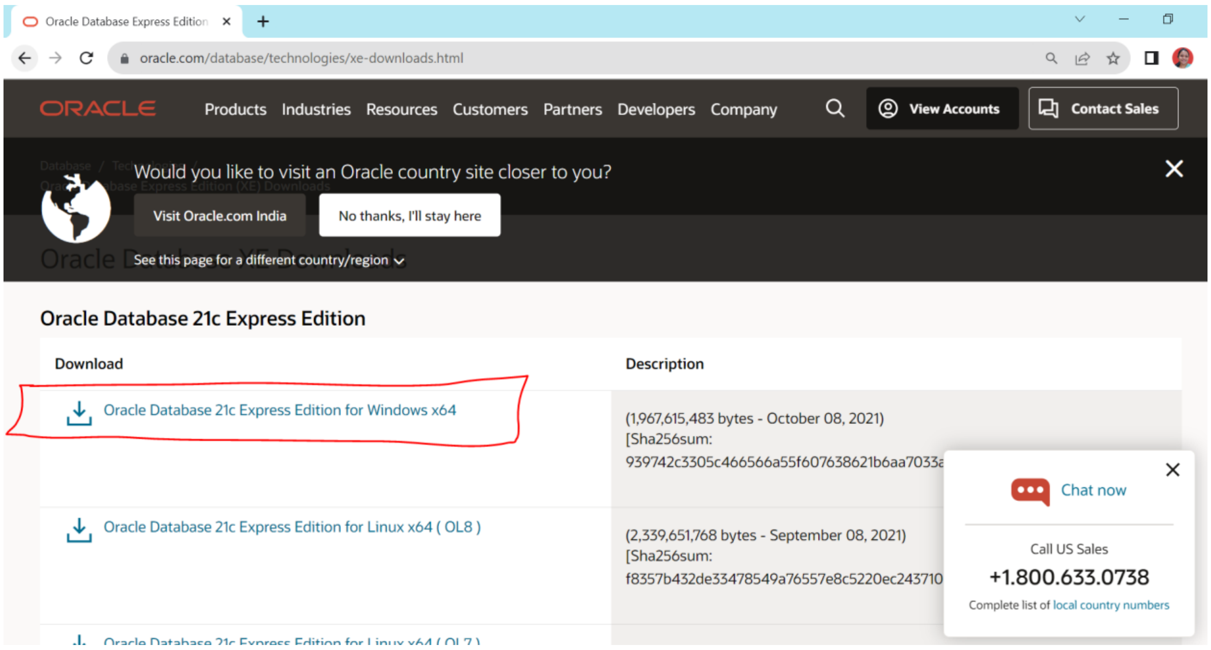
AISSMS
COLLEGE OF ENGINEERING
ज्ञानम् सकलजनहिताय
Accredited by NAAC with **"A+"** Grade

DEPARTMENT OF COMPUTER ENGINEERING
DBMS LAB REPORT

By Mrs. Vaishali Jorwekar

Download Oracle setup from below link:

https://www.oracle.com/database/technologies/xe-downloads.html


Oracle Database Express Edition
oracle.com/database/technologies/xe-downloads.html
ORACLE
Products
Industries
Resources
Customers
Partners
Developers
Company
View Accounts
Contact Sales
Visit Oracle.com India
No thanks, I'll stay here
See this page for a different country/region
Chat now
Call US Sales
+1.800.633.0738
Complete list of local country numbers

Would you like to visit an Oracle country site closer to you?

