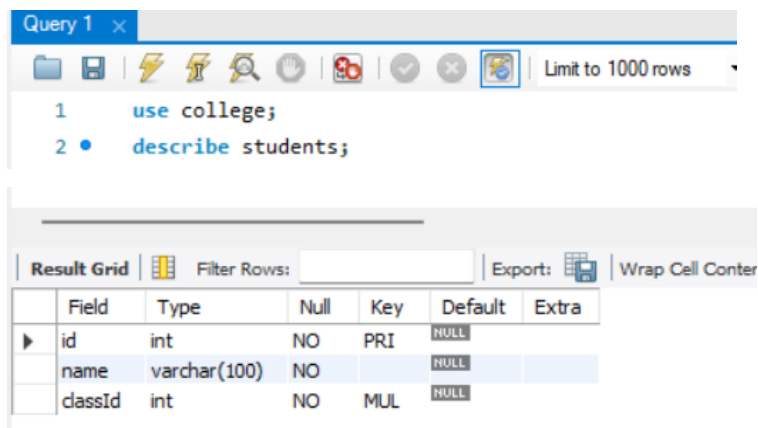


NOTE: I have added the screenshots of the SQL queries with the respective heading.

1. Create table

```
1 CREATE TABLE `college`.`students` (  
2   `id` INT NOT NULL,  
3   `name` VARCHAR(100) NOT NULL,  
4   `class` VARCHAR(45) NOT NULL,  
5   `email` VARCHAR(100) NOT NULL,  
6   PRIMARY KEY (`id`));  
7
```

2. Describe table



The screenshot shows a SQL IDE interface. At the top, there's a toolbar with various icons. Below it, a query window titled 'Query 1' contains the following SQL commands:

```
1 use college;  
2 describe students;
```

Below the query window, there's a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid itself is a table with the following columns: Field, Type, Null, Key, Default, and Extra.

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
name	varchar(100)	NO		NULL	
classId	int	NO	MUL	NULL	

3. Alter table

```
3 ALTER TABLE `college`.`students`  
4 ADD CONSTRAINT `classId`  
5 FOREIGN KEY (`classId`)  
6 REFERENCES `college`.`classes` (`classId`)  
7 ON DELETE CASCADE  
8 ON UPDATE CASCADE;  
9
```

4. Insert into Table

```
1 use college;  
2 insert into classes values(1,"A",30);  
3
```

5. Select statement

The screenshot shows two SQL queries and their corresponding result grids in a database IDE.

Query 1:

```
1 use college;
2 select * from classes;
3
```

Result Grid 1:

classId	className	maxCapacity
1	A	30
2	B	30
3	C	30
NULL	NULL	NULL

Query 2:

```
1 use college;
2 select * from students;
3
```

Result Grid 2:

id	name	classId
1	swara	1
2	mike	1
3	el	2
4	max	2
5	lucas	3
6	dustin	2
NULL	NULL	NULL

6. Distinct

The screenshot shows a SQL query using the DISTINCT keyword and its result grid.

Query:

```
1 use college;
2 select distinct(name) from students;
3
```

Result Grid:

name
swara
mike
el
max
lucas
dustin

7. Where

The screenshot shows a SQL query using the WHERE clause and its result grid.

Query:

```
1 use college;
2 select id, name from students where classid=2;
3
```

Result Grid:

id	name
3	el
4	max
6	dustin
NULL	NULL

8. Or

The screenshot shows a SQL IDE with a query editor and a result grid. The query is:

```
1 • use college;
2 • select id, name, classid from students where classid=2 or classid=3;
3
```

The result grid displays the following data:

	id	name	classid
▶	3	el	2
	4	max	2
	6	dustin	2
	5	lucas	3
*	NULL	NULL	NULL

9. And

The screenshot shows a SQL IDE with a query editor and a result grid. The query is:

```
1 • use college;
2 • select id, name, classid from students where classid=2 and id=3;
3
```

The result grid displays the following data:

	id	name	classid
▶	3	el	2
*	NULL	NULL	NULL

10. Order by

The screenshot shows a SQL IDE with a query editor and a result grid. The query is:

```
1 • use college;
2 • select id, name, classid from students order by name;
3
```

