SHOW DATABASE:

Example:

Show databases;

CREATE DATABASE:

Example:

CREATE DATABASE database_name;

DROP DATABASE:

Example:

DROP DATABASE database_name;

BACKUP DATABASE

mysqldump -u username -p database_name > backup.sql
mysqldump -u username -p database_name table_name > table_backup.sql
mysqldump -u username -p --all-databases > all_backup.sql
mysqldump -u username -p database_name > backup.sql
mysqldump -u root -p mydb > D:\Backup\mydb_backup.sql

CREATE TABLE:

Example:

CREATE TABLE employees (id INT, name VARCHAR(50), salary DECIMAL(10,2));

SHOW TABLE:

Example:- Show tables;

ALTER TABLE:

RENAME TABLE:

Example:

RENAME TABLE old_table_name TO new _table_name

ADD COLUMN

Example: ALTER TABLE table_name ADD COLUMN department VARCHAR(30);

RENAME COLUMN

ALTER TABLE table_name RENAME COLUMN old_column_name TO new_column_name;

MODIFY COLUMN

ALTER TABLE table_name MODIFY COLUMN column_name(data type_range);

DROP COLUMN

ALTER TABLE table_name DROP COLUMN column_name;

DROP TABLE:

DROP TABLE employees;

CREATE INDEX:

CREATE INDEX idx_name ON employees(name);

SHOW INDEX:

Example:

SHOW INDEX FROM table_name;

ALTER INDEX:

- ALTER TABLE table_name ADD INDEX index_name(column_name);

- ALTER TABLE your_table RENAME INDEX old_index_name TO new_index_name;

DROP INDEX:

DROP INDEX index_name ON table_name;

ALTER TABLE employees DROP INDEX idx_name;

CREATE VIEW:

CREATE VIEW high_salary_employees AS SELECT * FROM employees WHERE salary > 50000;

SHOW VIEW

```
SHOW CREATE VIEW ‘view_name’;
SHOW FULL TABLES WHERE TABLE_TYPE = 'VIEW';
```

USE VIEW

```
SELECT * FROM ‘view_name’;
```

DROP VIEW

DROP VIEW high_salary_employees;

TRUNCATE TABLE :

Example:

TRUNCATE TABLE table name;

DML

INSERT:

INSERT INTO table_name (employee_id, first_name, last_name, salary) VALUES (101, 'Abhiskek', 'Singh', 60000);
OR
INSERT INTO table_name VALUES (101, 'Abhiskek', 'Singh', 60000);

UPDATE:

Example:

UPDATE employees SET salary = 65000 WHERE employee_id = 101;

DELETE:

Example:

DELETE FROM table_name WHERE employee_id = 101;

Delete one or more rows

DELETE FROM ‘tablename’ WHERE ‘attributesname’ BETWEEN 18 AND 32;

DCL

Create user:

CREATE USER ‘username’@’localhost’ IDENTIFIED BY ‘password’;

SELECT USER:

Select host,user from mysql.user;

Login user:

Example:

System mysql -u ‘username’ -p ‘password’

Create user:

CREATE USER ‘username’@’localhost’ IDENTIFIED BY ‘password’;

Drop user:

Example:

Drop user ‘username@localhost’;

GRANT:

Example:

GRANT ALL ON *.* TO ‘username’@’localhost’ WITH GRANT OPTION;

Example;

GRANT SELECT,UPDATE ON new.* TO ‘username@localhost;
Flush Privileges;

Show grant:

Show grants for ‘username’@’localhost’;

Revoke:

Example

Revoke ‘name of permission’ on ‘database name.table name’ from ‘username’@’localhost’;

TCL

TCL IS WORK ONLY FOR DML
SELECT @@AUTOCOMMIT; for commit status
CHEK YOUR AUTOCOMMIT STATUS, ITS TURN ON OR OFF-‘
IF AUTOCOMMIT IS 1 THEN TURN OFF YOUR AUTO COMMIT

SET AUTOCOMMIT=0; FOR CHANGE YOUR COMMIT STATUS

IF YOUR AUTO COMMIT IS 0 THEN DON'T CHANGE YOUR COMMIT STATUS

COMMIT; FOR COMMIT YOUR TRANSACTION

ROLLBACK; FOR REVERSE YOUR TRANSACTION

SAVEPOINT "savepoin name"; using for rollback point

ROLLBACK TO "savepoint_name"; for rollback to save point

-- Example of proper transaction usage

```
SET AUTOCOMMIT=0;
START TRANSACTION;
UPDATE STUDENT SET NAME="TEST_TRANSACTION"
WHERE ROLL_NO=3;
SELECT * FROM STUDENT WHERE ROLL_NO=3; -- Should
show "TEST_TRANSACTION"
ROLLBACK;
SELECT * FROM STUDENT WHERE ROLL_NO=3; -- Should
revert to previous value
COMMIT;
```

