

# DBMS LAB FILE(Exp: 1-8)

---

## PRACTICAL ASSIGNMENT- 8

- Name: Hardik Arora
- Branch : Btech - CS
- Program: AIML
- University Roll No. : 2215500071
- Section: 2AC
- Class Roll No. : 28

---

```
CREATE DATABASE P6;  
USE P6;
```

```
CREATE TABLE IF NOT EXISTS Student (  
    sID INT PRIMARY KEY,  
    sName VARCHAR(50),  
    GPA FLOAT,  
    sizeHS INT NOT NULL,  
    DoB VARCHAR(50)  
);
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('123', 'Amy', '3.9',  
'1000', '1996-06-26');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('234', 'Bob', '3.6',  
'1500', '1995-04-07');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('345', 'Craig', '3.5',  
'500', '1995-02-04');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('456', 'Doris', '3.9',  
'1000', '1997-07-24');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('567', 'Edward', '2.9',  
'2000', '1996-12-21');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('678', 'Fay', '3.8',  
'200', '1996-08-27');  
  
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('789', 'Gary', '3.4',  
'800', '1996-10-08');  
  
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('987', 'Helen', '3.7',  
'800', '1997-03-27');  
  
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('876', 'Irene', '3.9',  
'400', '1996-03-07');  
  
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('765', 'Jay', '2.9',  
'1500', '1998-08-08');  
  
INSERT INTO student (sID, sName, GPA, sizeHS, DoB) VALUES ('654', 'Amy', '3.9',  
'1000', '1996-05-26');  
  
INSERT INTO student (sID, sName, GPA, sizeHS, DoB) VALUES ('543', 'Craig', '3.4',  
'2000', '1998-08-27');  
  
SELECT * FROM student;
```

localmysql: SELECT \* FROM st... X ...

sID	sName	GPA	sizeHS	DoB
abc Filter...	abc Filter...	abc Filter...	abc Filter...	abc Filter...
123	Amy	3.9	1000	1996-06-26
234	Bob	3.6	1500	1995-04-07
345	Craig	3.5	500	1995-02-04
456	Doris	3.9	1000	1997-07-24
543	Craig	3.4	2000	1998-08-27
567	Edward	2.9	2000	1996-12-21
654	Amy	3.9	1000	1996-05-26
678	Fay	3.8	200	1996-08-27
765	Jay	2.9	1500	1998-08-08
789	Gary	3.4	800	1996-10-08
876	Irene	3.9	400	1996-03-07
987	Helen	3.7	800	1997-03-27

```

CREATE TABLE IF NOT EXISTS College(
    cName VARCHAR(50) PRIMARY KEY,
    State VARCHAR(50),
    enrollment INT NOT NULL
);

INSERT INTO college(cName, State, enrollment) VALUES('Stanford', 'CA', '15000');
INSERT INTO college(cName, State, enrollment) VALUES('Berkeley', 'CA', '36000');
INSERT INTO college(cName, State, enrollment) VALUES('MIT', 'MA', '10000');
INSERT INTO college(cName, State, enrollment) VALUES('Cornell', 'NY', '21000');
INSERT INTO college(cName, State, enrollment) VALUES('Harvard', 'MA', '50040');
SELECT * FROM college;

```

localmysql: SELECT * FROM co... X		
cName	State	enrollment
abc Filter...	abc Filter...	abc Filter...
Berkeley	CA	36000
Cornell	NY	21000
Harvard	MA	50040
MIT	MA	10000
Stanford	CA	15000

```

CREATE TABLE IF NOT EXISTS Applied(
    sID INT NOT NULL,
    cName VARCHAR(50) NOT NULL,
    major VARCHAR(50) NOT NULL,
    decision VARCHAR(1) NOT NULL
);

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Stanford', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Stanford', 'EE', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Berkeley', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Cornell', 'EE', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('234', 'Berkeley', 'biology', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'MIT', 'bioengineering', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'bioengineering', 'N');

```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'EE', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('678', 'Stanford', 'history', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('987', 'Stanford', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('987', 'Berkeley', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('876', 'Stanford', 'CS', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('876', 'MIT', 'biology', 'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('876', 'MIT', 'marine biology', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('765', 'Stanford', 'history', 'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Stanford', 'history', 'N');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Cornell', 'history', 'N');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Cornell', 'psychology', 'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('543', 'MIT', 'CS', 'N');

SELECT * FROM applied;
```

localmysql: SELECT * FROM ap... X			
sID	cName	major	decision
abc Filter...	abc Filter...	abc Filter...	abc Filter...
123	Stanford	CS	Y
123	Stanford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Stanford	history	Y
987	Stanford	CS	Y
987	Berkeley	CS	Y
876	Stanford	CS	N
876	MIT	biology	Y
876	MIT	marine biology	N
765	Stanford	history	Y
765	Stanford	history	N
765	Cornell	history	N
765	Cornell	psychology	Y

Write SQL queries for the following:

Q1. As we need to notify in the system the birthday of each student, kindly create an index DoBIndex on the column DoB of the Student table.

```
CREATE INDEX DoBIndex ON Student(DoB);
```

- Q2. Which index would be more suitable for the major in Apply? Create a Bitmap Index named MAJORindex.

```
CREATE BITMAP INDEX MAJORindex ON Apply(major);
```

-- Q3. Remove the index on the DoB column.

```
DROP INDEX DoBindex;
```

-- Q4. Create a Unique index ENRindex on enrollment.

```
CREATE UNIQUE INDEX ENRindex ON College(enrollment);
```

-- Q5. Create a composite Unique index SCMindex on Apply using columns sID, cName, major.

```
CREATE UNIQUE INDEX SCMindex ON Apply(sID, cName, major);
```

-- Q6. Create a composite index on Apply using columns cName and major. Name this index as CMaplyINDX.

```
CREATE INDEX CMaplyINDX ON Apply(cName, major);
```

-- Q7. Create sequence sID\_seq with the following parameters: increment by 3, cycle, cache 4, and which will generate the numbers among 988 to 999.

```
CREATE SEQUENCE sID_seq  
INCREMENT BY 3  
START WITH 988  
MAXVALUE 999  
CYCLE  
CACHE 4;
```

-- Q8. Display the next value of Sequence sID\_seq.

```
SELECT sID_seq.nextval FROM dual;
```

-- Q9. A new student entered the database named Eric with the next sID from sequence sID\_seq having GPA 9.9, sizeHS 9999, DoB as '23-Apr-98' to the Student table.

```
INSERT INTO Student (sID, sName, GPA, sizeHS, DoB)
VALUES (sID_seq.nextval, 'Eric', 9.9, 9999, TO_DATE('23-Apr-98', 'DD-MON-YY'));
```

-- Q10. Now, another boy registered in our system named Troy with the next sID from sequence sID\_seq having GPA 9.8 and sizeHS 989 and Dob as '25-Nov-99'.

```
INSERT INTO Student (sID, sName, GPA, sizeHS, DoB)
VALUES (sID_seq.nextval, 'Troy', 9.8, 989, TO_DATE('25-Nov-99', 'DD-MON-YY'));
```

-- Q11. Display details of the Student table and observe the newly inserted Eric and Troy sID.

```
SELECT * FROM Student;
```

-- Q12. Create a sequence GPA\_seq having a maximum value of 5 and a minimum value of 3. You are supposed to start the sequence with 5 and decrement the sequence with -1, cycle and no cache.

```
CREATE SEQUENCE GPA_seq
START WITH 5
INCREMENT BY -1
MAXVALUE 5
MINVALUE 3
CYCLE
NOCACHE;
```

-- Q13. Update GPA of Eric to the next value of sequence GPA\_seq.

```
UPDATE Student
SET GPA = GPA_seq.nextval
WHERE sName = 'Eric';
```

-- Q14. Insert student Jack with sID from sID\_seq, GPA from GPA\_seq, sizeHS as 1500, and DoB as '22-OCT-97'.

```
INSERT INTO Student (sID, sName, GPA, sizeHS, DoB)
VALUES (sID_seq.nextval, 'Jack', GPA_seq.nextval, 1500, TO_DATE('22-OCT-97', 'DD-MON-YY'));
```



```
-- Q15. Display details of the Student Table and observe GPA and sID of Jack.  
  
SELECT * FROM Student WHERE sName = 'Jack';
```

```
-- Q16. Display the next value of sequence GPA_seq.  
  
SELECT GPA_seq.nextval FROM dual;
```

```
-- Q17. Drop sequence sID_seq.  
  
DROP SEQUENCE sID_seq;
```

```
-- Q18. Drop sequence GPA_seq.  
  
DROP SEQUENCE GPA_seq;
```

## PRACTICAL ASSIGNMENT- 7

---

```
CREATE DATABASE P6;  
USE P6;
```

```
CREATE TABLE IF NOT EXISTS Student (  
    sID INT PRIMARY KEY,  
    sName VARCHAR(50),  
    GPA FLOAT,  
    sizeHS INT NOT NULL,  
    DoB VARCHAR(50)  
);
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('123', 'Amy', '3.9',  
'1000', '1996-06-26');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('234', 'Bob', '3.6',  
'1500', '1995-04-07');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('345', 'Craig', '3.5',  
'500', '1995-02-04');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('456', 'Doris', '3.9',  
'1000', '1997-07-24');  
  
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('567', 'Edward', '2.9',  
'2000', '1996-12-21');  
  
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('678', 'Fay', '3.8',  
'200', '1996-08-27');  
  
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('789', 'Gary', '3.4',  
'800', '1996-10-08');  
  
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('987', 'Helen', '3.7',  
'800', '1997-03-27');  
  
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('876', 'Irene', '3.9',  
'400', '1996-03-07');  
  
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('765', 'Jay', '2.9',  
'1500', '1998-08-08');  
  
INSERT INTO student (sID, sName, GPA, sizeHS, DoB) VALUES ('654', 'Amy', '3.9',  
'1000', '1996-05-26');  
  
INSERT INTO student (sID, sName, GPA, sizeHS, DoB) VALUES ('543', 'Craig', '3.4',  
'2000', '1998-08-27');  
  
SELECT * FROM student;
```

localmysql: SELECT \* FROM st... X ...

sID	sName	GPA	sizeHS	DoB
abc Filter...	abc Filter...	abc Filter...	abc Filter...	abc Filter...
123	Amy	3.9	1000	1996-06-26
234	Bob	3.6	1500	1995-04-07
345	Craig	3.5	500	1995-02-04
456	Doris	3.9	1000	1997-07-24
543	Craig	3.4	2000	1998-08-27
567	Edward	2.9	2000	1996-12-21
654	Amy	3.9	1000	1996-05-26
678	Fay	3.8	200	1996-08-27
765	Jay	2.9	1500	1998-08-08
789	Gary	3.4	800	1996-10-08
876	Irene	3.9	400	1996-03-07
987	Helen	3.7	800	1997-03-27

```

CREATE TABLE IF NOT EXISTS College(
    cName VARCHAR(50) PRIMARY KEY,
    State VARCHAR(50),
    enrollment INT NOT NULL
);

INSERT INTO college(cName, State, enrollment) VALUES('Stanford', 'CA', '15000');
INSERT INTO college(cName, State, enrollment) VALUES('Berkeley', 'CA', '36000');
INSERT INTO college(cName, State, enrollment) VALUES('MIT', 'MA', '10000');
INSERT INTO college(cName, State, enrollment) VALUES('Cornell', 'NY', '21000');
INSERT INTO college(cName, State, enrollment) VALUES('Harvard', 'MA', '50040');
SELECT * FROM college;

```

localmysql: SELECT * FROM co... X		
cName	State	enrollment
abc Filter...	abc Filter...	abc Filter...
Berkeley	CA	36000
Cornell	NY	21000
Harvard	MA	50040
MIT	MA	10000
Stanford	CA	15000

```

CREATE TABLE IF NOT EXISTS Applied(
    sID INT NOT NULL,
    cName VARCHAR(50) NOT NULL,
    major VARCHAR(50) NOT NULL,
    decision VARCHAR(1) NOT NULL
);

