

SQL

error makes clever utube video

mysql workbench software

pw: 2530

oru oorla oru raja iruntharu, avarku keela oru mandhiri irukaru.

oru naal raja naatula yarulam romba kashta padra makkal irukangalo avangaluku help panalamnu decide pani mandhiri kita poverty people list collect pana soldranga within tamilnadu... mandhiri ella district um poi avanga data collect pani kondu varanga .. and list perisa iruku so raja chennai la iruka poverty list matum kekuranga.. mandhiri romba kashta patu chennai poverty list matum seperate pani edukuranga... apovum list perisa irukarathala ..entha family la more than 2 children irukangalo avanga list matum kekuranga.. thirumba ella papers um check pani list ready pandranga.. ithu mari vera vera soldranga so athu ready panave romba tired aidranga.....

then only madhiri came to know abt **database**

DATABASE :

once upon a time the information (data) , news has been stored in paper... bt now it has been stored in database

paper la iruka information elame digital data a convert pani ipo database la store pandrom

consider this database as memory card

example if i have 10 photos and i have stored it in memory card

memory card la iruka photos paka namaku oru phone venum ... ipo intha memory card eduthu phone la potu photos pakura

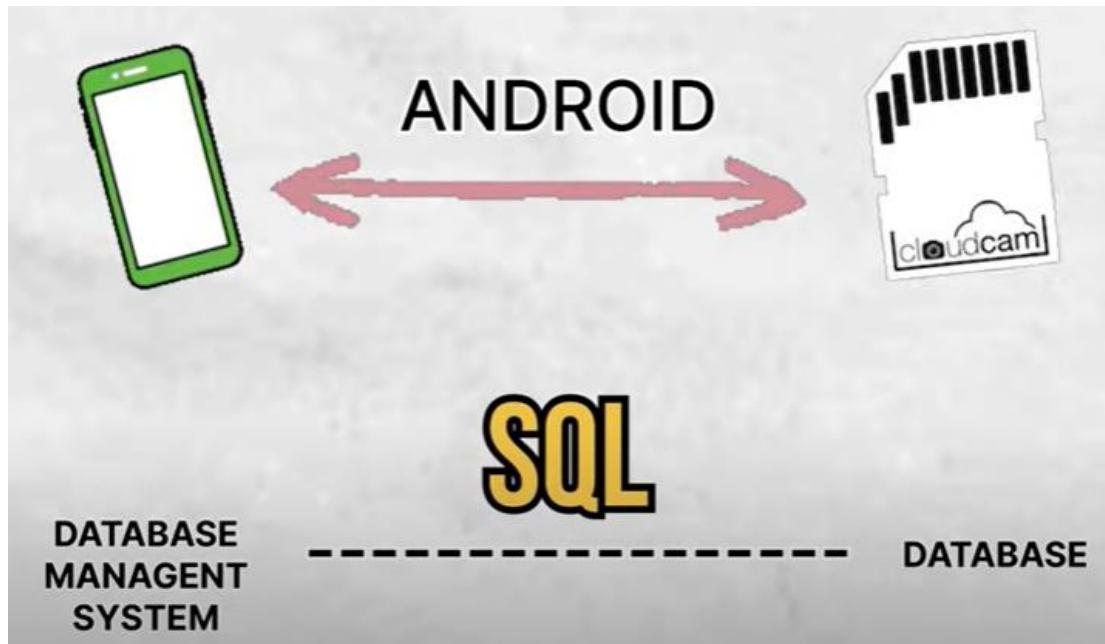
athe mari database la iruka data paaka, data retrieve pana namaku oru database management system venum -- then only we can see datas which is in database

epdi phone um memroy card um connect aaguthu.. epdi memory card la iruka photos paka mudithu -- bcoz phone la android or IOS nu oru OS irukum.. nadula sila coding pogum -- apo antha OS memory card la iruka photos kondu vanthu kamikum

athe mari database management system and database thaniya irukum..

DBMS la irunthu database kita pesa namaku oru language theva paduthu..

DBMS la irunthu SQL apdi endra oru language use pani na database kita pesura ethachu photos, data venuna DBMS la SQL use pani oru request raise pandrom database ku... then database athu puriju response panuthu



we have **different types of database**

Relational database

NOSQL database

Hierarchical database

Network database etc...

common a niraya per use pandrathu relational database

Relation database manage pana - **RELATIONAL DATABASE MANAGEMENT SYSTEM** iruku

different types of RDBMS

oracle, mysql , sql server, postgresql



here we gonna see **MYSQL**

if we learn this, we can easily learn others as well

only few syntax changes will be

Y should we use / learn SQL

=> it is a easy programming language with less syntax

=> data samanthamana ella velaikum SQL theva paduthu

eg: data science, data analyst etc ...

DATA - information - eg: name, age, place etc....

data can be divided into 3 types:

structured data

unstructured data

semistructured data



structured data -- will have defined structure

eg: oru table la name, age, salary iruku

athula age la dosa nu kuduka matom

it will have **text, number, date...**

Name	Age	Salary
John	23	20k
Praveen	Dosa	

unstructured data

eg: social media - it will have different types of data , comments, posts, videos , reels etc...

semi-structured data

combination of structured and unstructured data

eg: email

email la kandipa sender, receiver, date irukum - athula
structured data

athuku keela body irukum - athula namba photos, videos,
audios etc.. ethachu attach panuvom - athu **unstructured data**



Relational database VS Non - Relational database

Relational database

=> it will be in table form - structured

=> we will use **SQL** to work with relational database

=> here we will have only 1 type of data - which is in **table format**

Non - Relational database

=> unstructured

=> we will use **NOSQL** (also known as **not only sql**) to work with non-relational database - it means here we wont use sql

=> NOSQL is not a structured language like SQL

=> here we wont have data in table format -

 will have in document format, graph, key-value pairs, column oriented etc...

 will hv. data in different formats

syntax will differ b/w all

here we gonna see **RDBMS**

relational database kuda epdi structured query language (SQL) use pani work pandrathunu paka porom - here we will use **MySQL** database management system (DBMS)

here are the key points from the video:

- 1. SQL is a structured query language designed for relational databases.**
- 2. NoSQL is more of a paradigm or pattern than a specific language.**
- 3. SQL is typically used with relational databases.**
- 4. NoSQL is typically used with non-relational databases.**
- 5. SQL primarily employs tables for data organization.**
- 6. NoSQL databases can use various data structures, including columns, documents, and key-value pairs.**

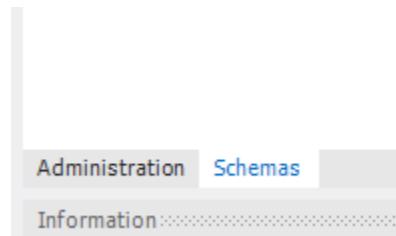
step 1:

download mysql workbench

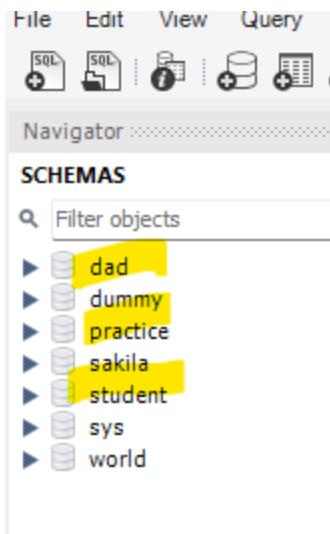
have to give password - everytime we login it will ask for password - 2530

we have 2 - administration, schemas

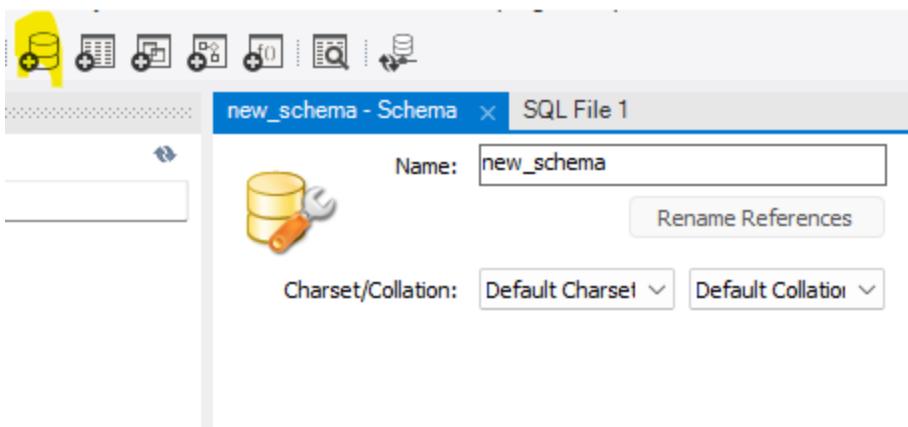
we will work in schemas



these are all schemas / also known as database



used to create new schema



SQL - Structured Query Language

it is used to work with data which is in database

it is a **query / command** - namba oru command solla porom - athu sql la solvom - antha sql command namba sonatha database la poi panum

we have 5 types of command

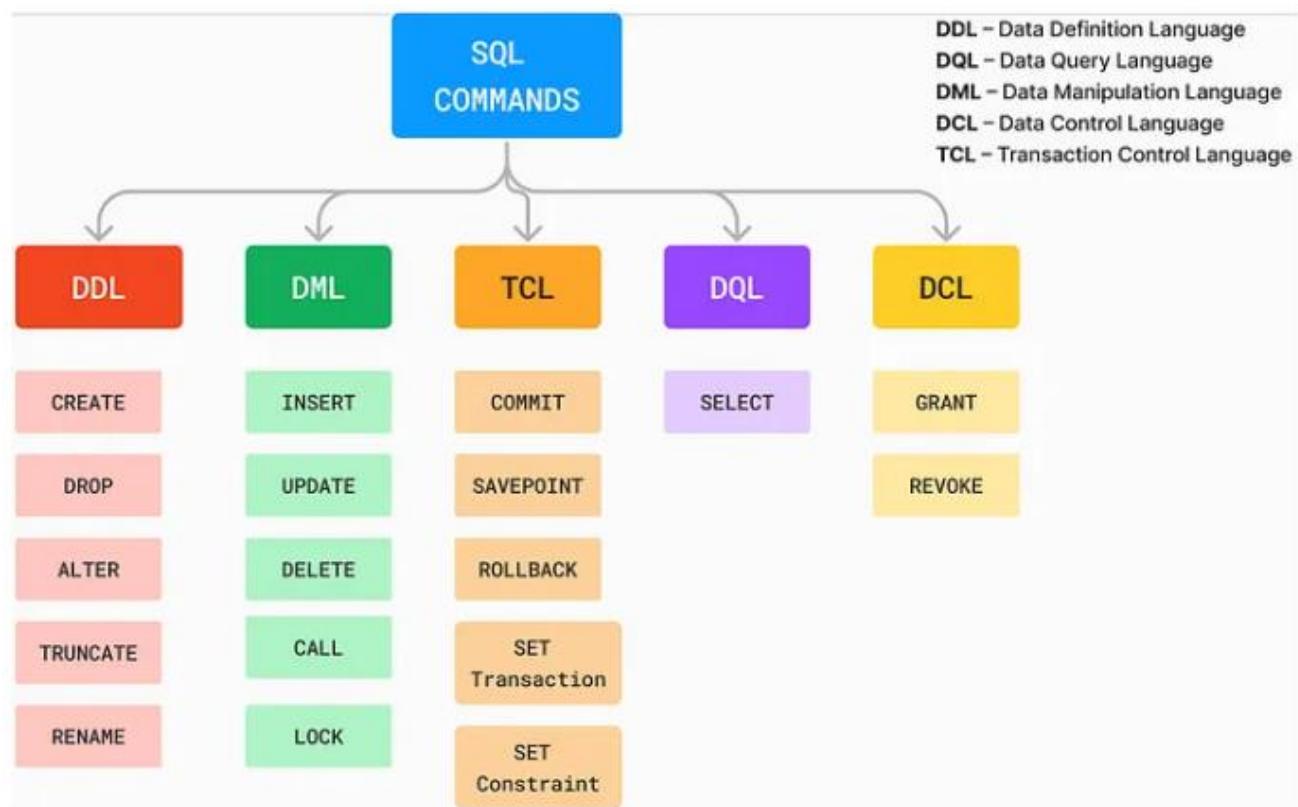
DDL command

DML

DCL

TCL

DQL



DDL command

DDL - DATA DEFINITION LANGUAGE

Relational database is based on table - will give datas within that defined table

Name	Age	Department	Salary
John	23	CSE	10000
Praveen	45	ECE	20000
Suresh	34	EEE	15000

intha mari table a define pana data definition language use pana porom

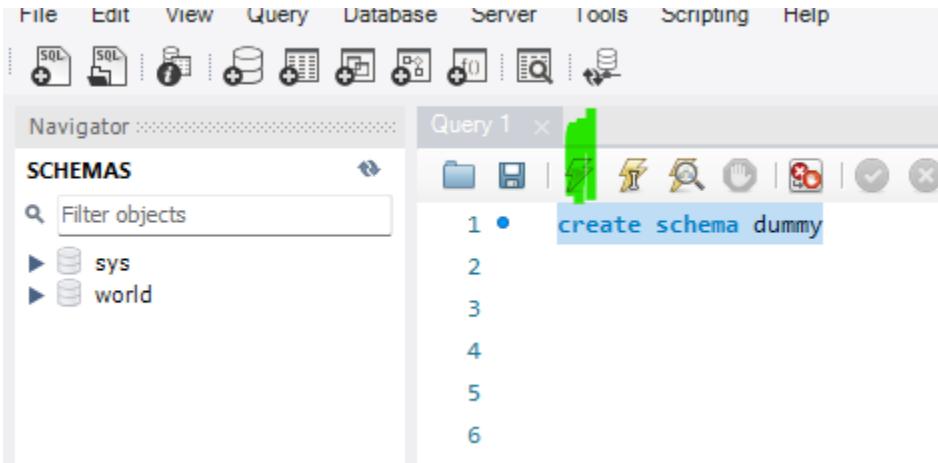
to create, alter, drop and truncate tables we use ddl command



