Assignment No: - 5

Title: - Write a Stored Procedure and function.

Problem Definition: -

To Design a PL/SQL stored procedure and function for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class

Learning Objectives:-

To write PL/SQL stored procedure and function.

Learning Outcomes:-

The ability write PL/SQL stored procedure and function

Software and Hardware Requirement

- OS-Linux
- Mysql
- 64 bit machine

Theory:

Stored Procedure:

A procedure (often called a stored procedure) is a subroutine like a subprogram in a regular computing language, stored in database.

A procedure has a name, a parameter list, and SQL statement(s). All most all relational database system supports stored procedure.

A Stored procedures are reusable and can be invoked by triggers, other storedprocedures, and applications.

Typically stored procedures help increase the performance of the applications.

Once created, stored procedures are compiled and stored in the database.

Stored procedures help reduce the traffic between application and database server because instead of sending multiple lengthy SQL statements, the application has to send only name and parameters of the stored procedure.

Types of parameters:

Optional. One or more parameters passed into the procedure. When creating aprocedure, there are three types of parameters that can be declared:

1. IN - The parameter can be referenced by the procedure. The value of the parameter can not be overwritten by the procedure.
2. OUT - The parameter can not be referenced by the procedure, but the value of the parameter can be overwritten by the procedure.
3. IN OUT - The parameter can be referenced by the procedure and the value of theparameter can be overwritten by the procedure.

Syntax:

<u>CREATING PROCEDURE</u>

```
CREATE PROCEDURE procedure_name [ (parameter datatype [,parameter datatype]) ]

BEGIN

declaration_section

executable_section

END;
```

<u>EXECUTING PROCEDURE</u>

```
CALL procedure name;
```

DROP PROCEDURE

```
DROP PRODECURE procedure name;
```

<u>Example:</u>

```
     DELIMITER //

   CREATE PROCEDURE GetAllProducts()

BEGIN
SELECT *  FROM products;


END //
 DELIMITER ;
```

Stored Function:

As same as Stored Procedures but Function always returns a result. Function can be called inside an SQL statement just like ordinary SQL functions,or within other stored procedure and functions.

Type of parameters:

When creating a function, all parameters are considered to be IN parameters (not OUT or INOUT parameters) where the parameters can be referenced by the function but can not be overwritten by the function.

A function parameter is the equivalent of the IN procedure parameter, as functions use the *RETURN* keyword to determine what is passed back.

Syntax:

CREATING FUNCTION

```
CREATE FUNCTION function_name [ (parameter datatype [, parameterdatatype]) ]

RETURNS return_datatype

BEGIN
declaration_sectionexecutable_section
END;
```

EXECUTING FUNCTION

```
SELECT function name();
```

DROP FUNCTION

```
DROP FUNCTION function name;
```

Example:

```
     CREATE FUNCTION WEIGHTED_AVERAGE (n1 INT, n2 INT, n3 INT, n4INT)
     RETURNS INT BEGIN
     DECLARE avg INT;

    SET avg = (n1+n2+n3*2+n4*4)/8;
    RETURN avg;

        END
```

Test Cases:-

Test Case no.	Input	Expected Output	Actual Output
TC_01	DELIMITER //		
	CREATE PROCEDURE		
	GetAllProducts()	select all	
	BEGIN	products from	
	SELECT * FROM products;	the products	
	END //	table.	
	DELIMITER ;		
TC_02	CREATE FUNCTION		
	WEIGHTED_AVERAGE (n1 INT, n2		
	INT, n3 INT, n4 INT)		
	RETURNS INT		
	DETERMINISTIC	Return	
	BEGIN	Weighted	
	DECLARE avg INT;	Average	
	SET avg =		
	(n1+n2+n3*2+n4*4)/8;		
	RETURN avg;		
	END		

Conclusion: Thus have successfully studied and implemented stored procedure and function in PL/SQL.

Questions:

1) What is the difference between "procedure" and "function"?
2) What is the use of package?

Assignment No: 02

Title: SQL Queries:

a. Design and Develop SQLDDL statements which demonstrate the use of SQL objects suchas Table, View, Index, Sequence, Synonym, different constraints etc.
b. Write at least 10 SQL queries on the suitable database application using SQL DMLstatements.