Oracle Database 21c Express Edition

Download	Description
Oracle Database 21c Express Edition for Windows x64	(1,967,615,483 bytes - October 08, 2021) [Sha256sum: 939742c3305c466566a55f607638621b6aa7033a
Oracle Database 21c Express Edition for Linux x64 (OL8)	(2,339,651,768 bytes - September 08, 2021) [Sha256sum: f8357b432de33478549a76557e8c5220ec243710
Oracle Database 21c Express Edition for Linux x64 (OL7)	

Assignment 2

Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym

```
SQL> create table client_master(client_no int, client_name
varchar(20),address varchar(50),city varchar(10), pincode int, state
varchar(20),bal_due float,primary key(client_no));

Table created.

SQL> insert into client_master
values('001','abhi','nasik','nasik','422004','MH','5000');

1 row created.

SQL> insert into client_master
values('002','piyu','nasik','nasik','422004','MH','10000');

1 row created.

SQL> insert into client_master
values('003','abd','nasik','nasik','422003','MH','5000');

1 row created.

SQL> insert into client_master
values('004','abd','nasik','nasik','422003','MH','5000');

1 row created.

SQL> insert into client_master
values('005','abc','nasik','nasik','422003','MH','5000');

1 row created.

SQL> select * from client_master;


CLIENT_NO   CLIENT_NAME ADDRESS     CITY  PINCODE     STATE BAL_DUE

-------------------------------------------------- ---------- ----------

1           abhi        nasik       nasik 422004      MH    5000

2           piyu        nasik       nasik 422004      MH    10000

3           abd         nasik       nasik 422003      MH    5000

4           abd         nasik       nasik 422003      MH    5000

5           abc         nasik       nasik 422003      MH    5000


SQL> select client_name, client_no from client_master;

CLIENT_NAME          CLIENT_NO

-------------------- ----------

abhi                          1
```

```
piyu                          2

abd                           3

abd                           4

abc                           5


SQL> insert into client_master
values('006','xyz','nasik','nasik','422004','MH','6000');

1 row created.

SQL> select client_name, client_no from client_master;

CLIENT_NAME           CLIENT_NO

-------------------- ----------

abhi                          1

piyu                          2

abd                           3

abd                           4

abc                           5

xyz                           6

6 rows selected.

SQL> create table product_master (product_no int, description varchar (20),
profit_per float, unit_measure varchar (10), quantity int, reorder int,
sell_price float, cost_price float, primary key(product_no));

Table created.

SQL> insert into product_master
values('001','shampoo','1','one','4','2','10','15');

1 row created.

SQL> insert into product_master
values('002','oil','13','one','4','2','11','16');

1 row created.

SQL> alter table client_master add telephone_no int;

Table altered.

SQL> select * from client_master;

CLIENT_NO CLIENT_NAME   ADDRESS     CITY  PINCODE     STATE BAL_DUE
      TELEPHONE_NO

1           abhi        nasik       nasik 422004      MH    5000

2           piyu        nasik       nasik 422004      MH    10000

3           abd         nasik       nasik 422003      MH    5000

4           abd         nasik       nasik 422003      MH    5000
```

```
5           abc         nasik       nasik 422003      MH    5000

6           xyz         nasik       nasik 422004      MH    6000

6 rows selected.

SQL> select * from product_master;


PRODUCT_NO DESCRIPTION PROFIT_PER UNIT_MEASU QUANTITY REORDER SELL_PRICE
COST_PRICE

001         shampoo     1           one          4           2     10
15

002         oil         13          one          4           2     11
16


SQL> CREATE TABLE auto (

  2      roll_no NUMBER GENERATED ALWAYS AS IDENTITY NOT NULL,

  3      name VARCHAR2(20),

  4      PRIMARY KEY (roll_no)

  5  );

Table created.

SQL> select * from auto;

no rows selected

SQL> INSERT INTO auto (name) VALUES ('abc');

1 row created.


SQL> INSERT INTO auto (name) VALUES ('adc');

1 row created.


SQL> CREATE SEQUENCE auto_sequence START WITH 100;

Sequence created.

SQL> select * from auto;

   ROLL_NO NAME

---------- --------------------

         1 abc

         2 adc

SQL> update client_master set client_name='nut' where client_no='4';

1 row updated.
```

```
SQL> select * from client_master;

CLIENT_NO CLIENT_NAME   ADDRESS     CITY  PINCODE     STATE BAL_DUE
      TELEPHONE_NO

1           abhi        nasik       nasik 422004      MH    5000

2           piyu        nasik       nasik 422004      MH    10000

3           abd         nasik       nasik 422003      MH    5000

4           nut         nasik       nasik 422003      MH    5000

5           abc         nasik       nasik 422003      MH    5000

6           xyz         nasik       nasik 422004      MH    6000

6 rows selected.


SQL> create index client_find on client_master(client_name, city);

Index created.

SQL> select * from product_master;


PRODUCT_NO DESCRIPTION PROFIT_PER UNIT_MEASU QUANTITY REORDER SELL_PRICE
COST_PRICE

001         shampoo     1           one         4           2     10
15

002         oil         13          one         4           2     11
16

SQL> desc product_master;

 Name                                      Null?    Type

 ----------------------------------------- -------- -----------------------
-----

 PRODUCT_NO                                NOT NULL NUMBER(38)

 DESCRIPTION                                        VARCHAR2(20)

 PROFIT_PER                                         FLOAT(126)

 UNIT_MEASURE                                       VARCHAR2(10)

 QUANTITY                                           NUMBER(38)

 REORDER                                            NUMBER(38)

 SELL_PRICE                                         FLOAT(126)

 COST_PRICE                                         FLOAT(126)


SQL> alter table client_master rename to c_master;

Table altered.
```

```
SQL> insert into product_master
values('003','nutela','15','three','40','5','110','123');

1 row created.


SQL> create view client as select client_no,client_name from c_master;

View created.


SQL> select * from client;

 CLIENT_NO CLIENT_NAME

---------- --------------------

         1 abhi

         2 piyu

         3 abd

         4 nut

         5 abc

         6 xyz

6 rows selected.
```

