→ SQL
→ show databases;
→ use databases;
→ create table tablename(A1D1, A1D2 ... A1Dn);

eg: create table student(idno int, name char(20),
gender char(10), address varchar(100),
marks float);

eg: insert into student values(10, "ram", "2", "3", 90);

Basic structure of SQL 3 clauses ① select ② from ③ where.

SQL Commands:→

① show databases;
② create db db_name;
③ drop db db_name;
④ use db_name;
⑤ show tables;
⑥ drop table tb_name;
⑦ create;
⑧ desc table_name;

You can use it when no programming interface is available to you
① Prepare ② execute ③ deallocate prepare statement

→ An alternative SQL interface to prepared statements is available
→ You can use it from any program when no programming interface is available to you
→ no programming is required.

① Prepare

mysql> PREPARE stmt1 FROM 'SELECT SQRT(POW(3,2)+POW(?,2)) AS hypotenuse';

→ Here Parsing means, checking if submitted query is syntactically & semantically valid
→ the textual protocol that returns the data to the client has serious performance issues. To overcome this issue
→ the prepared stmt or parameterized stmt is used to create the same statement repeatedly with high efficiency. It takes the advantage of client/server binary protocol
→ It parses the query that contains placeholder (?).

mysql > SELECT *FROM student WHERE(studentId = ?;

→ when mysql executes above statement by using different values of student id, it cannot parse the statement fully. mysql will execute the statement faster, especially when it executes the same query multiple times

→ The prepared statements contain placeholder (?)

Advantages of prepared statement:

→ can execute it multiple times. repeatedly.

→ open

Basic workflow of prepared statement:

1) Prepare stage

i) At prepare stage, statement template is sent to the db server.

syntax: PREPARE stmt_name FROM preparable_stmt;

2) EXECUTE stage:

The prepared statement prepares the query & it ready to execute

syntax: EXECUTE stmtname [USING @var_name[, @var_name]...]

3) DEALLOCATE | DROP stage:

{ DEALLOCATE | DROP } PREPARE stmt_name;

Example:

① mysql > PREPARE stmt1 FROM 'SELECT ?+? AS SUM';

② mysql> SET @a = 20;

③ mysql> SET @b = 30;

④ mysql> EXECUTE stmt1 USING @a, @b;

Employee table from sample database

EmpId	name	designation	phone	mob	other

① First we will prepare a statement that returns the employee name & designation specified by employee id:

```
mysql> PREPARE stmt1 FROM
'SELECT Name, Designation FROM employee
 WHERE Emp_id = ?';
```

② Next, we need to declare a variable named id & set its value to '1':

```
mysql > SET @id = 1;
```

③ Execute the prepared statement with the help of EXECUTE stmt

```
mysql > EXECUTE stmt1 USING @id;
```

→ After execution we will get the name & designation of the employee

→ Again we will assign another value to variable id:

```
mysql> SET @id = 3;
```

→ finally we can release the prepared stmt manually.

```
mysql > DEALLOCATE PREPARE stmt1;
```

→ Statement	VS	Prepared statement
① It is used when we want to execute the SQL query only once		① multiple times
② It is used for DDL statements		② It can be used for any SQL query
③ The performance of execution is slow		② The performance of execution is fast

NO SQL Introduction

→ A NOSQL Refers to non SQL or non relational. is a database that provides a mechanism for storage & retrieval of data.

→ NOSQL databases are used in real-time web application & big data & their use are increasing over time.

→ NOSQL databases are non-tabular databases & store data differently than relational tables.

→ It comes in a variety of types based on their data model. The main types are document, key-value, wide-column and graph.

→ NOSQL databases optimized for developer productivity

→ " allows developers to store huge amounts of unstructured data, giving them a lot of flexibility.

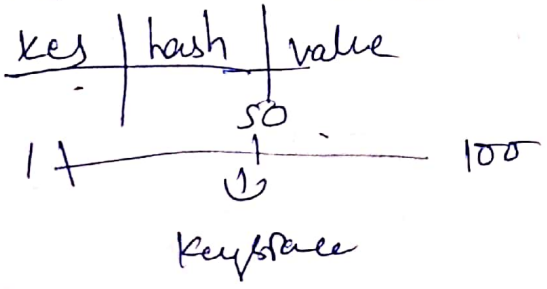
→ SQL scales vertically whereas NOSQL dbs horizontally (Eg: a building).

→ Every db has two field : a unique key & value

Key	Value
1942L	→ {name: —, price: —, desc: —}
1923456	usb microphone }

→ Apple runs nosql database which runs 75,000 + servers

→ (On the backend. we have large data dbs will be split into Partitions). and store the value. Then how will we know which key belongs to a value (specific) Then we will use hashes



→ nosql is schemaless

Diff blw SQL & noSQL

	SQL Databases	noSQL databases
Data storage model	Tables with fixed rows & columns	Document: JSON documents, Key-value: key-value pairs, wide-column: tables with rows, dynamic columns, Graphs: nodes & edges.
Development History	developed in the 1970s with a focus on reducing data duplication	late 2000s with a focus on scaling & allowing for rapid application change
Examples	Oracle, mysql, Microsoft SQL Server, & Postgresql	Document: mongoDB & couchDB, key-value: Redis & dynamoDB, wide-column: cassandra
Primary Purpose	General Purpose	Document: General-purpose, key-value: large amount of data with simple lookup queries, wide-column: large amounts of data with predictable query patterns, Graphs: analysing & traversing relationships blw connected data
Schemas	Rigid	flexible
Scaling	Vertical	horizontal
ACID Transaction	Supported	most do not support. some mongo-DB do.
Joins	Typically required	Typically not required
Data to object mapping	Requires ORM (Object Relational Mapping)	many don't require ORM. But mongodb can directly mail to ds. In many programming langs

Types of NoSQL Databases:

→ These are developed during the Internet era in response to the inability of SQL databases to address the needs of web scale applications that handled huge amounts of data & traffic.