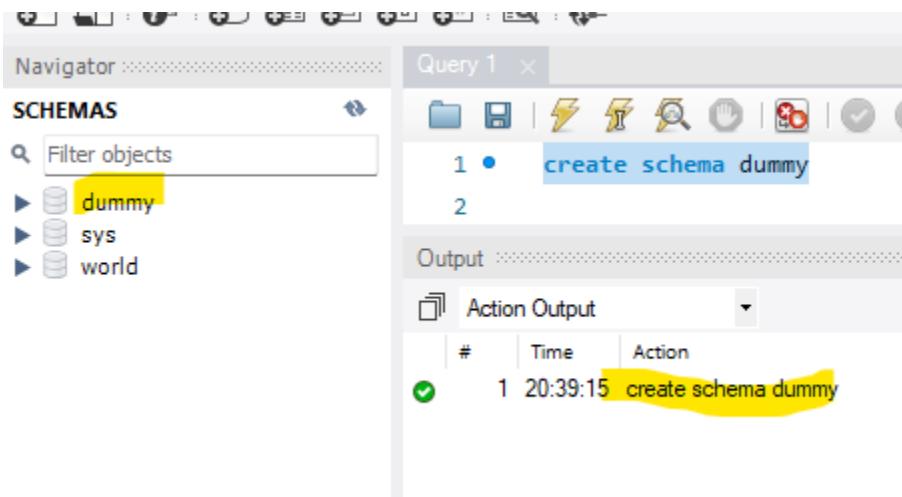
creating new schema named **dummy**

have to select the line and give star like symbol - so it will execute only the line which we selected

then refresh the schemas



schema dummy has been created



now we can create this type of table for teacher

fields - name, age, department, salary

syntax:

create table "table name" (always round bracket - inside this what are the fields required and its datatype (no. of characters), each field seperated by comma ,)

create table teacher-table name(

name - field varchar- datatype (50 - no of character)

```
create table teacher(  
    name varchar(50),  
    age int,  
    department varchar(50),  
    salary int)
```

Name	Age	Department	Salary

to run entire program in the page use this



```
1 • ⏪ create table teacher(  
2     name varchar(50),  
3     age int,  
4     department varchar(50),  
5     salary int)  
6
```

bt we received error - no database selected

means - namba ena database la work pananumnu mention panala

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under SCHEMAS, there is a 'dummy' schema listed. In the Query 1 editor, the following SQL code is present:

```
1 • create table teacher(
2   name varchar(50),
3   age int,
4   department varchar(50),
5   salary int)
6
```

The Output pane shows the results of the actions:

#	Time	Action	Message
1	20:39:15	create schema dummy	1 row(s) affected
2	20:47:28	create table teacher(name varchar(50), age int, department varchar(50), salary int)	Error Code: 1046. No database selected. Select the default DB to be used by

now on top of code - write which database to be used and run it alone

use dummy - select and run this alone

The screenshot shows the MySQL Workbench interface. The Query 1 editor contains the following SQL code:

```
1 • use dummy
2
3 • create table teacher(
4   name varchar(50),
5   age int,
6   department varchar(50),
```

The Output pane shows the results of the actions:

#	Time	Action
1	20:39:15	create schema dummy
2	20:47:28	create table teacher(name varchar(50),
3	20:56:21	use dummy

in output if it is green - success

if red - error

while running code remove the use database line and run otherwise error will occur

Query 1

```

1 • use dummy
2
3 ✘ create table teacher(
4     name varchar(50),
5     age int,
6     department varchar(50),

```

Output

#	Time	Action	Message
1	20:39:15	create schema dummy	1 row(s) affected
2	20:47:28	create table teacher(name varchar(50), age int, department varchar(50), salary int)	Error Code: 1046. No database selected Select the default DB to be used
3	20:56:21	use dummy	0 row(s) affected
4	20:58:06	use dummy create table teacher(name varchar(50), age int, department varchar(50), salary int)	Error Code: 1064. You have an error in your SQL syntax; check the manual

now table created - now we can see everything in the left

Navigator

SCHEMAS

- dummy
 - Tables
 - teacher
 - Columns: name, age, department, salary
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
- sys
- world

Administration Schemas

Information

Table: teacher

Columns:

name	varchar(50)
age	int
department	varchar(50)
salary	int

Query 1

```

1
2
3 • create table teacher(
4     name varchar(50),
5     age int,
6     department varchar(50),

```

Output

#	Time	Action
1	20:39:15	create schema dummy
2	20:47:28	create table teacher(name varchar(50), age int, department varchar(50), salary int)
3	20:56:21	use dummy
4	20:58:06	use dummy create table teacher(name varchar(50), age int, department varchar(50), salary int)
5	20:58:26	use dummy create table teacher(name varchar(50), age int, department varchar(50), salary int)
6	20:59:23	create table teacher(name varchar(50), age int, department varchar(50), salary int)

ipo teacher table dummy database kulla create aiduchu

now i want to add degree in teachers field

so alter command can be used

alter table teacher add column degree varchar(50)

SCHEMAS

- dummy
 - Tables
 - teacher
 - Columns
 - name
 - age
 - department
 - salary
 - degree**

```

1
2
3
4 • alter table teacher add column degree varchar(50)
5
6
7
8

```

Output :::::

Action Output

21:00:47 10 21:12:32 11 21:13:02

create table student(name varchar(50), department varchar(25), marks)

use dummy

alter table teacher add column degree varchar(50)

here we have a field named department

i want to change department into shortform as dept

department -> dept

syntax: **alter table "table name" change "oldname" "newname" datatype (no. of characters)**

alter table teacher change department dept varchar(25)

Navigator :::::::

SCHEMAS

- teacher
 - Columns
 - name
 - age
 - dept**
 - salary
 - degree

```

1
2 • alter table teacher change department dept varchar(25)
3
4
5
6
7

```

14 21:20:30 15 21:21:10

use dummy

alter table teacher change department dept varchar(25)

now the field name should be same bt need to change the datatype for age -- from int to varchar

same as above

```
alter table teacher change age age varchar(50)
```

The screenshot shows the MySQL Workbench interface. On the left, there is a tree view of the database schema. Under the 'dummy' schema, there is a 'Tables' node which contains a 'teacher' table. The 'Columns' node for the 'teacher' table is expanded, showing columns named 'name', 'age', 'dept', 'salary', and 'degree'. The 'age' column is highlighted with a yellow box. Other nodes like 'Indexes', 'Foreign Keys', and 'Triggers' are also visible under the 'teacher' table. Below the tree view, there are tabs for 'Administration' and 'Schemas', with 'Schemas' currently selected. At the bottom, there is a large 'Information' pane. In this pane, the title 'Column: age' is displayed in green. Below it, the definition 'age int' is shown, with the 'int' part also highlighted by a yellow box.

Column: age

Definition:
age int

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the schema structure. Under the 'dummy' schema, there is a 'Tables' folder containing a 'teacher' table. Within the 'teacher' table, there are columns named 'name', 'age', 'dept', and 'salary'. The 'age' column is highlighted with a yellow box. Below the Navigator, a tooltip provides details about the 'age' column: 'Column: age', 'Collation: utf8mb4_0900_ai_ci', and 'Definition: age varchar(50)'. On the right, the Query Editor pane contains a single line of SQL code: 'alter table teacher change age age varchar(50)'. This line is also highlighted with a yellow box.

```
1
2 • alter table teacher change age age varchar(50)
3
4
5
6
7
8
9
10
11
```

Column: age
Collation: utf8mb4_0900_ai_ci
Definition: age varchar(50)

now if we dont want the teacher table -- if it needs to be deleted **drop** can be used

The screenshot shows the Navigator pane again. The 'Tables' folder under the 'dummy' schema now contains a 'teacher' table, but it is empty (indicated by a small icon). Below the table, there are 'Columns', 'Indexes', and 'Foreign Keys' sections, all of which are currently empty.

teacher table has been deleted

```
1
2 • drop table teacher
3
```

Truncate

it is used to delete values / data in the fields / table

truncate table teacher

table will be as it is bt the datas inside the table will get deleted

Name	Age	Department	Salary
John	23	CSE	10000
Praveen	45	ECE	20000
Suresh	34	EEE	15000

after truncate

Name	Age	Department	Salary

task

1. Create a student table with 3 columns
 - .Name
 - .Department
 - .Marks_scored
2. Add new column called City to the student table
3. Rename column Marks_scored to Marks

1.

```
create table student(
```

```
    name varchar(50),
```

```
    department varchar(25),
```

```
    marks_scored int)
```

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the 'SCHEMAS' section with 'dummy' and 'student' schemas. The 'student' schema is selected, and its 'Tables' section contains a single table named 'student'. This table has three columns: 'name', 'department', and 'marks_scored'. Handwritten annotations in red highlight the 'student' schema and the 'Table' node under it. The main workspace shows the SQL editor with the following code:

```
1  
2  
3  
4 • ⊖ create table student(  
5     name varchar(50),  
6     department varchar(25),  
7     marks_scored int)
```

Below the SQL editor is the 'Output' pane, which displays the execution history:

#	Time	Action	Message
1	20:39:15	create schema dummy	1 row(s) affected
2	20:47:28	create table teacher(name varchar(50), age int, department varchar(50), salary int)	Error Code: 1046. No database selected
3	20:56:21	use dummy	0 row(s) affected
4	20:58:06	use dummy create table teacher(name varchar(50), age int, department varchar(50), salary int)	Error Code: 1064. You have defined a column with the same name as the table
5	20:58:26	use dummy create table teacher(name varchar(50), age int, department varchar(50), salary int)	Error Code: 1064. You have defined a column with the same name as the table
6	20:59:23	create table teacher(name varchar(50), age int, department varchar(50), salary int)	0 row(s) affected
7	21:01:57	Apply changes to student	Changes applied
8	21:04:37	use student	0 row(s) affected
9	21:05:47	create table student(name varchar(50), department varchar(25), marks_scored int)	0 row(s) affected

2.

```
alter table student add column city varchar(25)
```

The screenshot shows the MySQL Workbench interface. The 'SCHEMAS' pane indicates that the 'student' schema is selected. In the 'Tables' section, the 'student' table is shown with four columns: 'name', 'department', 'marks_scored', and 'city'. The 'city' column is highlighted with a green box. The main workspace shows the SQL editor with the following code:

```
1  
2  
3  
4 • ⊖ alter table student add column city varchar(25)
```

```
12 21:15:40 use student
13 21:16:25 alter table student add column city varchar(25)
```

3.

alter table student change marks_scored marks int

```
Navigator
Query 1 ×
SCHEMAS
student
Tables
student
Columns
name
department
marks
city
Indexes
1
2 • alter table student change marks_scored marks int
3
4
5
6
7
8
```

```
16 21:25:35 use student
17 21:26:15 alter table student change marks_scored marks int
```

4. change datatype for department as int

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tab is selected, displaying the 'student' schema. Under 'Tables', there is a single entry for 'student'. In the 'Columns' section, four columns are listed: 'name', 'department', 'marks', and 'city'. The 'department' column is highlighted with a yellow box. At the bottom of the interface, the tabs 'Administration' and 'Schemas' are visible, along with a 'Information' section.

Column: department

Collation:
utf8mb4_0900_ai_ci

Definition:
department varchar(25)

```
alter table student change department department int
```

The screenshot shows the MySQL Workbench interface after executing the 'alter table' command. The 'Schemas' tab is selected, and the 'student' schema is open. The 'Tables' and 'Columns' sections are identical to the initial state, but the 'Definition' for the 'department' column has been changed to 'int'. The SQL query editor at the top shows the executed command: '1 • alter table student change department department int'. The bottom section of the interface remains the same as the first screenshot.

