Nina Abdou (40003667), Maureen Adelson (40030871), Lu Han (40069298) and Weiwei Xiao (40197802)

COMP 5531/4 Bipin C. Desai Winter 2022 Assignment #4

Q1

1.1

```
CREATE TABLE Student
(SNUM INTEGER,
SName CHAR(50),
Major CHAR(50),
level CHAR(50),
age INTEGER,
PRIMARY KEY (SNUM))

CREATE TABLE Professor
(PID INTEGER,
FName CHAR (50),
DeptID INTEGER,
PRIMARY KEY (PID))

CREATE TABLE Class
(Name CHAR(50),
meets_at TIME,
room CHAR(50),
fid INTEGER,
PRIMARY KEY(Name),
```

```
FOREIGN KEY (fid) REFERENCES Professor (PID))


CREATE TABLE Enrolled

( SNUM INTEGER,

CName CHAR(50),

PRIMARY KEY (SNUM,CName),

FOREIGN KEY (SNUM) REFERENCES Student(SNUM) ,

FOREIGN KEY (CName) REFERENCES Class(Name))
```

1.2

a)
```
CREATE TABLE ENROLLED (
SNUM INTEGER,
CName CHAR(50),
PRIMARY KEY (SNUM,CName),
FOREIGN KEY (SNUM) References Student(SNUM),
FOREIGN KEY (CName) References Class(Name),
CHECK (( SELECT COUNT (E.SNUM) FROM
ENROLLED E
GROUP BY E.CName) >= 15),
CHECK (( SELECT COUNT (E.SNUM) FROM
Enrolled E
GROUP BY E.CName) <= 30));
```

b) This constraint is implied by the primary and foreign key constraint. In the Class relation, a room cannot be declared without having a class name (primary key) associated with it.

c)
```
CREATE ASSERTION TWOCOURSES
CHECK
(( SELECT COUNT (*) FROM
Professor P, Class C
WHERE P.PID = C.FID
GROUP BY C.FID HAVING COUNT (*) < 2 ) = 0 )
```

d)
```
CREATE TABLE Class
(Name CHAR(50),
meets_at TIME,
room CHAR(50),
fid INTEGER,
PRIMARY KEY(Name),
FOREIGN KEY (fid) REFERENCES Professor,
CHECK((
SELECT C.room,C.meets_at
FROM Class C
GROUP BY C.room,C.meets_at HAVING COUNT(*)>1)) = 0)
```

e)
```
CREATE ASSERTION DIFFERENTROOMS
CHECK (( SELECT COUNT (*)
FROM Professor P1,Professor P2, Class C1, Class C2
WHERE P1.PID = C1.fid
AND P2.PID = C2.fid
AND C1.room = C2.room
AND P1.DeptID <> P2.DeptID) = 0)
```

Q2

2.1
```
CREATE TABLE Emp (
        EID INT NOT NULL,
        EName STRING,
        Age INT,
        Salary REAL,
        PRIMARY KEY (EID),
        CHECK (Salary => 5000)
```

2.2
```
CREATE ASSERTION managerIsEmployee
        CHECK ((SELECT COUNT(*)
                FROM Dept D
                WHERE D.ManagerID NOT IN (SELECT * FROM Emp))
                = 0)
```

2.3
```
CREATE TABLE Works (
    EID INT,
    DID INT,
    Pct_time INT,
    PRIMARY KEY (EID, DID),
    CHECK (( SELECT COUNT (W.EID)
             FROM Works W
             GROUP BY W.EID
              HAVING Sum(Pct_time) > 100) = 100))
```

2.4
```
CREATION ASSERTION managerSalary
CHECK (SELECT E.EID
          FROM Emp E, Emp M, Works W, Dept D
          WHERE E.EID = W.EID
          AND W.DID = D.DID
          AND D.ManagerID = M.EID
          AND E.Salary > M.Salary))
```

2.5
```
CREATE TRIGGER raise AFTER UPDATE ON Emp
WHEN old.Salary < new.Salary
FOR EACH ROW
BEGIN
   UPDATE Emp M
   SET M.Salary = new.Salary
   WHERE M.salary < new.salary
   AND M.EID IN (SELECT D.ManagerID
                 FROM Emp E, Works W, Dept D
                 WHERE E.EID = new.EID
                 AND W.DID = D.DID);

END
```

2.6
```
CREATE TRIGGER raise AFTER UPDATE ON Emp
WHEN old.salary < new.salary
 FOR EACH ROW
DECLARE
   raise REAL;
BEGIN
   raise = new.salary -old.salary;
```

```
UPDATE Emp M
SET M.salary = new.salary
WHERE M.salary < new.salary
AND M.EID IN (SELECT D.ManagerID
                FROM Emp E, Works W, Dept D
                WHERE E.EID = new.EID
                AND E.EID = W.EID
                AND W.DID = D.DID);

UPDATE Dept D
SET D.budget = D.budet + raise
WHERE D.DID IN ( SELECT W.DID
                    FROM Emp E, Works W, Dept D
                    WHERE E.EID = new.eid
                    AND E.EID = W.EID
                    AND D.DID = W.DID
                    AND D.budget < (SELECT Sum(E2.salary)
                                    FROM Emp E2, Works W2
                                    WHERE E2.EID = W2.EID
                                    AND W2.dept = D.DID ));
END
```