Aim: Design and Develop SQL DDL statements which demonstrate the use of SQL objects suchasTable, View, Index, Sequence, Synonym. Design and Develop SQL DML statements

Prerequisites: Basics of database, data base Languages.

Objective: To learn the various DDL commands, view concepts, types of Indexes, sequence,Synonym and its implementation.To learn the various DML commands

Outcome: Develop the ability to handle basic operations on databases.

Theory:

DDL:

Data Definition Language helps you to define the database structure or schema. Let's learn about DDL commands with syntax.

Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.

The SQL CREATE TABLE Statement

The CREATE TABLE statement is used to create a new table in a database.

Syntax

CREATE TABLE *table_name* (

column1

datatype,*column2*

datatype,*column3*

datatype,

```
    ....
);
```

Example

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
   FirstName varchar(255),
   Address varchar(255),
   City varchar(255)
        );
```

View:

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

CREATE VIEW Syntax

```
 CREATE VIEW view_name
AS SELECT column1, column2, ... FROMtable_name
WHERE condition;
```

SQL CREATE VIEW Examples

If you have the Northwind database you can see that it has several views installed by default.

The view "Current Product List" lists all active products (products that are not discontinued) from the "Products" table. The view is created with the following SQL:

```
CREATE VIEW [Current Product List] AS
```

```
SELECT ProductID, ProductName
FROM Products
WHERE Discontinued = No;
```

Then, we can query the view as follows:

```
SELECT * FROM [Current Product List];
```

SQL Updating a View

You can update a view by using the following syntax:

SQL CREATE OR REPLACE VIEW SyntaxCREATE OR REPLACE VIEW view_name AS SELECT column1, column2, ...

```
FROM  table_name
WHERE condition;
```

Now we want to add the "Category" column to the "Current Product List" view. We will update the view with the following SQL:

```
CREATE OR REPLACE VIEW [Current Product List] AS
SELECT ProductID, ProductName, Category
FROM Products
WHERE Discontinued = No;
```

SQL Dropping a View

You can delete a view with the DROP VIEW command.

SQL DROP VIEW Syntax

```
DROP VIEW view_name;
```

<u>**Index:**</u>

SQL CREATE INDEX Statement

The CREATE INDEX statement is used to create indexes in tables.

Indexes are used to retrieve data from the database very fast. The users cannot see the indexes, they are just used to speed up searches/queries.

CREATE INDEX Syntax

Creates an index on a table. Duplicate values are allowed:

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

CREATE UNIQUE INDEX Syntax

Creates a unique index on a table. Duplicate values are not allowed:

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

CREATE INDEX Example

The SQL statement below creates an index named "idx_lastname" on the "LastName" column in the "Persons" table:

CREATE INDEX Example

The SQL statement below creates an index named "idx_lastname" on the "LastName" column in the "Persons" table:

If you want to create an index on a combination of columns, you can list the column names within the parentheses, separated by commas:

```
CREATE INDEX idx_pname
ON Persons (LastName, FirstName);
```

DROP INDEX Statement

The DROP INDEX statement is used to delete an index in a table.

DB2/Oracle:

```
DROP INDEX index_name;
```

MySQL:

```
ALTER TABLE table_name
DROP INDEX index_name;
```

Sequence:

A sequence is a set of integers 1, 2, 3, ... that are generated in order on demand. Sequences are frequently used in databases because many applications require each row in a table to contain a unique value and sequences provide an easy way to generate them. This chapter describes how to use sequences in MySQL.

Using AUTO_INCREMENT column:

The simplest way in MySQL to use Sequences is to define a column as AUTO_INCREMENT and leave rest of the things to MySQL to take care.

Example:

Try out the following example. This will create table and after that it will insert few rows in this table where it is not required to give record ID because it's auto incremented by MySQL.