The result grid displays the following data, sorted by name:

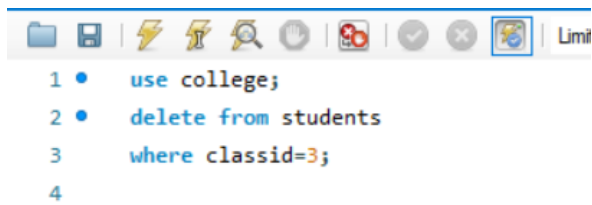
	id	name	classid
▶	6	dustin	2
	3	el	2
	5	lucas	3
	4	max	2
	2	mike	1
	1	swara	1
*	NULL	NULL	NULL

11. Update statement

The screenshot shows a SQL IDE with a query editor. The query is:

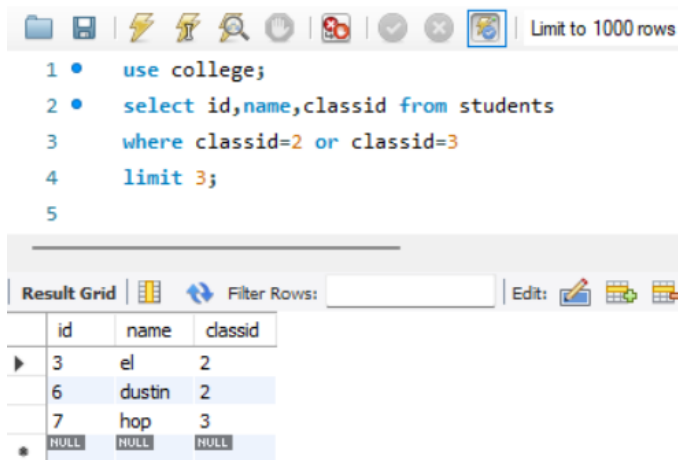
```
Query 1 x
1 • use college;
2 • update students
3   set classid=3 where id =4;
4
```

12. Delete row



```
1 • use college;
2 • delete from students
3   where classid=3;
4
```

13. Select TOP (limit)



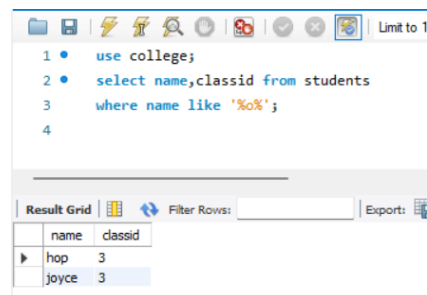
```
1 • use college;
2 • select id,name,classid from students
3   where classid=2 or classid=3
4   limit 3;
5
```

Result Grid

	id	name	classid
▶	3	el	2
	6	dustin	2
	7	hop	3
*	NULL	NULL	NULL

14. Like (Wildcards are % and _)

a. %

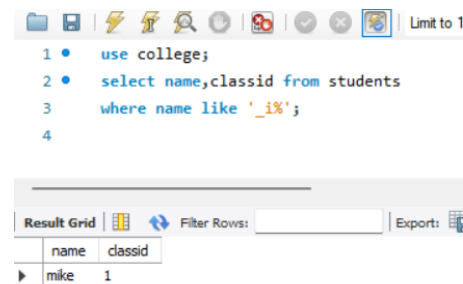


```
1 • use college;
2 • select name,classid from students
3   where name like '%o%';
4
```

Result Grid

	name	classid
▶	hop	3
	joyce	3

b. _

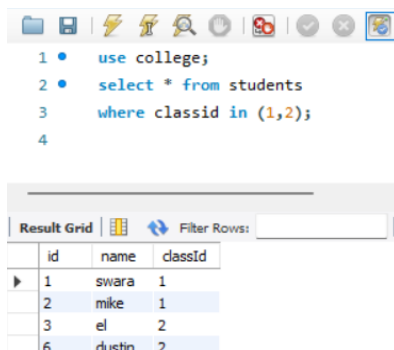


```
1 • use college;
2 • select name,classid from students
3   where name like '_i%';
4
```

