Table: Movies

| Id | Title | Director | Year | Length_minutes | Release date |
|---|---|---|---|---|---|
| 1 | Toy Story | John Lasseter | 1995 | 81 | |
| 2 | A Bug's Life | John Lasseter | 1998 | 95 | |
| 3 | Toy Story 2 | John Lasseter | 1999 | 93 | |
| 4 | Monsters, Inc. | Pete Docter | 2001 | 92 | |
| 5 | Finding Nemo | Andrew Stanton | 2003 | 107 | |
| 6 | The Incredibles | Brad Bird | 2004 | 116 | |

movieid is primary key

year >1970

length_minutes>15 min and < 240 mins

Release_date  >1990-12-31  default ‘1991-01-01’

```
create table movie(movieid int primary key,
title varchar(20),
director varchar(20),
year int chek(year>1990),
length_min int check(length_min between 15 and 240)
release_date date check(release_date>’1990-12-31’) default ‘1991-01-01’)

insert into movie values(1,'Toy story','John Lasseter',1995,81,'1995-03-02');
insert into movie values(2,'A Bugs Life','John Lasseter',1998,95,'1998-03-02');
```

Table: Boxoffice

| Movie_id | Rating | Domestic_sales | International_sales |
|---|---|---|---|
| 5 | 8.2 | 380843261 | 555900000 |
| 14 | 7.4 | 268492764 | 475066843 |
| 8 | 8 | 206445654 | 417277164 |
| 12 | 6.4 | 191452396 | 368400000 |
| 3 | 7.9 | 245852179 | 239163000 |
| 6 | 8 | 261441092 | 370001000 |

|  |  |  |  |
|---|---|---|---|
| 9 | 8.5 | 223808164 | 297503696 |

movie id ---primary key

movieid foreign key

rating >=1 and<=10

```
create table boxoffice(
movieid int primary key,
rating float(5,2) check(rating between 1 and 10),
domestic_sales int,
international_sales int,
constraint fk_mid foreign key(movieid) references movie(movieid))
insert into boxoffice values(1,7.8,380843261,55559988);
insert into boxoffice values(2,9,380843261,55559988);
```

to create table player (player_id, pname,speciality,date_of_joining,num_matches,team_id)

to create table team (team_id, tname,player_num)

to create table matches(match_id, team1_id,team2_id,match_date,winner,man_of_the match)

winner should be either team1_id or team2_id

```
create table team(
tid int primary key,
tname varchar(50),
player_num int check(player_num>0))

create table player(
player_id int primary key,
pname varchar(20),
speciality enum(‘bowler’,’batsman’,’allrounder’,’wicket_keeper’),
date_of_joining date,
num_matches int,
team_id int,
```

constraint fk_tid foreign key(team_id) references team(tid)

on delete set null

on update cascade);

create table matches(

match_id int primary key,

team1 int,

team2 int,

match_date date,

winner int,

man_of_match int,

constraint fk_pid foreign key(man_of_match) references player(player_id),

constraint fk_team1 foreign key(team1) references team(tid) ,

constraint fk_team2 foreign key(team2) references team(tid) ,

constraint chk_win check(winner in (team1,team2))

)

1. Display  first day of year and last day of year
   select dayname(concat(year(curdate()),'-01-01')) "first day", concat(year(curdate()),'-12-31') "last day";

Nested Query

if the output of the query is dependent on other query, then use nested query.

usually we need nested queries either in from clause or where clause to check conditions

nested queries are of 2 types

1. simple query
   a. if child query is independent, can execute by itself then it is called as simple query.
2. corelated query
   a. if child query is dependent on parent query, then it is called as corelated query.
   b. it gets executed once for each row in the outer query.
   c. data can be passed from parent query to child query but vice versa is not possible

1. list all employees who works in smith's department
   select deptno from emp where ename='SMITH'

select *
from emp
where deptno=( select deptno from emp where ename='SMITH')

