Write Query for create table
BRANCH (BID INTEGER, BRNAME CHAR (30), BRCITY CHAR (10))
CUSTOMER (CNO INTEGER, CNAME CHAR (20), CADDR CHAR (35), CITY CHAR(20))
LOAN_APPLICATION (LNO INTEGER, LAMTREQUIRED MONEY, LAMTAPPROVED MONEY, L_DATE DATE)

The relationship is as follows:
BRANCH, CUSTOMER, LOAN_APPLICATION are related with ternary relationship.
TERNARY (BID INTEGER, CNO INTEGER, LNO INTEGER).

Create table branch (bid int primary key, Brname varchar(30) not null, Brcity varchar(10) default “Pune”)	Create table customer(cno int primary key, Cname varchar(20) not null, Caddr varchar(35), City varchar(20))	Create table loan_application(Lno int primary key, Lamtrequired decimal(9,2), Lamtapproved decimal(9,2), L_date date)	Create table ternary(Brid int, Cno int, lno int, constraint br_f1 foreign key(brid) references branch(bid) on update cascade, constraint cno_f1 foreign key(cno) references customer(cno) on update cascade, constraint lno_f1 foreign key(lno) references loan_application(lno) on update cascade, constraint cno_pk primary key(lno,cno))

1. Find all employees who have not taken any loan

```
Select *
From customer c
Where not exists (select *
                  From ternary t
                  Where t.cno=c.cno)
```

To create a table using auto_increment

- In a table there can be only one auto_increment column
- It has to be defined as primary key
- While inserting the value use default keyword as a place holder in insert statement
- To change the starting value use alter table
- The new value will be always max value+1
- In case needed you can enter values manually also, but not syuggestable.

Create table mytable(

Id int auto_increament primary key,

Name varchar(20),

Address varchar(20));

insert into mytable values(default,'xxxx','Pune');

Query OK, 1 row affected (0.00 sec)

mysql> insert into mytable(name,address) values('Rajat','Pune');

Query OK, 1 row affected (0.01 sec)

To change the start value for auto_increment

ALTER TABLE mytable AUTO_INCREMENT=1001

Alter table

To modify table definition we need to use alter table

1. To add, delete and modify column
2. To rename table
3. Add or delete constraint

Add new column in the table

1. If the table contains data then while adding new column you cannot put not null constraint, so to add not null constraint
    a. Add new column
    b. Update all null values
    c. Then modify column to add not null constraint
2. But in mysql it allows to add a new column with not null constraint, then it assigns value 0 to all rows instead of null

ALTER TABLE table_name

ADD new_column_name column_definition

[FIRST | AFTER column_name];

Add multiple columns in table

Syntax

---------The syntax to add multiple columns in a table in MySQL (using the ALTER TABLE statement) is:

```
ALTER TABLE table_name
  ADD new_column_name column_definition
    [ FIRST | AFTER column_name ],
  ADD new_column_name column_definition
    [ FIRST | AFTER column_name ],
  ...
;
```

Add column	alter table mytable add lname varchar(20) after name;	alter table mytable add mobile int;
Delete column	alter table mytable Drop column lname	
Modify column	alter table mytable modify name varchar(50) not null	
Rename column	alter table mytable rename column name to fname	
Add constraint	alter table child_mytable add constraint child_pk primary key(child_id); alter table child_mytable add constraint child_pk foreign key(paret_id) references mytable(id)	To add table level constraint

Drop constraint	---- to delete the primary key alter table child_mytable drop primary key; -----To delete foreign key alter table child_mytable drop constraint child_fk;	To drop table level constraint
Rename the table	Alter table mytable Rename to newmytable	

DML--- to delete all rows

Delete from newmytable

DDL --- to delete all rows

Truncate table newmytable

Delete	Truncate
It is DML statement	It is DDL statement, so autocommit
Rollback is possible	Rollback is not possible, because it is auto committed
Where condition can be used, so only few rows can be deleted	Where condition cannot be used, so always used for deleting all rows