5. if needed to delete student table then use

```
drop table student
```

DDL - DATA DEFINITION LANGUAGE

to create and define table

=> create

=> alter

=> drop

=> truncate

now create pana table kulla information / data insert , update pananum....
information retrieve (thirumba edukanum) , delete pananum ... so to perform
these ---DML and DQL commands will be used

DML - DATA MANIPULATION LANGUAGE

=> used to insert, update , delete data

DQL - DATA QUERY LANGUAGE

=> using select command / query we can retrieve data from
database to view

b4 writing query we have to give which database we use

currently there is nothing in dummy database as we dropped it already
- now we can create table in it and work on it

1. Create table like this using DDL and add values using DML

Name	Age	Department
John	23	CSE
Praveen	45	ECE
Suresh	34	EEE

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the database schema. It shows a 'dummy' schema containing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. Below it is a 'student' schema containing a 'student' table. The 'student' table has two columns: 'name' and 'department'. On the right, the Query Editor pane is open with the following SQL command:

```
1 • use dummy
```

```
create table student (name varchar(50), age int, department varchar(50))
```

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree view shows a 'dummy' schema with a 'Tables' node expanded, revealing a 'student' table. This table has three columns: 'name', 'age', and 'department'. The 'age' column is highlighted with a yellow selection bar. On the right, the SQL editor window displays the creation of the 'student' table:

```
1  create table student (name varchar(50), age int, department varchar(50))
```

insert into student values(John,23, CSE) - if we give this it will give error

```
insert into student values(John,23, CSE)
```

```
24 15:15:26 use dummy
25 15:16:56 create table student (name varchar(50), age int, department varchar(50))
✖ 26 15:19:27 insert into student values(John,23, CSE)
```

bcoz varchar values should be given in double quotes

```
insert into student values("John",23, "CSE")
```

```
insert into student values("john",23,"CSE")
```

```
✖ 26 15:19:27 insert into student values(John,23, CSE)
✓ 27 15:21:28 insert into student values("john",23,"CSE")
```

values added bt we cant view it

so we have to use DQL - select command to view

```
select * from student
```

* means all values - table name

```
28 15:22:55 select *from student LIMIT 0, 1000 1 row(s) returned
29 15:25:26 insert values into student("praveen", 45, "ECE") Error Code: 1064. You have an error in your SQL syntax; check
```

give syntax correctly - the above is wrong

```
1 insert into student values("praveen",45, "ECE")
2 ✘ insert into student values("suresh",34,"EEE")
```

```
29 15:25:26 insert values into student("praveen", 45, "ECE") Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds .
30 15:28:20 insert into student values("praveen",45, "ECE") insert into student values("suresh",34,"EEE") Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds .
```

can't give at once - give separately and run

table created

```
1 •   select * from student  
2
```

The screenshot shows a MySQL Workbench interface. At the top, there is a command line with the following text:

```
1 •   select * from student  
2
```

Below the command line is a "Result Grid" table with three columns: name, age, and department. The data is as follows:

	name	age	department
▶	john	23	CSE
	praveen	45	ECE
	suresh	34	EEE

At the bottom of the interface, there is a log window displaying the following SQL statements and their results:

Statement ID	Time	SQL Statement	Result
31	15:29:20	insert into student values("praveen",45, "ECE")	1 row(s) affected
32	15:29:28	insert into student values("suresh",34, "EEE")	1 row(s) affected
33	15:29:46	select * from student LIMIT 0, 1000	3 row(s) returned

if any of the values needs to be updated

lets consider i have entered john age as 23 - bt need to change as 20

then update command can be used

update student set age=20 - if we give like this all the age column will change to 23

sometimes it will change all values in age as 20 bt here it shows error like condition not given where to change

The screenshot shows a log window with two error messages:

- Error 40 at 15:41:26: Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. To disable this check, add 'sql_mode=' at the start of your statement.
- Error 41 at 15:43:59: Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. To disable this check, add 'sql_mode=' at the start of your statement.

go to click EDIT , select PREFERENCE , click SQL EDITOR , scroll down SAFE UPDATE BOX click to DISABLE, OK click to QUERY option RECONNECT TO SERVER. **1170 error problem solved ..**



The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree view shows a 'dummy' schema with a 'Tables' node containing a 'student' table. The 'student' table has three columns: 'name', 'age', and 'department'. Below the table structure, there are sections for 'Indexes', 'Foreign Keys', and 'Triggers'. The 'Administration' tab is selected. In the center, a query editor window displays the SQL command 'select * from student'. To the right, a 'Result Grid' shows the data from the 'student' table:

	name	age	department
▶	John	23	CSE
	Praveen	23	ECE
	Suresh	23	EEE

the above ss is from utube - like if we set age all the values in age column will change

The screenshot shows the MySQL Workbench interface. A query editor window displays the SQL command 'update student set age=23'. The number '1' is highlighted in blue, indicating the current step.

The screenshot shows the MySQL Workbench history panel. It lists two entries:

	ID	Time	Operation
✖	48	17:00:59	update student set age=23
✓	49	17:01:57	update student set age=23

here all the datas for age has been changed to 23 bcoz we did not specifically mention to whom the age should be changed

```
1 • select * from student  
2  
3
```

The screenshot shows the MySQL Workbench 'Result Grid' window. The 'age' column is highlighted with a yellow box. The data shows that all three students now have an age of 23:

	name	age	department
▶	john	23	CSE
	praveen	23	ECE
	suresh	23	EEE

if any data needs to be updated in a table we should use **whereclass** condition

if specifically needs to be change then use where condition

eg: if we want to change department CSE - to computer

```
update student set department="computer" where department="CSE"
```

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, the query editor contains the following code:

```
1 •      select * from student
2
3
```

Below the editor is the "Result Grid" pane, which displays the following data:

	name	age	department
▶	john	23	computer
	praveen	23	ECE
	suresh	23	EEE

At the bottom of the interface, the history pane shows two recent queries:

- 52 17:07:11 update student set department="computer" where department="CSE"
- 53 17:07:27 select * from student LIMIT 0, 1000

change age of john to 30

```
update student set age=30 where name="john"
```

The screenshot shows the MySQL Workbench interface. The query editor contains the following code:

```
update student set age=30 where name="john"
```

Below the editor is the "Result Grid" pane, which displays the following data:

	name	age	department
▶	john	30	computer
	praveen	23	ECE
	suresh	23	EEE

At the bottom of the interface, the history pane shows two recent queries:

- 54 17:09:29 update student set age=30 where name="john"
- 55 17:09:47 select * from student LIMIT 0, 1000

delete

delete from student - all the datas in the student table will be deleted

so we have to give specific conditions which needs to be deleted by using where class

if we want to delete computer row --

delete from student where department="computer"

```
60 | 17:20:01 | delete from student where department="computer"
61 | 17:20:09 | select * from student LIMIT 0, 1000
```

```
1 •      select * from student
2
```

Result Grid			
	name	age	department
▶	praveen	23	ECE
	suresh	23	EEE

if i want to delete praveen suresh from table

delete from student where age=23

both age is common here so

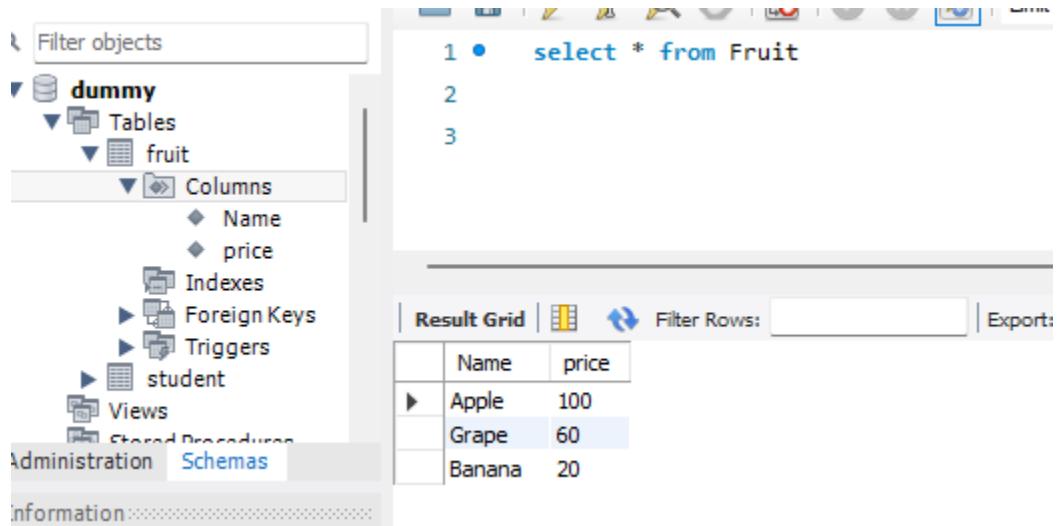
TASK

Fruit

Name	Price
Apple	100
Grape	60
Banana	20

the below is the same coding used to create

- ✓ 34 15:32:12 use dummy
- ✓ 35 15:32:58 create table Fruit(Name varchar(50), price int)
- ✓ 36 15:33:44 insert into Fruit values ("Apple", 100)
- ✓ 37 15:33:58 insert into Fruit values ("Grape",60)
- ✓ 38 15:34:16 insert into Fruit values ("Banana",20)
- ✓ 39 15:35:00 select * from Fruit LIMIT 0, 1000



Filter objects

dummy

Tables

fruit

Columns

Name

price

Indexes

Foreign Keys

Triggers

student

Views

Stored Procedures

Administration Schemas

Information

1 • select * from Fruit

2

3

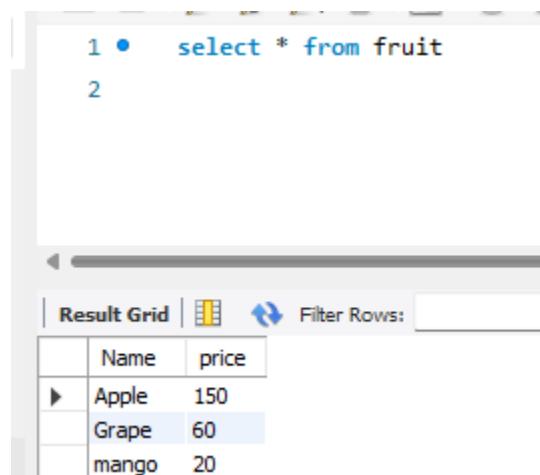
Result Grid | Filter Rows: Export:

	Name	price
▶	Apple	100
	Grape	60
	Banana	20

- Update Price of an apple to 150 in fruit table
- Update Banana to Mango in fruit table

update fruit set price=150 where price=100

✓	56	17:13:35	update fruit set price=150 where price=100
✓	57	17:15:27	update fruit set name="mango" where name="banana"



1 • select * from fruit

2

Result Grid | Filter Rows:

	Name	price
▶	Apple	150
	Grape	60
	mango	20

- Delete Apple from a table
- Delete everything from a table

```
delete from fruit where name="apple"
```

63	17:24:17	delete from fruit where name="apple"
64	17:24:26	select * from fruit LIMIT 0, 1000

Result Grid | Filter Row:

	Name	price
▶	Grape	60
	mango	20

delete everything = all datas will be deleted

```
delete from fruit
```

```
1 • select * from fruit
2
```

Result Grid | Filter Rows:

	Name	price

65	17:25:39	delete from fruit
66	17:25:46	select * from fruit LIMIT 0, 1000

```
*****
```

PRACTICE

Questions

1. How do you make a table in a database?

```
Customer(  
    customer_id,  
    customer_name,  
    customer_address,  
    city,  
    state,  
    ZIP_Code)
```

```
create table Customer(  
    customer_id int,  
    customer_name varchar(50),  
    customer_address varchar(100),  
    city varchar(50),  
    state varchar(50),  
    zip_code int  
)
```

SCHEMAS

Filter objects

dummy

practice

Tables

customer

Columns

- customer_id
- customer_name
- customer_address
- city
- state
- zip_code

Indexes

Foreign Keys

Triggers

Views

```
1 • 1 create table Customer(
2   customer_id int,
3   customer_name varchar(50),
4   customer_address varchar(100),
5
6   city varchar(50),
7   state varchar(50),
8   zip_code int
9 )
10
```

✓	67	19:13:27	create schema practice	1 row(s) affected
✗	68	19:13:55	use schema	Error Code: 1064. Y
✓	69	19:14:04	use practice	0 row(s) affected
✓	70	19:18:03	create table Customer(customer_id int, customer_name varchar(50), customer_address varchar(100), city varchar...)	0 row(s) affected

1 • 1 select * from customer

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

	customer_id	customer_name	customer_address	city	state	zip_code

2. How do you insert below records?

1. Customer ID: 1

- Name: John Doe
- Address: 392 Sunset Blvd.
- City: New York
- State: NT
- ZIP Code: 10059

3. Customer ID: 3

- Name: Richard Newman
- Address: 2040 Riverside Rd.
- City: San Diego
- State: CA
- ZIP Code: 92010

2. Customer ID: 2

- Name: Mary Smith
- Address: 6900 Main St.
- City: San Francisco
- State: CA
- ZIP Code: 94032

4. Customer ID: 4

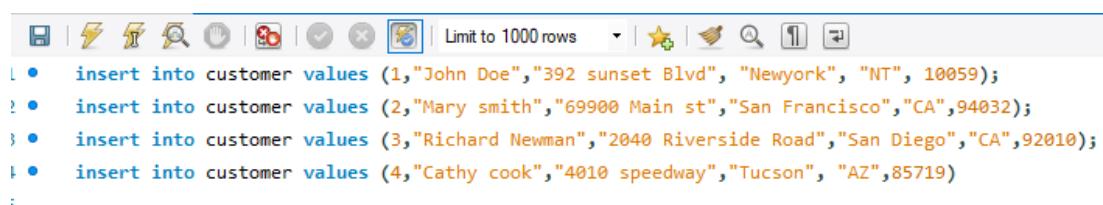
- Name: Cathy Cook
- Address: 4010 Speedway
- City: Tucson
- State: AZ
- ZIP Code: 85719

```
insert into customer values (1,"John Doe","392 sunset Blvd", "Newyork", "NT", 10059);
```

```
insert into customer values (2,"Mary smith","69900 Main st","San Francisco","CA",94032);
```

```
insert into customer values (3,"Richard Newman","2040 Riverside Road","San Diego","CA",92010);
```

```
insert into customer values (4,"Cathy cook","4010 speedway","Tucson", "AZ",85719)
```

