Lab Assignment -1

Experiment No: 1

Environment: Microsoft Windows
Tools/ Language: MySQL

Objective: Write the SQL queries for data definition and data manipulation language.

Theory & Concepts:

Introduction about SQL
SQL (Structured Query Language) is a nonprocedural language, you specify what you want, not how to get it. A block structured format of English key words is used in this Query language. It has the following components.
DDL (Data Definition Language)
DML (DATA Manipulation Language)
View definition
Transaction Control
Embedded SQL and Dynamic SQL
Integrity
Authorization

Data Definition Language
The SQL DDL allows specification of not only a set of relations but also information about each relation, including-

- Schema for each relation
- The domain of values associated with each attribute.
- The integrity constraints.
- The set of indices to be maintained for each relation.
- The security and authorization information for each relation.
- The physical storage structure of each relation on disk.

Domain types in SQL
The SQL standard supports a variety of built in domain types, including-

- Char (n)- A fixed length character length string with user specified length .
- Varchar (n)- A variable character length string with user specified maximum length n.
- Number (p, d)-A Fixed point number with user defined precision.
- Real, double precision- Floating point and double precision floating point numbers with machine dependent precision.
- Float (n)- A floating point number, with precision of at least n digits.
- Date- A calendar date containing a (four digit) year, month and day of the month.
- Time- The time of day, in hours, minutes and seconds Eg. Time ’09:30:00’.

DDL statement for creating a table

Syntax-

```
Create table tablename
(columnname datatype(size),  columnname datatype(size));
```

Creating a table from a table-

Syntax-

```
CREATE TABLE TABLENAME
[(columnname, columnname, ………)]
AS SELECT columnname, columnname……..FROM tablename;
```

Rules:

1. MySQL reserved words cannot be used.
2. Underscore, numerals, letters are allowed but not blank space.
3. Maximum length for the table name is 30 characters.
4. 2 different tables should not have same name.
5. We should specify a unique column name.
6. We should specify proper data type along with width.
7. We can include "not null" condition when needed. By default it is 'null'.

Insertion of data into tables

Syntax-

```
INSERT INTO tablename
[(columnname, columnname, ………)]
Values(expression, expression);
```

Inserting data into a table from another table:

Syntax-

```
INSERT INTO tablename
SELECT columnname, columnname, …….
FROM tablename;
```

Insertion of selected data into a table from another table:

Syntax-

```
INSERT INTO tablename
SELECT columnname, columnname……..
FROM tablename
WHERE columnname= expression;
```

Retrieving of data from the tables-

Syntax-`SELECT * FROM tablename;`

The retrieving of specific columns from a table-

Syntax-

```
SELECT  columnname, columnname, ….
FROM tablename;
```

Elimination of duplicates from the select statement-

Syntax-

```
SELECT DISTINCT columnname, columnname
FROM tablename;
```

Selecting a data set from table data-
Syntax
```
SELECT columnname, columnname
FROM tablenameWHERE search condition;
```

The SELECT DISTINCT * SQL syntax scans through entire rows, and eliminates rows that have exactly the same contents in each column.

Syntax:
```
SELECT DISTINCT *
FROM TableName;
```

DML (Data Manipulation Language)
Data manipulation is

- The retrieval of information stored in the database.
- The insertion of new information into the database.
- The deletion of information from the database.
- The modification of information stored by the appropriate data model. There are basically two types.

(i) **Procedural DML**: require a user to specify what data are needed and how to get those data.
(ii) **Non Procedural DML**: require a user to specify what data are needed without specifying how to get those data.

Updating the content of a table:
In creation situation we may wish to change a value in table without changing all values in the tuple . For this purpose the update statement can be used.
```
Update table name
Set columnname = expression, columnname =expression……
Where columnname = expression;
```

Deletion Operation:-
We can delete whole tuple (rows) we can delete values on only particulars attributes.
Deletion of all rows
Syntax:
```
Delete from tablename;
```

Deletion of specified number of rows
Syntax:
```
Delete from table name
where search condition;
```

Computation in expression lists used to select data:-

+	Addition	-	Subtraction
*	multiplication	**	exponentiation
/	Division	()	Enclosed operation

Renaming columns used with Expression Lists: - The default output column names can be renamed by the user if required

Syntax:
```
Select column name result_columnname, columnname result_columnname
From tablename;
```

Logical Operators:
The logical operators that can be used in SQL sentenced are

AND	all of must be included
OR	any of may be included
NOT	none of could be included

Range Searching: Between operation is used for range searching.

Pattern Searching:
The most commonly used operation on string is pattern matching using the operation ‘like’ we describe patterns by using two special characters.

- Percent (%): the % character matches any substring we consider the following examples.
    - ‘Perry %’ matches any string beginning with perry
    - ‘% idge % matches any string containing’ idge as substring.
    - ‘ - - - ‘ matches any string exactly three characters.
    - ‘ - - - % matches any string of at least of three characters.

Ordering tuples in a particular order:

- The ‘order by’ clause is used to sort the table data according to one or more columns of the table.
- The table rows are ordered in ascending order of the column values by default. The keyword used for the same is ‘asc’. For sorting the table data according to colname in descending order, keyword ‘desc’ is used.

Example: select colname1, colname2,… from tablename where search condition order by colname1 asc/desc, colname2 asc/desc,…;

Practical Assignment – 1

Create these tables which consist of following attributes

College

Column Name	Data type	Size
cName	varchar	10
state	varchar	10
enrollment	int	

Student

Column Name	Data type	Size
sID	int	
sName	varchar	10
GPA	number	2,1
sizeHS	int	
DoB	date	

Apply

Column Name	Data type	Size
sID	int	
cName	varchar	10
major	varchar	20
decision	char	1

About Database: This database is College Application Database which contain 3 tables.

First one is **Student** that contain information of student such as ID of Student, his name, GPA that he/she scored, size of his High School class i.e. number of students in his/her high school class and student's Date of Birth.

Second table in database is **College** this table contains college information such as name of college, state where it is situated, and its enrollment i.e. number seats in that college.