Primary key

- Create table 'table name'(id int primary key,name varchar(100),age int,mobile bigint(10));
- Create table 'table name'(id int,name varchar(100),age int,mobile bigint(10),primary key (id));

Foreign key

Create table 'ta ble name'(id int,city varchar(100),foreign key (id) references 'first table name'(first table primary key));

create table address(roll_no int,city varchar(100),state varchar(100),foreign key (roll_no) references student (roll_no));

JOIN

INNER JOIN

SELECT ‘column name’ FROM ‘table1 name’ INNER JOIN ‘table2 name’ ON table1.column=table2.column;

LEFT JOIN

SELECT ‘column name’ FROM ‘table1 name’ LEFT JOIN ‘table2 name’ ON table1.column=table2.column;

RIGHT JOIN

SELECT ‘column name’ FROM ‘table1 name’ RIGHT JOIN ‘table2 name’ ON table1.column=table2.column;

CROSS JOIN

SELECT ‘column name’ FROM ‘table1 name’ CROSS JOIN ‘table2 name’

SUB QUERY

Single row

SELECT *FROM student WHERE roll_no =(SELECT roll_no FROM student WHERE NAME ='value' and ADDRESS ='city';);

Multiple row

SELECT *FROM student WHERE roll_no IN (SELECT roll_no FROM student WHERE ADDRESS ='city';);

Correlated subquery

SELECT student.roll_no,marks.total FROM student INNER JOIN marks ON student.roll_no=marks.roll_no WHERE 2=(SELECT COUNT(DISTINCT total) FROM marks WHERE total >=marks.total);

AGGREGATE FUNCTIONS

COUNT

SELECT COUNT (column_name) FROM table name;

MIN

SELECT MIN(column_name) FROM table name;

MAX

SELECT MAX(column_name) FROM table name;

AVG

SELECT AVG(column_name) FROM table name;

SUM

SELECT SUM(column_name) FROM table name;

DELIMITER //

```
 CREATE PROCEDURE sp_name()
BEGIN
SELECT * FROM STUDENT;
END ;//
```

Call procedure syntax;

Call p_name( )//

Call p_name(type your input)//

SHOW PROCEDURE

SHOW CREATE PROCEDURE HELLO()//

SHOW PROCEDURE STATUS WHERE Db = 'karan'//

DROP PROCEDURE

DROP PROCEDURE name;

Example 2-

```
 CREATE PROCEDURE sp_name(parameter name and
type)
BEGIN
SELECT *FROM student WHERE NAME=
‘parametername’
```

```
END //
```

Example 3-

```
CREATE PROCEDURE name1(roll1 INT,roll2 INT)
BEGIN
SELECT * FROM student WHERE roll_no BETWEEN roll1 AND roll2;
  END //
```

Example 4 –

```
CREATE PROCEDURE name(roll1 INT,roll2 INT)
BEGIN
SELECT * FROM student WHERE roll_no=roll1 OR roll_no=roll2;
END //
```

Example 5 –

```
Create procedure p_name(num int)
Begin
If num%2=0 then
Select 'number is even' as result;
Else
Select 'number is odd' as result;
End if;
End ;
```

Example 6 –

```
Create procedure p_name(num1 int,num2 int,num3 int)
Begin
Declare result varchar(20);
If num1>=num2 and num1>=num3 then
```

```
Set result = “type your comment”;
Elseif num2>=num1 and num2>=num3 then
Set result= “type your comment 2”;
Else
Set result=”type your comment 3”;
End if;
Select result;
End ;
```

Example 7-

```
create procedure grade()
    begin
    select roll_no,total,
    case
    when total >= 80 then "A"
    when total >= 60 then "B"
    when total >= 40 then "C"
    else "FAIL"
    end as grade from marks3;
    end //
```

Example 8 –

```
create procedure grades(a int)
    begin
    declare b int;
    declare Y varchar (10);
    select total into b from marks3 where roll_no=a;
    if b>= 80 then
    set Y="A";
```

```
elseif b>=60 then
set Y="B";
elseif b>=40 then
set Y="c";
else
set Y="fail";
end if;
select a as roll_no,b as marks,Y as grade;
end//
```

Example 9 –

```
CREATE PROCEDURE sp_name(OUT countstudent INT)
BEGIN
SELECT COUNT(TOTAL) INTO countstudent FROM MARKS3;
END //
CALL sp_name(@B)//
SELECT @B//
```

Example 10 –

```
CREATE PROCEDURE name(num1 int,num2 int, OUT result INT)
BEGIN
set result =num1+num2;
END //
CALL NAME(5,5,@A)//
SELECT @A//
```