Result Grid

	name	classid
▶	mike	1

15. In

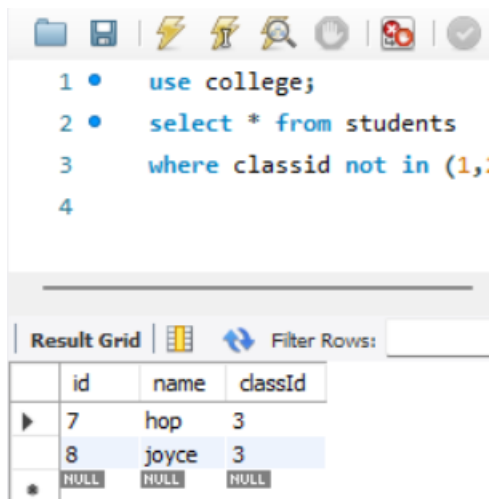


```
1 • use college;
2 • select * from students
3   where classid in (1,2);
4
```

Result Grid

	id	name	classid
▶	1	swara	1
	2	mike	1
	3	el	2
	6	dustin	2

16. NOT IN

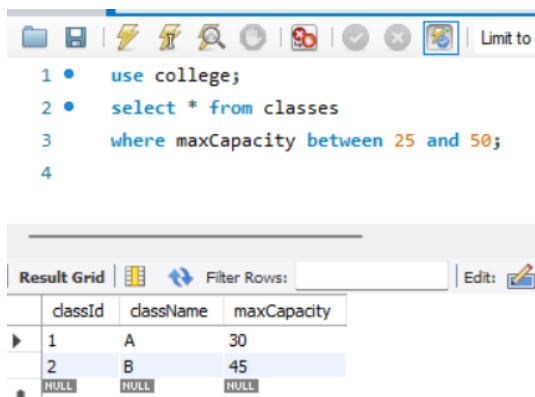


```
1 • use college;
2 • select * from students
3   where classid not in (1,2);
4
```

Result Grid

	id	name	classId
▶	7	hop	3
	8	joyce	3
*	NULL	NULL	NULL

17. Between

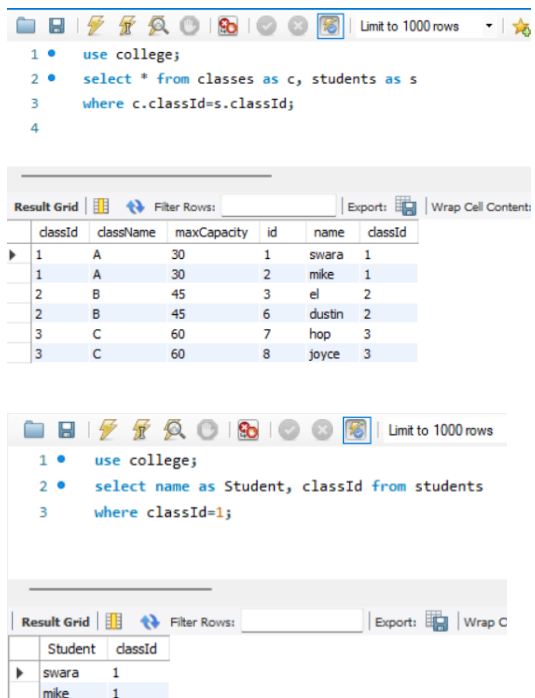


```
1 • use college;
2 • select * from classes
3   where maxCapacity between 25 and 50;
4
```

Result Grid

	classId	className	maxCapacity
▶	1	A	30
	2	B	45
*	NULL	NULL	NULL

18. Aliases



```
1 • use college;
2 • select * from classes as c, students as s
3   where c.classId=s.classId;
4
```

Result Grid

	classId	className	maxCapacity	id	name	classId
▶	1	A	30	1	swara	1
	1	A	30	2	mike	1
	2	B	45	3	el	2
	2	B	45	6	dustin	2
	3	C	60	7	hop	3
	3	C	60	8	joyce	3

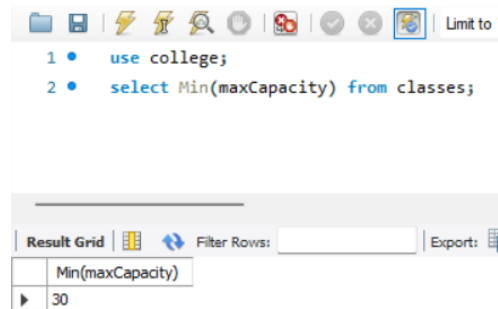

```
1 • use college;
2 • select name as Student, classId from students
3   where classId=1;
```