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Stanford', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Stanford', 'EE', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Berkeley', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Cornell', 'EE', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('234', 'Berkeley', 'biology', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'MIT', 'bioengineering', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'bioengineering', 'N');

```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'EE', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('678', 'Stanford', 'history', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('987', 'Stanford', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('987', 'Berkeley', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('876', 'Stanford', 'CS', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('876', 'MIT', 'biology', 'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('876', 'MIT', 'marine biology', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('765', 'Stanford', 'history', 'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Stanford', 'history', 'N');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Cornell', 'history', 'N');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Cornell', 'psychology', 'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('543', 'MIT', 'CS', 'N');

SELECT * FROM applied;
```

localmysql: SELECT \* FROM ap... X

sID	cName	major	decision
abc Filter...	abc Filter...	abc Filter...	abc Filter...
123	Stanford	CS	Y
123	Stanford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Stanford	history	Y
987	Stanford	CS	Y
987	Berkeley	CS	Y
876	Stanford	CS	N
876	MIT	biology	Y
876	MIT	marine biology	N
765	Stanford	history	Y
765	Stanford	history	N
765	Cornell	history	N
765	Cornell	psychology	Y

Write SQL queries for the following:

```
-- Q1. Create view WeakStudent on Student whose GPA is less than 3.7.
CREATE VIEW WeakStudent AS
SELECT *
FROM Student
WHERE GPA < 3.7;
```

```

-- Q2. Create a view cView on college (containing all the columns) and rename the
column cName as collegeName and enrollment as seats respectively.
CREATE VIEW cView AS
SELECT cName AS collegeName, enrollment AS seats, state, enrollment
FROM college;

-- Q3. Create view CSaccept having IDs and college of students who applied to CS
major and their application is accepted.
CREATE VIEW CSaccept AS
SELECT sID, cName
FROM Apply
WHERE major = 'CS' AND decision = 'Y';

-- Q4. Create view CSberkeley having IDs, name, GPA, sizeHS of those students who
are accepted in CS at Berkeley and come from a High School with more than 500
students.
CREATE VIEW CSberkeley AS
SELECT s.sID, s.sName, s.GPA, s.sizeHS
FROM Student s
JOIN CSaccept c ON s.sID = c.sID
WHERE c.cName = 'Berkeley' AND s.sizeHS > 500;

-- Q5. Display information about students in CSberk having GPA greater than 3.8.
SELECT *
FROM CSberkeley
WHERE GPA > 3.8;

-- Q6. Drop view CSaccept.
DROP VIEW CSaccept;

-- Q7. Display all students in CSberkeley.
SELECT *
FROM CSberkeley;

-- Q8. Update GPA by 0.8 of students in view WeakStudent who have a high school
size greater than 1000.
UPDATE Student
SET GPA = GPA + 0.8
WHERE sID IN (SELECT sID FROM WeakStudent WHERE sizeHS > 1000);

-- Q9. Create a view AppCount which contains sID of Student and the number of
applications they filed. Name the column sID and NoOfApp.
CREATE VIEW AppCount AS
SELECT sID, COUNT(*) AS NoOfApp
FROM Apply

```

```

GROUP BY sID;

-- Q10. Update NoOfApp so that sID 234 contains 4 applications.
UPDATE AppCount
SET NoOfApp = 4
WHERE sID = 234;

-- Q11. Create a view StuName containing student names and their GPA. Is this
View Updatable? If not, specify why.
CREATE VIEW StuName AS
SELECT sName, GPA
FROM Student;

-- This view is not updatable because it contains columns from the Student table
that are not unique, making it ambiguous for the database to determine which
records to update if changes are made.

-- Q12. Create view studentHS having details of students who come from a High
School of size more than 1000 using with check option.
CREATE VIEW studentHS AS
SELECT *
FROM Student
WHERE sizeHS > 1000
WITH CHECK OPTION;

-- Q13. Try to insert details of a new student Ram with sID 999 having GPA 9.9
and sizeHS 9999 to the newly created view studentHS. (Comment on whether this
view is updatable.)
-- This insert will fail because the check option is set, and the sizeHS value
for Ram (9999) does not meet the condition specified in the view (sizeHS > 1000).
Therefore, the view is not updatable in this context.

-- Q14. Clerk realizes he wrongly typed sizeHS of Ramu as 9999; it is actually
999. Can you help him update the value of sizeHS of Ramu?
UPDATE studentHS
SET sizeHS = 999
WHERE sID = 999;

-- Q15. Now, another boy registered in our system named Ramu with sID 998 having
GPA 9.8 and sizeHS 989.
INSERT INTO studentHS (sID, sName, GPA, sizeHS)
VALUES (998, 'Ramu', 9.8, 989);

```

---



# PRACTICAL ASSIGNMENT- 6

---

```
CREATE DATABASE P6;  
USE P6;
```

```
CREATE TABLE IF NOT EXISTS Student (  
    sID INT PRIMARY KEY,  
    sName VARCHAR(50),  
    GPA FLOAT,  
    sizeHS INT NOT NULL,  
    DoB VARCHAR(50)  
);
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('123', 'Amy', '3.9',  
'1000', '1996-06-26');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('234', 'Bob', '3.6',  
'1500', '1995-04-07');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('345', 'Craig', '3.5',  
'500', '1995-02-04');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('456', 'Doris', '3.9',  
'1000', '1997-07-24');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('567', 'Edward', '2.9',  
'2000', '1996-12-21');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('678', 'Fay', '3.8',  
'200', '1996-08-27');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('789', 'Gary', '3.4',  
'800', '1996-10-08');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('987', 'Helen', '3.7',  
'800', '1997-03-27');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('876', 'Irene', '3.9',  
'400', '1996-03-07');
```

```

INSERT INTO student(sID, sName, GPA, sizeHS, DoB) VALUES ('765', 'Jay', '2.9',
'1500', '1998-08-08');

INSERT INTO student (sID, sName, GPA, sizeHS, DoB) VALUES ('654', 'Amy', '3.9',
'1000', '1996-05-26');

INSERT INTO student (sID, sName, GPA, sizeHS, DoB) VALUES ('543', 'Craig', '3.4',
'2000', '1998-08-27');

SELECT * FROM student;

```

localmysql: SELECT \* FROM st... × ...

sID	sName	GPA	sizeHS	DoB
<input type="text" value="abc"/> Filter...	<input type="text" value="abc"/> Filter...	<input type="text" value="abc"/> Filter...	<input type="text" value="abc"/> Filter...	<input type="text" value="abc"/> Filter...
123	Amy	3.9	1000	1996-06-26
234	Bob	3.6	1500	1995-04-07
345	Craig	3.5	500	1995-02-04
456	Doris	3.9	1000	1997-07-24
543	Craig	3.4	2000	1998-08-27
567	Edward	2.9	2000	1996-12-21
654	Amy	3.9	1000	1996-05-26
678	Fay	3.8	200	1996-08-27
765	Jay	2.9	1500	1998-08-08
789	Gary	3.4	800	1996-10-08
876	Irene	3.9	400	1996-03-07
987	Helen	3.7	800	1997-03-27

```

CREATE TABLE IF NOT EXISTS College(
    cName VARCHAR(50) PRIMARY KEY,
    State VARCHAR(50),
    enrollment INT NOT NULL
);

INSERT INTO college(cName, State, enrollment) VALUES('Stanford', 'CA', '15000');

```

```

INSERT INTO college(cName, State, enrollment) VALUES('Berkeley', 'CA', '36000');
INSERT INTO college(cName, State, enrollment) VALUES('MIT', 'MA', '10000');
INSERT INTO college(cName, State, enrollment) VALUES('Cornell', 'NY', '21000');
INSERT INTO college(cName, State, enrollment) VALUES('Harvard', 'MA', '50040');
SELECT * FROM college;

```

localmysql: SELECT * FROM co... X		
cName	State	enrollment
abc Filter...	abc Filter...	abc Filter...
Berkeley	CA	36000
Cornell	NY	21000
Harvard	MA	50040
MIT	MA	10000
Stanford	CA	15000

```

CREATE TABLE IF NOT EXISTS Applied(
    sID INT NOT NULL,
    cName VARCHAR(50) NOT NULL,
    major VARCHAR(50) NOT NULL,
    decision VARCHAR(1) NOT NULL
);

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Stanford', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Stanford', 'EE', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Berkeley', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Cornell', 'EE', 'Y');

```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('234', 'Berkeley',
'biology', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'MIT',
'bioengineering', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell',
'bioengineering', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'CS',
'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'EE',
'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('678', 'Stanford',
'history', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('987', 'Stanford', 'CS',
'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('987', 'Berkeley', 'CS',
'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('876', 'Stanford', 'CS',
'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('876', 'MIT', 'biology',
'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('876', 'MIT', 'marine
biology', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('765', 'Stanford',
'history', 'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Stanford',
'history', 'N');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Cornell',
'history', 'N');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Cornell',
'psychology', 'Y');
```

```
INSERT INTO applied(sID, cName, major, decision) VALUES('543', 'MIT', 'CS', 'N');

SELECT * FROM applied;
```

localmysql: SELECT \* FROM ap... ✕

sID	cName	major	decision
abc Filter...	abc Filter...	abc Filter...	abc Filter...
123	Stanford	CS	Y
123	Stanford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Stanford	history	Y
987	Stanford	CS	Y
987	Berkeley	CS	Y
876	Stanford	CS	N
876	MIT	biology	Y
876	MIT	marine biology	N
765	Stanford	history	Y
765	Stanford	history	N
765	Cornell	history	N
765	Cornell	psychology	Y

Write SQL queries for the following:

Q1. Create a new column DoB in Student table. (Datatype will be date)

```
ALTER TABLE Student
ADD COLUMN DoB DATE;
```

Q2. Insert DoB for each Student in corresponding table using above instance of Student table.

```
UPDATE Student SET DoB = '1996-06-26' WHERE sID = '123';
UPDATE Student SET DoB = '1995-04-07' WHERE sID = '234';
UPDATE Student SET DoB = '1995-02-04' WHERE sID = '345';
UPDATE Student SET DoB = '1997-07-24' WHERE sID = '456';
UPDATE Student SET DoB = '1996-12-21' WHERE sID = '567';
UPDATE Student SET DoB = '1996-08-27' WHERE sID = '678';
UPDATE Student SET DoB = '1996-10-08' WHERE sID = '789';
UPDATE Student SET DoB = '1997-03-27' WHERE sID = '987';
UPDATE Student SET DoB = '1996-03-07' WHERE sID = '876';
UPDATE Student SET DoB = '1998-08-08' WHERE sID = '765';
UPDATE Student SET DoB = '1996-05-26' WHERE sID = '654';
UPDATE Student SET DoB = '1998-08-27' WHERE sID = '543';
```

Q3. Find average of GPA round off to 2 decimal places.

```
SELECT ROUND(AVG(GPA), 2) AS average_gpa FROM Student;
```

Q4. Find year of DoB of Student having less than 1000.

```
SELECT YEAR(DoB) AS birth_year FROM Student WHERE sizeHS < 1000;
```

Q5. Compute Age of each student. (Hint: take difference between year of sysdate and Student's DoB)

```
SELECT sName, YEAR(CURRENT_DATE()) - YEAR(DoB) - (DATE_FORMAT(CURRENT_DATE(), '%m%d') < DATE_FORMAT(DoB, '%m%d')) AS age FROM Student;
```

Q6. Display name of all Students in uppercase and name of college they applied in lower case.

```
SELECT UPPER(sName) AS student_name, LOWER(cName) AS college_name FROM Student
JOIN Applied ON Student.sID = Applied.sID;
```

Q7. Find fourth alphabet of each student. (Hint: use substring)

```
SELECT SUBSTRING(sName, 4, 1) AS fourth_alphabet FROM Student;
```

Q8. Find sID and sName of student whose sName has string length greater than 3.

```
SELECT sID, sName FROM Student WHERE LENGTH(sName) > 3;
```

Q9. Find floor, ceiling and truncate (to one decimal place) value of average GPA.

```
SELECT FLOOR(AVG(GPA)) AS floor_value, CEILING(AVG(GPA)) AS ceiling_value,  
TRUNCATE(AVG(GPA), 1) AS truncated_value FROM Student;
```

Q10. Display details of all students whose sID is even.

```
SELECT * FROM Student WHERE sID % 2 = 0;
```

Q11. Compute Square Root of 900 and  $24^7$ .