Q3

3.1 Select FLNO from FLIGHT where "FROM" ='YUL';

3.2 Select distinct "TO" from FLIGHT ;

3.3 This query can't be written as there is no data for connecting flights and the time gaps between connecting flights.

3.4 SELECT min(FlightDuration) from FLIGHT where "FROM"='YUL" and "TO" = "CCU";

Q4

q1

```
CREATE DATABASE A4Q4;
Create table Employees(
eID int,
dID int not null,-- will alter dID to a foreign key preference Departments(dID) later
svID int , -- will alter svID to NOT NULL later
```

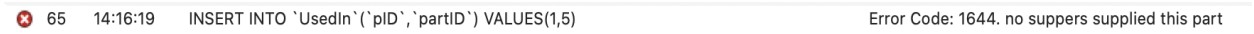
```
salary int , -- will alter salary to NOT NULL later
PRIMARY KEY (eID),
Foreign key (svID) REFERENCES Employees(eID))
ENGINE=InnoDB ;



Create table Departments(
dID int,
mID int,-- will alter mID to NOT NULL later
PRIMARY KEY (dID),
Foreign key(mID) REFERENCES Employees(eID))
ENGINE=InnoDB;


ALTER TABLE `Employees` ADD CONSTRAINT fk_dID FOREIGN KEY (dID) REFERENCES
Departments(dID);
select * from Departments;
INSERT INTO `Departments` (`dID`) VALUES ('1'), ('2');
-- ceo's eid = 1 in dpt2
INSERT INTO `employees` (`eID`,`dID`,`svID`,`salary`) VALUES ('1','2',1,1000000),
('2','1',1,300000), ('3','1',2,250000), ('4','2',1,400000);
 update Departments set mID = 1 where dID = 2;
 update Departments set mID = 2 where dID = 1;
Alter table Departments MODIFY mID int not null;
Alter table Employees MODIFY svID int not null;
Alter table Employees MODIFY salary int not null;

Create table Projects(
pID int,
PRIMARY KEY (pID)
)ENGINE=InnoDB;
INSERT INTO `Projects` (`pID`) VALUES ('1'), ('2'),('3'), ('4'), ('5'), ('6'), ('7');



Create table Assigned(
eID int,
pID int,
workhours int not null default '0',
PRIMARY KEY (eID, pID),
Foreign key(eID) REFERENCES Employees(eID),
Foreign key(pID) REFERENCES Projects(pID)
)ENGINE=InnoDB ;
```

```
select * from Assigned;

Create table Suppliers(
SupplierID int,
PRIMARY KEY (SupplierID)
)ENGINE=InnoDB;
INSERT INTO `Suppliers` (`SupplierID`) VALUES ('1'), ('2'), ('3'), ('4'), ('5');

select * from Suppliers;

Create table Parts(
partID int,
PRIMARY KEY (partID)
)ENGINE=InnoDB;
INSERT INTO `parts` (`partID`) VALUES ('1'), ('2'), ('3'), ('4'), ('5') ;

select * from Parts;

Create table UsedIn(
pID int,
partID int,
PRIMARY KEY (pID, partID),
Foreign key(pID) REFERENCES Projects(pID),
Foreign key(partID) REFERENCES Parts(partID)
)ENGINE=InnoDB;

select * from UsedIn;

Create table Supplied(
SupplierID int,
partID int,
PRIMARY KEY (SupplierID, partID),
Foreign key(SupplierID) REFERENCES Suppliers(SupplierID),
Foreign key(partID) REFERENCES Parts(partID)
)ENGINE=InnoDB;
INSERT INTO `Supplied` (`SupplierID`, `partID`) VALUES ('1', '2'), ('1', '3'), ('2', '3'), ('5', '1');

select * from Supplied;

—----------------------------------------------------------------------------------------------------

-- q2
delimiter //
```

```
create trigger Check_supCount before insert on UsedIn
for each row
begin
DECLARE howmany int;
set howmany=(SELECT count(*) from supplied where partID = NEW.partID);
if(howmany<1) THEN
SIGNAL SQLSTATE '45000'
set MESSAGE_TEXT = 'no suppers supplied this part';
end if;
end;
//
```

INSERT INTO `UsedIn`(`pID`,`partID`) VALUES(1,5);

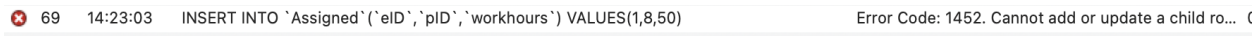
Error Code: 1644. no suppers supplied this part



-- q3

INSERT INTO `Assigned`(`eID`,`pID`,`workhours`) VALUES(1,8,50);