2. list all employees with salary > jones sal
   select *
   from emp
   where sal>(select sal from emp where ename='Jones')

3. list all employees with salary > jones sal or smith's sal

select *

from emp

where sal >all  (select sal from emp where ename in ('Jones','smith'))

4. list all employees who works either in smiths department or king's department
   select * from emp
   where deptno in (select deptno from emp where ename in('smith','king'))

5. to find all employees with sal > maximum sal of smiths department

select * from emp

where sal > (select max(sal) from emp

where deptno=(select deptno from emp where ename='SMITH')

)

6. to find all employees with salary < avg sal of accounting department
   select * from emp
   where sal< (select avg(sal) from emp where deptno=(select deptno from dept where dname='ACCOUNTING'))
7. to find all employees with salary < avg sal of either accounting or sales department
   select * from emp
   where sal< all (select avg(sal) from emp where deptno in(select deptno from dept where dname in ('ACCOUNTING','sales'))
   group by deptno
   )

8. list all employees whose sal < avg salary of its own department.
   select *
   from emp e
   where sal < (select avg(sal) from emp m where m.deptno=e.deptno)
9. list all employees whose sal < avg sal of all  employess working under the same mgr
   select * from emp e
   where sal<(select avg(sal) from emp m where m.mgr=e.mgr)

10. 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)
    order by deptno

11. find all employees who are  mgr of som employee
    select empno,ename

from emp e

where exists ( select * from emp m where m.mgr=e.empno)

12. display all employees with job manager and if sal > avg sal  of its own department
    select * from emp e
    where job='manager' and sal >(select avg(sal) from emp m where m.deptno=e.deptno)

13. display all employees whose sal > smith's sal and < ward sal
    select *
    from emp
    where sal between (select sal from emp where ename='smith' ) and  (select sal from emp where ename='WARD')
14. to update sal to salary of jones+1000 for employee smith
    update emp
    set sal=(select sal from emp e where e.ename='jones')+1000
    where ename='smith'
15. update sal to jones sal+500 and job to 'king's job for ward
    update emp
    set sal=(select sal from (select * from emp) e where e.ename='jones')+500,job=(select job from (select * from emp) k where k.ename='KING')
    where ename='ward'

we may use nested query in create table

| | |
|---|---|
| 1. create table emp_10 with all the employee working in department 10<br>create table emp_10<br>as<br>(select empno,ename from emp<br>where deptno=10) | It will create a table emp_10, which has 2 columns empno,ename and it will also get data for all employees with deptno 10 |
| 1. create table emp_20 table same as emp but no records should be added in it<br>create table emp_20<br>as<br>(select * from emp where 1=2) | It will create a empty table emp_20, which has all columns emp table |
| 2. insert into CLERK_tab as | |

| | |
|---|---|
| (select * from emp where job='CLERK') | |

use following table to solve the questions

| fid | fname | skill |
|---|---|---|
| 100 | Narendra | Database |
| 101 | sonali | java |
| 102 | Sarika | security |

course

| cid | cname | description | rid | fid |
|---|---|---|---|---|
| 121 | DAC | 6 months full stack | 10 | 100 |
| 131 | DBDA | Data anyasts | 11 | null |
| 141 | DTISS | security | null | null |
| | | | | |

Room

| rid | rname | location |
|---|---|---|
| 10 | Lotus | 1st floor |
| 11 | Rose | 2nd floor |
| 12 | jasmin | 1st floor |
| 13 | Mogra | 2nd floor |

1. find which room can be assigned to DTISS
   select * from room r
   where not exists (select * from course c where r.rid=c.rid)

2. find which faculty can be assigned to DTISS, faculty needs security skill for it
   select * from faculty f
   where skill='security' and not exists (select * from course c where f.fid=c.fid)

3. find all rooms which are assigned to some course.
   select * from room r
   where exists select * from course c where r.rid=c.rid

4. list all faculties who are assigned to some course
   select * from faculty f
   where exists (select * from course c where f.fid=c.fid)