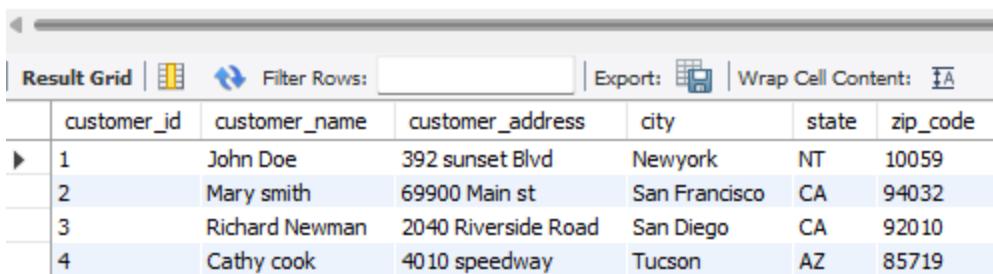


```
L •  insert into customer values (1,"John Doe","392 sunset Blvd", "Newyork", "NT", 10059);
? •  insert into customer values (2,"Mary smith","69900 Main st","San Francisco","CA",94032);
3 •  insert into customer values (3,"Richard Newman","2040 Riverside Road","San Diego","CA",92010);
4 •  insert into customer values (4,"Cathy cook","4010 speedway","Tucson", "AZ",85719)
;
```

selected and ran one by one line

✓	77	19:32:50	insert into customer values (1,"John Doe","392 sunset Blvd", "Newyork", "NT", 10059)
✓	78	19:32:54	insert into customer values (2,"Mary smith","69900 Main st","San Francisco","CA",94032)
✓	79	19:32:57	insert into customer values (3,"Richard Newman","2040 Riverside Road","San Diego","CA",92010)
✓	80	19:33:04	insert into customer values (4,"Cathy cook","4010 speedway","Tucson", "AZ",85719)

```
1 •  select * from customer  
2
```



	customer_id	customer_name	customer_address	city	state	zip_code
▶	1	John Doe	392 sunset Blvd	Newyork	NT	10059
	2	Mary smith	69900 Main st	San Francisco	CA	94032
	3	Richard Newman	2040 Riverside Road	San Diego	CA	92010
	4	Cathy cook	4010 speedway	Tucson	AZ	85719

3. How can we change the 'customer_address' column to just 'address'?

```
alter table customer change customer_address address varchar(100)
```

```
alter table customer change column customer_address address varchar(100)
```

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

	customer_id	customer_name	address	city	state	zip_code
▶	1	John Doe	392 sunset Blvd	Newyork	NT	10059
	2	Mary smith	69900 Main st	San Francisco	CA	94032
	3	Richard Newman	2040 Riverside Road	San Diego	CA	92010
	4	Cathy cook	4010 speedway	Tucson	AZ	85719


```

① 89 19:40:58 alter customer change customer_address address varchar(100)
② 90 19:41:48 alter table customer change customer_address address varchar(100)
③ 91 19:42:16 select * from customer LIMIT 0, 1000

```

4. How do you add a new column called 'mobile_number'?

`alter table customer add column mobile_number varchar(100)`

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

	customer_id	customer_name	address	city	state	zip_code	mobile_number
▶	1	John Doe	392 sunset Blvd	Newyork	NT	10059	NULL
	2	Mary smith	69900 Main st	San Francisco	CA	94032	NULL
	3	Richard Newman	2040 Riverside Road	San Diego	CA	92010	NULL
	4	Cathy cook	4010 speedway	Tucson	AZ	85719	NULL

a new column mobile_number has been added, bt we didnt give values to it... so it has null data

5.How do you delete a column where mobile_number is null?

NULL



delete from customer where mobile_number=null

Limit to 1000 rows
1 • delete from customer where mobile_number=null

✖	92	19:45:36	alter table add column mobile_number int	Error Code: 1064. You
✖	93	19:46:00	alter table add column mobile_number varchar(100)	Error Code: 1064. You
✓	94	19:46:37	alter table customer add column mobile_number varchar(100)	0 row(s) affected Rec
✓	95	19:47:09	select * from customer LIMIT 0, 1000	4 row(s) returned
✓	96	19:50:20	delete from customer where mobile_number=null	0 row(s) affected

it has executed without error bt 0 rows affected means no changes

bcoz null is not a datatype like int, varchar

so we have to give **is null**

✓	95	19:47:09	select * from customer LIMIT 0, 1000
✓	96	19:50:20	delete from customer where mobile_number=null
✓	97	19:52:34	delete from customer where mobile_number is null

→ no change

4 delete from customer where mobile_number is null
all data's deleted

Customer Data						
Customer Information		Address Details				
Customer ID	Name	Address	City	State	Zip Code	Mobile Number
1	John Doe	123 Main St	New York	NY	10001	987-654-3210

6. How do you insert below records?

1	'John Doe'	'392 Sunset Blvd.'	'New York'	'NT'	'10059'	'555-123-4567'
2	'Mary Smith'	'6900 Main St.'	'San Francisco'	'CA'	'94032'	'555-987-6543'
3	'Richard Newman'	'2040 Riverside Rd.'	'San Diego'	'CA'	'92010'	'555-555-5555'
4	'Cathy Cook'	'4010 Speedway'	'San Diego'	'CA'	'85719'	'555-321-7890'
5	'Alice Johnson'	'123 Oak Street'	'San Diego'	'CA'	'90001'	'555-111-2222'
6	'Bob Williams'	'456 Elm Avenue'	'Chicago'	'IL'	'60601'	'555-444-7777'

```
insert into customer values (1,"John Doe","392 sunset Blvd", "Newyork", "NT", 10059,"555-123-4567");
```

```
insert into customer values (2,"Mary smith","6900 Main st","San Francisco","CA",94032,"555-987-6543");
```

```
insert into customer values (3,"Richard Newman","2040 Riverside Road","San Diego","CA",92010,"555-555-5555");
```

```
insert into customer values (4,"Cathy cook","4010 speedway","Tucson", "AZ",85719,"555-321-7890");
```

```
insert into customer values (5,"Alice Johnson","123 Oak street", "san Diego","CA", 90001,"555-111-2222");
```

```
insert into customer values (6,"Bob Williams","456 Elm Avenue", "Chicago", "IL",60601,"555-444-7777");
```

7. How to show all Records

SHOW



```
3 • select * from customer
```

customer 23

Output

	customer_id	customer_name	address	city	state	zip_code	mobile_number
▶	1	John Doe	392 sunset Blvd	Newyork	NT	10059	555-123-4567
	2	Mary smith	6900 Main st	San Francisco	CA	94032	555-987-6543
	3	Richard Newman	2040 Riverside Road	San Diego	CA	92010	555-555-5555
	4	Cathy cook	4010 speedway	Tucson	AZ	85719	555-321-7890
	5	Alice Johnson	123 Oak street	san Diego	CA	90001	555-111-2222
	6	Bob Williams	456 Elm Avenue	Chicago	IL	60601	555-444-7777

Action Output

#	Time	Action	Message
✓ 105	20:11:34	insert into customer values (1,"John Doe","392 sunset Blvd", "Newyork", "NT", 10059,"555-123-4567")	1 row(s) affected
✓ 106	20:11:38	insert into customer values (2,"Mary smith","6900 Main st", "San Francisco", "CA",94032,"555-987-6543")	1 row(s) affected
✓ 107	20:11:42	insert into customer values (3,"Richard Newman","2040 Riverside Road", "San Diego", "CA",92010,"555-555-5555")	1 row(s) affected
✓ 108	20:11:45	insert into customer values (4,"Cathy cook","4010 speedway", "Tucson", "AZ",85719,"555-321-7890")	1 row(s) affected
✓ 109	20:11:49	insert into customer values (5,"Alice Johnson","123 Oak street", "san Diego", "CA", 90001,"555-111-2222")	1 row(s) affected
✓ 110	20:11:52	insert into customer values (6,"Bob Williams","456 Elm Avenue", "Chicago", "IL",60601,"555-444-7777")	1 row(s) affected
✓ 111	20:12:06	select * from customer LIMIT 0, 1000	6 row(s) returned

6 - just insert, 7 - view (select *) - both questions similar

---> to view only one row

```
2
3 • select customer_name from customer
```

customer_name
John Doe
Mary smith
Richard Newman
Cathy cook
Alice Johnson
Bob Williams

```
2
3 • select address from customer
```

address
392 sunset Blvd
6900 Main st
2040 Riverside Road
4010 speedway
123 Oak street
456 Elm Avenue

customer 25 ×

to view 2 columns

2

3 • `select address,mobile_number from customer`

address	mobile_number
392 sunset Blvd	555-123-4567
6900 Main st	555-987-6543
2040 Riverside Road	555-555-5555
4010 speedway	555-321-7890
123 Oak street	555-111-2222
456 Elm Avenue	555-444-7777

1 • `select customer_name,address,mobile_number from customer`

customer_name	address	mobile_number
John Doe	392 sunset Blvd	555-123-4567
Mary smith	6900 Main st	555-987-6543
Richard Newman	2040 Riverside Road	555-555-5555
Cathy cook	4010 speedway	555-321-7890
Alice Johnson	123 Oak street	555-111-2222
Bob Williams	456 Elm Avenue	555-444-7777

Customer 27 ×

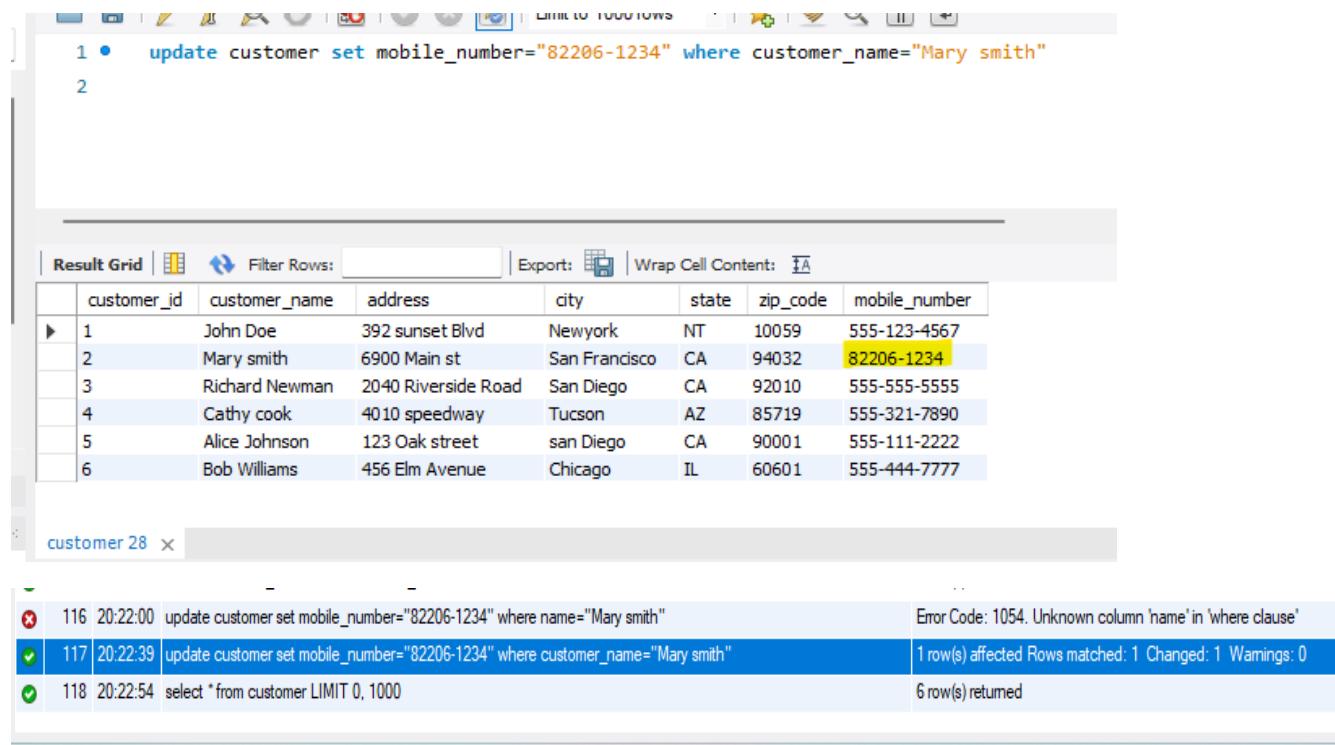
8. How can you update the phone number for 'Mary Smith' to '82206-1234'?



