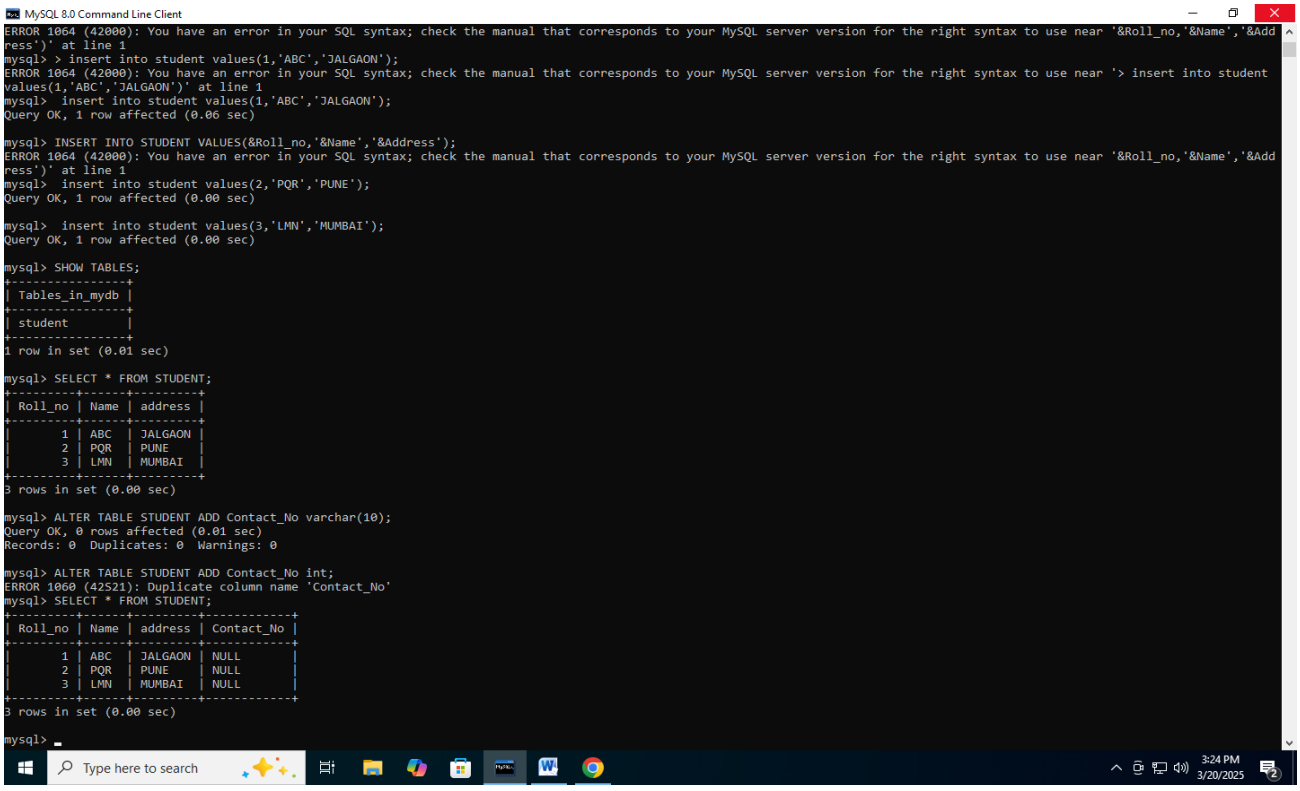
➢ Delete Statement:

The MySQL **DELETE** statement deletes one or more existing records from a table. It is commonly used with the WHERE or JOIN clause. It is a **Data Manipulation Language** (DML) statement. Generally, you cannot ROLLBACK (undo) after performing the DELETE statement. You can delete the entire table data using DELETE or delete only specific rows.