```
mysql> CREATE TABLE insect
-> (
-> id INT UNSIGNED NOT NULL AUTO_INCREMENT,
-> PRIMARY KEY (id),
-> name VARCHAR(30) NOT NULL, # type of insect
-> date DATE NOT NULL, # date collected
-> origin VARCHAR(30) NOT NULL # where collected
);
Query OK, 0 rows affected (0.02 sec)
```

DML:
Data Manipulation Language (DML) allows you to modify the database instance by inserting, modifying, and deleting its data. It is responsible for performing all types of data modification in a database.

DML Commands

The SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)VALUES (*value1*, *value2*, *value3*, ...);

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

INSERT INTO *table_name*

VALUES (*value1*, *value2*, *value3*, ...);

The SQL UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

UPDATE Syntax

UPDATE *table_name*

SET *column1* = *value1*, *column2* = *value2*, ...WHERE *condition*;

The SQL DELETE Statement

The DELETE statement is used to delete existing records in a table.

DELETE Syntax

DELETE FROM *table_name* WHERE *condition*;

Conclusion: Thus we have implemented the DDL and DML commands

Questions:

1. Explain create view and drop view.
2. Explain concept of Index in Mysql.
3. Explain DML statements with syntax

Group B: Assignment No : 1

Title of Assignment: MongoDB Queries

Assignment Name: -.

Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators)

Theory: -

What is NoSQL?

NoSQL is a non-relational DBMS, that does not require a fixed schema, avoids joins, and is easy to scale. The purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook, Google collect terabytes of user data every single day.

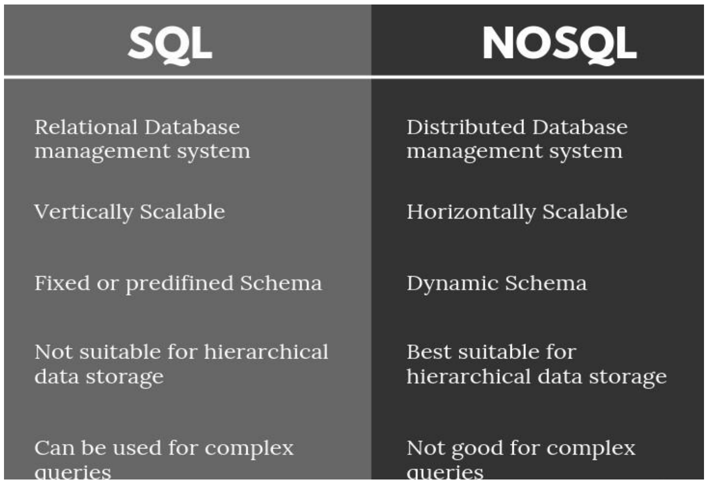
NoSQL database stands for "Not Only SQL" or "Not SQL." Though a better term would be "NoREL", NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

Why NoSQL?

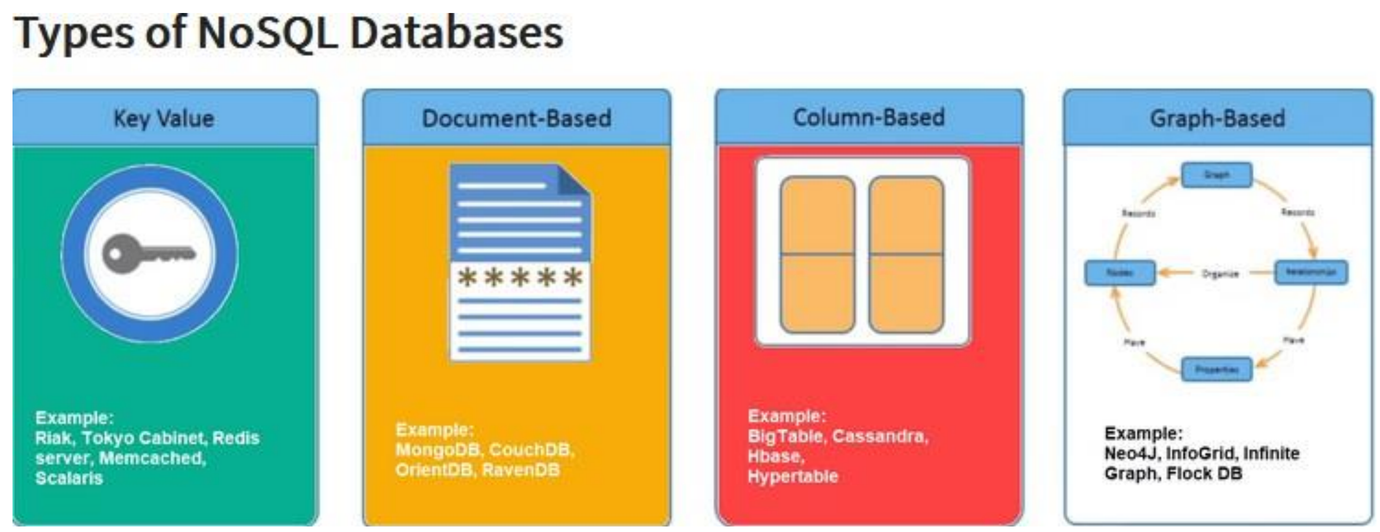
The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow whenyou use RDBMS for massive volumes of data.

To resolve this problem, we could "scale up" our systems by upgrading our existing hardware. This process is expensive.

Difference Between SQL and NoSQL



SQL	NOSQL
Relational Database management system	Distributed Database management system
Vertically Scalable	Horizontally Scalable
Fixed or predifined Schema	Dynamic Schema
Not suitable for hierarchical data storage	Best suitable for hierarchical data storage
Can be used for complex queries	Not good for complex queries


Types of NoSQL Databases
Key Value
Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris
Document-Based
Example:
MongoDB, CouchDB, OrientDB, RavenDB
Column-Based
Example:
BigTable, Cassandra, Hbase, Hypertable
Graph-Based
Graph
Records
Nodes
Organize
Relationships
Have
Properties
Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Document-Oriented:
Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood bythe DB and can be queried.

MongoDB

Scalable High-Performance Open-source, Document-orientated database.

• Built for Speed

• Rich Document based queries for Easy readability.

• Full Index Support for High Performance.

• Replication and Failover for High Availability.

• Auto Sharding for Easy Scalability.

• Map / Reduce for Aggregation**.**

Advantages of MongoDB