UPDATE



```
update customer set mobile_number="82206-1234" where  
customer_name="Mary smith"
```



The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a SQL editor window containing the following code:

```
1 • update customer set mobile_number="82206-1234" where customer_name="Mary smith"
2
```

Below the SQL editor is a Result Grid table with columns: customer_id, customer_name, address, city, state, zip_code, and mobile_number. The data is as follows:

	customer_id	customer_name	address	city	state	zip_code	mobile_number
▶	1	John Doe	392 sunset Blvd	Newyork	NT	10059	555-123-4567
	2	Mary smith	6900 Main st	San Francisco	CA	94032	82206-1234
	3	Richard Newman	2040 Riverside Road	San Diego	CA	92010	555-555-5555
	4	Cathy cook	4010 speedway	Tucson	AZ	85719	555-321-7890
	5	Alice Johnson	123 Oak street	san Diego	CA	90001	555-111-2222
	6	Bob Williams	456 Elm Avenue	Chicago	IL	60601	555-444-7777

At the bottom of the interface, there's a log window titled "customer 28" showing the following entries:

ID	Time	Query	Message
116	20:22:00	update customer set mobile_number="82206-1234" where name="Mary smith"	Error Code: 1054. Unknown column 'name' in 'where clause'
117	20:22:39	update customer set mobile_number="82206-1234" where customer_name="Mary smith"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
118	20:22:54	select * from customer LIMIT 0, 1000	6 row(s) returned

9.How can you delete the record where the ZIP_Code is "60601"?



```
delete from customer where zip_code=60601
```

```
1 •   select * from customer LIMIT 0, 1000
2
```

Result Grid							
	customer_id	customer_name	address	city	state	zip_code	mobile_number
▶	1	John Doe	392 sunset Blvd	Newyork	NT	10059	555-123-4567
	2	Mary smith	6900 Main st	San Francisco	CA	94032	82206-1234
	3	Richard Newman	2040 Riverside Road	San Diego	CA	92010	555-555-5555
	4	Cathy cook	4010 speedway	Tucson	AZ	85719	555-321-7890
	5	Alice Johnson	123 Oak street	san Diego	CA	90001	555-111-2222

119	20:26:33	delete from customer where zip_code=60601
120	20:26:42	select * from customer LIMIT 0, 1000

10. How can you select distinct data where the user is from a state ="CA"?

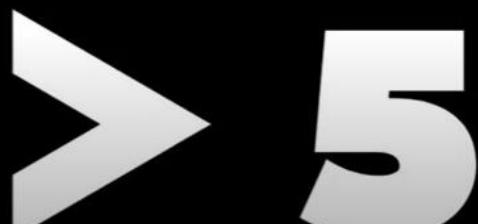


```
select * from customer where state="CA"
```

```
1 •  select * from customer where state="CA"  
2
```

Result Grid							
	customer_id	customer_name	address	city	state	zip_code	mobile_number
▶	2	Mary smith	6900 Main st	San Francisco	CA	94032	82206-1234
	3	Richard Newman	2040 Riverside Road	San Diego	CA	92010	555-555-5555
	5	Alice Johnson	123 Oak street	san Diego	CA	90001	555-111-2222

11. How do you print customer IDs who are greater than 5?



no records bcoz we have total of 5 id's -- greater than 5 - then no values

The screenshot shows a MySQL query window. The SQL query is:

```
1 •   select * from customer where customer_id>5
2
```

The results pane shows a table header for "Result Grid" with columns: customer_id, customer_name, address, city, state, zip_code, mobile_number. There are no rows displayed.

lets try >2

```
1 •  select * from customer where customer_id>2  
2
```

	customer_id	customer_name	address	city	state	zip_code	mobile_number
▶	3	Richard Newman	2040 Riverside Road	San Diego	CA	92010	555-555-5555
▶	4	Cathy cook	4010 speedway	Tucson	AZ	85719	555-321-7890
▶	5	Alice Johnson	123 Oak street	san Diego	CA	90001	555-111-2222

13. How do you delete the mobile number column?



```
alter table customer drop column mobile_number
```

- 123 20:32:27 select * from customer where customer_id>2 LIMIT 0, 1000
- 124 20:55:50 delete from customer where column mobile_number
- 125 20:56:35 delete from customer column mobile_number
- 126 20:57:35 drop customer column mobile_number
- 127 20:58:01 drop from customer column mobile_number
- 128 20:58:29 drop from customer where column mobile_number
- 129 20:58:47 drop table customer where column mobile_number
- 130 20:58:55 drop table customer column mobile_number
- 131 21:00:36 alter table customer drop column mobile_number

mobile_number column has been deleted

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

	customer_id	customer_name	address	city	state	zip_code
▶	1	John Doe	392 sunset Blvd	Newyork	NT	10059
	2	Mary smith	6900 Main st	San Francisco	CA	94032
	3	Richard Newman	2040 Riverside Road	San Diego	CA	92010
	4	Cathy cook	4010 speedway	Tucson	AZ	85719
	5	Alice Johnson	123 Oak street	san Diego	CA	90001

14. How do you truncate a table?



```
truncate table customer
```

all datas in the table deleted

The screenshot shows a MySQL Workbench interface. In the top-left pane, there is a query editor window containing the following SQL code:

```
1 •  truncate table customer
2
```

Below the query editor is a results pane titled "Result Grid". It displays a table structure with the following columns:

customer_id	customer_name	address	city	state	zip_code

AGGREGATE FUNCTIONS

- 1. sum()**
- 2. min()**
- 3. max()**
- 4. avg()**
- 5. count()**

oru table la iruka columns la inthe aggregate functions use pani sum, min value, max value, average, count find panalam

```
create table student1(
```

student_name varchar(100),

student_mark int,

department varchar(20))

insert into student1 values("sangeetha",50,"CSE");

insert into student1 values("sindhu", 45,"Commerce");

insert into student1 values("sarасwathi", 68,"cooking");

insert into student1 values("anbazagan",75,"EB");

insert into student1 values("vijaya",46,"cooking")

SCHEMAS

Filter objects

- dummy
 - Tables
 - fruit
 - student
 - student1**
 - Views
 - Stored Procedures
 - Functions
- practice
 - Tables
 - customer
 - Views
 - Stored Procedures
 - Functions
- student

Administration Schemas

Information

No object selected

```
1 • select * from student1
2
```

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

	student_name	student_mark	department
▶	sangeetha	50	CSE
	sindhu	45	Commerce
	saraswathi	68	cooking
	anbazagan	75	EB
	vijaya	46	cooking

student1 1 ×

Output

Action Output

#	Time	Action
1	22:39:08	use dummy
2	22:44:43	create table student1(student_name varchar(100), student_mark int, department varchar(2...)
3	22:45:14	create table student1(student_name varchar(100), student_mark int, department varchar(2...)
4	22:45:19	insert into student1 values("sangeetha",50,"CSE")
5	22:45:19	insert into student1 values("sindhu", 45, "Commerce")
6	22:45:19	insert into student1 values("sarawathi", 68, "cooking")
7	22:45:19	insert into student1 values("anbazagan",75,"EB")
8	22:45:19	insert into student1 values("vijaya",46,"cooking")
9	22:45:57	select * from student1 LIMIT 0, 1000

Object Info Session

select student_mark from student1

```
1 • select student_mark from student1
2
```

Result Grid | Filter Rows: Export:

student_mark
50
45
68
75
46

total mark of all the student

```
select sum(student_mark) from student1
```

```
1 •   select sum(student_mark) from student1
2
```

Result Grid		Filter Rows:	Export:
	sum(student_mark)		
▶	284		

here we can change column name

after sum - the column name will be sum(student_mark)

now we can change it to totalmark

```
select sum(student_mark) as totalmark from student1
```

```
1 •   select sum(student_mark) as totalmark from student1
2
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	totalmark			
▶	284			

```
9 22:45:57 select * from student1 LIMIT 0, 1000
10 22:49:29 select student_mark from student1 LIMIT 0, 1000
11 22:50:49 select sum(student_mark) from student1 LIMIT 0, 1000
12 22:53:19 select sum(student_mark) as totalmark from student1 LIMIT 0, 1000
```

to find minimum mark

```
select min(student_mark) as minmark from student1
```

```
1 •   select min(student_mark) as minmark from student1
2
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	minmark			

▶ 45

to find maximum mark

```
select max(student_mark) as maxmark from student1
```

```
1 •   select max(student_mark) as maxmark from student1
2
```

Result Grid		Filter Rows:	Export:	Wrap Cell Co
	maxmark			

▶ 75

to find average mark

```
select avg(student_mark) as avgmark from student1
```

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a query editor window containing the following text:

```
1 • select avg(student_mark) as avgmark from student1
2
```

The screenshot shows the results of the query in a grid format. The grid has two columns: 'Result Grid' and 'avgmark'. The 'avgmark' column contains the value '56.8000'. The 'Result Grid' column is highlighted with a yellow background.

Result Grid	avgmark
	56.8000

to find count

```
select count(student_name) from student1
```

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a query editor window containing the following text:

```
1 • select count(student_name) from student1
2
```

Below the query editor is a results grid. The grid has two columns: 'Result Grid' and 'count(student_name)'. The 'count(student_name)' column contains the value '5'. The 'Result Grid' column is highlighted with a yellow background.

Result Grid	count(student_name)
	5

At the bottom of the screen, there is a history of previous queries:

- 13 22:55:55 select min(student_mark) as minmark from student1 LIMIT 0, 1000
- 14 22:56:50 select max(student_mark) as maxmark from student1 LIMIT 0, 1000
- 15 22:59:04 select avg(student_mark) as avgmark from student1 LIMIT 0, 1000
- 16 23:00:57 select count(student_name) from student1 LIMIT 0, 1000

```
select student_name from student1
```

```
1 •   select student_name from student1
2
```

Result Grid | Filter Rows: Export

student_name
sangeetha
sindhu
saraswathi
anbazagan
vijaya

select student_name from student1 where department="cooking"

```
1 •   select student_name from student1 where department="cooking"
2
```

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

student_name
saraswathi
vijaya

select * from student1 where department="cooking"

```
1 •   select * from student1 where department="cooking"
2
```

Result Grid			
	student_name	student_mark	department
▶	saraswathi	68	cooking
	vijaya	46	cooking

```
select count(student_name) from student1 where
department="cooking"
```

```
1 •   select count(student_name) from student1 where department="cooking"
2
```

Result Grid			
	count(student_name)		
▶	2		

- 17 23:03:09 select student_name from student1 LIMIT 0, 1000
- 18 23:04:30 select student_name from student1 where department="cooking" LIMIT 0, 1
- 19 23:05:25 select * from student1 where department="cooking" LIMIT 0, 1000
- 20 23:06:10 select * from student1 LIMIT 0, 1000

```
select count(student_name) from student1 where
student_mark>40
```

```
1 •  select count(student_name) from student1 where student_mark>40
2
```

Result Grid	
	count(student_name)
▶	5

**select count(student_name) from student1 where
student_mark>60**

Result Grid	
	count(student_name)
▶	2

- ✖ 21 23:09:04 select count() from student1 where department="cooking" LIMIT 0, 1000
- ✓ 22 23:09:22 select count(student_name) from student1 where department="cooking" LIMIT 0, 1000
- ✓ 23 23:10:47 select count(student_name) from student1 where student_mark>40 LIMIT 0, 1000
- ✓ 24 23:11:30 select count(student_name) from student1 where student_mark>60 LIMIT 0, 1000
- ✓ 25 23:12:13 select * from student1 LIMIT 0, 1000

group by , order by

this is also aggregate function

```
use dummy;
```

```
create table student2(  
student_name varchar(100),  
student_marks int,  
department varchar(50));
```

```
insert into student2(student_name, student_marks, department)
```

```
values
```

```
("ranjani", 78,"CSE"),  
("viki", 57,"ECE"),  
("sangeetha",89,"CSE"),  
("sindhu",69,"commerce"),  
("divya", 72,"ECE"),  
("swathi",47,"commerce"),  
("lilly",84,"ECE"),  
("nivetha",61,"CSE")
```

```
select * from student2
```

```
1 • select * from student2
```

	student_name	student_marks	department
▶	ranjani	78	CSE
	viki	57	ECE
	sangeetha	89	CSE
	sindhu	69	commerce
	divya	72	ECE
	swathi	47	commerce
	lilly	84	ECE
	nivetha	61	CSE

✓	36	15:56:42	insert into student2(student_name, student_marks, department) values ("ranjani", 78,"CSE")	8 row(s) affected Records: 8
✓	37	15:57:36	select * from student2 LIMIT 0, 1000	8 row(s) returned

if we want to view only CSE department

```
select * from student2 where department="CSE"
```

```
1 select * from student2 where department="CSE"
```

	student_name	student_marks	department
▶	ranjani	78	CSE
	sangeetha	89	CSE
	nivetha	61	CSE

select student_name,student_marks from student2

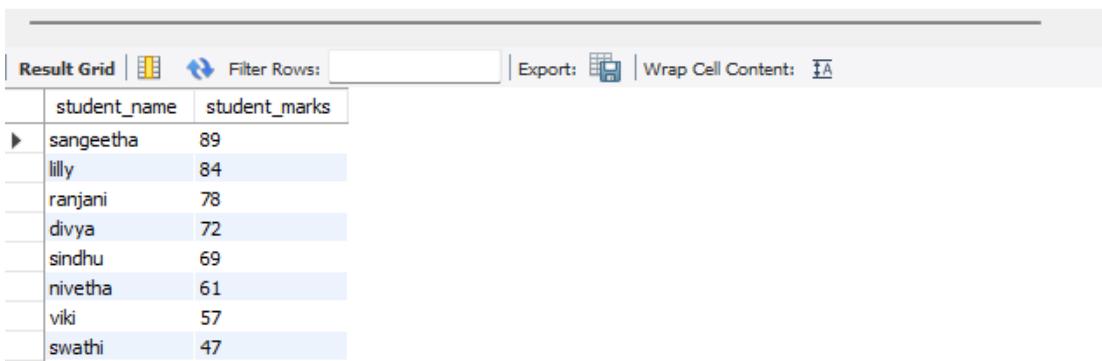
```
1 select student_name,student_marks from student2
```

	student_name	student_marks
▶	ranjani	78
	viki	57
	sangeetha	89
	sindhu	69
	divya	72
	swathi	47
	lilly	84
	nivetha	61

order by -- now we gonna align marks from highest to lowest - by using order by asc or desc

```
select student_name,student_marks from student2 order by student_marks desc
```

```
1      select student_name,student_marks from student2 order by student_marks desc
```



A screenshot of a database query results grid. The grid has two columns: 'student_name' and 'student_marks'. The data rows are: sangeetha (89), lilly (84), ranjani (78), divya (72), sindhu (69), nivetha (61), viki (57), and swathi (47). The grid includes standard database navigation buttons like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'.