Third table contains data about applications and this table is named **Apply** each row of this table will contain information about one application. Each application has

- **sID:** This will contain ID (similar to Roll No.) of student who is applying. [Each Application only contain sID of Applicant all other information about student such as his name or GPA can be check from S**tudent** table]
- **cName:** Name of college where applicant is applying [similar to sID all the other information about college such as its state or enrollment can be retrieve from **College** table]
- **major:** it is the major in which applicant is applying e.g. CS, EE, biology etc.
- **decision:** it is Y or N shows Acceptance or Rejection of Application

Insert the following data to these tables:

Apply

Student

sID	sName	GPA	sizeHS	DoB
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
567	Edward	2.9	2000	1996-12-21
678	Fay	3.8	200	1996-08-27
789	Gary	3.4	800	1996-10-08
987	Helen	3.7	800	1997-03-27
876	Irene	3.9	400	1996-03-07
765	Jay	2.9	1500	1998-08-08
654	Amy	3.9	1000	1996-05-26
543	Craig	3.4	2000	1998-08-27

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

College

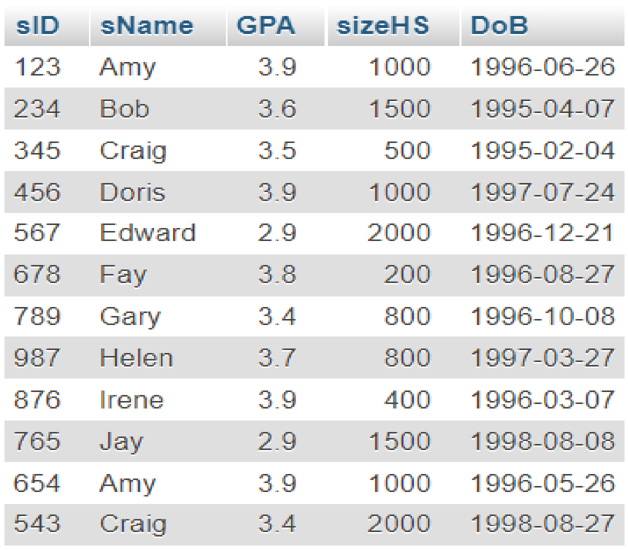
sID	cName	major	decision
123	Stanford	CS	Y
123	Stanford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Stanford	history	Y
987	Stanford	CS	Y
987	Berkeley	CS	Y
876	Stanford	CS	N
876	MIT	biology	Y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

❖ ***Creating Table 1- Student ->***

```
CREATE TABLE Student(sID INTEGER, sName VARCHAR(10), GPA NUMERIC(2,1),
sizeHS INTEGER,DoB DATE);
```

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

```
SELECT * from Student;
```



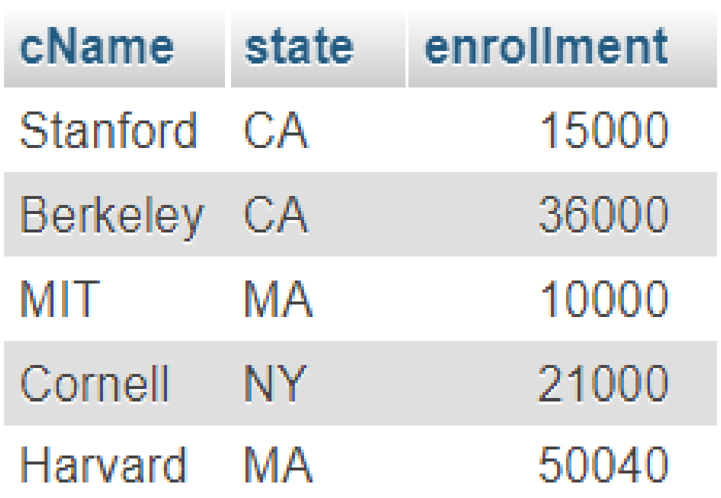
sID	sName	GPA	sizeHS	DoB
123	Amy	3.9	1000	1996-06-26
234	Bob	3.6	1500	1995-04-07
345	Craig	3.5	500	1995-02-04
456	Doris	3.9	1000	1997-07-24
567	Edward	2.9	2000	1996-12-21
678	Fay	3.8	200	1996-08-27
789	Gary	3.4	800	1996-10-08
987	Helen	3.7	800	1997-03-27
876	Irene	3.9	400	1996-03-07
765	Jay	2.9	1500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

❖ ***<u>Creating Table 2- College -></u>***

```
CREATE TABLE College(cName VARCHAR(10), state VARCHAR(10), enrolment
INTEGER);
```

```
Insert into college values
('Stanford' ,'CA' ,15000),
('Berkeley', 'CA', 36000),
('MIT', 'MA', 10000),
('Cornell', 'NY', 21000),
('Harvard', 'MA', 50040);
```

```
SELECT * from College;
```



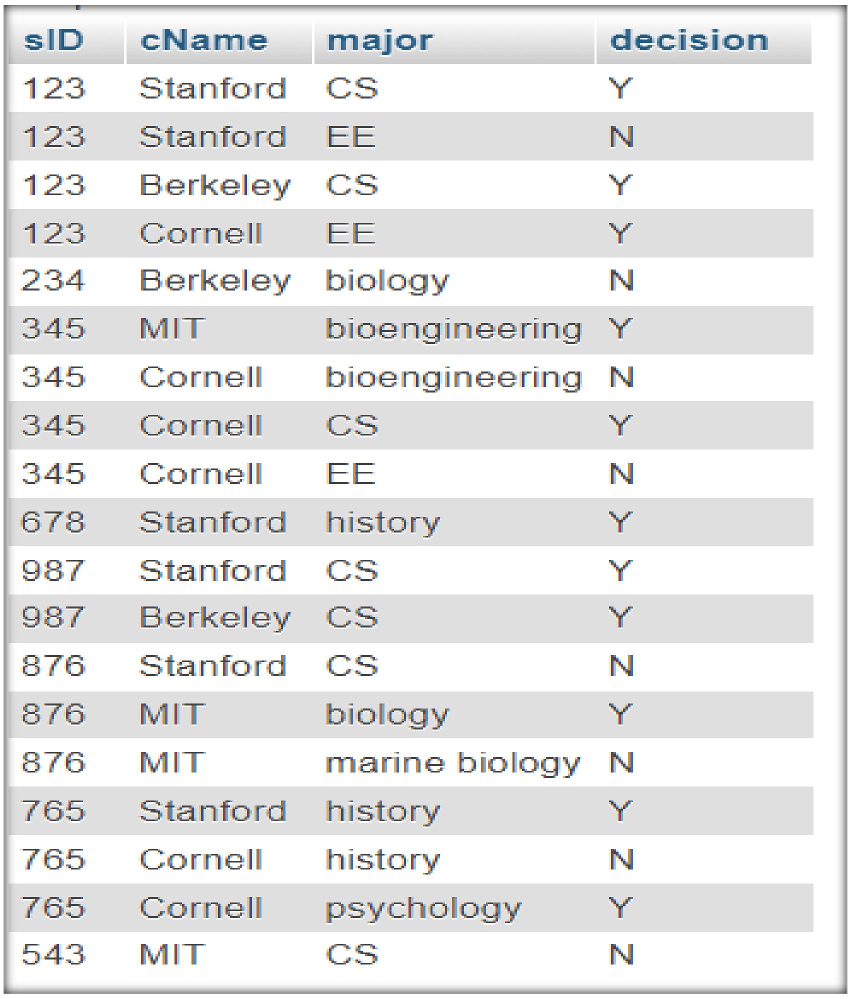
cName	state	enrollment
Stanford	CA	15000
Berkeley	CA	36000
MIT	MA	10000
Cornell	NY	21000
Harvard	MA	50040