Example-

```
CREATE PROCEDURE Top5Records()
BEGIN
   SELECT *
   FROM STUDENT1
   LIMIT 5;
END//
```

Example -

```
 CREATE PROCEDURE Top5ByMarks()
 BEGIN
   SELECT *
   FROM STUDENT1
   ORDER BY marks DESC
   LIMIT 5;
   END//
```

TRIGGER

Ex1-

```
 CREATE TRIGGER trigger_name
 {BEFORE | AFTER} {INSERT | UPDATE | DELETE} ON table_name
 FOR EACH ROW
  BEGIN
 SQL statements;
 END //23
```

Ex-

```
DELIMITER //

CREATE TRIGGER before_insert_marks
BEFORE INSERT ON marks
FOR EACH ROW
BEGIN
   SET NEW.total = NEW.math + NEW.science + NEW.english;
END//
```

Ex2-

```
DELIMITER $$
CREATE TRIGGER strigger
AFTER UPDATE ON student
FOR EACH ROW
BEGIN
   INSERT INTO supdate
   VALUES (NOW());
END$$
DELIMITER ;
```

Time

```
DELIMITER //
 CREATE TRIGGER trigger_name
   AFTER INSERT  ON table_name
   FOR EACH ROW
   BEGIN
   INSERT INTO emp_audit VALUES (null, concat('A row is  inserted in
employee table at ', date_format(now(), '%d-     %m-%y%h:%i:%s %p')));
  END //
```

Time Formate

```
CREATE TRIGGER tr_k
AFTER UPDATE on student
FOR EACH ROW
BEGIN
DECLARE new_name varchar(50);
DECLARE old_name varchar(50);
set new_name = new.name;
set old_name = old.name;
INSERT into audit (id,details)VALUES(null, concat('A name
',old_name,' is replaced with name ',new_name, 'at',
date_format(now(), '%d-%m-%v %h:%i:%s %p')));
END//
```

Example-

```
CREATE TRIGGER T1
BEFORE INSERT ON STUDENT1
FOR EACH ROW
BEGIN
   IF NEW.age = 17 THEN
      SET NEW.remark = 'UNDER AGE';
   END IF;
END//
```

Example -

```
CREATE TRIGGER T1
BEFORE INSERT ON STUDENT1
FOR EACH ROW
BEGIN
    IF NEW.age = 17 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'UNDER AGE STUDENT';
```

```
    END IF;
END;
//

DELIMITER ;
```

CURSOR

```
create procedure cur25()
 begin
 declare c1 varchar(50);
 declare c2 varchar(50);
 declare cursor1 cursor for select name, address from
student1;
 open cursor1;
 fetch cursor1 into c1,c2;
 insert into S_table_name
(name,address)values(c1,c2);
 close cursor1;
 end//
```

example-

```
create procedure cur10()
 begin
```

```
 declare c1 int;
 declare c2 varchar(50);
declare done int default false;
 declare cursor1 cursor for select roll_no,
 address from student1;
declare continue handler for not found set
done=true;
 open cursor1;
lp: loop
 fetch cursor1 into c1,c2;
if (done) then leave lp;
end if;
 insert into student1 (roll_no,address)values(c1,c2);
end loop;
 close cursor1;
 end//
```

example

```
CREATE PROCEDURE cur41()
BEGIN
DECLARE c1 INT;
DECLARE c2 VARCHAR(50);
DECLARE done INT DEFAULT 0;
DECLARE counter INT DEFAULT 0;
```

```
DECLARE cursor1 CURSOR FOR SELECT roll_no,
address FROM student;
DECLARE CONTINUE HANDLER FOR NOT FOUND
SET done = 1;
OPEN cursor1;
lp: LOOP
FETCH cursor1 INTO c1, c2;
IF done OR counter >= 15 THEN
LEAVE lp;
END IF;
INSERT INTO db.student (id, name) VALUES (done, c2);
SET counter = counter + 1;
END LOOP;
CLOSE cursor1;
END//
```

example

```
CREATE PROCEDURE cur4()
BEGIN
DECLARE c1 INT;
DECLARE c2 VARCHAR(50);
DECLARE done INT DEFAULT 0;
DECLARE counter INT DEFAULT 0;
```

```
DECLARE cursor1 CURSOR FOR SELECT roll_no, address
FROM student;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET
done = 1;
OPEN cursor1;
lp: LOOP
FETCH cursor1 INTO c1, c2;
IF done OR counter >= 15 THEN
LEAVE lp;
END IF;
IF counter <= 10 THEN
SET counter = counter + 1;
ITERATE lp;
END IF;
INSERT INTO user.db.student (id, name) VALUES (c1, c2);
SET counter = counter + 1;
END LOOP;
CLOSE cursor1;
END;
//
```