Syntax:

DELETE FROM table_name WHERE condition;





Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

---

3. Implement following Constraints.

- NOT NULL, Primary Key Constraint, Foreign Key Constraint
- Unique Constraint, Check Constraint, Default Constraint

➢ **NOT NULL Constraint**

The NOT NULL constraint in MySQL serves as a **validation rule** for the columns. In NOT NULL constraints the column cannot contain the NULL values. This constraint guarantees that each row in the table must have a value for that particular column.

```
CREATE TABLE Persons (
   ID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255) NOT NULL,
   Age int
 );
```

➢ **Primary Key Constraint**

A MySQL Primary Key is a unique column/field in a table that should not contain duplicate or NULL values and is used to identify each record in the table uniquely.

Syntax

```
CREATE TABLE table_name (
   column1 datatype,
   column2 datatype,
   PRIMARY KEY (column1)
);
```

➢ **Foreign Key Constraint**

A FOREIGN KEY is a field/column(or collection of fields) in a table that refers to a PRIMARY KEY in another table. It is used for linking one or more than one table together. FOREIGN KEY is also called referencing key. A Foreign key creates a link (relation) between two tables thus creating referential integrity.

The FOREIGN KEY creates a relationship between the columns in the current table or let's say table A (the one with the foreign key) and the referenced table or table B (the one with the unique key).

Syntax

```
CREATE TABLE child_table (
child_id INT PRIMARY KEY,
parent_id INT,
other_columns DATATYPE,
FOREIGN KEY (parent_id) REFERENCES parent_table(parent_id)
);
```

➢ **Unique Constraint**

A UNIQUE constraint in **MySQL** prevents two records from having identical values in a column. A UNIQUE constraint can contain null values as long as the combination of values is unique. This makes it different from a **PRIMARY KEY** as the primary key constraint cannot contain null values. There can be multiple UNIQUE constraints in a table, but only one PRIMARY KEY constraint.

Syntax

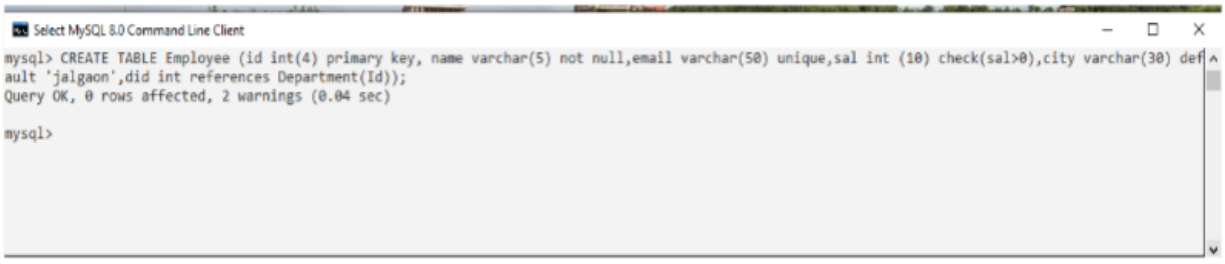
```
CREATE TABLE table_name (
  column_name data_type UNIQUE,
  ...
);
```

Check Constraint

In MYSQL, the Check constraint is used to impose conditions on what type of data to be inserted into our table. It helps in maintaining the accuracy and consistency of the data. It helps in avoiding the entry of data that does not follow our specified conditions.

Example:

```
CREATE TABLE Persons (
   ID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int,
   CHECK (Age>=18)
);
```



Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

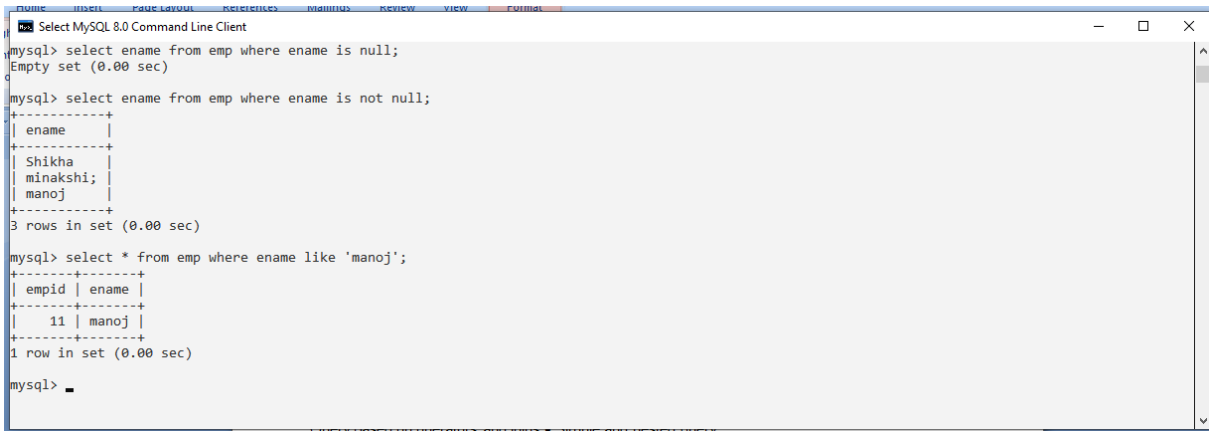
---

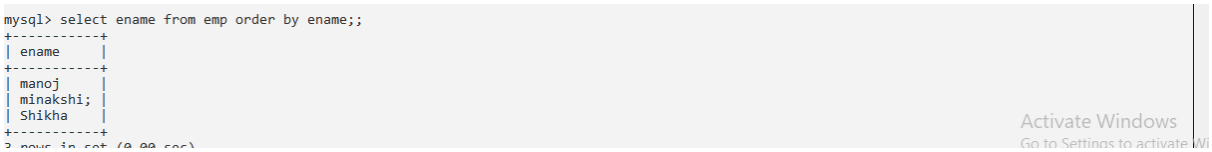
4. Implement following clauses.

- **Simple select clause**
- **Accessing specific data with Where Clause**
- **Ordered By/ Distinct/Group By Clause**

```
CREATE TABLE Employee ( Id INT,Name VARCHAR(50),Salary MONEY, DepartmentId INT);

INSERT INTO Employee VALUES (1001, 'Leena Lele', 20000, 101);

INSERT INTO Employee VALUES (1002, 'Pawan Pawar', 30000, 102);

INSERT INTO Employee VALUES (1003, 'Sunil Nikumbh', 40000, 103);

INSERT INTO Employee VALUES (1003, 'Shankar Marathe', 40000, 106);

CREATE TABLE Department (Id INT PRIMARY KEY,Name VARCHAR(30) NOT NULL UNIQUE,
Location VARCHAR(50) DEFAULT 'MUMBAI',CONSTRAINT locationChk CHECK (Location
IN('MUMBAI','DELHI','BANGLORE')));

INSERT INTO Department VALUES (101,'TESTING','BANGLORE') ; INSERT INTO Department
VALUES (102,'CODING','MUMBAI') ; INSERT INTO Department VALUES
(103,'MAINTAINANCE','DELHI') ; INSERT INTO Department VALUES
(104,'DEPLOYMENT','MUMBAI');

SELECT * FROM Department;
SELECT * FROM Department WHERE Location like ‘MUMBAI’ ;
SELECT * FROM Employee WHERE Salary <= 30000;
SELECT * FROM Employee ORDER BY Salary;
SELECT * FROM Employee ORDER BY Salary DESC;
SELECT Distinct(City) FROM Employee;
SELECT COUNT(*), Department   FROM Employee GROUP BY Department;
SELECT COUNT(*), City   FROM Employee GROUP BY City HAVING count(*) > 2;
```