❖ ***Creating Table 3- Apply ->***

```
CREATE TABLE Apply(sID INTEGER, cName VARCHAR(10), major VARCHAR(20),
decision char(1));
```

```
INSERT INTO apply VALUES
(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');
```

```
SELECT * from Apply;
```

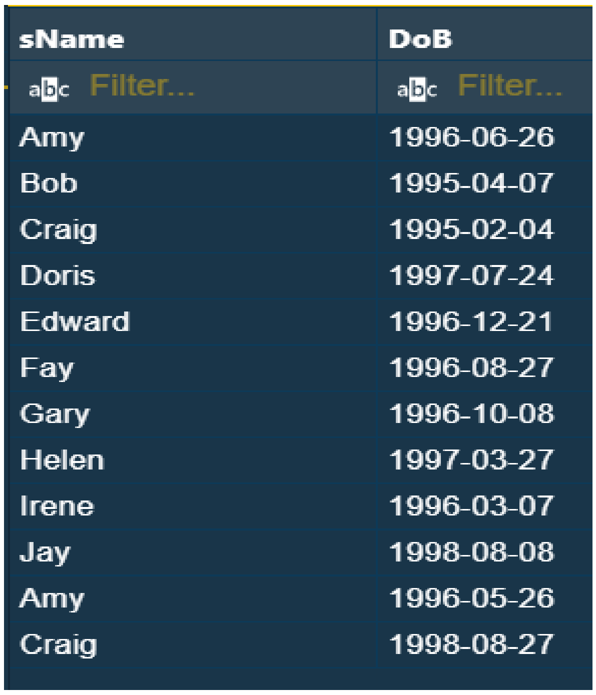


sID	cName	major	decision
123	Stanford	CS	Y
123	Stanford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Stanford	history	Y
987	Stanford	CS	Y
987	Berkeley	CS	Y
876	Stanford	CS	N
876	MIT	biology	Y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

Queries

1. **List the student name, DoB from student table.**

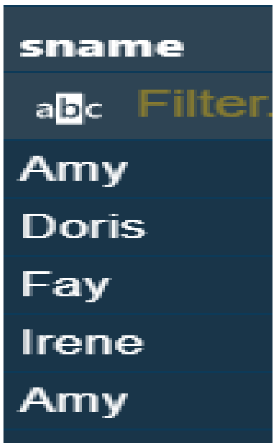
```
SELECT sName,DoB FROM student;
```



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

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

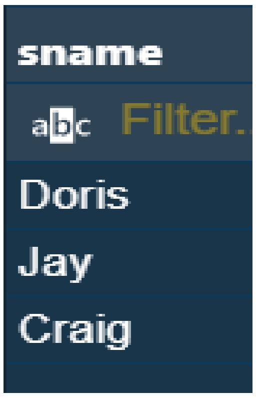
```
SELECT sName from Student WHERE GPA > 3.7;
```



sname
abc Filter
Amy
Doris
Fay
Irene
Amy

3. **List the name of student whose High School size is atleast 1000 and born after 1996. [Hint: check DoB greater than 31st December, 1996]**

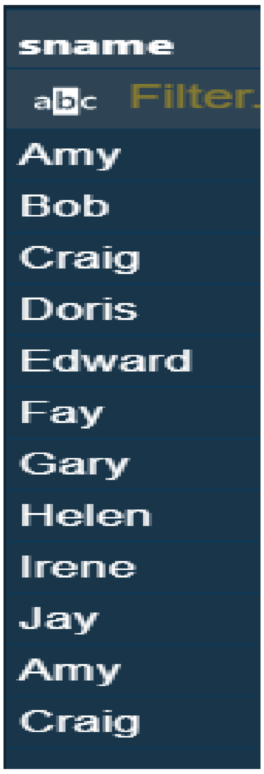
```
Select sName from Student where sizeHS >= 1000 and DoB > '1996-12-31';
```



```
sname
abc Filter...
Doris
Jay
Craig
```

4. **List the name of student who are scoring GPA in between 2.9 and 3.9.**

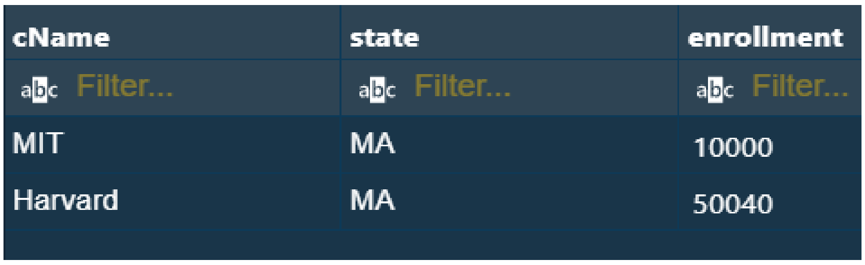
```
SELECT sName from Student where GPA BETWEEN 2.9 and 3.9;
```



```
sname
abc Filter...
Amy
Bob
Craig
Doris
Edward
Fay
Gary
Helen
Irene
Jay
Amy
Craig
```