Result Grid

	Student	classId
▶	swara	1
	mike	1

19. Aggregate Functions

a. Min

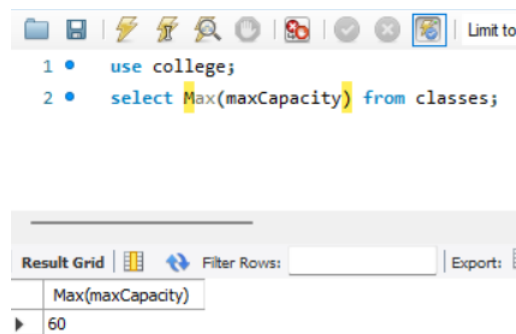


The screenshot shows a SQL IDE with a toolbar at the top. The query editor contains two lines of SQL code:

```
1 • use college;  
2 • select Min(maxCapacity) from classes;
```

Below the query editor, the 'Result Grid' tab is active. It displays a table with one column, 'Min(maxCapacity)', and one row with the value '30'.

b. Max

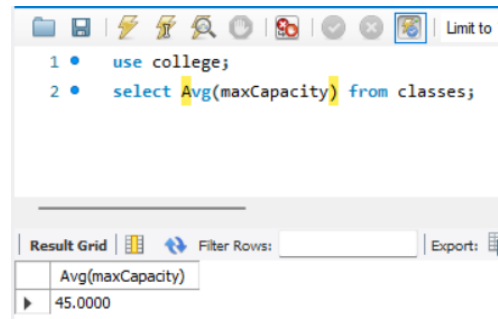


The screenshot shows a SQL IDE with a toolbar at the top. The query editor contains two lines of SQL code:

```
1 • use college;  
2 • select Max(maxCapacity) from classes;
```

Below the query editor, the 'Result Grid' tab is active. It displays a table with one column, 'Max(maxCapacity)', and one row with the value '60'.

c. Avg

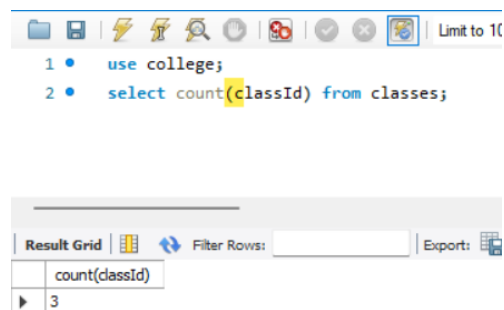


The screenshot shows a SQL IDE with a toolbar at the top. The query editor contains two lines of SQL code:

```
1 • use college;  
2 • select Avg(maxCapacity) from classes;
```

Below the query editor, the 'Result Grid' tab is active. It displays a table with one column, 'Avg(maxCapacity)', and one row with the value '45.0000'.

d. Count



The screenshot shows a SQL IDE with a toolbar at the top. The query editor contains two lines of SQL code:

```
1 • use college;  
2 • select count(classId) from classes;
```

Below the query editor, the 'Result Grid' tab is active. It displays a table with one column, 'count(classId)', and one row with the value '3'.

e. Sum

```
1 • use college;
2 • select sum(maxCapacity) as TotalCapacity
3 • from classes;
```

Result Grid		Filter Rows:	Export:
TotalCapacity			
▶	135		

20. Joins

a. Inner Join

```
1 • use college;
2 • select s.id,s.name,c.className from classes as c
3 • inner join students as s on c.classId=s.classId;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Co
id		name	className	
▶	1	swara	A	
	2	mike	A	
	3	el	B	
	6	dustin	B	
	7	hop	C	
	8	joyce	C	

b. Left join