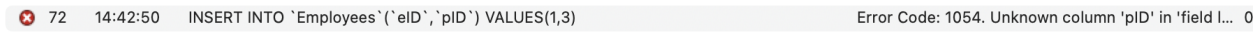
Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`a4q4`.`assigned`, CONSTRAINT `assigned_ibfk_2` FOREIGN KEY (`pID`) REFERENCES `projects` (`pID`))



-- q4

INSERT INTO `Employees`(`eID`,`dID`) VALUES(1,3);

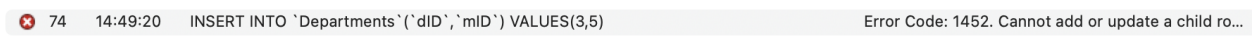
Error Code: 1054. Unknown column 'pID' in 'field list'



-- q5

INSERT INTO `Departments`(`dID`,`mID`) VALUES(3,5);

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`a4q4`.`departments`, CONSTRAINT `departments_ibfk_1` FOREIGN KEY (`mID`) REFERENCES `employees` (`eID`))



-- q6

```
delimiter //
create trigger checkSupervisor
BEFORE INSERT ON employees
FOR EACH ROW
begin
declare salary_sup int;
declare managerID_Dept int;
declare dept_sup int;
set salary_sup = (select salary from Employees where eID=NEW.svID);
set managerID_Dept=(select mID from Departments where dID = NEW.dID);
set dept_sup = (select dID from Employees where eID=NEW.svID);
if((NEW.eID <> '0')AND(NEW.salary*1.1>salary_sup))then
SIGNAL SQLSTATE '45000'
set MESSAGE_TEXT = 'there is some error on salary';
ELSEIF((NEW.eID<>managerID_Dept)AND(NEW.dID<>dept_sup))THEN
SIGNAL SQLSTATE '45000'
set MESSAGE_TEXT = 'employee and supervisor shuld be in same Department';
end if;
end;
//
```

-- eid5 has the same salary as his supervisor eid2.
-- does not meet [the salary of a supervisor is at least 10% higher any employees the s/he supervises].

```
INSERT INTO `Employees`(`eID`, `dID`, `svID`, `salary`) VALUES (5,1,2,300000);
```

Error Code: 1644. there is some error on salary

```
81  15:25:58  INSERT INTO `employees`(`eID`, `dID`, `svID`, `salary`) VALUES (5,1,2,300000)  Error Code: 1644. there is some error on salary
```

-- eid5 is in dID2, but it's supervisor eid3 is in dID1.
-- does not meet [The supervisor of all employees, except the manager of the department, must be in the same department].

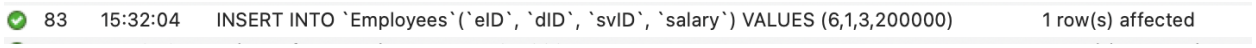
```
INSERT INTO `Employees`(`eID`, `dID`, `svID`, `salary`) VALUES (5,2,3,200000);
```

Error Code: 1644. employee and supervisor shuld be in same Department

```
101  21:44:58  INSERT INTO `Employees`(`eID`, `dID`, `svID`, `salary`) VALUES (5,2,3,200000)  Error Code: 1644. employee and supervisor shuld...
```

-- successful example
-- eid6's salary is 200000 and it's supervisor eid3's salary is 250000.
-- eid6 is in dID1 and it's supervisor eid3 is in dID1.

```
INSERT INTO `Employees`(`eID`, `dID`, `svID`, `salary`) VALUES (6,1,3,200000);
```



-- q7

```
delimiter //
Create trigger checkoverwork
after insert on Assigned
for each row
BEGIN
Declare howmany int;
Declare workhour int;
set howmany=(select count(*) from Assigned where eID = NEW.eID);
set workhour=(select SUM(workhours) from Assigned where eID = NEW.eID order by
eID);
if(howmany>5) then
SIGNAL SQLSTATE '45000'
set MESSAGE_TEXT = 'Too many projects for this employee!';
ELSEIF(workhour>70) then
SIGNAL SQLSTATE '45001'
set MESSAGE_TEXT = 'Too many workhours for this employee!';
end if;
end;
//
```

-- let eID1 be assigned to 3 more projects who has alrealy be assigned to 2. the total projects that eID has will the maximum 5

```
INSERT INTO `Assigned` (`eID`, `pID`, `workhours`) VALUES ('1', '2', '10'), ('1', '4', '15'), ('1',
'1','15');
```

-- assign one more to eID1, cause error['Too many projects for this employee!']

```
INSERT INTO `Assigned` (`eID`, `pID`, `workhours`)  VALUES ('1', '6', '7');
```

Error Code: 1644. employee and supervisor shuld be in same Department

-- assign a 700 workhours project to eID2, cause error['Too many workhours for this employee!']

```
INSERT INTO `Assigned` (`eID`, `pID`, `workhours`)  VALUES ('2', '6', '700');
```

Error Code: 1644. Too many workhours for this employee!