5. **List all the details of colleges who situated in MA.**

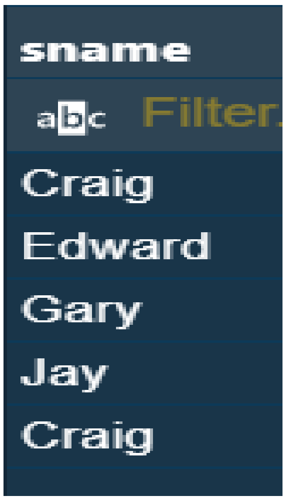
```
Select * from College where state = 'MA';
```



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

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

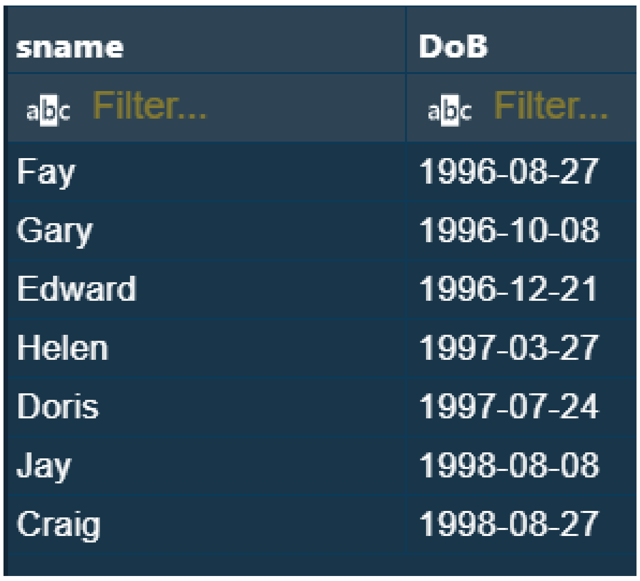
```
Select sName from Student where GPA >= 2.0 and GPA <= 3.5;
```



sname
abc Filter.
Craig
Edward
Gary
Jay
Craig

7. **List the students who have born after 1st Jul 96 in the order of the Date of Birth.**

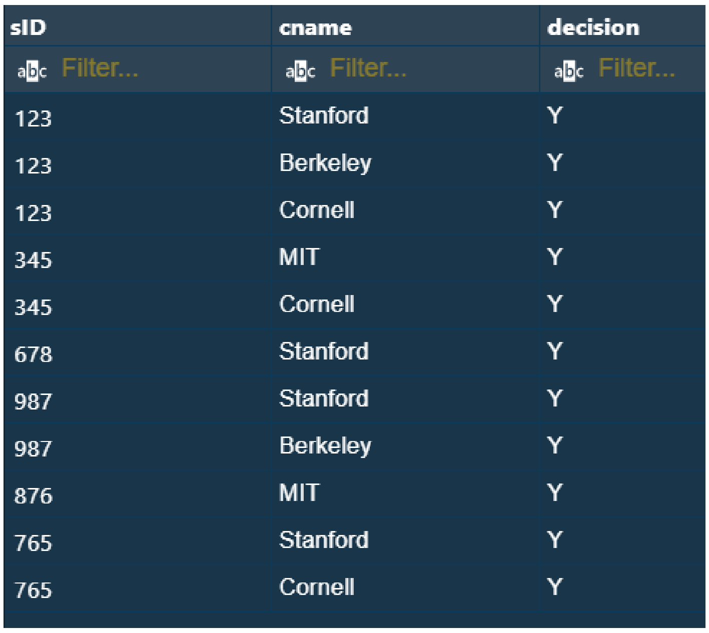
```
SELECT sName,DoB from Student where DoB > '1996-07-01' ORDER BY DoB;
```



sname	DoB
abc Filter...	abc Filter...
Fay	1996-08-27
Gary	1996-10-08
Edward	1996-12-21
Helen	1997-03-27
Doris	1997-07-24
Jay	1998-08-08
Craig	1998-08-27

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

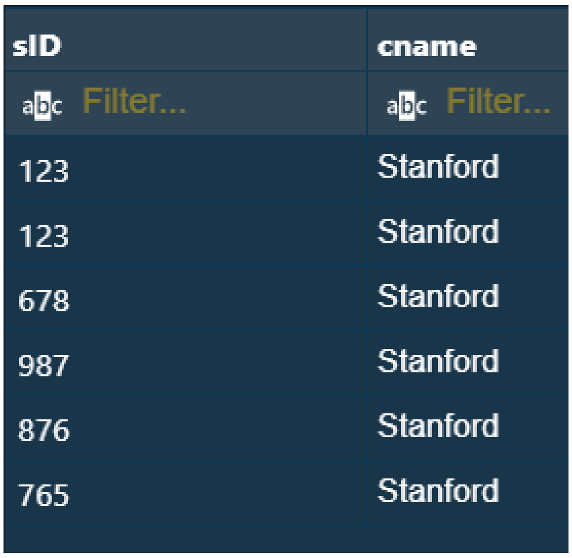
```
Select sID,cName,decision from Apply where decision = 'Y';
```



sID	cname	decision
Filter...	Filter...	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 which are filled at Stanford.**

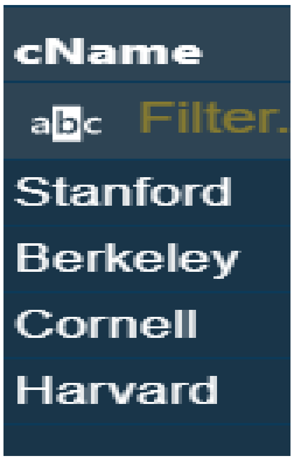
```
Select sID,cName from apply where cName = 'Stanford';
```



sID	cname
Filter...	Filter...
123	Stanford
123	Stanford
678	Stanford
987	Stanford
876	Stanford
765	Stanford