```
SELECT SQRT(900) AS square_root_900, SQRT(247) AS square_root_247;
```

Q12. Consider the string “Peter Piper picked a peck of pickled peppers. A peck of pickled peppers Peter Piper picked. If Peter Piper picked a peck of pickled peppers, Where the peck of pickled peppers Peter Piper picked?” Find 6th occurrence of string ‘pick’. (Hint: use INSTR)

```
SELECT INSTR('Peter Piper picked a peck of pickled peppers. A peck of pickled  
peppers Peter Piper picked. If Peter Piper picked a peck of pickled peppers,  
Where the peck of pickled peppers Peter Piper picked?', 'pick', 1, 6) AS  
sixth_occurrence_position;
```

Q13. Consider String ‘Satya Nadella’ replace this using the key (Hint: use translate)

```
SELECT TRANSLATE('Satya Nadella', 'aeINST123456y7', 'eaeINS1t234567y7') AS  
replaced_string;
```

Q14. Display sID, sname and DoB in this format ‘February 26, 2014’

```
SELECT sID, sname, DATE_FORMAT(DoB, '%M %e, %Y') AS formatted_date FROM Student;
```

Q15. Convert the text ‘26/02/2014’ to date.

```
SELECT STR_TO_DATE('26/02/2014', '%d/%m/%Y') AS converted_date;
```

Q16. Compute on which date is next Saturday and last day of this month?

```
SELECT DATE_ADD(CURRENT_DATE(), INTERVAL (7 - DAYOFWEEK(CURRENT_DATE())) DAY) AS next_saturday;  
SELECT LAST_DAY(CURRENT_DATE()) AS last_day_of_month;
```

## Exercise:

Q1. Display sID, sname and DoB in this format '26th February, 2014'

```
SELECT sID, sname, DATE_FORMAT(DoB, '%D %M, %Y') AS formatted_date FROM Student;
```

Q2. Display sID, sname and DoB in this format '26/02/2014'

```
SELECT sID, sname, DATE_FORMAT(DoB, '%d/%m/%Y') AS formatted_date FROM Student;
```

Q3. Add 5 months to DoB of Edward?

```
UPDATE Student SET DoB = DATE_ADD(DoB, INTERVAL 5 MONTH) WHERE sName = 'Edward';
```

Q4. Display last day of DoB of Amy?

```
SELECT LAST_DAY(DoB) AS last_day_of_birth FROM Student WHERE sName = 'Amy';
```

Q5. Display next Sunday of DoB of Doris?

```
SELECT DATE_ADD(DoB, INTERVAL (8 - DAYOFWEEK(DoB)) DAY) AS next_sunday FROM Student WHERE sName = 'Doris';
```

# PRACTICAL ASSIGNMENT - 5

---

```
CREATE DATABASE p4;  
USE p4;
```

```
CREATE TABLE IF NOT EXISTS Student (  
    sID INT PRIMARY KEY,  
    sName VARCHAR(50),  
    GPA FLOAT,
```



```
sizeHS INT NOT NULL
);

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('123', 'Amy', '3.9',
'1000');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('234', 'Bob', '3.6',
'1500');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('345', 'Craig', '3.5',
'500');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('456', 'Doris', '3.9',
'1000');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('567', 'Edward', '2.9',
'2000');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('678', 'Fay', '3.8', '200');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('789', 'Gary', '3.4',
'800');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('987', 'Helen', '3.7',
'800');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('876', 'Irene', '3.9',
'400');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('765', 'Jay', '2.9',
'1500');

INSERT INTO student (sID, sName, GPA, sizeHS) VALUES ('654', 'Amy', '3.9',
'1000');

INSERT INTO student (sID, sName, GPA, sizeHS) VALUES ('543', 'Craig', '3.4',
'2000');

SELECT * FROM student;
```

localmysql: SELECT \* FROM st... X

sID	sName	GPA	sizeHS
abc Filter...	abc Filter...	abc Filter...	abc Filter...
123	Amy	3.9	1000
234	Bob	3.6	1500
345	Craig	3.5	500
456	Doris	3.9	1000
543	Craig	3.4	2000
567	Edward	2.9	2000
654	Amy	3.9	1000
678	Fay	3.8	200
765	Jay	2.9	1500
789	Gary	3.4	800
876	Irene	3.9	400
987	Helen	3.7	800

```
CREATE TABLE IF NOT EXISTS College(
  cName VARCHAR(50) PRIMARY KEY,
  State VARCHAR(50),
  enrollment INT NOT NULL
);
```

```
INSERT INTO college(cName, State, enrollment) VALUES('Stanford', 'CA', '15000');
INSERT INTO college(cName, State, enrollment) VALUES('Berkeley', 'CA', '36000');
INSERT INTO college(cName, State, enrollment) VALUES('MIT', 'MA', '10000');
INSERT INTO college(cName, State, enrollment) VALUES('Cornell', 'NY', '21000');
INSERT INTO college(cName, State, enrollment) VALUES('Harvard', 'MA', '50040');
SELECT * FROM college;
```

localmysql: SELECT \* FROM co... X

cName	State	enrollment
abc Filter...	abc Filter...	abc Filter...
Berkeley	CA	36000
Cornell	NY	21000
Harvard	MA	50040
MIT	MA	10000
Stanford	CA	15000

```
CREATE TABLE IF NOT EXISTS Applied(
```

```
    sID INT NOT NULL,
    cName VARCHAR(50) NOT NULL,
    major VARCHAR(50) NOT NULL,
    decision VARCHAR(1) NOT NULL
```

```
);
```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Stanford', 'CS', 'Y');
```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Stanford', 'EE', 'N');
```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Berkeley', 'CS', 'Y');
```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('123', 'Cornell', 'EE', 'Y');
```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('234', 'Berkeley', 'biology', 'N');
```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'MIT', 'bioengineering', 'Y');
```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'bioengineering', 'N');
```

```
INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('345', 'Cornell', 'EE', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('678', 'Stanford', 'history', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('987', 'Stanford', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('987', 'Berkeley', 'CS', 'Y');

INSERT INTO Applied(sID, cName, major, decision) VALUES('876', 'Stanford', 'CS', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('876', 'MIT', 'biology', 'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('876', 'MIT', 'marine biology', 'N');

INSERT INTO Applied(sID, cName, major, decision) VALUES('765', 'Stanford', 'history', 'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Stanford', 'history', 'N');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Cornell', 'history', 'N');

INSERT INTO applied(sID, cName, major, decision) VALUES('765', 'Cornell', 'psychology', 'Y');

INSERT INTO applied(sID, cName, major, decision) VALUES('543', 'MIT', 'CS', 'N');

SELECT * FROM applied;
```

localmysql: SELECT * FROM ap... X				
sID	cName	major	decision	
a b c Filter...	a b c Filter...	a b c Filter...	a b c Filter...	
123	Stanford	CS	Y	
123	Stanford	EE	N	
123	Berkeley	CS	Y	
123	Cornell	EE	Y	
234	Berkeley	biology	N	
345	MIT	bioengineering	Y	
345	Cornell	bioengineering	N	
345	Cornell	CS	Y	
345	Cornell	EE	N	
678	Stanford	history	Y	
987	Stanford	CS	Y	
987	Berkeley	CS	Y	
876	Stanford	CS	N	
876	MIT	biology	Y	
876	MIT	marine biology	N	
765	Stanford	history	Y	
765	Stanford	history	N	
765	Cornell	history	N	
765	Cornell	psychology	Y	

### Write SQL queries for the following:

Q1. IDs and names of students who have applied to major in CS at some college.

```
SELECT DISTINCT s.sID, s.sName
FROM Student s
JOIN Applied a ON s.sID = a.sID
WHERE a.major = 'CS';
```

Q2. Find ID and name of student having same high school size as Jay.

```
SELECT sID, sName
FROM Student
WHERE sizeHS = (SELECT sizeHS FROM Student WHERE sID = '765') AND sID != '765';
```

Q3. Find ID and name of student having same high school size as Jay but result should not include Jay.

```
SELECT sID, sName
FROM Student
WHERE sizeHS = (SELECT sizeHS FROM Student WHERE sID = '765') AND sID != '765';
```

Q4. Find the name of student with their GPA and Sid whose GPA not equal to GPA of Irene?

```
SELECT sID, sName, GPA
FROM Student
WHERE GPA != (SELECT GPA FROM Student WHERE sName = 'Irene');
```

Q5. Find college where any student having their name started from J have applied?

```
SELECT DISTINCT cName
FROM Applied
WHERE sID IN (SELECT sID FROM Student WHERE sName LIKE 'J%');
```

Q6. Find all different major where Irene has applied?

```
SELECT DISTINCT major
FROM Applied
WHERE sID = (SELECT sID FROM Student WHERE sName = 'Irene');
```

Q7. Find IDs of student and major who applied in any of major Irene had applied to?

```
SELECT DISTINCT a.sID, a.major
FROM Applied a
WHERE a.major IN (SELECT DISTINCT major FROM Applied WHERE sID = (SELECT sID FROM Student WHERE sName = 'Irene'));
```

Q8. Find IDs of student and major who applied in any of major Irene had applied to? But this time exclude Irene sID from the list.

```
SELECT DISTINCT a.sID, a.major
FROM Applied a
WHERE a.major IN (SELECT DISTINCT major FROM Applied WHERE sID = (SELECT sID FROM Student WHERE sName = 'Irene'))
AND a.sID != (SELECT sID FROM Student WHERE sName = 'Irene');
```

Q9. Give the number of colleges Jay applied to? (Remember count each college once no matter if he applied to same college twice with different major)

```
SELECT COUNT(DISTINCT cName) AS NumColleges
```

```
FROM Applied
WHERE sID = '765';
```

Q10. Find sID of student who applied to more or same number of college where Jay has applied?

```
SELECT sID
FROM Applied
GROUP BY sID
HAVING COUNT(DISTINCT cName) >= (SELECT COUNT(DISTINCT cName) FROM Applied WHERE sID = '765');
```

Q11. Find details of Students who applied to major CS but not applied to major EE? (sID 987, 876, 543 should only be include in result)

```
SELECT sID, sName, GPA, sizeHS
FROM Student
WHERE sID IN (
    SELECT DISTINCT a.sID
    FROM Applied a
    WHERE a.major = 'CS'
)
AND sID NOT IN (
    SELECT DISTINCT a.sID
    FROM Applied a
    WHERE a.major = 'EE'
)
AND sID IN ('987', '876', '543');
```

Q12. All colleges such that some other college is in same state. (Cornell should not be part of result as no other college in New York Hint: use exists)

```
SELECT cName, State
FROM College c1
WHERE EXISTS (
    SELECT 1
    FROM College c2
    WHERE c1.State = c2.State
    AND c1.cName != c2.cName
)
AND cName != 'Cornell';
```

Q13. Find the college with highest enrollment.

```
SELECT cName, enrollment
FROM College
WHERE enrollment = (SELECT MAX(enrollment) FROM College);
```

Q14. Find name of student having lowest GPA.

```
SELECT sName
FROM Student
```

```
WHERE GPA = (SELECT MIN(GPA) FROM Student);
```

Q15. Find the most popular major.

```
SELECT major, COUNT(*) AS popularity
FROM Applied
GROUP BY major
ORDER BY popularity DESC
LIMIT 1;
```

Q16. Find sID, sName, sizeHS of all students NOT from smallest HS

```
SELECT sID, sName, sizeHS
FROM Student
WHERE sizeHS != (SELECT MIN(sizeHS) FROM Student);
```

Q17. Find the name of student who applies to all the colleges where sID 987 has applied? (Hint: see Query Find IDs of student applied to all colleges)

```
insert into Applied
select s1.sID, 'Berkeley', 'CSE', 'Y'
from Student s1
where s1.sID IN (select s.sID from Student s
MINUS
select a.sID from Applied a where a.cName = 'Berkeley');
```

Q18. . Find sid of student who have not applied to Stanford.

```
SELECT sID
FROM Student
EXCEPT
SELECT sID
FROM Applied
WHERE cName = 'Stanford';
```

Q19. Find sid of Student that applied to both Stanford and Berkeley.

```
SELECT sID
FROM Applied
WHERE cName = 'Stanford'
INTERSECT
SELECT sID
FROM Applied
WHERE cName = 'Berkeley';
```