→ These are designed to scale horizontally across many servers.

4 types of noSQL databases:

1) Document databases
2) key-value stores
3) column-oriented databases
4) Graph databases

① Document-databases:

→ It stores data in JSON, BSON or XML docs (not word or google docs)

→ Documents can be stored & retrieved in a form that is much closer to the data objects used in apps., which means less translation is required to use the data in an application.

→ Document databases are popular with developers because they have the flexibility to rework their document structures as needed to suit their application, shaping their data structures.

→ Use cases include ecommerce platforms, trading platforms, & mobile app development across industries

key-value stores

→ simplest way is a key-value store

→ Every data element in the db is stored as a key value pair consisting of an attribute name (or "key") and a value.

→ use cases include shopping carts, user preferences & user profiles.

Column-Oriented databases:

→ relational db store data in rows & reads row by row

→ A column store is organized as a set of columns

→ It means that when you want to run analytics on a small no. of columns you can read those columns directly without consuming memory with the unwanted data.

→ columns are often of the same type & benefit from more efficient compression, making reads even faster.

Use case include analytics.

Graph Databases:

→ It focuses on the relationship b/w data elements. Each element is stored as a node (such as a person in a social media group).

→ The connections b/w elements are called links or relationships

Uses cases include fraud detection, social news & knowledgable graph.

saving the user object in Mongodb (Query Mechanisms for NoSQL database)

→ to store the user object, use save:

```
mongo> db.user.save({
  "_id":1234,
  "name":{
        "first": "Ram",
        "last": "K"
       },
  "topics": ["skating","music"]
});
```

Querying the user object in MongoDB

→ Use find to filter on any attribute or sub-attributes:

```
mongo > db.user.find({
    "_id" : 1234
});
mongo > db.user.find({
    "name.first" : "ram"
});
```

Querying using $query $operators:

```
mongo > db.user.find({
    "$or" : [
        { "name.first" : "ram" },
        { "topics" : {
            "$in" : [ "skating" ]
        }
        }
    ]
});
```

→ find queries can be combined with count(), limit(), skip(), sort() etc. functions.

Ex-1:

Query documents that belong to a specific customer

→ We use find method to query documents from a mongoDB database. If used without any arguments on collections, find method retrieves all documents.

→ If we want to see the document belongs to customer Ram so the name field needs to be specified in the find method.

> db.customer.find ({ name: "Ram" })

{ "_id": ObjectId ("60ocia"), "name": "Ram", "age": 20,
"gender": "male" }.

We can attach pretty method to make the document seem more appealing.

> db.customer.find ({ name: "Ram" }). pretty()

{
"_id": -
"name": -
}

Ex-2:

Query documents that belong to customer older than 40.

> db.customer.find ({ age: { $gt: 40 } }). pretty()

Unit-4

jQuery Plugin

-> Plugin → had extra functionality of anything.

→ Let us take an h/w component (In our laptop we have touchPad ← left click right mouse).

Instead of touchPad, we can use separate mouse.)

→ In order to move cursor or mouse (we need extra drivers).

→ jQuery → IS library contains → collection of methods. (Reduce development time)

→ for example we are having $(selector).Parent();
there is separate code for this ↗ (in the jQuery library)

→ To simplify the process of developer and also reduces the time.

(It is commonly used by most of the people).

→ for example in java, we are having Packages (userdefined Packages), we will create methods, instead of writing them separately, we will write it in a separate location and it can be used, wherever it is required. concept about Packages only for jquery.

→ Any kind of functionality is similar to so many people.

→ for example in our college we are having 1000 students, and each of them have been allotted individual projects. (Now their task is to identify the "getelementname of every name") so that they can write the code.

Prop("tagname") It needs to write by every student, which increases the burden on every student. for this purpose we can write a separate code and call it as a "plugin" and this can be used by all

all the students, without writing the code separately.

→ Plugin is a piece of code which is written in a standard JS file

→ In the previous Traversing DOM concept we have used first(), last() method, In this there will be an inbuilt code in js file.

→ we are already having JS library, but by adding some extra functionality to this JS library is called as JQuery Plugin. (without affecting JS library).

Rules to write JQPlugin:

① .js extension (jquery.name.js)
eg: jquery.gNE.js
getname element

② jv.html
→ In this file we will use .js

② Functions/methods; (end with semicolon)
② Return in the fun of jquery object.
④ Always use this.each() (method)
⑤ To write a method → jquery.fn.display = function() { }
functionname.

→ eg: Here the Requirement is, we have 1000 students & every student. display the every element name which are used by themselves

Now we are going to create a plugin which is used by all students.

→ ① jquery.display.js

② jquery.fn.display = function(){
return this.each(function(){
alert("element name:" + $(this).prop("tagname"));
});
});

To display every elem.

Now coming to html file.

→ ready()
  click()
  { $("a").

→ Installation link ① <script " src = " cdn link " >.

② Link to plugin <script src = " jquery.display.js " >

③ normal script tag.

Eg: <html>
  <head>
  <script type = "text/js" src = " cdn link " >
→ <script type = "text/js" src = "jquery.display.js" >
  <script type = "text/js" >

$(document).ready(function(){
  $("#b1").click(function(){
  $("a").display();
  $("div").display();

});
});

</script> </head>
<body> RGUKT SKLM
 Hi
 CSE <button id = "b1" > click here </button>

① create jquery.display.js
  Plugin file
jquery fn
→ display = function(){
  return this.each(function(){
  alert("Element name:" +
    $(this).prop("tagname"));
});
});
(Note: we can use elementname & text).