10. **List the colleges that that has enrollment greater than 10001.**

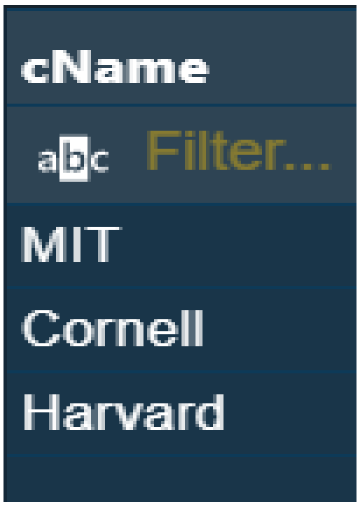
```
Select cName from College where enrollment > 10001;
```



cName
Filter.
Stanford
Berkeley
Cornell
Harvard

11. **List the colleges not in California.**

```
Select cName from College where state != 'CA';
```



cName
Filter...
MIT
Cornell
Harvard

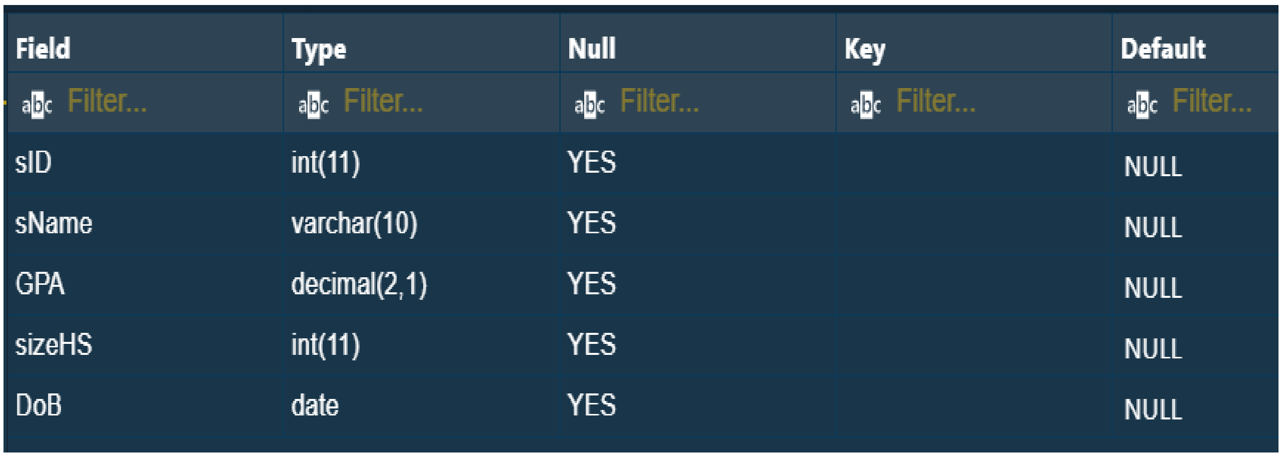
12. **List names of all student who came from high school having size greater than 17000 and scored GPA less than 3.8.**

```
Select sName from Student WHERE sizeHS > 17000 and GPA < 3.8;
```

Query returned 0 rows

13. **Display the description of the Student table.**

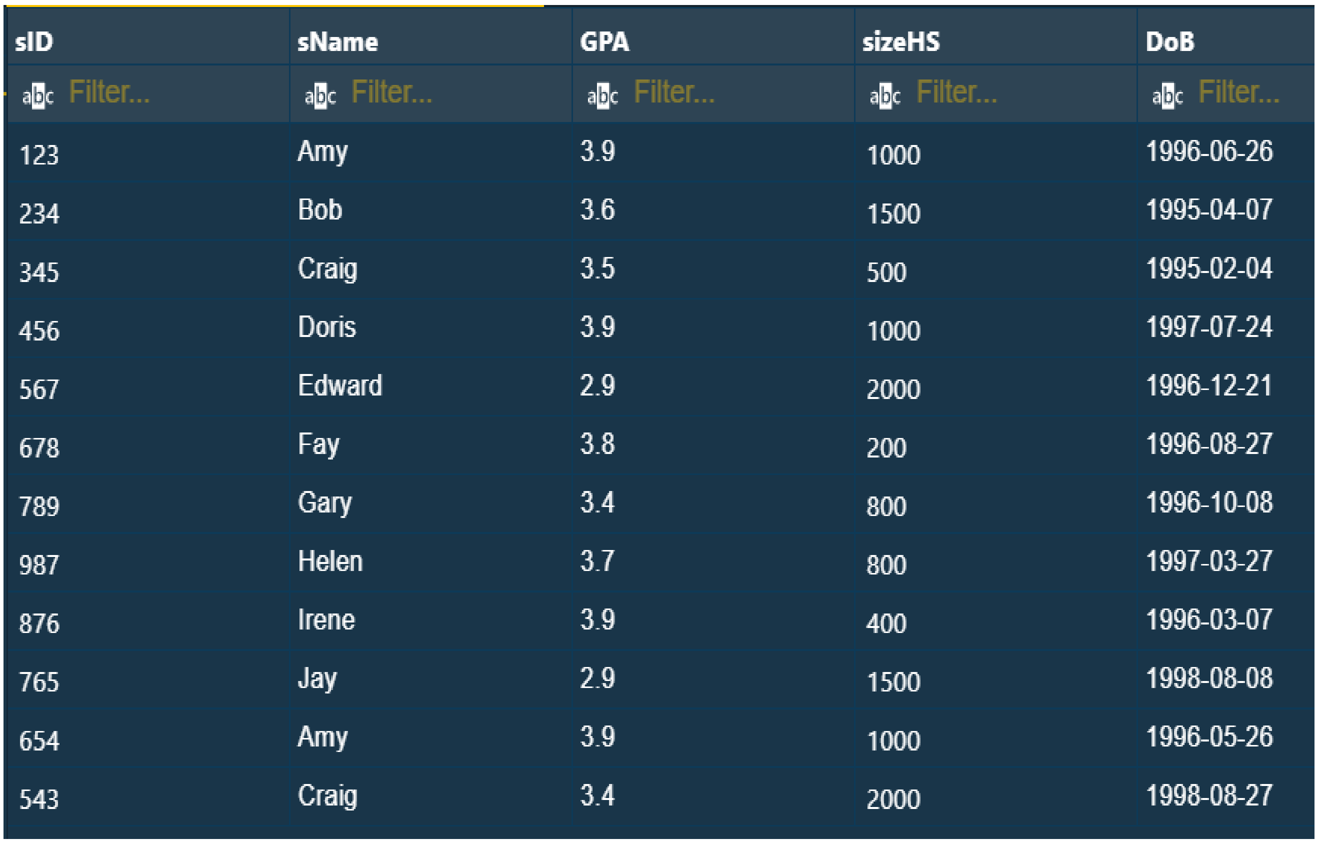
```
DESCRIBE student;
```



Field	Type	Null	Key	Default
sID	int(11)	YES		NULL
sName	varchar(10)	YES		NULL
GPA	decimal(2,1)	YES		NULL
sizeHS	int(11)	YES		NULL
DoB	date	YES		NULL