	student_name	student_marks
▶	sangeetha	89
	lilly	84
	ranjani	78
	divya	72
	sindhu	69
	nivetha	61
	viki	57
	swathi	47

lowest to highest

```
select student_name,student_marks from student2 order by student_marks asc
```

```
1      select student_name,student_marks from student2 order by student_marks asc
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	student_name	student_marks		
▶	swathi	47		
▶	viki	57		
▶	nivetha	61		
▶	sindhu	69		
▶	divya	72		
▶	ranjani	78		
▶	lilly	84		
▶	sangeetha	89		

```
1      select Student_marks from student2
```

Result Grid		Filter Rows:	Export:
	Student_marks		
▶	78		
▶	57		
▶	89		
▶	69		
▶	72		
▶	47		
▶	84		
▶	61		

```
1      select sum(Student_marks) from student2
```

Result Grid		Filter Rows:	Export:
sum(Student_marks)			
▶ 557			

```
1      select avg(Student_marks) from student2
```

Result Grid		Filter Rows:	Export:
avg(Student_marks)			
▶ 69.6250			

if we want to view all department by departmentwise then we have to use group by

after groupby we have to mention based on which field - here department will be common so we can use department

to find average mark in each department

```
select department,avg(Student_marks) from student2 group by  
department
```

```
1 select department,avg(Student_marks) from student2 group by department
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	department	avg(Student_marks)		
▶	CSE	76.0000		
	ECE	71.0000		
	commerce	58.0000		

to find student count in each department

here we can give anything inside count () function

```
select department,count(Student_marks) from student2 group by department
```

```
1 select department,count(Student_marks) from student2 group by department
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	department	count(Student_marks)		
▶	CSE	3		
	ECE	3		
	commerce	2		

```
select department,count(Student_name) from student2 group by department
```

```
1 select department,count(Student_name) from student2 group by department
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
department	count(Student_name)			
CSE	3			
ECE	3			
commerce	2			

with group by we can use order by as well

after order by whatever the aggregate function (avg(), sum() etc...) we used should be given after order by also followed by asc or desc

```
select department,count(Student_name) from student2 group by  
department order by count(Student_name) asc
```

```
1 • select department,count(Student_name) from student2 group by department order by count(Student_name) asc  
2
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
department	count(Student_name)			
commerce	2			
CSE	3			
ECE	3			

```
select department,count(Student_name) from student2 group by  
department order by count(Student_name) desc
```

```
1 •   select department,count(Student_name) from student2 group by department order by count(Student_name) desc  
2
```

Result Grid	
department	count(Student_name)
CSE	3
ECE	3
commerce	2

1. CREATE A EMPLOYEE TABLE .

```
(1, 'JOHN', 'DOE', 'HR', 55000),  
(2, 'JANE', 'SMITH', 'IT', 60000),  
(3, 'BOB', 'JOHNSON', 'IT', 62000),  
(4, 'ALICE', 'WILLIAMS', 'HR', 54000),  
(5, 'EVA', 'DAVIS', 'FINANCE', 58000),  
(6, 'MIKE', 'BROWN', 'FINANCE', 59000);
```

- `use dummy;`
- `create table employee(`
`E_ID int,`
`F_name varchar(50),`
`L_name varchar(50),`
`dept varchar(35),`
`salary int);`
- `insert into employee (E_ID,F_name,L_name,dept,salary)`
`values`
`(1,"John", "Doe","HR",55000),`
`(2,"Jane","Smith","IT",60000),`
`(3,"Bob","Johnson","IT",62000),`
`(4,"Alice","Williams","HR",54000),`
`(5,"Eva","Davis","Finance",58000),`
`(6,"Mike","Brown","Finance",59000)`

```
use dummy;
```

```
create table employee(  

E_ID int,  

F_name varchar(50),  

L_name varchar(50),  

dept varchar(35),  

salary int );
```

```
insert into employee (E_ID,F_name,L_name,dept,salary)  

values  

(1,"John", "Doe","HR",55000),  

(2,"Jane","Smith","IT",60000),  

(3,"Bob","Johnson","IT",62000),  

(4,"Alice","Williams","HR",54000),
```

(5,"Eva","Davis","Finance",58000),

(6,"Mike","Brown","Finance",59000)

```
✓ 55 16:43:51 use dummy 0 row(s) affected
✓ 56 16:47:39 create table employee( E_ID int, F_name varchar(50), L_name varchar(50), dept varchar(3... 0 row(s) affected
✓ 57 16:52:34 insert into employee (E_ID,F_name,L_name,dept,salary) values (1,"John", "Doe","HR",55... 6 row(s) affected
```

```
19 • select * from employee
```

```
20
```

Result Grid					
	E_ID	F_name	L_name	dept	salary
▶	1	John	Doe	HR	55000
	2	Jane	Smith	IT	60000
	3	Bob	Johnson	IT	62000
	4	Alice	Williams	HR	54000
	5	Eva	Davis	Finance	58000
	6	Mike	Brown	Finance	59000

QUESTION 1

LIST ALL EMPLOYEES IN ALPHABETICAL ORDER BY LAST NAME:

```
select * from employee order by L_name asc
```

```
10  
19 • select * from employee order by L_name asc  
20
```

	E_ID	F_name	L_name	dept	salary
▶	6	Mike	Brown	Finance	59000
	5	Eva	Davis	Finance	58000
	1	John	Doe	HR	55000
	3	Bob	Johnson	IT	62000
	2	Jane	Smith	IT	60000
	4	Alice	Williams	HR	54000

QUESTION 2

LIST ALL EMPLOYEES IN THE IT DEPARTMENT AND SORT THEM BY SALARY IN DESCENDING ORDER :

```
select * from employee where dept="IT" order by salary desc
```

```
21 • select * from employee where dept="IT" order by salary desc  
22
```

	E_ID	F_name	L_name	dept	salary
▶	3	Bob	Johnson	IT	62000
	2	Jane	Smith	IT	60000

```
60 17:01:27 select * from employee order by salary desc LIMIT 0, 1000  
✖ 61 17:01:59 select *from employee order by salary desc where department="IT"  
✓ 62 17:02:55 select *from employee where dept="IT" order by salary desc LIMIT 0, 1000
```

QUESTION 3

FIND THE TOTAL NUMBER OF EMPLOYEES IN EACH DEPARTMENT :

```
select dept,count(*) from employee group by dept
```

```
23 • select dept,count(*) from employee group by dept
```

```
24
```

Result Grid | Filter Rows: Export: Wrap Cell Content

dept	count(*)
HR	2
IT	2
Finance	2

if they ask for particular dept -- eg: IT .. then we can use where
for all dept - to find count - use groupby

QUESTION 4

CALCULATE THE AVERAGE SALARY FOR EACH DEPARTMENT ,SORTED IN ASCENDING ORDER BY DEPARTMENT NAME :

```
select dept,avg(salary) from employee group by dept order by avg(salary) asc
```

```
25 • select dept,avg(salary) from employee group by dept order by avg(salary) asc  
26  
27
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
dept	avg(salary)			
HR	54500.0000			
Finance	58500.0000			
IT	61000.0000			

the above is for order by of salary - based on its ascending order it printed

bt the question is to sort by ascending order by dept name so..
based on alphabetical order

```
24  
25 • select dept,avg(salary) from employee group by dept order by dept asc  
26
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
dept	avg(salary)			
Finance	58500.0000			
HR	54500.0000			
IT	61000.0000			

QUESTION 5

FIND THE DEPARTMENT WITH THE HIGHEST AVERAGE SALARY:

```
select dept,avg(salary) from employee group by dept order by avg(salary) desc;
```

```
29 • select dept,avg(salary) from employee group by dept order by avg(salary) desc;  
30  
31  
32  
33
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	dept	avg(salary)		
▶	IT	61000.0000		
	Finance	58500.0000		
	HR	54500.0000		

the qn. is to find **dept with highest avg salary**

limit 1 - prints only 1st row

```
29 • select dept,avg(salary) from employee group by dept order by avg(salary) desc  
30     limit 1  
31
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	dept	avg(salary)			
▶	IT	61000.0000			

to find 2 dept with lowest avg. salary

limit 2 - prints first 2 rows

```

29 •   select dept,avg(salary) from employee group by dept order by avg(salary) asc
30     limit 2
31
32
33

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	dept	avg(salary)			
▶	HR	54500.0000			
	Finance	58500.0000			

Having clause

to find what are the department is earning less than 60000

```

71 20:13:16 select avg(salary)from employee where avg(salary)<60000 group by dept l
72 20:15:07 select avg(salary)from employee group by dept where avg(salary)<60000

```

select avg(salary) from employee group by dept where avg(salary)<60000

select avg(salary) from employee where avg(salary)<60000 group by dept

```

select avg(salary) from employee group by dept where avg(salary)<60000
select avg(salary) from employee where avg(salary)<60000 group by dept |

```

group by ku aprum where poda kudathu, so used b4 group by bt still getting error

we are getting error when we use where

bcoz inga aggregate function la condition use pana try pandrom

aggregate function la condition use pananum apdina having clause use pananum

```
select department,avg(salary) from employee group by dept having avg(salary)<60000
```

```
36
37 • select department,avg(salary) from employee group by dept having avg(salary)<60000
38
```

	avg(salary)
▶	54500.0000
	58500.0000



```
select dept,avg(salary) from employee group by dept having avg(salary)>55000
```

```
38
39 • select dept,avg(salary) from employee group by dept having avg(salary)>55000
40
```

dept	avg(salary)
IT	61000.0000
Finance	58500.0000

more than 1 employee

```
41
42 •   select dept,count(*) from employee group by dept having count(*)>1
43
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	dept	count(*)		
▶	HR	2		
	IT	2		
	Finance	2		

more than 2 employees - no more than 2 - so no values

```
42 •   select dept,count(*) from employee group by dept having count(*)>2
43
44
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	dept	count(*)		

```
42 •   select dept,count(*) from employee group by dept having count(*)>=2
43
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	dept	count(*)		
▶	HR	2		
	IT	2		
	Finance	2		

in question they asked to find both avg salary greater , more than 2 employees at once

select dept, avg(salary), count(*) from employee

```
select dept, avg(salary), count(*) from employee
```

```
85 20:52:48 select dept,count(*) from employee group by dept having count(*)>:  
86 20:55:55 select dept, avg(salary), count(*) from employee LIMIT 0, 1000
```

here we have not given group by so error

select dept, avg(salary), count(*) from employee group by dept having avg(salary)>55000 or count(*)>=2

dept	avg(salary)	count(*)
HR	54500.0000	2
IT	61000.0000	2
Finance	58500.0000	2

select dept, avg(salary), count(*) from employee group by dept having avg(salary)>55000 and count(*)>=2

dept	avg(salary)	count(*)
IT	61000.0000	2
Finance	58500.0000	2

25 Practice Questions | DDL, DML, DQL, Group by, Having

1

Create a table named "Students" with columns for ID (INT) , Name (VARCHAR) and Age (INT).

ID	Name	Age

```
create schema practice1;
```

```
use practice1;
```

```
create table students (
```

```
    ID int,
```

```
    name varchar(30),
```

```
    age int);
```

```
select * from students;
```

Filter objects

- dummy
- practice
- practice1

Tables

- students
 - Columns
 - ID
 - name
 - age
 - Indexes
 - Foreign Keys
 - Triggers
- Views
- Stored Procedures
- Functions

student Administration Schemas

Information

```
1 •  create schema practice1;
2
3 •  use practice1;
4
5 •  create table students (
6     ID int,
7     name varchar(30),
8     age int);
9
10 •   select * from students;
11
```

Result Grid | Filter Rows:

	ID	name	age
--	----	------	-----

Action Output

#	Time	Action
1	21:14:02	create schema practice1
2	21:16:18	create table students (ID int, name varchar(30), age int)
3	21:16:47	use practice1 create table students (ID int, name varchar(30), age int)
4	21:17:02	use practice1
5	21:17:21	create table students (ID int, name varchar(30), age int)
6	21:17:59	select *from students LIMIT 0, 1000

2

Add a new column "Grade" (VARCHAR) to the "Students" table.



```
alter table students add column grade varchar(20)
```

```
1 • alter table students add column grade varchar(20)
2
```

Result Grid | Filter Rows: [] Export: [] Wrap Cell Cor

	ID	name	age	grade
7	21:21:08	alter table students add column grade varchar(20)		
8	21:21:20	select * from students LIMIT 0, 1000		

3

Rename the "Grade" column to "FinalGrade."