Q20. Give list of all names including all names of colleges and students.

```
SELECT sName AS Name
FROM Student
UNION
SELECT cName AS Name
FROM College;
```



Q21. Create a table ApplicationInfo having columns sID: int, sName: varchar2(10) and number\_of\_applications: number(2) they filed? Populate this table with appropriate data using insert command. (Remember to include details of ALL students that have applied or not applied)

```
-- Create the ApplicationInfo table
CREATE TABLE ApplicationInfo (
    sID INT,
    sName VARCHAR(10),
    number_of_applications INT
);

-- Populate the ApplicationInfo table with data
INSERT INTO ApplicationInfo (sID, sName, number_of_applications)
SELECT s.sID, s.sName, COUNT(a.sID) AS number_of_applications
FROM Student s
LEFT JOIN Applied a ON s.sID = a.sID
GROUP BY s.sID, s.sName;

-- Insert data for students who have not applied
INSERT INTO ApplicationInfo (sID, sName, number_of_applications)
SELECT sID, sName, 0 AS number_of_applications
FROM Student
WHERE sID NOT IN (SELECT sID FROM Applied);
```

Q22. Create table ApplicationData and load with ID, name of college where they applied with state of college on runtime using single query.

```
-- Create the ApplicationData table
CREATE TABLE ApplicationData (
    sID INT,
    cName VARCHAR(50),
    State VARCHAR(50)
);

-- Populate the ApplicationData table with data
INSERT INTO ApplicationData (sID, cName, State)
SELECT a.sID, a.cName, c.State
FROM Applied a
JOIN College c ON a.cName = c.cName;
```

Q23. Stanford decide not to take any student who have also applied to its rival Berkeley turn their application decision to N. (Hint: Turn decision to N only if any student applied to both Stanford and Berkeley, Update decision of only Stanford Application )

```
UPDATE Applied
SET decision = 'N'
WHERE cName = 'Stanford'
```

```
AND sID IN (  
    SELECT sID  
    FROM Applied  
    WHERE cName = 'Stanford'  
    INTERSECT  
    SELECT sID  
    FROM Applied  
    WHERE cName = 'Berkeley'  
);
```

Q24. Delete applications that are filed Colleges situated at city 'New York'.

```
DELETE FROM Applied  
WHERE cName IN (  
    SELECT cName  
    FROM College  
    WHERE City = 'New York'  
);
```

## PRACTICAL ASSIGNMENT – 4

---

```
CREATE DATABASE p4;  
USE p4;  
  
CREATE TABLE IF NOT EXISTS Student (  
    sID INT PRIMARY KEY,  
    sName VARCHAR(50),  
    GPA FLOAT,  
    sizeHS INT NOT NULL  
);  
  
INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('123', 'Amy', '3.9',  
'1000');  
  
INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('234', 'Bob', '3.6',  
'1500');  
  
INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('345', 'Craig', '3.5',  
'500');  
  
INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('456', 'Doris', '3.9',  
'1000');
```

```
INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('567', 'Edward', '2.9', '2000');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('678', 'Fay', '3.8', '200');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('789', 'Gary', '3.4', '800');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('987', 'Helen', '3.7', '800');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('876', 'Irene', '3.9', '400');

INSERT INTO student(sID, sName, GPA, sizeHS) VALUES ('765', 'Jay', '2.9', '1500');

INSERT INTO student (sID, sName, GPA, sizeHS) VALUES ('654', 'Amy', '3.9', '1000');

INSERT INTO student (sID, sName, GPA, sizeHS) VALUES ('543', 'Craig', '3.4', '2000');

SELECT * FROM student;
```

sID	sName	GPA	sizeHS
abc Filter...	abc Filter...	abc Filter...	abc Filter...
123	Amy	3.9	1000
234	Bob	3.6	1500
345	Craig	3.5	500
456	Doris	3.9	1000
543	Craig	3.4	2000
567	Edward	2.9	2000
654	Amy	3.9	1000
678	Fay	3.8	200
765	Jay	2.9	1500
789	Gary	3.4	800
876	Irene	3.9	400
987	Helen	3.7	800

```

CREATE TABLE IF NOT EXISTS College(
    cName VARCHAR(50) PRIMARY KEY,
    State VARCHAR(50),
    enrollment INT NOT NULL
);

INSERT INTO college(cName, State, enrollment) VALUES('Stanford', 'CA', '15000');
INSERT INTO college(cName, State, enrollment) VALUES('Berkeley', 'CA', '36000');
INSERT INTO college(cName, State, enrollment) VALUES('MIT', 'MA', '10000');
INSERT INTO college(cName, State, enrollment) VALUES('Cornell', 'NY', '21000');
INSERT INTO college(cName, State, enrollment) VALUES('Harvard', 'MA', '50040');

SELECT * FROM college;

```

localmysql: SELECT \* FROM co... X

cName	State	enrollment
abc Filter...	abc Filter...	abc Filter...
Berkeley	CA	36000
Cornell	NY	21000
Harvard	MA	50040
MIT	MA	10000
Stanford	CA	15000

```

CREATE TABLE IF NOT EXISTS Apply(
  sID INT PRIMARY KEY,
  cName VARCHAR(50) NOT NULL,
  major VARCHAR(50) NOT NULL,
  decision VARCHAR(1) NOT NULL
);

INSERT INTO apply(sID, cName, major, decision) VALUES('123', 'Stanford', 'CS',
'Y');

INSERT INTO apply(sID, cName, major, decision) VALUES('123', 'Stanford', 'EE',
'N');

INSERT INTO apply(sID, cName, major, decision) VALUES('123', 'Berkeley', 'CS',
'Y');

INSERT INTO apply(sID, cName, major, decision) VALUES('123', 'Cornell', 'EE',
'Y');

INSERT INTO apply(sID, cName, major, decision) VALUES('234', 'Berkeley',
'biology', 'N');

INSERT INTO apply(sID, cName, major, decision) VALUES('345', 'MIT',
'bioengineering', 'Y');

```

```
INSERT INTO apply(sID, cName, major, decision) VALUES('345', 'Cornell',  
'bioengineering', 'N');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('345', 'Cornell', 'CS',  
'Y');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('345', 'Cornell', 'EE',  
'N');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('678', 'Stanford',  
'history', 'Y');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('987', 'Stanford', 'CS',  
'Y');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('987', 'Berkeley', 'CS',  
'Y');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('876', 'Stanford', 'CS',  
'N');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('876', 'MIT', 'biology',  
'Y');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('876', 'MIT', 'marine  
biology', 'N');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('765', 'Stanford',  
'history', 'Y');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('765', 'Stanford',  
'history', 'N');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('765', 'Cornell',  
'history', 'N');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('765', 'Cornell',  
'psychology', 'Y');  
  
INSERT INTO apply(sID, cName, major, decision) VALUES('543', 'MIT', 'CS', 'N');
```

---

## Q. Solve the following :

Q1. Count the total number of Students.

```
SELECT COUNT(*) AS total_students FROM Student;
```

Q2. Calculate the average GPA of all Student.

```
SELECT AVG(GPA) AS average_gpa FROM Student;
```

Q3. Determine the minimum and maximum GPA. Rename the titles as 'max\_GPA' and 'min\_GPA' respectively.

```
SELECT MAX(GPA) AS max_GPA, MIN(GPA) AS min_GPA FROM Student;
```

Q4. Count the number of students having GPA greater than or equal to 3.7.

```
SELECT COUNT(*) AS students_above_3_7 FROM Student WHERE GPA >= 3.7;
```

Q5. Find Maximum, Average, Minimum, total GPA of all student.

```
SELECT MAX(GPA) AS max_gpa, AVG(GPA) AS avg_gpa, MIN(GPA) AS min_gpa, SUM(GPA) AS total_gpa FROM Student;
```

Q6. Find total number of colleges in our Application Database.

```
SELECT COUNT(*) AS total_colleges FROM College;
```

Q7. Find how many different majors student had applied in.

```
SELECT COUNT(DISTINCT major) AS different_majors FROM Apply;
```

Q8. Find total no. of Applications in our Application System's Database.

```
SELECT COUNT(*) AS total_applications FROM Apply;
```

Q9. Find average of all distinct GPA.

```
SELECT AVG(DISTINCT GPA) AS avg_distinct_gpa FROM Student;
```

Q10. Display the total number of application accepted.

```
SELECT COUNT(*) AS applications_accepted FROM Apply WHERE decision = 'Y';
```

Q11. Find number of students having GPA>3.4 and coming from high school having size>1000.

```
SELECT COUNT(*) AS students_criteria FROM Student WHERE GPA > 3.4 AND sizeHS > 1000;
```

Q12. Find how many student applied to 'marine biology'.

```
SELECT COUNT(DISTINCT sID) AS students_marine_biology FROM Apply WHERE major = 'marine biology';
```

Q13. Find how many applications were rejected and accepted by the colleges.

```
SELECT  
    SUM(CASE WHEN decision = 'Y' THEN 1 ELSE 0 END) AS accepted,  
    SUM(CASE WHEN decision = 'N' THEN 1 ELSE 0 END) AS rejected  
FROM Apply;
```

Q14. Find how many students applied to a particular major. (show count(sid) as No\_of\_applications).

```
SELECT major, COUNT(sID) AS No_of_applications FROM Apply GROUP BY major;
```

Q15. Find number of applications received by particular college.

```
SELECT cName, COUNT(*) AS applications_received FROM Apply GROUP BY cName;
```

Q16. Find number of applications received in a particular major at a particular college.

```
SELECT cName, major, COUNT(*) AS applications_received FROM Apply GROUP BY cName, major;
```

Q17. Give the college name and major, where number of applications received are greater than or equal to 2.

```
SELECT cName, major FROM Apply GROUP BY cName, major HAVING COUNT(*) >= 2;
```

Q18. Give the name and no of applications of all those colleges which receives applications from 3 or more students.



```
SELECT cName, COUNT(DISTINCT sID) AS no_of_applications
FROM Apply
GROUP BY cName
HAVING COUNT(DISTINCT sID) >= 3;
```

Q19. Give state and number of colleges of a state that has more than 1 college.

```
SELECT state, COUNT(*) AS no_of_colleges
FROM College
GROUP BY state
HAVING COUNT(*) > 1;
```

Q20. Find the name of students that are duplicate.

```
SELECT sName
FROM Student
GROUP BY sName
HAVING COUNT(*) > 1;
```

Q21. Find how many applications are filed by each student. [Hint: use left join as we need information about all 12 students here. If they applied nowhere than show zero in front of them]

```
SELECT s.sID, s.sName, COALESCE(COUNT(a.sID), 0) AS no_of_applications
FROM Student s
LEFT JOIN Apply a ON s.sID = a.sID
GROUP BY s.sID;
```

Q22. Provide name of students that file 3 or more applications.

```
SELECT s.sName
FROM Student s
JOIN Apply a ON s.sID = a.sID
GROUP BY s.sID
HAVING COUNT(a.sID) >= 3;
```

Q23. Provide name of student who have not applied to any college.

```
SELECT sName
FROM Student
WHERE sID NOT IN (SELECT DISTINCT sID FROM Apply);
```

Q24. Find maximum GPA, Average GPA, and minimum GPA among applicants of each college. (i.e. say sID 123, 324 and 987 had applied to Berkley then compute and display max GPA among these three)

```
SELECT a.cName, MAX(s.GPA) AS max_GPA, AVG(s.GPA) AS avg_GPA, MIN(s.GPA) AS min_GPA
FROM Apply a
JOIN Student s ON a.sID = s.sID
GROUP BY a.cName;
```

Q25. Find how many student have same GPA among all students. (provide this frequency in two column table as GPA 3.9 is 4 times, GPA 2.9 is 2 times )

```
SELECT GPA, COUNT(*) AS frequency
FROM Student
GROUP BY GPA;
```

Q26. Find how many application of each major are rejected and accepted.

```
SELECT major, decision, COUNT(*) AS no_of_applications
FROM Apply
GROUP BY major, decision;
```

Q27. Find out the acceptance rate for each college. (Acceptance Rate is percentage of number application accepted w. r. t. number of application received)

```
SELECT cName,
       ROUND((COUNT(CASE WHEN decision = 'Y' THEN 1 END) / CAST(COUNT(*) AS
FLOAT)) * 100, 2) AS acceptance_rate
FROM Apply
GROUP BY cName;
```