14. **Display the details of all students.**

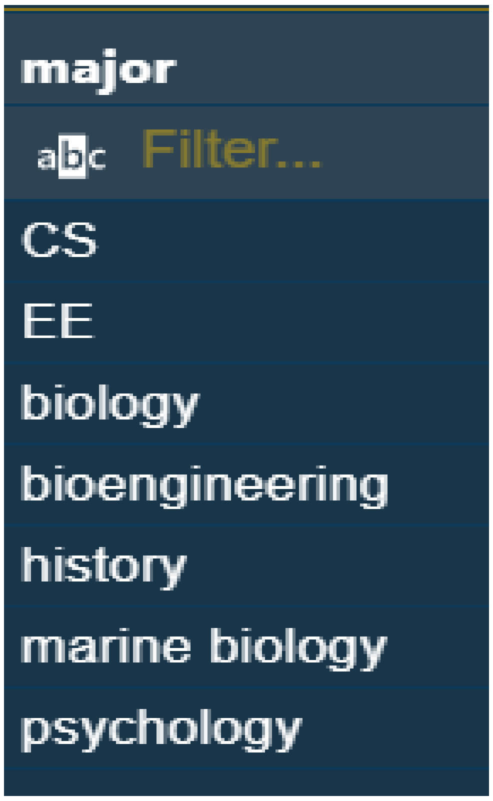
```
SELECT * from Student;
```



sID	sName	GPA	sizeHS	DoB
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
567	Edward	2.9	2000	1996-12-21
678	Fay	3.8	200	1996-08-27
789	Gary	3.4	800	1996-10-08
987	Helen	3.7	800	1997-03-27
876	Irene	3.9	400	1996-03-07
765	Jay	2.9	1500	1998-08-08
654	Amy	3.9	1000	1996-05-26
543	Craig	3.4	2000	1998-08-27

15. Display unique majors.

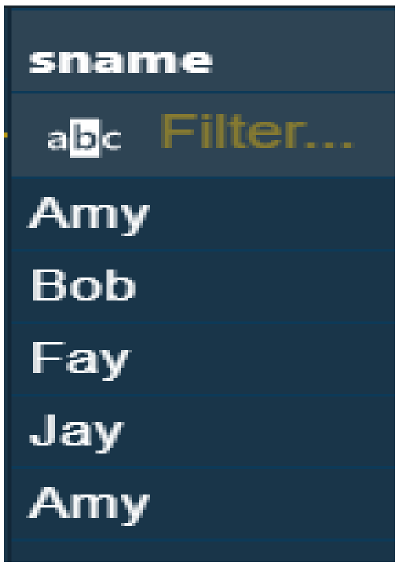
```sql
Select DISTINCT major from Apply;
```



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

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

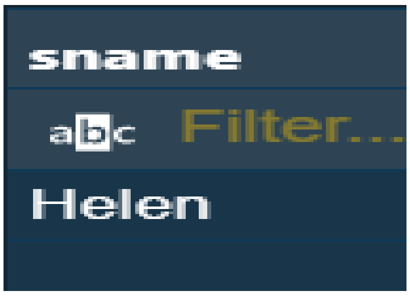
```sql
Select sName from Student where sName LIKE '___';
```



```
sname
Filter...
Amy
Bob
Fay
Jay
Amy
```

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

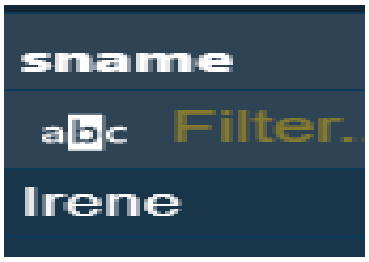
```
Select sName from Student where sName LIKE 'H____';
```



```
sname
abc Filter...
Helen
```

18.List the student names those are having third character and fifth char. must be ‘e’.

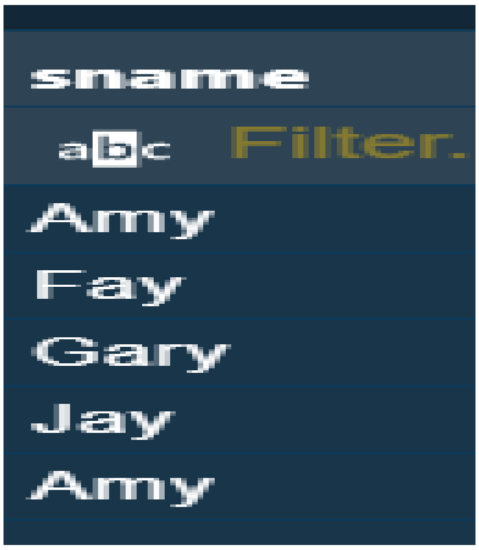
```
Select sName from Student where sName LIKE '__e_e';
```



```
sname
abc Filter...
Irene
```