Select MySQL 8.0 Command Line Client
Query OK, 0 rows affected, 2 warnings (0.04 sec)
mysql> select * from emp;
+-------+-----------+
| empid | ename |
+-------+-----------+
1	Shikha
4	minakshi;
11	manoj
+-------+-----------+
3 rows in set (0.00 sec)
mysql> select ename from emp;
+-----------+
| ename |
+-----------+
| Shikha |
| minakshi; |
| manoj |
+-----------+
3 rows in set (0.00 sec)
mysql> select * from emp where ename='Shikha';
+-------+--------+
| empid | ename |
+-------+--------+
| 1 | Shikha |
+-------+--------+
1 row in set (0.00 sec)
mysql> select ename from emp order by ename;;
+-----------+
| ename |
+-----------+
| manoj |
| minakshi; |
| Shikha |
+-----------+
Activate Windows


Select MySQL 8.0 Command Line Client
mysql> select ename from emp where ename is null;
Empty set (0.00 sec)
mysql> select ename from emp where ename is not null;
+-----------+
| ename |
+-----------+
| Shikha |
| minakshi; |
| manoj |
+-----------+
3 rows in set (0.00 sec)
mysql> select * from emp where ename like 'manoj';
+-------+-------+
| empid | ename |
+-------+-------+
| 11 | manoj |
+-------+-------+
1 row in set (0.00 sec)
mysql>


mysql> select ename from emp order by ename;;
+-----------+
| ename |
+-----------+
| manoj |
| minakshi; |
| Shikha |
+-----------+
Activate Windows

Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

---

5. Implement Aggregate Functions.

• AVG, COUNT, MAX, MIN, SUM, CUBE

Aggregate functions

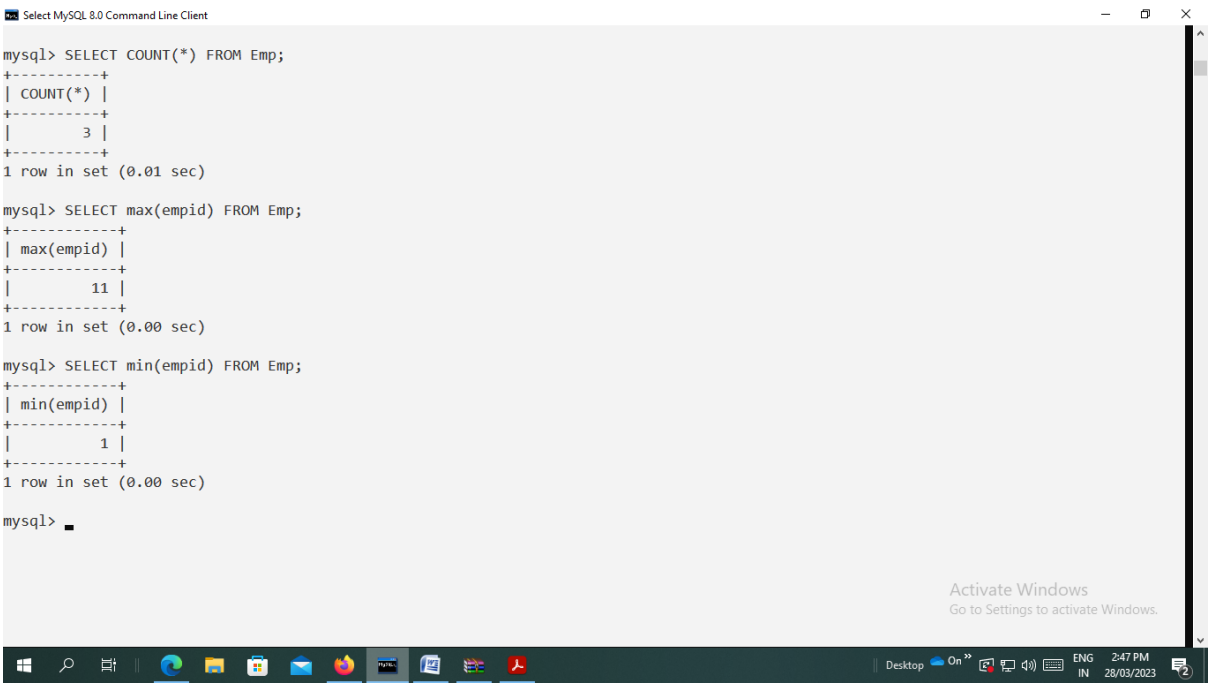
COUNT(Column Name) –
SELECT COUNT(*)FROM Customers;