---

## Practical Assignment – 3

---

**Imagine you are managing a comprehensive database system for an academic institution that tracks essential information about colleges, students, and their application records. Create Student, Apply and College tables using script.**

**Run the following Script:**

```
Create DATABASE p3;

USE p3;

CREATE TABLE College (
    CollegeID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Location VARCHAR(255)
);

CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Age INT,
    GPA FLOAT
);

CREATE TABLE Apply (
    StudentID INT,
    CollegeID INT,
    Decision VARCHAR(50),
    CONSTRAINT PK_Apply PRIMARY KEY (StudentID, CollegeID),
    CONSTRAINT FK_StudentID FOREIGN KEY (StudentID) REFERENCES Student
(StudentID),
    CONSTRAINT FK_CollegeID FOREIGN KEY (CollegeID) REFERENCES College
(CollegeID)
);

DROP TABLE IF EXISTS Apply;
DROP TABLE IF EXISTS Student;
DROP TABLE IF EXISTS College;

CREATE TABLE College(
    collegeName VARCHAR(10) PRIMARY KEY,
```

```

    state VARCHAR(10),
    enrollment INT
);

CREATE TABLE Student(
    sID INT PRIMARY KEY,
    sName VARCHAR(10),
    GPA DECIMAL(3, 1),
    sizeHS INT
);

CREATE TABLE Apply(
    sID INT,
    cName VARCHAR(10),
    major VARCHAR(20),
    decision CHAR(1),
    PRIMARY KEY(sID, major, cName),
    FOREIGN KEY(sID) REFERENCES Student(sID),
    FOREIGN KEY(cName) REFERENCES College(collegeName)
);

INSERT INTO College VALUES ('Stanford', 'CA', 15000);
INSERT INTO College VALUES ('Berkeley', 'CA', 36000);
INSERT INTO College VALUES ('MIT', 'MA', 10000);
INSERT INTO College VALUES ('Cornell', 'NY', 21000);
INSERT INTO College VALUES ('Harvard', 'MA', 50040);

INSERT INTO Student VALUES (123, 'Amy', 3.9, 1000);
INSERT INTO Student VALUES (234, 'Bob', 3.6, 1500);
INSERT INTO Student VALUES (345, 'Craig', 3.5, 500);
INSERT INTO Student VALUES (456, 'Doris', 3.9, 1000);
INSERT INTO Student VALUES (567, 'Edward', 2.9, 2000);
INSERT INTO Student VALUES (678, 'Fay', 3.8, 200);
INSERT INTO Student VALUES (789, 'Gary', 3.4, 800);
INSERT INTO Student VALUES (987, 'Helen', 3.7, 800);
INSERT INTO Student VALUES (876, 'Irene', 3.9, 400);
INSERT INTO Student VALUES (765, 'Jay', 2.9, 1500);
INSERT INTO Student VALUES (654, 'Amy', 3.9, 1000);
INSERT INTO Student VALUES (543, 'Craig', 3.4, 2000);

INSERT INTO Apply VALUES (123, 'Stanford', 'CS', 'Y');
INSERT INTO Apply VALUES (123, 'Stanford', 'EE', 'N');
INSERT INTO Apply VALUES (123, 'Berkeley', 'CS', 'Y');
INSERT INTO Apply VALUES (123, 'Cornell', 'EE', 'Y');

```

```

INSERT INTO Apply VALUES (234, 'Berkeley', 'biology', 'N');
INSERT INTO Apply VALUES (345, 'MIT', 'bioengineering', 'Y');
INSERT INTO Apply VALUES (345, 'Cornell', 'bioengineering', 'N');
INSERT INTO Apply VALUES (345, 'Cornell', 'CS', 'Y');
INSERT INTO Apply VALUES (345, 'Cornell', 'EE', 'N');
INSERT INTO Apply VALUES (678, 'Stanford', 'history', 'Y');
INSERT INTO Apply VALUES (987, 'Stanford', 'CS', 'Y');
INSERT INTO Apply VALUES (987, 'Berkeley', 'CS', 'Y');
INSERT INTO Apply VALUES (876, 'Stanford', 'CS', 'N');
INSERT INTO Apply VALUES (876, 'MIT', 'biology', 'Y');
INSERT INTO Apply VALUES (876, 'MIT', 'marine biology', 'N');
INSERT INTO Apply VALUES (765, 'Stanford', 'history', 'Y');
INSERT INTO Apply VALUES (765, 'Cornell', 'history', 'N');
INSERT INTO Apply VALUES (765, 'Cornell', 'psychology', 'Y');
INSERT INTO Apply VALUES (543, 'MIT', 'CS', 'N');

SELECT * FROM apply;

```

---

## Q. Exercise:-

**Write SQL queries for each of the following:**

Q1. Produce a combine table in which each student is combine with every other application.

```

SELECT
    s.sID AS student_id,
    s.sName AS student_name,
    a.cName AS college_name,
    a.major AS major,
    a.decision AS decision
FROM
    Student s
CROSS JOIN
    Apply a
ORDER BY
    s.sID,
    a.cName;

```

Q2. Give Student ID, name, GPA and name of college and major each student applied to.

```
SELECT
    s.sID AS student_id,
    s.sName AS student_name,
    s.GPA,
    a.cName AS college_name,
    a.major
FROM
    Student s
JOIN
    Apply a ON s.sID = a.sID
ORDER BY
    s.sID,
    a.cName;
```

Q3. Find detail of applications who applied to California State.

```
SELECT
    s.sID AS student_id,
    s.sName AS student_name,
    s.GPA,
    c.collegeName AS college_name,
    a.major
FROM
    Apply a
JOIN
    Student s ON a.sID = s.sID
JOIN
    College c ON a.cName = c.collegeName
WHERE
    c.state = 'CA';
```

Q4. IDs, name, GPA of students and name of college with GPA > 3.7 applying to Stanford

```
SELECT
    s.sID AS student_id,
    s.sName AS student_name,
    s.GPA,
    c.collegeName AS college_name
```

```

FROM
    Student s
JOIN
    Apply a ON s.sID = a.sID
JOIN
    College c ON a.cName = c.collegeName
WHERE
    s.GPA > 3.7
    AND c.collegeName = 'Stanford';

```

Q5. Find detail of Student who apply to CS major and their application are rejected

```

SELECT
    s.sID AS student_id,
    s.sName AS student_name,
    s.GPA,
    c.collegeName AS college_name,
    a.major,
    a.decision
FROM
    Student s
JOIN
    Apply a ON s.sID = a.sID
JOIN
    College c ON a.cName = c.collegeName
WHERE
    a.major = 'CS'
    AND a.decision = 'N';

```

Q6. Find detail of student and application who applied to colleges at New York.

```

SELECT
    s.sID AS student_id,
    s.sName AS student_name,
    s.GPA,
    c.collegeName AS college_name,
    a.major,
    a.decision
FROM
    Student s
JOIN
    Apply a ON s.sID = a.sID
JOIN

```

```
College c ON a.cName = c.collegeName
WHERE
    c.state = 'NY';
```

Q7. Find detail of student who have not applied to any of college.

```
SELECT
    s.sID AS student_id,
    s.sName AS student_name,
    s.GPA
FROM
    Student s
LEFT JOIN
    Apply a ON s.sID = a.sID
WHERE
    a.sID IS NULL
```

Q8. Find college where no student have applied

```
SELECT
    c.collegeName AS college_name
FROM
    College c
LEFT JOIN
    Apply a ON c.collegeName = a.cName
WHERE
    a.cName IS NULL;
```

Q9. Find sID who have only one application.

```
SELECT
    sID
FROM
    Apply
GROUP BY
    sID
HAVING
    COUNT(*) = 1;
```

Q10. Find name and GPA of applicants who apply to any college whose enrollment is not more than 25000.



```

SELECT
    s.sName AS student_name,
    s.GPA
FROM
    Student s
JOIN
    Apply a ON s.sID = a.sID
JOIN
    College c ON a.cName = c.collegeName
WHERE
    c.enrollment <= 25000;

```

Q11. Find pair of students (sID) having same GPA. (*each pair should occur just once in result*)

```

SELECT
    DISTINCT s1.sID AS student1_id,
    s2.sID AS student2_id
FROM
    Student s1
JOIN
    Student s2 ON s1.sID < s2.sID AND s1.GPA = s2.GPA

```

---

## Exercise:

For each of the following you need to write three queries i.e. three version first using :CROSS Join Second using: Natural Join And third using: Inner Join

Q12. Find student and major he / she applied to.

```

-- Query using CROSS JOIN
SELECT s.sID, s.sName, a.major
FROM Student s
CROSS JOIN Apply a;

-- Query using NATURAL JOIN
SELECT s.sID, s.sName, a.major
FROM Student s
NATURAL JOIN Apply a;

```

```
-- Query using INNER JOIN
SELECT s.sID, s.sName, a.major
FROM Student s
INNER JOIN Apply a ON s.sID = a.sID;
```

Q13. Find detail of student who came from high school have size less than 20000 and applied to CSat Stanford.

```
-- Query using CROSS JOIN
SELECT s.sID, s.sName, s.GPA, s.sizeHS, a.cName, a.major
FROM Student s
CROSS JOIN Apply a
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS < 20000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND c.collegeName = 'Stanford';
```

```
-- Query using NATURAL JOIN
SELECT s.sID, s.sName, s.GPA, s.sizeHS, a.cName, a.major
FROM Student s
NATURAL JOIN Apply a
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS < 20000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND c.collegeName = 'Stanford';
```

```
-- Query using INNER JOIN
SELECT s.sID, s.sName, s.GPA, s.sizeHS, a.cName, a.major
FROM Student s
INNER JOIN Apply a ON s.sID = a.sID
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS < 20000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND c.collegeName = 'Stanford';
```

Q14. Provide complete detail of each student where they applied what major they applied to what was the decision and complete detail of college they applied.

```
-- Query using CROSS JOIN
SELECT s.sID, s.sName, s.GPA, s.sizeHS, a.cName, a.major
FROM Student s
CROSS JOIN Apply a
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS < 20000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND c.collegeName = 'Stanford';
```

```
-- Query using NATURAL JOIN
```

```

SELECT s.sID, s.sName, s.GPA, s.sizeHS, a.cName, a.major
FROM Student s
NATURAL JOIN Apply a
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS < 20000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND c.collegeName = 'Stanford';

-- Query using INNER JOIN
SELECT s.sID, s.sName, s.GPA, s.sizeHS, a.cName, a.major
FROM Student s
INNER JOIN Apply a ON s.sID = a.sID
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS < 20000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND c.collegeName = 'Stanford';

```

Q15. Names and GPAs of students with HS>1000 who applied to CS and were rejected

```

-- Query using CROSS JOIN
SELECT s.sName, s.GPA
FROM Student s
CROSS JOIN Apply a
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS > 1000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND a.decision = 'N';

-- Query using NATURAL JOIN
SELECT s.sName, s.GPA
FROM Student s
NATURAL JOIN Apply a
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS > 1000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND a.decision = 'N';

-- Query using INNER JOIN
SELECT s.sName, s.GPA
FROM Student s
INNER JOIN Apply a ON s.sID = a.sID
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS > 1000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND a.decision = 'N';

```

Q16. Names and GPAs of students with HS>1000 who applied to CS at college with enr>20,000 and were rejected.

```

-- Query using CROSS JOIN
SELECT s.sName, s.GPA
FROM Student s
CROSS JOIN Apply a
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS > 1000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND a.decision = 'N'
AND c.enrollment > 20000;

-- Query using NATURAL JOIN
SELECT s.sName, s.GPA
FROM Student s
NATURAL JOIN Apply a
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS > 1000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND a.decision = 'N'
AND c.enrollment > 20000;

-- Query using INNER JOIN
SELECT s.sName, s.GPA
FROM Student s
INNER JOIN Apply a ON s.sID = a.sID
JOIN College c ON a.cName = c.collegeName
WHERE s.sizeHS > 1000 AND s.sizeHS IS NOT NULL
AND a.major = 'CS' AND a.decision = 'N'
AND c.enrollment > 20000;