19. List the student names ending with ‘y’.

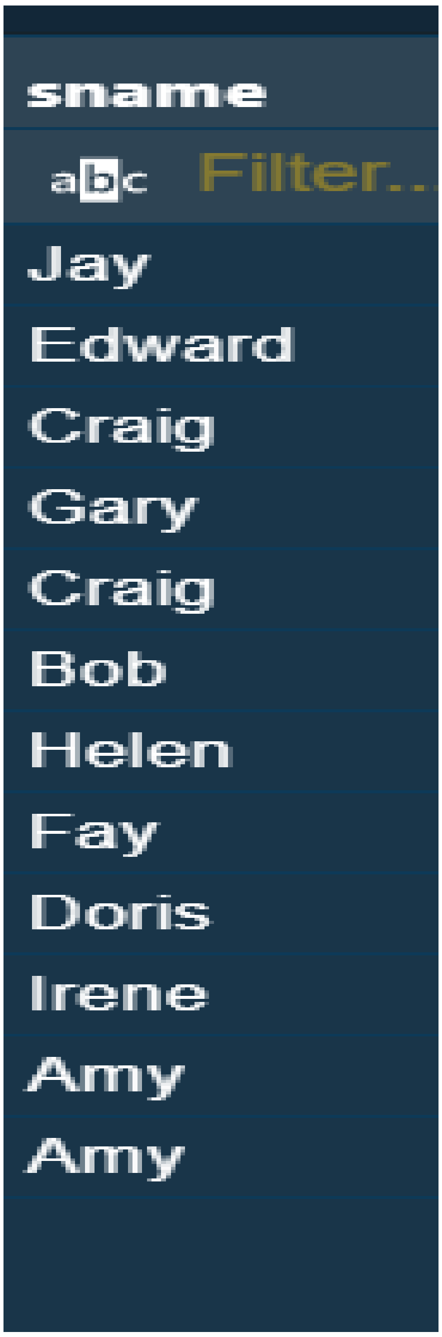
```
Select sName from Student where sName LIKE '%y';
```



```
sname
abc Filter...
Amy
Fay
Gary
Jay
Amy
```

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

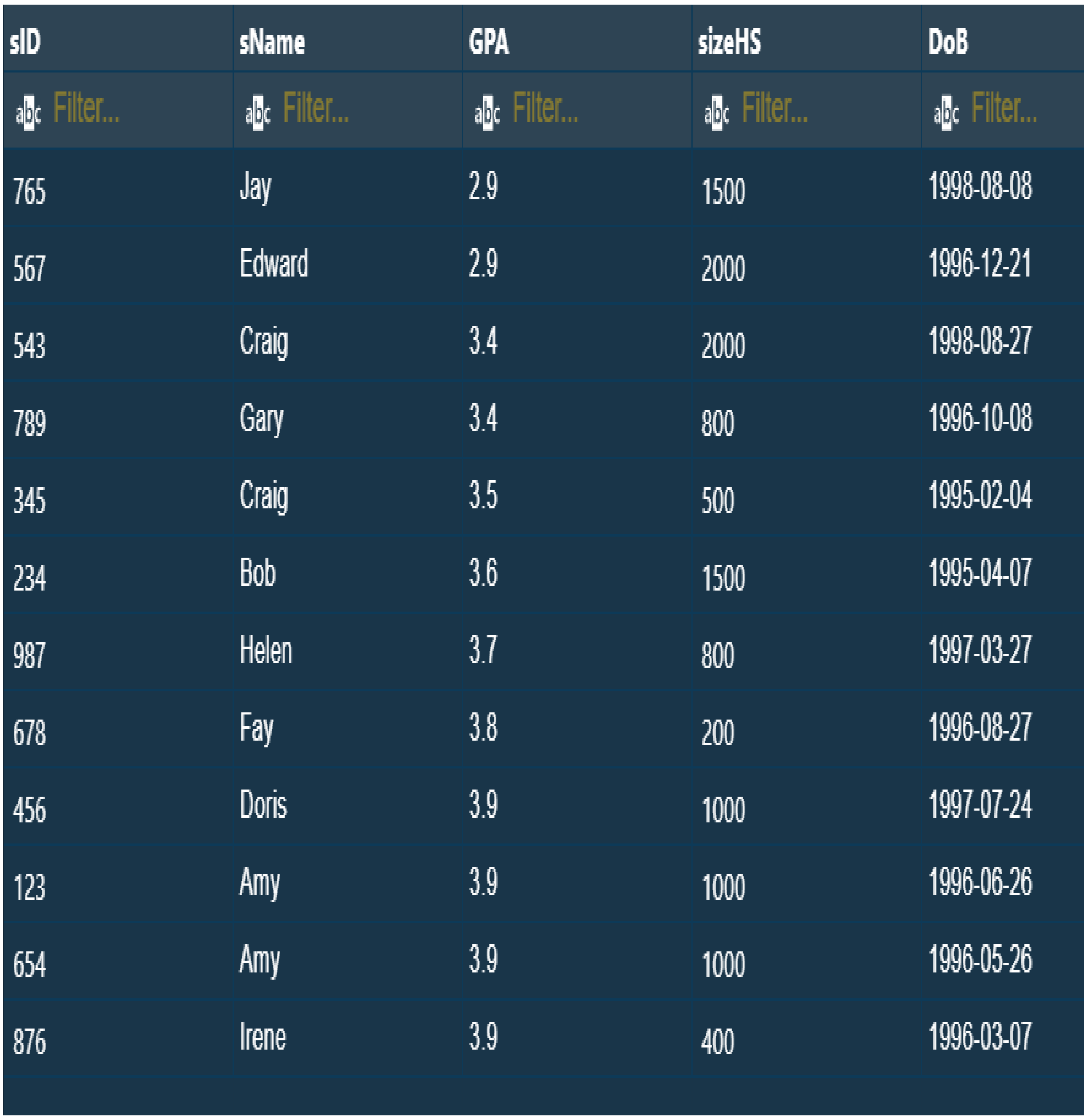
```
Select sName from Student order by GPA;
```



```
sname
abc Filter...
Jay
Edward
Craig
Gary
Craig
Bob
Helen
Fay
Doris
Irene
Amy
Amy
```