- Schema less : Number of fields, content and size of the document can be differ from one document to another.
- No complex joins
- Data is stored as JSON style
- Index on any attribute
- Replication and High availability

Mongo DB Terminologies for RDBMS concepts

RDBMS	MongoDB
Database	Database
Table, View	Collection
Row	Document (JSON, BSON)
Column	Field
Index	Index
Join	Embedded Document
Foreign Key	Reference
Partition	Shard

Data Types of MongoDB

- String : This is most commonly used datatype to store the data. String in mongodb must be UTF-8 valid.
- Integer : This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- Boolean : This type is used to store a boolean (true/ false) value.
- Double : This type is used to store floating point values.
- Min/ Max keys : This type is used to compare a value against the lowest and highest BSON elements.
- Arrays : This type is used to store arrays or list or multiple values into one key.
- Timestamp : ctimestamp. This can be handy for recording when a document has been modified or added.
- Object : This datatype is used for embedded documents.
- Null : This type is used to store a Null value.
- Symbol : This datatype is used identically to a string however, it's generally reserved for languages that use a specific symbol type.
- Date : This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
- Object ID : This datatype is used to store the document’s ID.
- Binary data : This datatype is used to store binay data.
- Code : This datatype is used to store javascript code into document.
- Regular expression : This datatype is used to store regular expression

Basic Database Operations

- use <*database name*>
  switched to database provided with command
- db
  To check currently selected database use the command db
- show dbs
  Displays the list of databases
- db.dropDatabase()
  To Drop the database

- db.createCollection (name)
- Ex:- db.createCollection(Stud)
  - To create collection
- >show collections
  - List out all names of collection in current database
- db.*databasename*.insert
- ({Key : Value})
- Ex:- db.Stud.insert({{Name:"Jiya"})
  - In mongodb you don't need to create collection. MongoDB creates collection automatically, when you insert some document.
- db.collection.drop() Example:- db.Stud.drop()

MongoDB's db.collection.drop() is used to drop a collection from the database.

CRUD Operations:

- Insert
- Find
- Update
- Delete

CRUD Operations – Insert

The insert() Method:- To insert data into MongoDB collection, you need to use MongoDB's insert()or save()method.

Syntax

>db.COLLECTION_NAME.insert(document)

Example

>db.stud.insert({name: “Jiya”, age:15})

_id Field

- If the document does not specify an *_id* field, then MongoDB will add the _id field and assigna unique *ObjectId* for the document before inserting.
- The _id value must be unique within the collection to avoid duplicate key error.

Insert a Document without Specifying an _id Field

- db.stud.insert({ Name : “Reena", Rno: 15 })
- db.stud.find()
  { "_id" : "5063114bd386d8fadbd6b004”, “Name” : “Reena", “Rno”: 15 }

Insert a Document Specifying an _id Field

- db.stud.insert({ _id: 10, Name : “Reena", Rno: 15 })
- db.stud.find()

{ "_id" : 10, “Name” : “Reena", “Rno”: 15 }

Insert Single Documents

db.stud.insert ({Name: “Ankit”, Rno:1, Address: “Pune”})

Insert Multiple Documents