Update salary

```
CREATE PROCEDURE cur4()
BEGIN
    DECLARE v_emp_id INT;
    DECLARE v_salary DECIMAL(10,2);
    DECLARE done INT DEFAULT 0;
```

```
    DECLARE cursor1 CURSOR FOR
        SELECT emp_id, salary FROM employees;
    DECLARE CONTINUE HANDLER FOR NOT FOUND
SET done = 1;
    OPEN cursor1;
    read_loop: LOOP
        FETCH cursor1 INTO v_emp_id, v_salary;
        IF done = 1 THEN
            LEAVE read_loop;
        END IF;
        UPDATE employees
        SET salary = v_salary * 1.20
        WHERE emp_id = v_emp_id;
    END LOOP;
    CLOSE cursor1;
END;
//
```

Example

```
Create procedure holi()
begin
declare c1 int;
declare c2 varchar(100);
declare rang int default 0;
```

```
declare blue int default 0;
declare cur1 cursor for select roll_no,
name from student;
declare continue handler for
not found set rang=1;
open cur1;
lp: loop
fetch cur1 into c1,c2;
set blue=blue+1;
if blue<=8 then iterate lp;

end if;
if rang then leave lp;
end if;
insert into audit_t values(c1,c2);
end loop;
close cur1;
end //
```

Example

```
CREATE PROCEDURE UpdateSalaries
AS
BEGIN
    DECLARE @EmpID INT, @Salary DECIMAL(10, 2);
```

```
    DECLARE cursor_Emp CURSOR FOR

    OPEN cursor_Emp;

    FETCH NEXT FROM cursor_Emp INTO @EmpID,
@Salary;

    WHILE @@FETCH_STATUS = 0
    BEGIN

        UPDATE Employees
        SET Salary = @Salary * 1.2
        WHERE CURRENT OF cursor_Emp;

        FETCH NEXT FROM cursor_Emp INTO @EmpID,
@Salary;
    END

    CLOSE cursor_Emp;
   DEALLOCATE cursor_Emp;
END;
```

Example

```
DELIMITER //

CREATE PROCEDURE cur3()
BEGIN
   DECLARE c1 INT;
```

```
    DECLARE c2 VARCHAR(50);
    DECLARE done INT DEFAULT 0;
    DECLARE counter INT DEFAULT 0;
    DECLARE cursor1 CURSOR FOR SELECT roll_no, address
FROM student;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET
done = 1;
    OPEN cursor1;
    lp: LOOP
        FETCH cursor1 INTO c1, c2;
        IF done OR counter >= 15 THEN
            LEAVE lp;
        END IF;
        IF counter <= 10 THEN
            SET counter = counter + 1;
            ITERATE lp;
        END IF;
        INSERT INTO triggers.student (id, name) VALUES (c1,
c2);
        SET counter = counter + 1;
    END LOOP;

    CLOSE cursor1;
END; //
```

Example

```
DELIMITER ;
CREATE PROCEDURE cur12()
    BEGIN
     DECLARE c1 INT;
```

```
        DECLARE c2 VARCHAR(50);
        DECLARE done INT DEFAULT 0;
        DECLARE counter INT DEFAULT 0;
        DECLARE cursor1 CURSOR FOR SELECT roll_no,
address FROM student;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET
done = 1;
        OPEN cursor1;
        lp: LOOP
            FETCH cursor1 INTO c1, c2;
            IF done OR counter = 9 THEN LEAVE lp;
            END IF;
            SET counter = counter + 1;
            IF counter < 5 THEN
            ITERATE lp;

        END IF;
        INSERT INTO triggers.student (id, name) VALUES (c1,
c2);
        END LOOP;
        CLOSE cursor1;
    END;
    //
```

Example

```
DELIMITER //

CREATE PROCEDURE cur3()
```

```
    BEGIN
        DECLARE c1 INT;
        DECLARE c2 VARCHAR(50);
        DECLARE done INT DEFAULT 0;
        DECLARE counter INT DEFAULT 0;
        DECLARE cursor1 CURSOR FOR SELECT roll_no,
address FROM student;
DECLARE CONTINUE HANDLER FOR NOT FOUND
SET done = 1;
  OPEN cursor1;
lp: LOOP
 FETCH cursor1 INTO c1, c2;
  IF done THEN
   LEAVE lp;
   END IF;
 SET counter = counter + 1;
  IF counter < 11 THEN
  ITERATE lp;
  END IF;
  IF counter > 15 THEN
  LEAVE lp;
END IF;
INSERT INTO triggers.student (id, name) VALUES (c1,
c2);
        END LOOP;
        CLOSE cursor1;
    END//
```