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

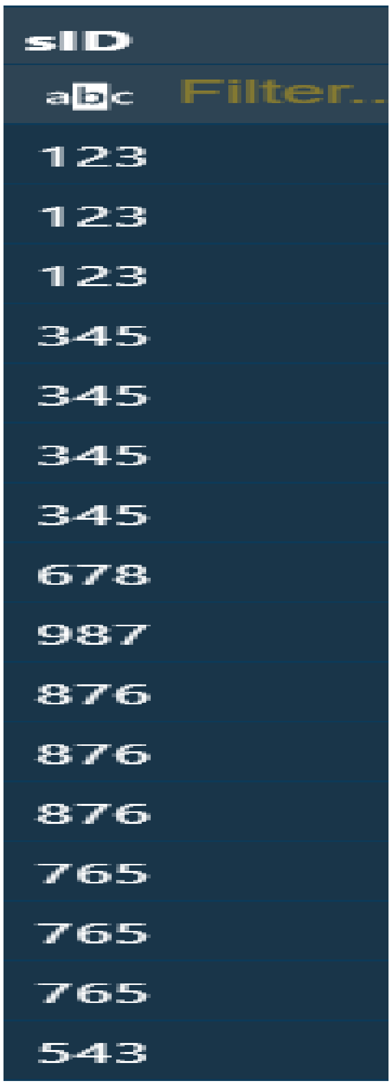
```
Select * from Student order BY GPA,DoB DESC;
```



sID	sName	GPA	sizeHS	DoB
Filter...	Filter...	Filter...	Filter...	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 student 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
678
987
876
876
876
765
765
765
543
```

23. Delete all applications filled at Stanford (*Choose table wisely*)

```
delete from Apply where cName = 'Stanford';
```

```
6 row(s) deleted.
```

24.Delete the college Stanford from college table.

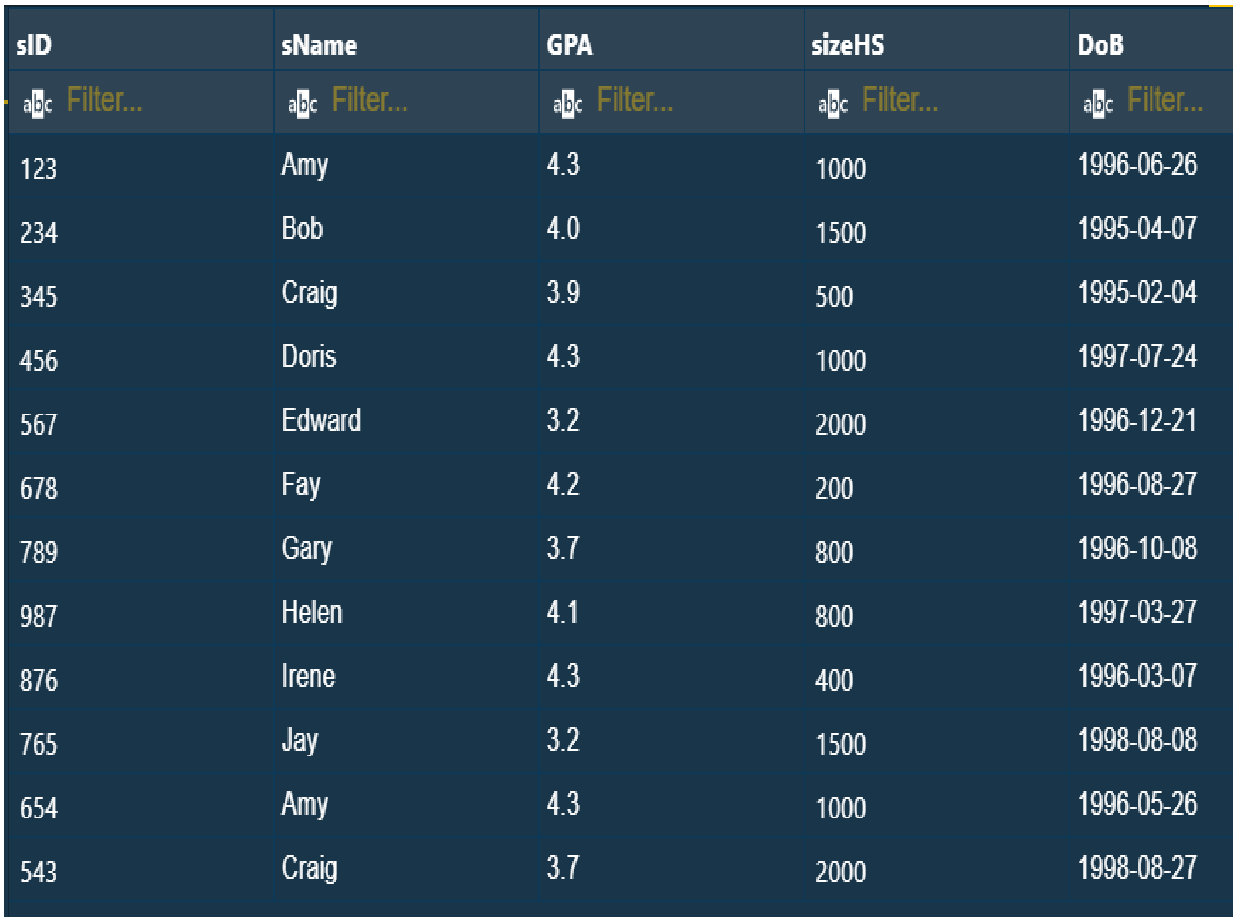
```
delete from College where cName = 'Stanford';
```

```
1 row(s) deleted.
```

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

```
update Student set GPA = GPA + 10*GPA/100;
```

```
12 row(s) updated.
```



sID	sName	GPA	sizeHS	DoB
123	Amy	4.3	1000	1996-06-26
234	Bob	4.0	1500	1995-04-07
345	Craig	3.9	500	1995-02-04
456	Doris	4.3	1000	1997-07-24
567	Edward	3.2	2000	1996-12-21
678	Fay	4.2	200	1996-08-27
789	Gary	3.7	800	1996-10-08
987	Helen	4.1	800	1997-03-27
876	Irene	4.3	400	1996-03-07
765	Jay	3.2	1500	1998-08-08
654	Amy	4.3	1000	1996-05-26
543	Craig	3.7	2000	1998-08-27

26. Increment the GPA of the students by 1.5 whose GPA is less than 3.5 and belong to High School having size greater than 1500.

```
update student set GPA = GPA + 1.5 where GPA < 3.5 and SizeHS > 1500;
```

```
2 row(s) updated.
```

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

```
DELETE from student where GPA < 3.2;
```

```
2 row(s) deleted.
```