```
1 • use college;
2
3 • select s.id,s.name,c.className from classes as c
4 • left join students as s on c.classId=s.classId;
```

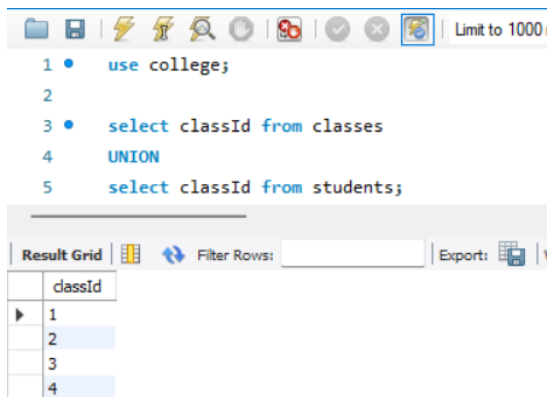
Result Grid		Filter Rows:	Export:	Wrap Cell Co
id		name	className	
▶	2	mike	A	
	1	swara	A	
	6	dustin	B	
	3	el	B	
	8	joyce	C	
	7	hop	C	
	NULL	NULL	D	

c. Right join

```
1 • use college;
2
3 • select s.id,s.name,c.className from classes as c
4 • right join students as s on c.classId=s.classId;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Cont
id		name	className	
▶	1	swara	A	
	2	mike	A	
	3	el	B	
	6	dustin	B	
	7	hop	C	
	8	joyce	C	

21. UNION



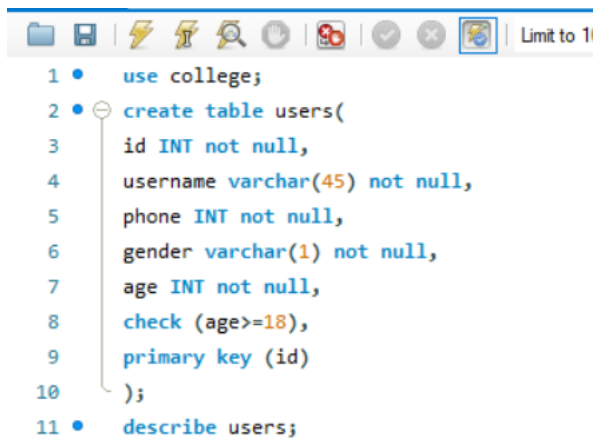
The screenshot shows a SQL IDE with a toolbar at the top. The SQL editor contains the following code:

```
1 • use college;
2
3 • select classId from classes
4   UNION
5   select classId from students;
```

Below the editor is the "Result Grid" tab. It shows a table with one column, "classId", and four rows with values 1, 2, 3, and 4.

classId
1
2
3
4

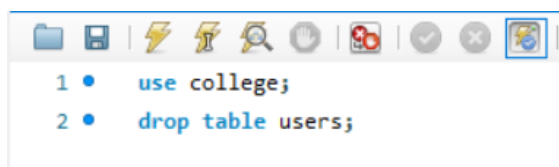
22. Create table (check, primary key , not null)



The screenshot shows a SQL IDE with a toolbar at the top. The SQL editor contains the following code:

```
1 • use college;
2 • create table users(
3     id INT not null,
4     username varchar(45) not null,
5     phone INT not null,
6     gender varchar(1) not null,
7     age INT not null,
8     check (age>=18),
9     primary key (id)
10 );
11 • describe users;
```

23. Drop table



The screenshot shows a SQL IDE with a toolbar at the top. The SQL editor contains the following code:

```
1 • use college;
2 • drop table users;
```


24. Auto increment

```
1 • use college;
2 • create table users(
3     id INT not null auto_increment,
4     username varchar(45) not null,
5     phone INT not null,
6     gender varchar(1) not null,
7     age INT not null,
8     check (age>=18),
9     primary key (id)
10 );
11 • describe users;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	HULL	auto_increment
username	varchar(45)	NO		HULL	
phone	int	NO		HULL	
gender	varchar(1)	NO		HULL	
age	int	NO		HULL	

25. Default

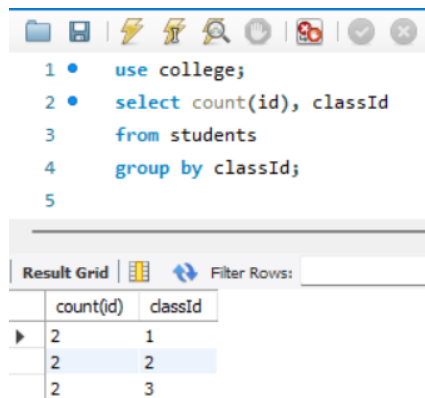
```
1 • use college;
2 • create table users(
3     id INT not null auto_increment,
4     username varchar(45) not null,
5     phone INT not null,
6     gender varchar(1) not null,
7     age INT not null,
8     check (age>=18),
9     primary key (id),
10    city varchar(45) default "Mumbai"
11 );
12 • describe users;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	HULL	auto_increment
username	varchar(45)	NO		HULL	
phone	int	NO		HULL	
gender	varchar(1)	NO		HULL	
age	int	NO		HULL	
city	varchar(45)	YES		Mumbai	