```
alter table students change column grade FinalGrade varchar(20)
```

```
1 • alter table students change column grade FinalGrade varchar(20);
2 • select * from students
3
```

Result Grid			
	ID	name	age
			FinalGrade

- 9 21:27:59 alter table students change column grade FinalGrade varchar(20)
- 10 21:28:13 select * from students LIMIT 0, 1000

4

Insert 10 new students as dummy Records.

101	'John Doe'	20	'A'
102	'Jane Smith'	22	'B'
103	'Bob Johnson'	19	'C'
104	'Alice Brown'	21	'A'
105	'Charlie Davis'	20	'B'
106	'Emma Wilson'	23	'A'
107	'Michael Lee'	20	'C'
108	'Olivia Moore'	19	'B'
109	'William Turner'	21	'A'
110	'Sophia Rodriguez'	22	'C'

insert into students(ID,name, age,FinalGrade)

values

(101,"John Doe", 20,"A"),
(102,"Jane Smith",22,"B"),
(103,"Bob Johnson",19,"C"),
(104,"Alice Brown", 21,"A"),
(105,"Charlie Davis",20,"B"),
(106,"Emma Wilson",23,"A"),
(107,"Michael Lee",20,"C"),
(108,"Olivia Moore",19,"B"),
(109,"William Turner",21,"A"),
(110,"Sophia Rodriguez",22,"C")

```
1 •  insert into students(ID,name, age,FinalGrade)
2   values
3     (101,"John Doe", 20,"A"),
4     (102,"Jane Smith",22,"B"),
5     (103,"Bob Johnson",19,"C"),
6     (104,"Alice Brown", 21,"A"),
7     (105,"Charlie Davis",20,"B"),
8     (106,"Emma Wilson",23,"A"),
9     (107,"Michael Lee",20,"C"),
10    (108,"Olivia Moore",19,"B"),
11    (109,"William Turner",21,"A"),
12    (110,"Sophia Rodriguez",22,"C")
```

Result Grid | Filter Rows: | Export: | Wrap

	ID	name	age	FinalGrade
▶	101	John Doe	20	A
	102	Jane Smith	22	B
	103	Bob Johnson	19	C
	104	Alice Brown	21	A
	105	Charlie Davis	20	B
	106	Emma Wilson	23	A
	107	Michael Lee	20	C
	108	Olivia Moore	19	B
	109	William Turner	21	A
	110	Sophia Rodriguez	22	C

5

Update the age of student with ID 101 to 21.



update students set age=21 where ID=101

```
14 • update students set age=21 where ID=101;  
15  
16 • select * from students
```

Result Grid | Filter Rows: Export: \

ID	name	age	FinalGrade
101	John Doe	21	A
102	Jane Smith	22	B
103	Bob Johnson	19	C

24 22:07:10 update students set age=21 where ID=101
25 22:07:30 select * from students LIMIT 0, 1000

6

Delete the student with ID 101 from the "Students" table.

101		'John Doe'	20	'A'
102		'Jane Smith'	22	'B'
103		'Bob Johnson'	19	'C'
104		'Alice Brown'	21	'A'
105		'Charlie Davis'	20	'B'
106		'Emma Wilson'	23	'A'
107		'Michael Lee'	20	'C'
108		'Olivia Moore'	19	'B'
109		'William Turner'	21	'A'
110		'Sophia Rodriguez'	22	'C'

delete from students where ID=101

18 • delete from students where ID=101

Result Grid				
	ID	name	age	FinalGrade
▶	102	Jane Smith	22	B
▶	103	Bob Johnson	19	C
▶	104	Alice Brown	21	A
▶	105	Charlie Davis	20	B
▶	106	Frank Miller	22	A

✖	26	22:10:43	drop from students where ID=101
✖	27	22:11:37	drop from table students where ID=101
✓	28	22:12:23	delete from students where ID=101
✓	29	22:12:31	select * from students LIMIT 0, 1000

7

Retrieve all students aged 19 or older.



select * from students where age>=19

3 • `select * from students where age>=19`

	ID	name	age	FinalGrade
▶	102	Jane Smith	22	B
	103	Bob Johnson	19	C
	104	Alice Brown	21	A
	105	Charlie Davis	20	B
	106	Emma Wilson	23	A
	107	Michael Lee	20	C
	108	Olivia Moore	19	B
	109	William Turner	21	A
	110	Sophia Rodriguez	22	C

greater than 21

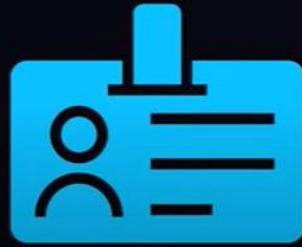
`select * from students where age>21`

3 • `select * from students where age>21`

	ID	name	age	FinalGrade
▶	102	Jane Smith	22	B
	106	Emma Wilson	23	A
	110	Sophia Rodriguez	22	C
	102	Jane Smith	22	B
	106	Emma Wilson	23	A
	110	Sophia Rodriguez	22	C

8

Retrieve students named 'William Turner' or "Alice Brown"



```
select * from students where age=21
```

here we have only less data and we can see age 21 only for these 2 so we used this..if lot of data we cant use this-- find below

```
1 •  select * from students where age=21
2
3
```

Result Grid				
	ID	name	age	FinalGrade
▶	104	Alice Brown	21	A
	109	William Turner	21	A

```
select * from students where name="William Turner" or name="Alice Brown"
```

```
1 •  select * from students where name="William Turner" or name="Alice Brown"  
2  
3
```

ID	name	age	FinalGrade
104	Alice Brown	21	A
109	William Turner	21	A

we are using **in** operator to perform this

this we used in python operations samewise

here we will have some list of jobs, we will check if the given job is in the list.. if so it will be printed -- like wise in operator works here

ID	name	age	finalgrade
104	Alice Brown	21	A
109	William Turner	21	A

```
select * from students where name in ("William Turner", "Alice Brown")
```

```
1 • select * from students where name in ("William Turner" , "Alice Brown")
2
3
```

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

ID	name	age	FinalGrade
104	Alice Brown	21	A
109	William Turner	21	A

38 18:29:14 select * from students where name="William Turner" or name="Alice Brown" LIMIT 0, 1000
39 18:33:41 select * from students where name in ("William Turner" , "Alice Brown") LIMIT 0, 1000

9

Retrieve students with grades 'A' or 'B' and aged 20 or older.



```
select * from students where FinalGrade in ("A","B") and age>=20
```

```
1 •  select * from students where FinalGrade in ("A","B") and age>=20
2
3
```

Result Grid				
	ID	name	age	FinalGrade
▶	102	Jane Smith	22	B
	104	Alice Brown	21	A
	105	Charlie Davis	20	B
	106	Emma Wilson	23	A
	109	William Turner	21	A

```
select * from students where FinalGrade="A" or FinalGrade="B" and
age>=20
```

```
1 •  select * from students where FinalGrade="A" or FinalGrade="B" and age>=20
2
3
```

Result Grid				
	ID	name	age	FinalGrade
▶	102	Jane Smith	22	B
	104	Alice Brown	21	A
	105	Charlie Davis	20	B
	106	Emma Wilson	23	A
	109	William Turner	21	A

40	18:39:39	select * from students where FinalGrade in ("A","B") and age>=20 LIMIT 0, 1000
41	18:42:55	select * from students where FinalGrade="A" or FinalGrade="B" and age>=20 LIMIT 0, 1000

10

Retrieve students aged between 18 and 25.

18

25

for 18 to 21

```
select * from students where age>=18 and age<=21
```

```
1 •  select * from students where age>=18 and age<=21  
2  
3
```

ID	name	age	FinalGrade
103	Bob Johnson	19	C
104	Alice Brown	21	A
105	Charlie Davis	20	B
107	Michael Lee	20	C
108	Olivia Moore	19	B
109	William Turner	21	A

between key is used here

```
select * from students where age between 18 and 20
```

```
1 •  select * from students where age between 18 and 20  
2  
3
```

ID	name	age	FinalGrade
103	Bob Johnson	19	C
105	Charlie Davis	20	B
107	Michael Lee	20	C
108	Olivia Moore	19	B
.	.	.	.

select * from students where age between 18 and 25

```
1 •  select * from students where age between 18 and 25  
2  
3
```

ID	name	age	FinalGrade
102	Jane Smith	22	B
103	Bob Johnson	19	C
104	Alice Brown	21	A
105	Charlie Davis	20	B
106	Emma Wilson	23	A
107	Michael Lee	20	C
108	Olivia Moore	19	B
109	William Turner	21	A
110	Sophia Rodriguez	22	C

11

Retrieve students with ages less than 18.

< 18

no student is less than 18

```
select * from students where age<18
```

```
1 •  select * from students where age<18
2
3
```

Result Grid			
ID	name	age	FinalGrade

less than 21

```
select * from students where age<21
```

```
1 •  select * from students where age<21  
2  
3
```

ID	name	age	FinalGrade
103	Bob Johnson	19	C
105	Charlie Davis	20	B
107	Michael Lee	20	C
108	Olivia Moore	19	B

less than 20

```
select * from students where age<20
```

```
1 •  select * from students where age<20  
2  
3
```

ID	name	age	FinalGrade
103	Bob Johnson	19	C
108	Olivia Moore	19	B

12

Retrieve students with grades greater than 'C'.

<C

no student is greater than C

```
select * from students where FinalGrade>"C"
```

```
1 •   select * from students where FinalGrade>"C"
2
3
```

ID	name	age	FinalGrade

```
select * from students where FinalGrade>"B"
```

```
1 •  select * from students where FinalGrade>"B"  
2  
3
```

ID	name	age	FinalGrade
103	Bob Johnson	19	C
107	Michael Lee	20	C
110	Sophia Rodriguez	22	C

```
select * from students where FinalGrade>"A"
```

```
1 •  select * from students where FinalGrade>"A"  
2  
3
```

ID	name	age	FinalGrade
102	Jane Smith	22	B
103	Bob Johnson	19	C
105	Charlie Davis	20	B
107	Michael Lee	20	C
108	Olivia Moore	19	B
110	Sophia Rodriguez	22	C

13

Count the total number of students.



select count(ID) from students

```
1 •  select count(ID) from students
2
3
```

Result Grid	
	count(ID)
▶	10

select count(name) from students

```
1 •   select count(name) from students  
2  
3
```

Result Grid		Filter Rows:
	count(name)	
▶	10	

14

Calculate the average age of students.



```
select avg(age) as average_age from students
```

```
1 •   select avg(age) as average_age from students  
2  
3
```

Result Grid		Filter Rows:	Export:	Wrap
	average_age			
▶	20.7000			

15

Find the sum of ages for students with grades
'A' or 'B'.



**select sum(age) from students where FinalGrade="A" or
FinalGrade="B"**

```
1 •   select sum(age) from students where FinalGrade="A" or FinalGrade="B"
2
3
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	sum(age)			
▶	146			

select sum(age) from students where FinalGrade in ("A","B")

```
1 •   select sum(age) from students where FinalGrade in ("A","B")
2
3
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	sum(age)			
▶	146			

16

Group students by grade and count the number of students in each grade.



select FinalGrade,count(*) from students group by FinalGrade

```
1 • select FinalGrade,count(*) from students group by FinalGrade
```

FinalGrade	count(*)
A	4
B	3
C	3

when we used aggregate function and group by it worked

bt when i want to find only group by it throws error

no output - error

select FinalGrade,count(*) from students group by FinalGrade

Recent Activity			
1 •	select * from students group by FinalGrade		
67	19:17:42	select count(*) from students group by FinalGrade LIMIT 0, 1000	
68	19:18:04	select FinalGrade,count(*) from students group by FinalGrade LIMIT 0, 1000	
69	19:20:52	select * from students group by FinalGrade LIMIT 0, 1000	

the error is -- u want to group by bt wat function u wanna perform on group by - use that aggregate function - so when we given aggregate function it worked

```
select FinalGrade,count(*) from students group by FinalGrade order  
by count(*) asc
```

The screenshot shows a MySQL command-line interface. At the top, there is a command line with the following text:

```
1 • select FinalGrade,count(*) from students group by FinalGrade order by count(*) asc  
2
```

Below the command line is a result grid titled "Result Grid". The grid has two columns: "FinalGrade" and "count(*)". The data is as follows:

FinalGrade	count(*)
B	3
C	3
A	4

17

Group students by age and calculate the average age in each group.



```
select FinalGrade,avg(age) from students group by FinalGrade order  
by avg(age) asc
```

```
1 •  select FinalGrade,avg(age) from students group by FinalGrade order by avg(age) asc  
2
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	FinalGrade	avg(age)		
▶	B	20.3333		
	C	20.3333		
	A	21.2500		

18

Find the grade with the highest number of students.



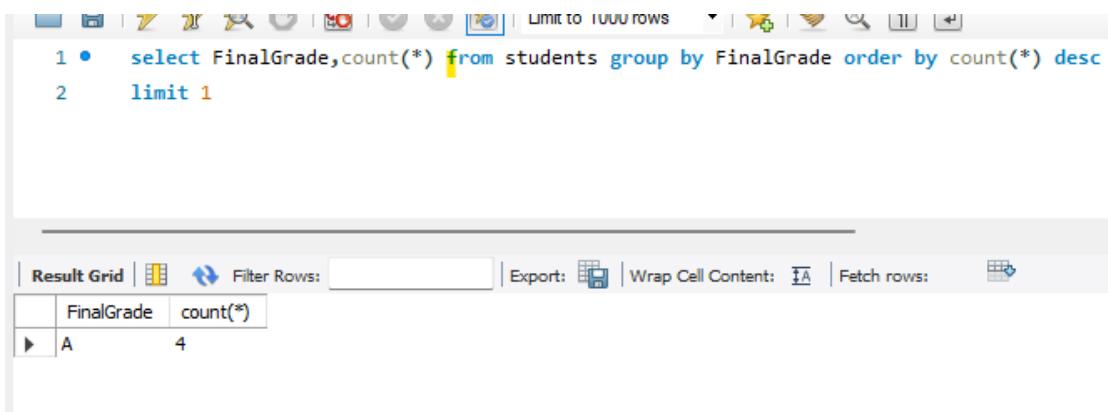
```
1 •  select FinalGrade,count(*) from students group by FinalGrade order by count(*) desc  
2
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	FinalGrade	count(*)		
▶	A	4		
	B	3		
	C	3		

highest

```
select FinalGrade,count(*) from students group by FinalGrade order  
by count(*) desc
```

```
limit 1
```



A screenshot of a MySQL query editor window. The query entered is:

```
1 •  select FinalGrade,count(*) from students group by FinalGrade order by count(*) desc  
2     limit 1
```

The result grid shows one row of data:

	FinalGrade	count(*)
▶	A	4

19

Find grades with an average age greater than 22.