TCL ---transaction control language

Rollback, commit, savepoint

commit	It is used to make changes done by insert, update and delete statement permanent.
rollback	It is used to undo the changes done by insert, update and delete statement upto previous commit
Rollback to A	It is used to undo the changes done by insert, update and delete statement upto savepoint A
savepoint	To add markers in between

10 rows are available commit Insert ------ rollback will be upto this line Insert Insert Deleted 2 rows Rollback ------- 10 rows will remain in the table	10 rows are available commit Insert Insert DDL statement Insert ------ rollback will be upto this line Deleted 2 rows Rollback ------ 12 rows will be in the table	10 rows are available commit Insert Insert Insert insert Deleted 2 rows commit ------ rollback will be upto this line Rollback ------ 12 rows will be in the table
10 rows are available commit Insert Insert Savepoint A Insert Insert Savepoint B Deleted 2 rows Rollback to B ------- 10 rows will remain in the table		

Nested Query

- If you are writing a query inside another query, then it is called as nested query
- In nested query, child query gets executed only once, and then parent query will get executed
- If nested query is used in create table, then the new table will contain the columns same as result of nested query and also copy the result of nested query.
    - Columns name in new table will be same as the nested query's columns
    - Constraint of table will not get copied
- If child query is dependent on patent query, then it is called as co-related query
- In corelated query child query will get executed n times if parent query table contains n rows

1. Find all employees who are working in smith's department

```
Select * from emp
Where deptno=( Select deptno
                From emp
                Where ename='SMITH'
                );
```

2. Find all employees with salary > blake's salary

```
select *
from emp
where sal>(select sal
                from emp
                where ename='BLAKE')
```

3. To find all employees with sal > avg salary of dept 10

```
Select *
From emp
Where sal>(select avg(sal)
                From emp
                Where deptno=10)
```

4. To find all employees who are working in accounting department.

```
select * from emp
  -> where deptno=(select deptno
  -> from dept
  -> where dname='ACCOUNTING');
```

5. To display all employees with salary >= smith's salary and <= blake's salary.

```
Select *
From emp
Where sal  between  (select sal
                From emp
                Where ename='SMITH') and (select sal
                                                From emp
                                                Where ename='BLAKE')
```

6. Find all employees whose salary is = either SMITH or BLAKE.

```
Select *
From emp
Where sal in (select sal
                From emp
                Where ename in ('SMITH','BLAKE))
```

7. To create a table employee_history with same columns and data available in emp table

```
Create table employee_history
As
(select empno,ename,sal,sal+ifnull(comm) from emp);
```

8. To create a table employee_history with same columns and data available in emp table for employees working as analyst

```
Create table employee_history
As
(select * from emp
Where job='ANALYST');
```

9. To create an empty table employee_history with same columns available in emp table

```
Create table employee_history
As
(select * from emp
Where 1=2)
```

10. To create an empty table employee_history with only empno,ename,sal columns available in emp table

```
Create table employee_history
As
(select empno,ename,sal from emp
Where 1=2 );
```

11. Insert values from emp table to existing emp_history table

```
Insert into emp_history
(select *
From emp
Where empno >7500;
)
```

12. Delete all employees who are working in Allen's department.

```
Delete from emp
Where deptno=(select deptno
                From (select * from emp) e
                Where ename='ALLEN')
```

13. Update smith's salary with blake's salary

```
Update emp
Set sal=(select sal
        from (select * from emp ) e
        where ename='BLAKE')
where ename='SMITH';
```

14. Find all employees with salary > avg salary of their own department

```
Select *
```

```
From emp e
Where sal >(select avg(sal)
        From emp m
        Where m.deptno=e.deptno)
```

15. Find all departments in which no employees are there

```
Select *
From dept d
Where  not  exists    (select *
                From emp e
                Where e.deptno=d.deptno)
```

14. find all employees whose sal < avg salary of ALLEN's department.

```
Select *
From emp
Where sal < (select avg(sal)
                From emp
                Where deptno=(select deptno
                                From emp
                                Where ename='ALLEN'))
```

16. Select all employees with salary > avg salary of either dept 10 or dept 20

```
Select *
From emp
Where sal > any   (select avg(sal)
                From emp
                Where deptno in (10,20))
```