MIN(Column Name)
SELECT MIN(Age) FROM Customers;

MAX(Column Name)
SELECT MAX(Age) FROM Customers;

AVG(Column Name)
SELECT AVG(Age) FROM Customers;

SUM(Column Name)
SELECT SUM(purchasedAmount) FROM Customers;



Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

---

6. Implement all String functions.

- ASCII (character_expression) , returns integer SELECT ASCII('A');
  65

- CHAR (integer_expression) , returns string SELECT CHAR(65)
  example: SELECT FirstName + ' ' + LastName, + CHAR(13) + EmailAddress
  A
- CONCAT (string_value1, string_value2 [, string_valueN])
- SELECT CONCAT ('Happy ', 'Birthday Prashant! Today is', 25, '/', '1/2023') AS Result; Happy Birthday Prashant! Today is 25/1/2023
- LEFT (character_expression , integer_expression)
- SELECT LEFT('abcdefg',2);
- SELECT LEFT(Name, 5) FROM Product; ab
- RIGHT (character_expression , integer_expression) SELECT RIGHT('abcdefg',2);
- SELECT RIGHT(Name, 5) FROM Product ; fg
- LEN (string_expression)
- SELECT LEN(FirstName) AS Length, FirstName FROM Customers;
- LOWER (character_expression)
- SELECT LOWER(FirstName) FROM Customers;
- UPPER (character_expression)
- SELECT UPPER(LastName) FROM Customers;
- SELECT UPPER('leena');
- SELECT REPLICATE('wow! ',3) Wow! Wow! Wow!

7. Implement Date and Time Functions.

1. Return the current date and time:

SELECT CURRENT_TIMESTAMP

2. DATEADD(interval, number, date) example

Add two months to a date, then return the date:

SELECT DATEADD(month, 2, '2022/01/25') AS DateAdd;

Add one year to a date, then return the date:

SELECT DATEADD(year, 1, '2022/01/25') AS DateAdd;

3. The DATEDIFF() function returns the difference between two dates.
   DATEDIFF(interval, date1, date2)
   SELECT DATEDIFF(month, '2017/08/25', '2022/08/25') AS DateDiff;
   SELECT DATEDIFF(hour, '2022/08/25 07:00', '2022/08/25 12:45') AS DateDiff;

4. Return a specified part of a date, returns String DATENAME(interval, date)

SELECT DATENAME(yy, '2017/08/25') AS DatePartString;

5. The DATEPART() function returns a specified part of a date, returns integer

SELECT DATEPART(yy, '2017/08/25') AS DatePartInt;

6. Return the day of the month for a date: DAY(date)

SELECT DAY('2017/08/13 09:08') AS DayOfMonth;

7. Return the month part of a date:

SELECT MONTH('2017/08/25') AS Month;