```
db.stud.insert ( [
 { Name: “Ankit”, Rno:1, Address: “Pune”} ,
 { Name: “Sagar”, Rno:2},
 { Name: “Neha”, Rno:3}
 ] )
```

Insert Multicolumn attribute

```
db.stud.insert( {
   Name: “Ritu",
   Address: { City: “Pune", State: “MH” },
  Rno: 6
 })
```

Insert Multivalued attribute

```
db.stud.insert( {
    Name : “Sneha",
    Hobbies: [“Singing”, “Dancing” , “Cricket”] ,
    Rno:8
 })
```

Insert Multivalued with Multicolumn attribute

```
db.stud.insert( {
    Name : “Sneha",
     Awards: [ { Award : “Dancing”, Rank: “1st”, Year: 2008 },
                {Award : “Drawing”, Rank: “3rd”, Year: 2010 } ,
                {Award : “Singing”, Rank: “1st”, Year: 2015 } ],
   Rno: 9 })
```

CRUD Operations – **Find**

The find() Method- To display data from MongoDB collection. Displays all the documents in a non structured way.

Syntax

>db.COLLECTION_NAME.find()

The pretty() Method- To display the results in a formatted way, you can use **pretty()** method.

Syntax

>db. COLLECTION_NAME.find().pretty()

Specify Equality Condition

use the query document { <field>: <value> }

Examples:

- db.stud.find(name: "Jiya" })
- db.stud.find({ _id: 5 })

Comparison Operators

Operator	Description
$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	values that are greater than or equal to a specified value.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$in	Matches any of the values specified in an array.
$nin	Matches none of the values specified in an array.

Find Examples with comparison operators

- db.stud.find({ rno: { $gt:5} }) *Shows all documents whose rno>5*
- db.stud.find({ rno: { $gt: 0, $lt: 5} }) *Shows all documents whose rno greater than 0 and less than 5*

Examples to show only particular columns

- db.stud.find({name: "Jiya"},{Rno:1}) *To show the rollno of student whose name is equal to Jiya (by default _id is also shown)*
- db.stud.find({name: "jiya"},{_id:0,Rno:1}) *show the rollno of student whose name is equal to Jiya (_id is not shown)*

Examples for Sort function

- db.stud.find().sort({ Rno: 1 })
  Sort on age field in Ascending order (1)
- db.stud.find().sort({ Rno: -1 })
  Sort on age field in Ascending order(-1)

Examples of Count functions

- db.stud.find().count()
  Returns no of documents in the collection

Examples of limit and skip

- db.stud.find().limit(2)
  Returns only first 2 documents
- db.stud.find().skip(5)
  Returns all documents except first 5 documents

CRUD Operations – Update

Syntax

```
db.CollectionName.update (
  <query/Condition>,
  <update with $set or $unset>,
  {
   upsert: <boolean>,
   multi: <boolean>,
  } )
```

upsert

- If set to *True*, creates new document if no matches found.

multi

- If set to *True*, updates multiple documents that matches the query criteria

CRUD Operations – Update Examples

1> Set age = 25 where id is 100, First Whole document is replaced where condition is matchedand only one field is remained as age:25

```
db.stud.update(
{ _id: 100 },
{ age: 25})
```

2> Set age = 25 where id is 100, Only the age field of one document is updated where conditionis matched .