```
1 •   select FinalGrade,avg(age) from students group by FinalGrade
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	FinalGrade	avg(age)		
▶	A	21.2500		
	B	20.3333		
	C	20.3333		

select FinalGrade,avg(age) from students group by FinalGrade having avg(age)>22

```
1 •   select FinalGrade,avg(age) from students group by FinalGrade having avg(age)>22
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	FinalGrade	avg(age)		

select FinalGrade,avg(age) from students group by FinalGrade having avg(age)>21

```
1 • select FinalGrade,avg(age) from students group by FinalGrade having avg(age)>21
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	FinalGrade	avg(age)		
▶	A	21.2500		

select FinalGrade,avg(age) from students group by FinalGrade having avg(age)>20.5

```
1 • select FinalGrade,avg(age) from students group by FinalGrade having avg(age)>20.5
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	FinalGrade	avg(age)		
▶	A	21.2500		

select FinalGrade,avg(age) from students group by FinalGrade having avg(age)>20

```
1 • select FinalGrade,avg(age) from students group by FinalGrade having avg(age)>20
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	FinalGrade	avg(age)		
▶	A	21.2500		
	B	20.3333		
	C	20.3333		

20

Find grades with fewer than 3 students.



```
select FinalGrade, count(*) from students group by FinalGrade having  
count(*)<3
```

```
1 • select FinalGrade, count(*) from students group by FinalGrade having count(*)<3
```

		Result Grid	Filter Rows:	Export:	Wrap Cell Content:
		FinalGrade	count(*)		

```
select FinalGrade, count(*) from students group by FinalGrade having  
count(*)<4
```

```
1 • select FinalGrade,count(*) from students group by FinalGrade having count(*)<4
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	FinalGrade	count(*)		
▶	B	3		
	C	3		

greater than 3

```
select FinalGrade,count(*) from students group by FinalGrade having count(*)>3
```

```
1 • select FinalGrade,count(*) from students group by FinalGrade having count(*)>3
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	FinalGrade	count(*)		
▶	A	4		

21

Find grades with an average age between 20 and 25.



`select FinalGrade,avg(age) from students group by FinalGrade having avg(age) between 20 and 25`

```
1 • select FinalGrade,avg(age) from students group by FinalGrade having avg(age) between 20 and 25
```

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

	FinalGrade	avg(age)
▶	A	21.2500
	B	20.3333
	C	20.3333

`select FinalGrade,avg(age) from students group by FinalGrade having avg(age) between 20 and 21`

```
1 • select FinalGrade,avg(age) from students group by FinalGrade having avg(age) between 20 and 21
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	FinalGrade	avg(age)		
▶	B	20.3333		
	C	20.3333		

22

List all students in ascending order of age.



```
select * from students order by age asc
```

```
1 •  select * from students order by age asc
```

	ID	name	age	FinalGrade
▶	103	Bob Johnson	19	C
	108	Olivia Moore	19	B
	101	John Doe	20	A
	105	Charlie Davis	20	B
	107	Michael Lee	20	C
	104	Alice Brown	21	A
	109	William Turner	21	A
	102	Jane Smith	22	B
	110	Sophia Rodriguez	22	C
	106	Emma Wilson	23	A

select name,age from students order by age asc

```
1 •  select name,age from students order by age asc
```

	name	age
▶	Bob Johnson	19
	Olivia Moore	19
	John Doe	20
	Charlie Davis	20
	Michael Lee	20
	Alice Brown	21
	William Turner	21
	Jane Smith	22
	Sophia Rodriguez	22
	Emma Wilson	23

23

List students with grades 'A' or 'B' in descending order of age.



```
select name,age,FinalGrade from students where FinalGrade="A" or  
FinalGrade="B" order by age desc
```

```
1 • select name,age,FinalGrade from students where FinalGrade="A" or FinalGrade="B" order by age desc
```

Result Grid			
	name	age	FinalGrade
▶	Emma Wilson	23	A
	Jane Smith	22	B
	Alice Brown	21	A
	William Turner	21	A
	John Doe	20	A
	Charlie Davis	20	B
	Olivia Moore	19	B

24

Sort students by grade in alphabetical order.



here i have used name to order by - so name will be in alphabetical order

```
select name,FinalGrade from students order by name asc
```

A screenshot of a database query results window. The query is:

```
1 • select name,FinalGrade from students order by name asc
```

The results grid shows the following data:

	name	FinalGrade
▶	Alice Brown	A
	Bob Johnson	C
	Charlie Davis	B
	Emma Wilson	A
	Jane Smith	B
	John Doe	A
	Michael Lee	C
	Olivia Moore	B
	Sophia Rodriguez	C
	William Turner	A

bt qn. is to sort by finalgrade in alphabetical order

```
select * from students order by FinalGrade asc
```

The screenshot shows a MySQL query results window. At the top, there is a toolbar with icons for back, forward, refresh, and other database operations. Below the toolbar, the query `select * from students order by FinalGrade asc` is displayed. The main area is a "Result Grid" table with the following data:

ID	name	age	FinalGrade
101	John Doe	20	A
104	Alice Brown	21	A
106	Emma Wilson	23	A
109	William Turner	21	A
102	Jane Smith	22	B
105	Charlie Davis	20	B
108	Olivia Moore	19	B
103	Bob Johnson	19	C
107	Michael Lee	20	C
110	Sophia Rodriguez	22	C

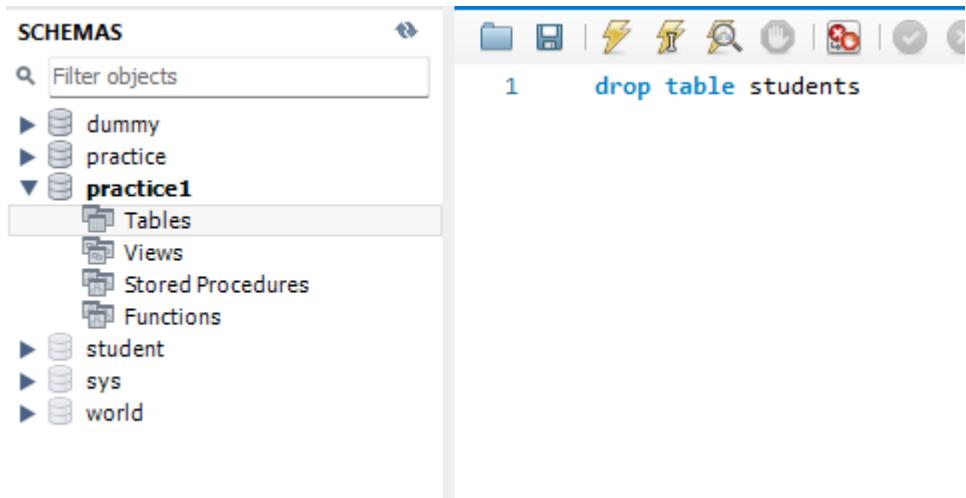
25

Delete the "Students" table.

The screenshot shows a dark-themed interface with a table containing the same data as the previous table. The table has the following structure:

101	'John Doe'	20	'A'
102	'Jane Smith'	22	'B'
103	'Bob Johnson'	19	'C'
104	'Alice Brown'	21	'A'
105	'Charlie Davis'	20	'B'
106	'Emma Wilson'	23	'A'
107	'Michael Lee'	20	'C'
108	'Olivia Moore'	19	'B'
109	'William Turner'	21	'A'
110	'Sophia Rodriguez'	22	'C'

```
drop table students
```



entire table deleted

PRIMARY KEY

we have created 2 table and inserted values

✓	2	12:35:03	use dummy
✓	3	12:35:03	create table books(b_ID int, title varchar(50))
✓	4	12:35:03	create table author(a_ID int, name varchar(50))
✓	5	12:35:19	select * from books LIMIT 0, 1000
✓	6	12:35:41	select * from author LIMIT 0, 1000
✓	7	12:38:09	insert into books (b_ID,title) values (1,"bookone"), (2,"booktwo"), (3,"bookthree")
✓	8	12:38:09	insert into author (a_ID,name) values (1,"sangee"), (2,"sindhu"), (3,"sarasa")
✓	9	12:38:23	select * from books LIMIT 0, 1000
✓	10	12:38:31	select * from author LIMIT 0, 1000

```
3 •  select * from books  
4  
5
```

Result Grid	
b_ID	title
1	bookone
2	booktwo
3	bookthree

now i am adding extra 1 data by giving b_ID as 3 which already exists

here we got same 3 as b_Id

```
3 •  select * from books;  
4  
5 •  insert into books values(3,"bookfour")
```

Result Grid	
b_ID	title
1	bookone
2	booktwo
3	bookthree
3	bookfour

bt for ID we need unique values , it should not get repeated

then we use primary key

now we can delete id=3 - both will be deleted

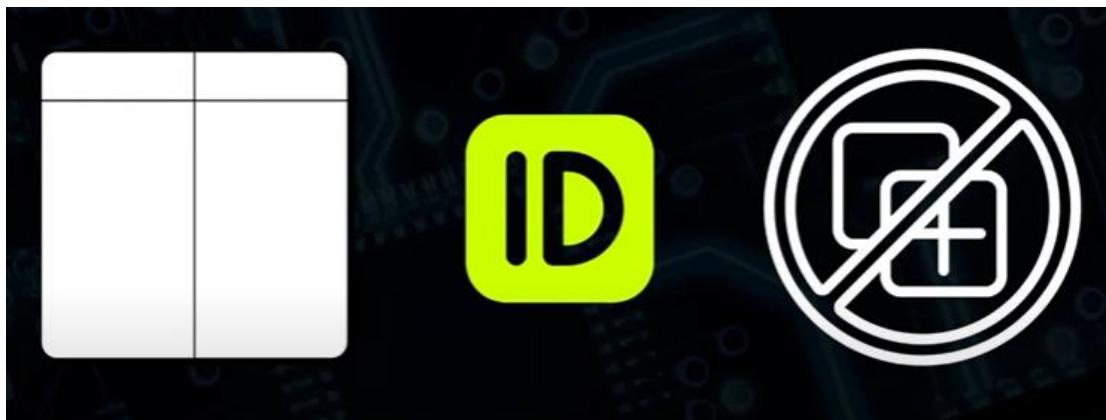
```
1 •  delete from books where b_id=3
```

Result Grid	
b_ID	title
1	bookone
2	booktwo

now we can add primary key

primary key - set of rules

eg: oru column la oru ID once tha irukanum , duplicate data ethum iruka kudathunu rules set pandrom



book ID la tha duplicate id iruka kudathu so rule b_id ku poduvom by using primary key

constraint - means rules in database - what to do, wat not to do

we hv to give name to constraint - anything

we have to decide primary key rule entha column ku podanumnu - there we have to use primary key

alter table books add constraint pk_bid primary key(b_ID);

```
1 • alter table books add constraint pk_bid primary key(b_ID)
2
```

Result Grid		Filter Rows:	Edit:	Export/Import:
b_ID	title			
1	bookone			
2	booktwo			
*	NULL			

now we will try to insert values by using existing b_ID

```
insert into books values(2,"bookthree")
```

✖ 19 12:59:38 insert into books values(2,"bookthree")

Error Code: 1062. Duplicate entry '2' for key 'books.PRIMARY'

when we used primary key rule - and inserted 2 which is already existing it gives error = duplicate entry

..... now we can update b_id column with null values

```
update books set b_id=null
```

✖ 20 13:05:03 update books set b_id=null

Error Code: 1048. Column 'b_ID' cannot be null

if we added primary key, then after we cant add null values to that column

primary key - no duplicates, no null values

for one table only one primary key can be used

here already we have given primary key for b_Id for books table

when we try to give for same table-for other column title- error

```
alter table books add constraint pk_bid primary key(b_title);
```

✖ 21 13:11:01 alter table books add constraint pk_bid primary key(b_title)

Error Code: 1068. Multiple primary key defined

here we 1st created table and then altered by adding constraint as primary key to avoid duplicate

we can also use primary key while creating table itself - better

```
create table books1(  
    id int primary key,  
    title varchar(30))
```



22 13:17:21 create table books1(id int primary key, title varchar(30))

0 row(s) affected

```
insert into books1 (id, title)
```

```
values
```

```
(1,"abc"),  
(2,"def"),  
(1,"ghi")
```



23 13:19:39 insert into books1 (id, title) values (1,"abc"), (2,"def"), (2,"ghi")



24 13:19:53 insert into books1 (id, title) values (1,"abc"), (2,"def"), (1,"ghi")

Error Code: 1062. Duplicate entry '2' for key 'books1.PRIMARY'

Error Code: 1062. Duplicate entry '1' for key 'books1.PRIMARY'

while creating itself if we use primary key by default it wont accept duplicate values-- if we missed while creating then we can alter it by adding constraints rules

TASK

use primary key for authour table - ID

```
select * from author;
```

```
alter table author add constraint author_ID primary key(a_ID);
```

```
insert into author values(3,"anbu");
```

```
insert into author values(4,"anbu")
```

```
select * from author;
```

```
alter table author add constraint author_ID primary key(a_ID);
```

```
insert into author values(3,"anbu");
```

```
insert into author values(4,"anbu")
```

26	13:24:39	select * from author LIMIT 0, 1000	3 row(s) returned
27	13:28:45	alter table author add constraint author_ID primary key(a_ID)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
28	13:29:17	insert into author values(3,"anbu")	Error Code: 1062. Duplicate entry '3' for key 'author.PRIMARY'
29	13:29:30	insert into author values(4,"anbu")	1 row(s) affected

*****Error makes Clever SQL tutorial completed