26. Select Into

```
1 • use college;
2 • select * into StudentsDivA
3   from students where classId = (select classId from classes where className="A");
```

27. Group by

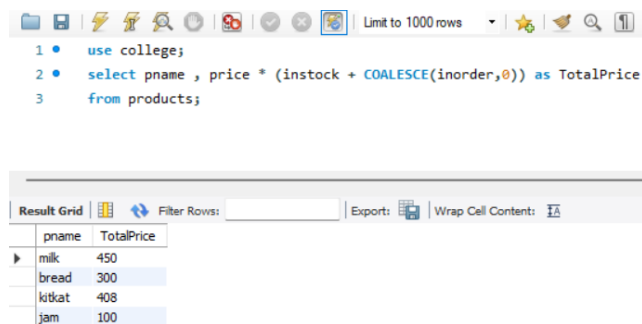


```
1 • use college;
2 • select count(id), classId
3   from students
4  group by classId;
```

Result Grid

	count(id)	classId
▶	2	1
	2	2
	2	3

28. Null Functions

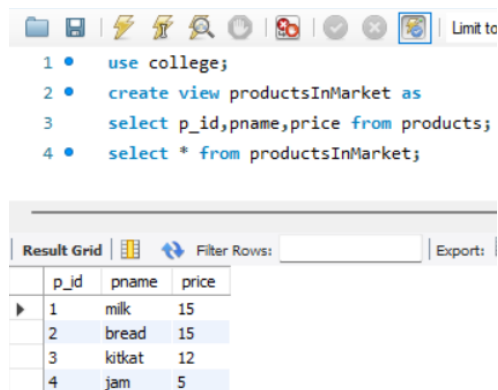


```
1 • use college;
2 • select pname , price * (instock + COALESCE(inorder,0)) as TotalPrice
3   from products;
```

Result Grid

	pname	TotalPrice
▶	milk	450
	bread	300
	kitkat	408
	jam	100

29. Views




```
1 • use college;
2 • create view productsInMarket as
3   select p_id,pname,price from products;
4 • select * from productsInMarket;
```


Result Grid

	p_id	pname	price
▶	1	milk	15
	2	bread	15
	3	kitkat	12
	4	jam	5

30. Stored Procedure




```
1 • use college;
2   DELIMITER &&
3 • CREATE PROCEDURE get_Students()
4   BEGIN
5     SELECT * FROM students;
6   END &&
7   DELIMITER ;
8 • call get_Students();
```



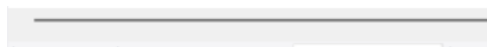
Result Grid | Filter Rows: | Export: | Wrap Cell Col

	id	name	classId
▶	1	swara	1
	2	mike	1
	3	el	2
	6	dustin	2
	7	hop	3
	8	joyce	3

31. Having




```
1 • use college;
2 • select count(id),classId
3   from students
4   group by classId
5   having count(classId)>1;
```



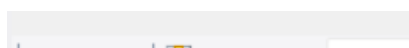
Result Grid | Filter Rows: | Exp

	count(id)	classId
▶	2	1
	2	2
	2	3

32. Prepared Statement



```
1 • use college;
2 • PREPARE stmt1 FROM
3   'SELECT name from students
4   where classId=?';
5 • SET @id=2;
6 • execute stmt1 using @id;
```



Result Grid | Filter Rows: | Exp

	name
▶	el
	dustin