```
db.stud.update(
{ _id: 100 },
{ $set:{age: 25}})
```

3> To remove a age column from single document where id=100

```
db.stud.update(
{ _id: 100 },
{ $unset:{age: 1}})
```

CRUD Operations – Remove

- **Remove All Documents**
  - db.inventory.remove({})
- **Remove All Documents that Match a Condition**
  - db.inventory.remove ({ type : "food" })
- **Remove a Single Document that Matches a Condition**
  - db.inventory.remove ({ type : "food" }, 1)

Conclusion:
Here we performed CRUD operations of mongoDB.

Questions:
1. What makes MongoDB the best?
2. What are the key features of mongodb?

Group B: Assignment No : 3

Assignment Name: -. MongoDB Map Reduce

Title of Assignment: Implement Map reduce operation with suitable example using MongoDB.

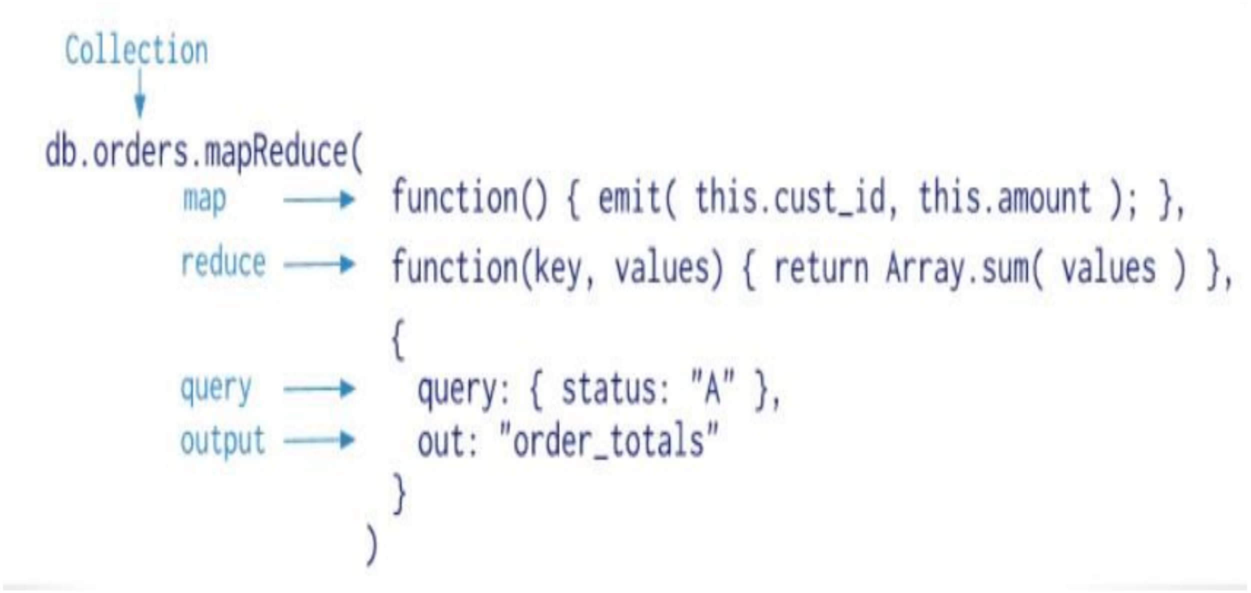
Theory: -

Map-Reduce

• **Map-reduce is a data processing paradigm for condensing**

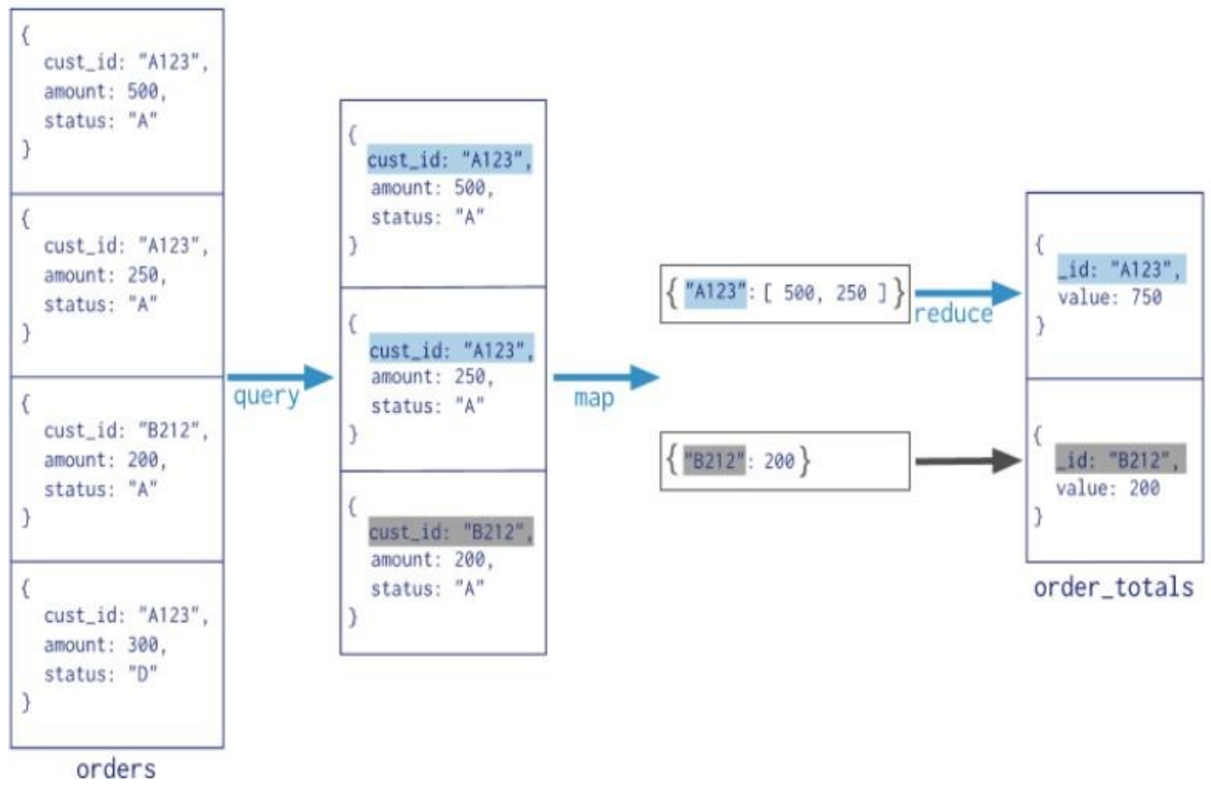
large volumes of data into useful aggregated results. For mapreduce operations, MongoDB provides the mapReduce database command.

• **Consider the following map-reduce operation:**