```

## Practical Assignment – 2

---

1. Create the following tables and specify constraints at the time of creation:

```

CREATE TABLE IF NOT EXISTS Department(
    Deptno INT PRIMARY KEY,
    Dname VARCHAR(20) UNIQUE,
    Location VARCHAR(20) NOT NULL,

```

```
CONSTRAINT location_check CHECK (Location IN ('Delhi', 'Pune', 'Agra'))
);
```

```
CREATE TABLE IF NOT EXISTS Employee (
    Empno VARCHAR(5) PRIMARY KEY,
    Ename VARCHAR(20) UNIQUE,
    Designation VARCHAR(20) NOT NULL,
    Salary INT DEFAULT 25000,
    DOB DATE NOT NULL,
    Dno INT,
    CONSTRAINT Ename_check CHECK (SUBSTRING(Ename, 1, 1) = 'E'),
    CONSTRAINT Salary_check CHECK (Salary BETWEEN 15000 AND 50000),
    CONSTRAINT fk_department_Dno FOREIGN KEY (Dno) REFERENCES Department(Deptno)
);
```

```
CREATE TABLE IF NOT EXISTS Candidate(
    Candidate_ID INT PRIMARY KEY,
    Candidate_Name VARCHAR(20) NOT NULL,
    Candidate_Email VARCHAR(20) UNIQUE,
    Candidate_Dept VARCHAR(20) DEFAULT 'HR',
    Manager_Id INT,
    CONSTRAINT email_format_check CHECK (
        POSITION('@' IN Candidate_Email) > 0 AND
        POSITION('.') IN Candidate_Email > POSITION('@' IN Candidate_Email)
    ),
    CONSTRAINT fk_manager_candidate FOREIGN KEY (Manager_ID) REFERENCES
Candidate(Candidate_ID)
);
```

---

## 2. Create the schemas as specified above without specifying any constraints.

```
CREATE TABLE IF NOT EXISTS College(
    cName VARCHAR(10),
    state VARCHAR(10),
    enrollment INT
```

```
);
```

```
CREATE TABLE IF NOT EXISTS Student(  
    sID INT,  
    sName VARCHAR(10),  
    GPA FLOAT,  
    sizeHS INT  
);
```

```
CREATE TABLE IF NOT EXISTS Apply(  
    sID INT,  
    cName VARCHAR(10),  
    major VARCHAR(20)  
);
```

Q1. Add cName as Primary key in College.

```
ALTER TABLE College ADD PRIMARY KEY (cName);
```

Q2. Add sID as Primary Key in student.

```
ALTER TABLE Student ADD PRIMARY KEY (sID);
```

Q3. Add sID, cName, major as Primary Key in Apply.

```
ALTER TABLE Apply ADD PRIMARY KEY (sID, cName, major);
```

Q4. Make sID in Apply foreign key referring table student and cName referring table college.

```
ALTER TABLE Apply ADD CONSTRAINT fk_student FOREIGN KEY (sID) REFERENCES  
student(sID);
```

```
ALTER TABLE Apply ADD CONSTRAINT fk_college FOREIGN KEY (cName) REFERENCES  
college(cName);
```

Q5. Increase data type size of major from 20 to 25.

```
ALTER TABLE Apply MODIFY COLUMN major VARCHAR(25);
```

Q6. Add a new column decision in the Apply table keeping a constraint of not null for this column with data type varchar(3).

```
ALTER TABLE Apply ADD COLUMN decision VARCHAR(3) NOT NULL;
```

Q7. Change data type of decision in Apply to char(1).

```
ALTER TABLE Apply MODIFY COLUMN decision CHAR(1);
```

Q8. Drop foreign key on column name cName from Apply table.

```
ALTER TABLE Apply DROP FOREIGN KEY fk_college;
```

Q9. Remove Column sizeHS from Student table.

```
ALTER TABLE Student DROP COLUMN sizeHS;
```

Q10. Drop primary key from college.

```
ALTER TABLE College DROP PRIMARY KEY;
```

Q11. Make cName, major unique pairwise such as Stanford CS, Stanford EE.

```
ALTER TABLE Apply ADD CONSTRAINT uc_cName_major UNIQUE (cName, major);
```

Q12. Add cName as Foreign Key in Apply table referring table College using on delete cascade.

```
ALTER TABLE College ADD INDEX idx_cName (cName);  
ALTER TABLE Apply ADD CONSTRAINT fk_college FOREIGN KEY (cName) REFERENCES  
College(cName) ON DELETE CASCADE;
```

Q13. Rename Column enrollment to enroll in college table.

```
ALTER TABLE Student CHANGE COLUMN enrollment enroll VARCHAR(255);
```

---

## Exercise:

Customer:

```
CREATE TABLE IF NOT EXISTS CUSTOMER(  
    CustomerId VARCHAR(6) PRIMARY KEY,  
    CustomerName VARCHAR(30) NOT NULL,  
    DateOfReg DATE,  
    UserId VARCHAR(15) UNIQUE,  
    Password VARCHAR(15) NOT NULL,  
    CONSTRAINT CustomerId_check CHECK (SUBSTRING(CustomerId, 1, 1) = 'C')  
);
```

BankInfo:

```
CREATE TABLE IF NOT EXISTS BankInfo(  
    AccountNo INT,  
    CustomerId VARCHAR(6),  
    CONSTRAINT PK_BankInfo PRIMARY KEY (AccountNo, CustomerId),  
    CONSTRAINT CustomerId_fk FOREIGN KEY(CustomerId) REFERENCES  
Customer(CustomerId) ON DELETE CASCADE  
);
```

Billing:

```
CREATE TABLE IF NOT EXISTS Billing(  
    BillId INT PRIMARY KEY,  
    AccountNo INT,  
    CustomerId VARCHAR(6),  
    BillDate DATE DEFAULT CURRENT_DATE,  
    PaymentType ENUM('creditcard', 'debitcard'),  
    CONSTRAINT FK_AccountNo_CustomerId FOREIGN KEY(AccountNo, CustomerId)  
REFERENCES BankInfo(AccountNo, CustomerId) ON DELETE CASCADE  
);
```



```
INSERT INTO Billing (BillId, AccountNo, CustomerId, PaymentType) VALUES (1, 12345, 'ABCDEF', 'creditcard');
```

Item:

```
CREATE TABLE IF NOT EXISTS Item(  
    ItemId VARCHAR(6) PRIMARY KEY,  
    ItemName VARCHAR(30) NOT NULL,  
    QtyOnHand INT CHECK (QtyOnHand > 0),  
    UnitPrice INT CHECK (UnitPrice > 0),  
    Class CHAR(1),  
    UnitOfMeasurement VARCHAR(12),  
    ReOrderLevel INT CHECK (ReOrderLevel > 0),  
    ReOrderQty INT CHECK (ReOrderQty > 0),  
    Discount INT,  
    CONSTRAINT QtyOnHand_check CHECK(QtyOnHand > ReOrderLevel),  
    CONSTRAINT Class_check CHECK(  
        (Class = 'A' AND UnitPrice < 100) OR  
        (Class = 'B' AND UnitPrice < 1000 AND UnitPrice >= 100) OR  
        (Class = 'C' AND UnitPrice >= 1000)  
    )  
);
```

## Practical Assignment - 1

---

**Q. Create the following tables and fill the given data in them:-**

Table-1

```
CREATE DATABASE kt;  
  
USE kt;  
  
Create Table IF NOT EXISTS College(  
    cName VARCHAR(10),  
    state VARCHAR(10),
```

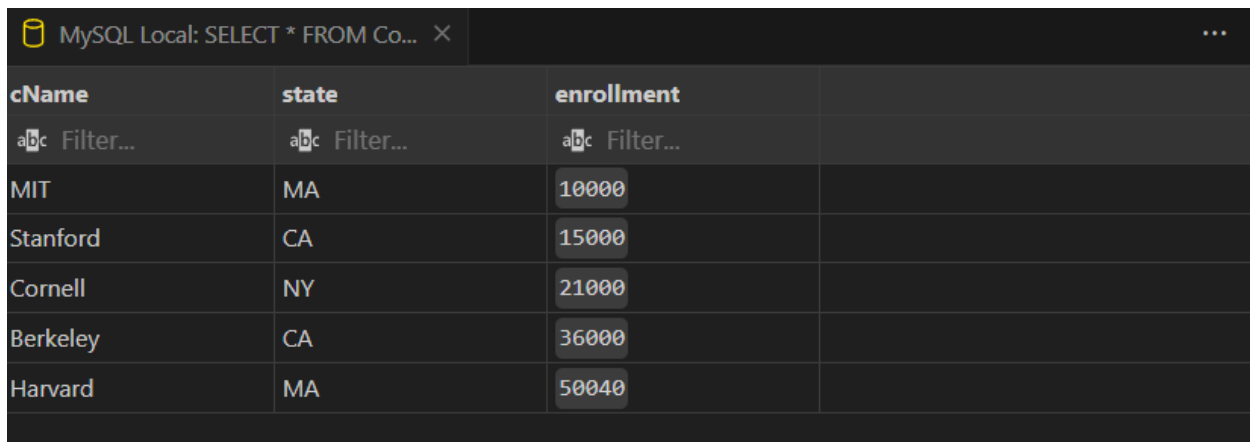
```

    enrollment INT PRIMARY KEY NOT NULL
);

INSERT INTO College(cName, state, enrollment) VALUES('Stanford', 'CA', '15000');
INSERT INTO College(cName, state, enrollment) VALUES('Berkeley', 'CA', '36000');
INSERT INTO College(cName, state, enrollment) VALUES('MIT', 'MA', '10000');
INSERT INTO College(cName, state, enrollment) VALUES('Cornell', 'NY', '21000');
INSERT INTO College(cName, state, enrollment) VALUES('Harvard', 'MA', '50040');
SELECT * FROM College;

```

OUTPUT:



cName	state	enrollment	
abc Filter...	abc Filter...	abc Filter...	
MIT	MA	10000	
Stanford	CA	15000	
Cornell	NY	21000	
Berkeley	CA	36000	
Harvard	MA	50040	

Table -2

```

Create Table IF NOT EXISTS Student(
    sID INT PRIMARY KEY,
    sName VARCHAR(10),
    GPA FLOAT,
    sizeHS INT NOT NULL,
    DoB DATE
);

```

```
INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('123', 'Amy', '3.9',
'1000', '1996-06-26');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('234', 'Bob', '3.6',
'1500', '1995-04-07');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('345', 'Craig', '3.5',
'500', '1995-02-04');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('456', 'Doris', '3.9',
'1000', '1997-07-24');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('567', 'Edward', '2.9',
'2000', '1996-12-21');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('678', 'Fay', '3.8',
'200', '1996-08-27');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('789', 'Gary', '3.4',
'800', '1996-10-08');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('987', 'Helen', '3.7',
'800', '1997-03-27');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('876', 'Irene', '3.9',
'400', '1996-03-07');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('765', 'Jay', '2.9',
'1500', '1998-08-08');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('654', 'Amy', '3.9',
'1000', '1996-05-26');

INSERT INTO Student(sID, sName, GPA, sizeHS, DoB) VALUES('543', 'Craig', '3.4',
'2000', '1998-08-27');

SELECT * FROM Student;
```

OUTPUT:

sID	sName	GPA	sizeHS	DoB	
123	Amy	3.9000000953674316	1000	1996-06-26T00:00:00.000Z	
234	Bob	3.5999999046325684	1500	1995-04-07T00:00:00.000Z	
345	Craig	3.5000000000000004	500	1995-02-04T00:00:00.000Z	
456	Doris	3.9000000953674316	1000	1997-07-24T00:00:00.000Z	
543	Craig	3.4000000953674316	2000	1998-08-27T00:00:00.000Z	
567	Edward	2.9000000953674316	2000	1996-12-21T00:00:00.000Z	
654	Amy	3.9000000953674316	1000	1996-05-26T00:00:00.000Z	
678	Fay	3.799999952316284	200	1996-08-27T00:00:00.000Z	
765	Jay	2.9000000953674316	1500	1998-08-08T00:00:00.000Z	
789	Gary	3.4000000953674316	800	1996-10-08T00:00:00.000Z	
876	Irene	3.9000000953674316	400	1996-03-07T00:00:00.000Z	
987	Helen	3.7000000476837163	800	1997-03-27T00:00:00.000Z	