8. Return the current database system date and time:
   SELECT GETDATE();
9. Return the date and time of the SQL Server:

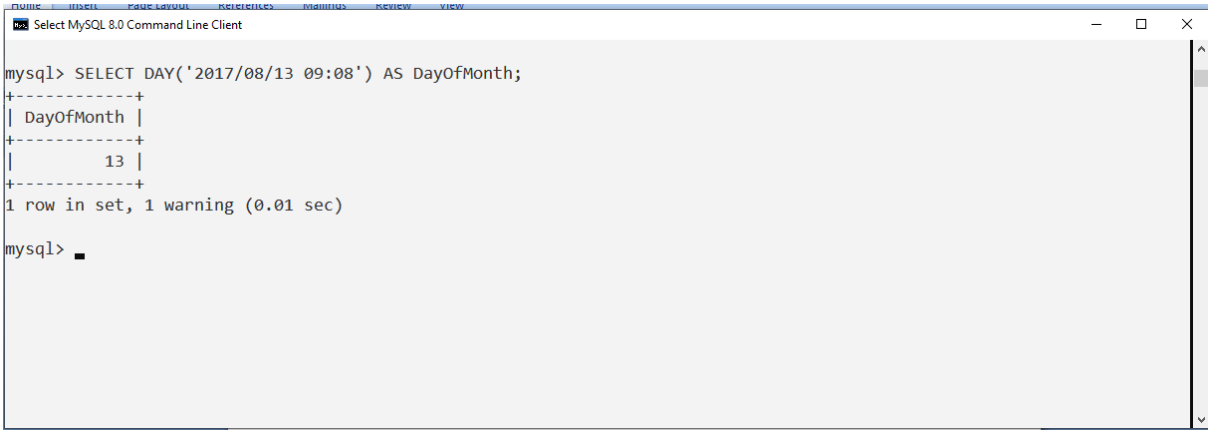
SELECT SYSDATETIME() AS SysDateTime;

10. Return the year part of a date:
    SELECT YEAR('2017/08/25') AS Year;
11. Return the current UTC date and time:

SELECT GETUTCDATE();

12. Check if the expression is a valid date:

SELECT ISDATE('2022-01-25');


Select MySQL 8.0 Command Line Client
mysql> SELECT DAY('2017/08/13 09:08') AS DayOfMonth;
+------------+
| DayOfMonth |
+------------+
| 13 |
+------------+
1 row in set, 1 warning (0.01 sec)
mysql>

Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

---

8. Implement use of UNION, INTERSECTION, SET DIFFERENCE.

UNION:-
The UNION operator is used to combine the result-set of two or more SELECT statements.

select first name,last name from Student UNION select first name,last name from Student1;

UNION ALL:

select first name,last name from Student UNION All select first name,last name from Student1;

INTERSECT:-
The INTERSECT operator in SQL is used to retrieve the records that are identical/common between the result sets of two or more tables.

select first name,last name from Student intersect select first name,last name from Student1;

SET DIFFERENCE:-
Set difference in SQL is accomplished by the keyword EXCEPT.
select first name,last name from Student except select first name,last name from Student1;

Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

---

9. Implement Nested Queries & all types of JOIN operation.

Create table emp(empid int primary key,name varchar, salary int, departmentid int);

insert into emp (001,' riya marathe', 50000, 101);
insert into emp (002,' rina patil', 40000, 101);
insert into emp (003,' Piyush marathe', 50000, 102);
insert into emp (004,' Satish patil', 50000, 101);
insert into emp (005,' sagar koli', 50000, 101);
insert into emp (006,' sima pendase', 50000, 101);

create table dept (deptid int primary key,name varchar(25), location varchar(30) default 'mumbai', constraint locationchk check (location in('mumbai', 'delhi', 'banglore'));

insert into dept values (101, 'testing','banglore');
insert into dept values (102, 'coding ','mumbai');

Nested Query
select * from employee where departmentid not in (select deptid from department where name='testing');

INNER JOIN-

select emp.empid,emp.departmentid,emp.name,emp.salary,dept.name, dept.location from emp inner join dept on emp.departmentid= dept.deptid;

RIGHT JOIN-

select emp.empid,emp.departmentid, emp.name,emp.salary,dept.name,dept.location from emp right join dept on e.departmentid =dept.deptid;

Full Join-
select emp.empid,emp.departmentid,e.name,e.salary,dept.name,dept.location from emp full join dept on emp.departmentid=dept.deptid;

Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

10. Implement practical performing different operations on a view.

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition. In this article we will learn about creating , deleting and updating Views.

```
CREATE VIEW view_name AS
SELECT column1, column2..... FROM table_name      WHERE condition;
```

```
create table dept
(id int primary key,name varchar(100)not null,hod varchar(100),capacity int);

insert into dept values(101,'cs',' akanksha', 20);
insert into dept values(102,' Biology','Rakhi', 40);

create table Faculty
(id int primary key,name varchar(100) not null,dept int foreign key references dep( id));

insert into Faculty values(01,'Ramesh',101);
insert into Faculty values(02,'Nilesh',102);
insert into Faculty values(03,'Hiren',101);

create view data(teacherid,teachername,teacherdept,deptid,deptname,,deptcapacity) as
selectt faculty.id, faculty.name, faculty.department,dept.id,dept.name,dept.capacity from Faculty  inner join
dept on faculty.department=dept.id;


alter view data(teacherid,teachername,teacherdept,deptid deptname)
as select f.id,f. name,f.department,d.id,d.name from Faculty f inner join dept d on f.department=d.id


insert into data(teacherid,teachername, teacherdept) values(205,' chaudhari', 102);


update data set teachername='bonde' where teacherid=201;

Delete from data where teacherid=201;

Drop view data;
```

Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

---

11. Implement use of Procedures.

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

```
CREATE PROCEDURE SelectFaculty
AS
SELECT * FROM Faculty
GO;
```

```
EXEC procedure_name;
```

```
Exec ;
```

```
Create  table customers1( id int , name varchar (80),city varchar(70),PurchasedAmount int);
```

```
Select * from customers1;
```

```
Insert into customers1 values(1,' vaishu',' jalgaon ',500)
```

```
CREATE PROCEDURE SelectAllCustomers
AS SELECT FROM Customers
Go;
EXEC SelectAllCustomers;
EXEC SelectAllCustomers@city='Jalgaon';
```

```
CREATE PROCEDURE SelectAllCustomers8 @City nvarchar(30) @purchased Amount int=3000
as
SELECT * FROM Customers WHERE city @City AND purchasedAmount>=@purchasedamount
Go;
Exec SelectAllCustomers8@city=’jalgaon’,@purchasedamount=4000;
```

Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

---

12. Implement use of Triggers.

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers can be defined on the table, view, schema, or database with which the event is associated. Triggers can be written for the following purposes −

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

Create table student2(s_id int PRIMARY KEY IDENTITY,name varchar(50),computer int, java int,DBMS int,total int, per float);

insert into student values ('anju',70,90,80);
insert into dhe student values ('seema',80,90,50);
insert into dbo student values ('Vaishnavi',50,80,70);

```
CREATE trigger studmarksON Student2
FOR INSERT
AS
DECLARE @Id int
DECLARE @studName varchar(50)
DECLARE @s1 int
DECLARE @s2 int
DECLARE @s3 int
DECLARE @total INT
DECLARE @perc INT


SELECT @d=@@IDENTITY from inserted;
 SELECT @$1 = computer from inserted;
 SELECT @s2 = java from inserted;
SELECT @s3=DBMS from inserted;
 SELECT @studName =name from inserted;
 SET @total=@s1+@s3;
 SET @perc= @total *100/300

UPDATE student SET total= @total,perc @per WHERE sid=@id;

Drop trigger studmarksON;
```

Godavari Institute of Management & Research, Jalgaon
Master of Computer Application (MCA)

Name: _ Roll No: _ _ _ _ _

Date of Performance: _ _ /_ _/20_ _ Batch: _ _ _ _ _ _

Class: M.C.A. (I) Practical no:_ _ Sign. of Teacher :_ _ _ _ _ _

Subject: 432 Lab on Database Management System

---

13. Implement use of Cursor.

```
create table Dept
(
id int, name varchar(100), hod VARCHAR(100),



insert into dept values (1, 'khushi','Durga');
insert into dept values (2,'preeti, bindiya');
insert into dept values (3,’Manoj’, 'pooja');
insert into dept values (4,'sayali', 'himanshu');

declare
@name varchar(100);
@hod varchar(100);

declare deptcursor cursor for select name, hod form dept;
open deptcursor;

fetch next from deptcursor into @name,@hod;
while @@fetch_status=0

begin
print 'dept name:' + @name;

print 'hod:' + @hod

fetch  next  from  deptcursor into @name,@hod;

end;

close deptcursor;

deallocate deptcursor;
```