```
db.orders.mapReduce(
                function() { emit( this.cust_id, this.amount ); },
                function(key, values) { return Array.sum( values ) },
                {
                  query: { status: "A" },
                  out: "order_totals"
                }
              )
```

MapReduce:



Map-Reduce:

In very simple terms, the mapReduce command takes 2 primary inputs, the mapper function andthe reducer function.

A Mapper will start off by reading a collection of data and building a Map with only the required fields we wish to process and group them into one array based on the key.

And then this key value pair is fed into a Reducer, which will process the values.

Map-Reduce Syntax:

```
db.collection.mapReduce(
 function() {emit(key, value);},
function(key,values) {return reduceFunction},
 {
 out: collection,
 query: document,
 sort: document,
 limit: number
 }
)
```

Map-Reduce Syntax Explanation:

The above map-reduce function will query the collection, and then mapthe output documents to the emit key-value pairs. After this, it is reduced based on the keys that have multiple values. Here, we have used the following functions and parameters.

- Map: – It is a JavaScript function. It is used to map a value with a key and produces a key-value pair.
- Reduce: – It is a JavaScript function. It is used to reduce or group together all the documents which have the same key.
- Out: – It is used to specify the location of the map-reduce query output.
- Query: – It is used to specify the optional selection criteria for selecting documents.
- Sort: – It is used to specify the optional sort criteria.
- Limit: – It is used to specify the optional maximum number of documents which are desired to be returned.

Conclusion: Here we performed Mapreduce operation with suitable example using MongoDB.

Question:

1. Define and Explain mapreduce in MongoDB with examples.
2. Why to use Mapreduce in MongoDB