Table – 3

```

Create Table IF NOT EXISTS Apply(
    sID INT NOT NULL,
    cName VARCHAR(10),
    major VARCHAR(20),
    decision VARCHAR(1)
);

INSERT INTO Apply(sID, cName, major, decision) VALUES('123', 'Stanford', 'CS',
'Y');

INSERT INTO Apply(sID, cName, major, decision) VALUES('123', 'Stanford', 'EE',
'N');

INSERT INTO Apply(sID, cName, major, decision) VALUES('123', 'Berkeley', 'CS',
'Y');

INSERT INTO Apply(sID, cName, major, decision) VALUES('123', 'Cornell', 'EE',
'Y');

INSERT INTO Apply(sID, cName, major, decision) VALUES('234', 'Berkeley',
'biology', 'N');

INSERT INTO Apply(sID, cName, major, decision) VALUES('345', 'MIT',
'bioengineering', 'Y');

```

```
INSERT INTO Apply(sID, cName, major, decision) VALUES('345', 'Cornell',
'bioengineering', 'N');

INSERT INTO Apply(sID, cName, major, decision) VALUES('345', 'Cornell', 'CS',
'Y');

INSERT INTO Apply(sID, cName, major, decision) VALUES('345', 'Cornell', 'EE',
'N');

INSERT INTO Apply(sID, cName, major, decision) VALUES('678', 'Stanford',
'history', 'Y');

INSERT INTO Apply(sID, cName, major, decision) VALUES('987', 'Stanford', 'CS',
'Y');

INSERT INTO Apply(sID, cName, major, decision) VALUES('987', 'Berkeley', 'CS',
'Y');

INSERT INTO Apply(sID, cName, major, decision) VALUES('876', 'Stanford', 'CS',
'N');

INSERT INTO Apply(sID, cName, major, decision) VALUES('876', 'MIT', 'biology',
'Y');

INSERT INTO Apply(sID, cName, major, decision) VALUES('876', 'MIT', 'marine
biology', 'N');

INSERT INTO Apply(sID, cName, major, decision) VALUES('765', 'Stanford',
'history', 'Y');

INSERT INTO Apply(sID, cName, major, decision) VALUES('765', 'Cornell',
'history', 'N');

INSERT INTO Apply(sID, cName, major, decision) VALUES('765', 'Cornell',
'psychology', 'Y');

INSERT INTO Apply(sID, cName, major, decision) VALUES('543', 'MIT', 'CS', 'N');

SELECT * FROM Apply;
```

OUTPUT:

sID	cName	major	decision	
Filter...	Filter...	Filter...	Filter...	
123	Stanford	CS	Y	
123	Stanford	EE	N	
123	Berkeley	CS	Y	
123	Cornell	EE	Y	
234	Berkeley	biology	N	
345	MIT	bioengineering	Y	
345	Cornell	bioengineering	N	
345	Cornell	CS	Y	
345	Cornell	EE	N	
678	Stanford	history	Y	
987	Stanford	CS	Y	
987	Berkeley	CS	Y	
876	Stanford	CS	N	
876	MIT	biology	Y	
876	MIT	marine biology	N	
765	Stanford	history	Y	
765	Cornell	history	N	
765	Cornell	psychology	Y	
543	MIT	CS	N	

## Q. State of SQL \*PLUS Queries for each of the following:

1. List the student name, dob from student table

```
SELECT sName, DoB FROM student;
```

sName	DoB
Filter...	Filter...
Amy	1996-06-26T00:00:00.000Z
Bob	1995-04-07T00:00:00.000Z
Craig	1995-02-04T00:00:00.000Z
Doris	1997-07-24T00:00:00.000Z
Craig	1998-08-27T00:00:00.000Z
Edward	1996-12-21T00:00:00.000Z
Amy	1996-05-26T00:00:00.000Z
Fay	1996-08-27T00:00:00.000Z
Jay	1998-08-08T00:00:00.000Z
Gary	1996-10-08T00:00:00.000Z
Irene	1996-03-07T00:00:00.000Z
Helen	1997-03-27T00:00:00.000Z

2. List of the name of student scoring more than 3.7 in GPA.

```
SELECT sName FROM student WHERE GPA>3.7;
```

sName
abc Filter...
Amy
Doris
Amy
Fay
Irene
Helen

- List the name of student whose High School size is atleast 1000 and born after 1996.

```
SELECT sName FROM student WHERE sizeHS >= 1000 AND DoB > '1996-12-31';
```

sName
abc Filter...
Doris
Craig
Jay

- List the name of student who are scoring GPA between 2.9 and 3.9.

```
SELECT sName FROM student WHERE GPA >= 2.9 AND GPA <= 3.9;
```

sName
abc Filter...
Bob
Craig
Craig
Edward
Fay
Jay
Gary
Helen

- List all the details of colleges who are situated in MA

```
SELECT * FROM college WHERE state = 'MA' ;
```

cName	state	enrollment
abc Filter...	abc Filter...	abc Filter...
MIT	MA	10000
Harvard	MA	50040

6. List the students who have scored more than 2.0 but less than 3.5.

```
SELECT sName FROM student WHERE GPA >= 2.0 AND GPA <= 3.5;
```

sName
abc Filter...
Craig
Craig
Edward
Jay
Gary

7. List the name of students who are born after 1 st Jul 1996 in the order of Date of Birth.

```
SELECT sName, DoB FROM student WHERE DoB > '1996-07-01' ORDER BY DoB ASC;
```

sName	DoB
abc Filter...	abc Filter...
Fay	1996-08-27T00:00:00.000Z
Gary	1996-10-08T00:00:00.000Z
Edward	1996-12-21T00:00:00.000Z
Helen	1997-03-27T00:00:00.000Z
Doris	1997-07-24T00:00:00.000Z
Jay	1998-08-08T00:00:00.000Z
Craig	1998-08-27T00:00:00.000Z

8. List the sID, cName, decision of applications that are accepted.

```
SELECT sID, cName, decision FROM apply WHERE decision = 'Y';
```



sID	cName	decision
abc Filter...	abc Filter...	abc Filter...
123	Stanford	Y
123	Berkeley	Y
123	Cornell	Y
345	MIT	Y
345	Cornell	Y
678	Stanford	Y
987	Stanford	Y
987	Berkeley	Y
876	MIT	Y
765	Stanford	Y
765	Cornell	Y

9. List the sID, cName of applications that are filed at Stanford.

```
SELECT sID,cName FROM apply WHERE cName = 'Stanford';
```

sID	cName
abc Filter...	abc Filter...
123	Stanford
123	Stanford
678	Stanford
987	Stanford
876	Stanford
765	Stanford

10. List all the colleges that has enrollment greater than 1000.

```
SELECT cName FROM college WHERE enrollment > '1000';
```

cName
abc Filter...
MIT
Stanford
Cornell
Berkeley
Harvard

11. List the colleges not in California.

```
SELECT cName FROM college WHERE state != 'CA';
```

cName
abc Filter...
MIT
Cornell
Harvard

12. List names of all students who came from high school having size greater than 1700 and scored GPA less than 3.8.

```
SELECT sName FROM student WHERE sizeHS > '1700' AND GPA < 3.8;
```

sName
abc Filter...
Craig
Edward

13. Display the description of student table.

```
DESCRIBE student;
```

Field	Type	Null	Key	Default	Extra
abc Filter...	abc Filter...	abc Filter...	abc Filter...	abc Filter...	abc Filter...
sID	int	NO	PRI	NULL	
sName	varchar(10)	YES		NULL	
GPA	float	YES		NULL	
sizeHS	int	NO		NULL	
DoB	date	YES		NULL	

14. Display the details of all students.

```
SELECT * FROM student;
```

sID	sName	GPA	sizeHS	DoB
abc Filter...	abc Filter...	abc Filter...	abc Filter...	abc Filter...
123	Amy	3.9000000953674316	1000	1996-06-26T00:00:00.000Z
234	Bob	3.5999999046325684	1500	1995-04-07T00:00:00.000Z
345	Craig	3.5000000000000004	500	1995-02-04T00:00:00.000Z
456	Doris	3.9000000953674316	1000	1997-07-24T00:00:00.000Z
543	Craig	3.4000000953674316	2000	1998-08-27T00:00:00.000Z
567	Edward	2.9000000953674316	2000	1996-12-21T00:00:00.000Z
654	Amy	3.9000000953674316	1000	1996-05-26T00:00:00.000Z
678	Fay	3.799999952316284	200	1996-08-27T00:00:00.000Z
765	Jay	2.9000000953674316	1500	1998-08-08T00:00:00.000Z
789	Gary	3.4000000953674316	800	1996-10-08T00:00:00.000Z
876	Irene	3.9000000953674316	400	1996-03-07T00:00:00.000Z
987	Helen	3.7000000476837163	800	1997-03-27T00:00:00.000Z

15. Display unique majors.

```
SELECT DISTINCT major FROM apply;
```

major
abc Filter...
CS
EE
biology
bioengineering
history
marine biology
psychology

16. List the student names those are having three characters in their Names

```
SELECT sName FROM student WHERE CHAR_LENGTH(sName) = 3;
```

sName
abc Filter...
Amy
Bob
Amy
Fay
Jay

17. List the student names those are starting with 'H' and with five characters.

```
SELECT sName FROM student WHERE sName LIKE 'H%' AND CHAR_LENGTH(sName)=5;
```

sName
abc Filter...
Helen

18. List the student names that are having third character and fifth character must be 'e'.

```
SELECT sName FROM student WHERE CHAR_LENGTH(sName) >= 5 AND SUBSTRING(sName, 3, 1) = 'e' AND SUBSTRING(sName, 5, 1) = 'e';
```

sName
abc Filter...
Irene

19. List the student names ending with 'Y'

```
SELECT sName FROM student WHERE RIGHT(sName , 1) = 'Y';
```

sName
abc Filter...
Amy
Amy
Fay
Jay
Gary

20. List the students in the order of their GPA.

```
SELECT sName,GPA FROM student ORDER BY GPA ASC;
```

sName	GPA
abc Filter...	abc Filter...
Edward	2.9
Jay	2.9
Craig	3.4
Gary	3.4
Craig	3.5
Bob	3.6
Helen	3.7
Fay	3.8
Amy	3.9
Doris	3.9
Amy	3.9
Irene	3.9

21. List the details of students in order of the ascending of GPA and descending of DoB.

```
SELECT * FROM student ORDER BY GPA ASC , DoB DESC;
```

sID	sName	GPA	sizeHS	DoB
abc Filter...	abc Filter...	abc Filter...	abc Filter...	abc Filter...
765	Jay	2.9	1500	1998-08-08
567	Edward	2.9	2000	1996-12-21
543	Craig	3.4	2000	1998-08-27
789	Gary	3.4	800	1996-10-08
345	Craig	3.5	500	1995-02-04
234	Bob	3.6	1500	1995-04-07
987	Helen	3.7	800	1997-03-27
678	Fay	3.8	200	1996-08-27
456	Doris	3.9	1000	1997-07-24
123	Amy	3.9	1000	1996-06-26
654	Amy	3.9	1000	1996-05-26
876	Irene	3.9	400	1996-03-07

22. List the sIDs of students who apply in either 'Stanford', 'Cornell' or 'MIT' college.

```
SELECT sID FROM apply WHERE cName IN ('Stanford','Cornell','MIT');
```

sID
abc Filter...
123
123
123
345
345
345
345
345
678
987
876
876
876
765
765
765
543

23.

23. Delete all applications filed at Stanford.



```
DELETE FROM apply WHERE cName = 'Stanford';
SELECT * FROM apply;
```

sID	cName	major	decision
abc Filter...	abc Filter...	abc Filter...	abc Filter...
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
987	Berkeley	CS	Y
876	MIT	biology	Y
876	MIT	marine biology	N
765	Cornell	history	N
765	Cornell	psychology	Y
543	MIT	CS	N

24. Modify the GPA of all students by giving 10% raise in GPA.

```
UPDATE student SET GPA = GPA + (0.10*GPA);
SELECT * FROM student;
```






sID	sName	GPA	sizeHS	DoB
abc Filter...	abc Filter...	abc Filter...	abc Filter...	abc Filter...
123	Amy	4.29	1000	1996-06-26
234	Bob	3.96	1500	1995-04-07
345	Craig	3.85	500	1995-02-04
456	Doris	4.29	1000	1997-07-24
543	Craig	3.74	2000	1998-08-27
567	Edward	3.19	2000	1996-12-21
654	Amy	4.29	1000	1996-05-26
678	Fay	4.18	200	1996-08-27
765	Jay	3.19	1500	1998-08-08
789	Gary	3.74	800	1996-10-08
876	Irene	4.29	400	1996-03-07
987	Helen	4.07	800	1997-03-27

25. Delete the college Stanford from the table.

```
DELETE FROM college WHERE cName = 'Stanford';
```

26. Increment the GPA of students by 1.5 whose GPA is less than 3.5 and whose High School Size is greater than 1500.

```
Update student SET GPA = GPA + 1.5 WHERE GPA < 3.5 AND sizeHS > 1500;  
SELECT * FROM student;
```

sID	sName	GPA	sizeHS	DoB
 Filter...	 Filter...	 Filter...	 Filter...	 Filter...
123	Amy	4.29	1000	1996-06-26
234	Bob	3.96	1500	1995-04-07
345	Craig	3.85	500	1995-02-04
456	Doris	4.29	1000	1997-07-24
543	Craig	3.74	2000	1998-08-27
567	Edward	4.69	2000	1996-12-21
654	Amy	4.29	1000	1996-05-26
678	Fay	4.18	200	1996-08-27
765	Jay	3.19	1500	1998-08-08
789	Gary	3.74	800	1996-10-08
876	Irene	4.29	400	1996-03-07
987	Helen	4.07	800	1997-03-27

27. Delete the students who have scored less than 3.2 GPA.

```
DELETE * FROM student WHERE GPA < 3.2;  
SELECT * FROM student;
```

---

## Exercise:

- Q. Determine the appropriate datatype:

- Deptno – INT



- Dname – VARCHAR(255)
- Loc – VARCHAR(255)

```
CREATE TABLE IF NOT EXISTS Dept(
    deptno INT PRIMARY KEY,
    dname VARCHAR(255),
    loc VARCHAR(255)
);

INSERT INTO Dept(deptno, dname, loc) VALUES('1', 'ACCOUNTING', 'ST LOUIS');

INSERT INTO Dept(deptno, dname, loc) VALUES('2', 'RESEARCH', 'NEW YORK');

INSERT INTO Dept(deptno, dname, loc) VALUES('3', 'SALES', 'ATLANTA');

INSERT INTO Dept(deptno, dname, loc) VALUES('4', 'OPERATIONS', 'SEATTLE');

SELECT * FROM Dept;
```

deptno	dname	loc
abc Filter...	abc Filter...	abc Filter...
1	ACCOUNTING	ST LOUIS
2	RESEARCH	NEW YORK
3	SALES	ATLANTA
4	OPERATIONS	SEATTLE

Q. Determine the appropriate datatype:

- empno – INT
- ename – VARCHAR(255)
- job – VARCHAR(255)
- mgr – VARCHAR(255)
- hiredate – DATE
- sal – INT
- comm – VARCHAAR(255)
- dept - INT

```
CREATE TABLE IF NOT EXISTS Employee(  
    empno INT PRIMARY KEY,  
    ename VARCHAR(255),  
    job VARCHAR(255),  
    mgr VARCHAR(255),  
    hierdate DATE,  
    sal INT NOT NULL,  
    comm VARCHAR(255),  
    dept INT NOT NULL  
);  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('1', 'JOHNSON', 'ADMIN', '6', '1990-12-17', '18000', '(null)', '4');  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('2', 'HARDING', 'MANAGER', '9', '1998-02-02', '52000', '300', '3');  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('3', 'TAFT', 'SALES 1', '2', '1996-01-02', '25000', '500', '3');  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('4', 'HOOVER', 'SALES 1', '2', '1990-04-02', '27000', '(null)', '3');  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('5', 'LINCOLN', 'TECH', '6', '1994-06-23', '22500', '1400', '4');  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('6', 'GARFIELD', 'MANAGER', '9', '1993-05-01', '54000', '(null)', '4');  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('7', 'POLK', 'TECH', '6', '1997-09-27', '25000', '(null)', '4');  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('8', 'GRANT', 'ENGINEER', '10', '1997-03-30', '32000', '(null)', '2');  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('9', 'JACKSON', 'CEO', '(null)', '1990-01-01', '75000', '(null)', '4');  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('10', 'FILLMORE', 'MANAGER', '9', '1994-08-09', '56000', '(null)', '2');  
  
INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES  
('11', 'ADAMAS', 'ENGINEER', '10', '1996-03-15', '34000', '(null)', '2');
```

```

INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES
('12', 'WASHINGTON', 'ADMIN', '6', '1998-04-16', '18000', '(null)', '4');

INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES
('13', 'MONROE', 'ENGINEER', '10', '2000-12-03', '30000', '(null)', '2');

INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES
('14', 'ROOSEVELT', 'CPA', '9', '1995-10-12', '35000', '(null)', '1');

INSERT INTO Employee(empno, ename, job, mgr, hierdate, sal, comm, dept) VALUES
('15', 'HANCOCK', 'SALES 1', '2', '1990-03-02', '27500', '(null)', '3');

SELECT * FROM employee;


```

empno	ename	job	mgr	hierdate	sal	comm	dept
1	JOHNSON	ADMIN	6	1990-12-17	18000	(null)	4
2	HARDING	MANAGER	9	1998-02-02	52000	300	3
3	TAFT	SALES 1	2	1996-01-02	25000	500	3
4	HOOVER	SALES 1	2	1990-04-02	27000	(null)	3
5	LINCOLN	TECH	6	1994-06-23	22500	1400	4
6	GARFIELD	MANAGER	9	1993-05-01	54000	(null)	4
7	POLK	TECH	6	1997-09-27	25000	(null)	4
8	GRANT	ENGINEER	10	1997-03-30	32000	(null)	2
9	JACKSON	CEO	(null)	1990-01-01	75000	(null)	4
10	FILLMORE	MANAGER	9	1994-08-09	56000	(null)	2
11	ADAMAS	ENGINEER	10	1996-03-15	34000	(null)	2
12	WASHINGTON	ADMIN	6	1998-04-16	18000	(null)	4
13	MONROE	ENGINEER	10	2000-12-03	30000	(null)	2
14	ROOSEVELT	CPA	9	1995-10-12	35000	(null)	1
15	HANCOCK	SALES 1	2	1990-03-02	27500	(null)	3

## Solve the following queries:

Q1. Employee Name and Hire Date sorted by Hire Date(Recent to Old).

```
SELECT ename,hierdate FROM employee ORDER BY hierdate DESC;
```

ename	hiredate
 Filter...	 Filter...
MONROE	2000-12-03
WASHINGTON	1998-04-16
HARDING	1998-02-02
POLK	1997-09-27
GRANT	1997-03-30
ADAMAS	1996-03-15
TAFT	1996-01-02
ROOSEVELT	1995-10-12
FILLMORE	1994-08-09
LINCOLN	1994-06-23
GARFIELD	1993-05-01
JOHNSON	1990-12-17
HOOVER	1990-04-02
HANCOCK	1990-03-02
JACKSON	1990-01-01

Q2. Employee Name and Job sorted by Job(Alphabetically).

```
SELECT ename,job FROM employee ORDER BY job ASC;
```

ename	job
 Filter...	 Filter...
JOHNSON	ADMIN
WASHINGTON	ADMIN
JACKSON	CEO
ROOSEVELT	CPA
GRANT	ENGINEER
ADAMAS	ENGINEER
MONROE	ENGINEER
HARDING	MANAGER
GARFIELD	MANAGER
FILLMORE	MANAGER
TAFT	SALES 1
HOOVER	SALES 1
HANCOCK	SALES 1
LINCOLN	TECH
POLK	TECH





Q3. Employee Name and Job for all Engineers, sorted by Employee Name Alphabetically.

```
SELECT ename,job FROM employee ORDER BY ename ASC;
```

ename	job
 Filter...	 Filter...
ADAMAS	ENGINEER
FILLMORE	MANAGER
GARFIELD	MANAGER
GRANT	ENGINEER
HANCOCK	SALES 1
HARDING	MANAGER
HOOVER	SALES 1
JACKSON	CEO
JOHNSON	ADMIN
LINCOLN	TECH
MONROE	ENGINEER
POLK	TECH
ROOSEVELT	CPA
TAFT	SALES 1
WASHINGTON	ADMIN

Q4. Job, Employee Name , Salary and Commission for employees with salary over 50000 sorted by Salary (Largest to Smallest).

```
SELECT job,ename,sal,comm FROM employee WHERE sal > '50000' ORDER BY sal DESC;
```

job	ename	sal	comm
 Filter...	 Filter...	 Filter...	 Filter...
CEO	JACKSON	75000	(null)
MANAGER	FILLMORE	56000	(null)
MANAGER	GARFIELD	54000	(null)
MANAGER	HARDING	52000	300

Q5. Job, Employee Name, Salary and Commission for employees with a Commission sorted by Salary (Largest to Smallest).

```
SELECT job,ename,sal,comm FROM employee WHERE comm != '(null)' ORDER BY sal DESC;
```

job	ename	sal	comm
abc Filter...	abc Filter...	abc Filter...	abc Filter...
MANAGER	HARDING	52000	300
SALES 1	TAFT	25000	500
TECH	LINCOLN	22500	1400

Q6. Job, Employee Name, Salary and Commission for employees whose name starts with the letter 'H'.

```
SELECT job,ename,sal,comm FROM employee WHERE SUBSTRING(ename, 1, 1) = 'H';
```

job	ename	sal	comm
abc Filter...	abc Filter...	abc Filter...	abc Filter...
MANAGER	HARDING	52000	300
SALES 1	HOOVER	27000	(null)
SALES 1	HANCOCK	27500	(null)

Q7. Job, Employee Name, Salary and Commission for employees whose name starts with the letter 'H' and who do not get any Commission.

```
SELECT job,ename,sal,comm FROM employee WHERE SUBSTRING(ename, 1, 1) = 'H' AND comm = '(null)';
```

job	ename	sal	comm
abc Filter...	abc Filter...	abc Filter...	abc Filter...
SALES 1	HOOVER	27000	(null)
SALES 1	HANCOCK	27500	(null)

Q8. Job, Employee Name for employees in Dept No. 3.

```
SELECT job,ename FROM employee WHERE dept = '3';
```

job	ename
abc Filter...	abc Filter...
MANAGER	HARDING
SALES 1	TAFT
SALES 1	HOOVER
SALES 1	HANCOCK

Q9. Dept Name and Loc for employees in Dept No. 3.

```
SELECT dname,loc FROM dept WHERE deptno = '3';
```

dname	loc
abc Filter...	abc Filter...
SALES	ATLANTA

Q10. Job, Employee Name, Dept, Salary sorted first by Dept(Smallest to Largest) and then Salary(Largest to Smallest).

```
SELECT job,ename,dept,sal FROM employee ORDER BY dept ASC,sal DESC;
```

job	ename	dept	sal
abc Filter...	abc Filter...	abc Filter...	abc Filter...
CPA	ROOSEVELT	1	35000
MANAGER	FILLMORE	2	56000
ENGINEER	ADAMAS	2	34000
ENGINEER	GRANT	2	32000
ENGINEER	MONROE	2	30000
MANAGER	HARDING	3	52000
SALES 1	HANCOCK	3	27500
SALES 1	HOOVER	3	27000
SALES 1	TAFT	3	25000
CEO	JACKSON	4	75000
MANAGER	GARFIELD	4	54000
TECH	POLK	4	25000
TECH	LINCOLN	4	22500
ADMIN	JOHNSON	4	18000
ADMIN	WASHINGTON	4	18000

- 
- Submitted By : Hardik Arora
  - Branch : Btech - CS
  - Program: AIML
  - University Roll No. : 2215500071
  - Section: 2AC
  - Class Roll No. : 28
  - Submitted to: Ayushi Mam