Assignment 3

Design at least 10 SQL queries for suitable database application using SQL DML statements: all types of Joins, Sub-Query and View

```
SQL> CREATE TABLE name(roll_no INT NOT NULL, name VARCHAR(30) NOT
NULL,PRIMARY KEY (roll_no));

Table created.


SQL> INSERT INTO name VALUES(37 ,'INDRANEEL');

1 row created.


SQL> INSERT INTO name VALUES (38 ,'SHUBHAM');

1 row created.


SQL> INSERT INTO name VALUES (39 ,'AKSHAY');

1 row created.


SQL> INSERT INTO name VALUES (40 ,'SAKSHI');

1 row created.


SQL> INSERT INTO name VALUES (41 ,'KETAN');

1 row created.


SQL> SELECT * FROM name;

   ROLL_NO NAME

---------- ------------------------------

        37 INDRANEEL

        38 SHUBHAM

        39 AKSHAY

        40 SAKSHI

        41 KETAN


SQL> CREATE TABLE submission(sr_no INT NOT NULL, assgn_id
VARCHAR(30),roll_no INT NOT NULL,PRIMARY KEY(sr_no));

Table created.
```

```
SQL> INSERT INTO submission VALUES (1,’YYYY’,37);

1 row created.


SQL> INSERT INTO submission VALUES (2,’YXYY’,37);

1 row created.


SQL> INSERT INTO submission VALUES (3,’YXYY’,38);

1 row created.


SQL> INSERT INTO submission VALUES (4,'YXYY’,39);

1 row created.


SQL> SELECT * FROM submission;

     SR_NO ASSGN_ID                          ROLL_NO

---------- ------------------------------ ----------

         1 YYYY                                   37

         2 YXYY                                   37

         3 YXYY                                   38

         4 YXYY                                   39


SQL> ALTER TABLE submission ADD FOREIGN KEY (roll_no) REFERENCES
name(roll_no);

Table altered.


SQL> desc submission

 Name                                      Null?    Type

 ----------------------------------------- -------- -----------------------
-----

 SR_NO                                     NOT NULL NUMBER (38)

 ASSGN_ID                                           VARCHAR2(30)

 ROLL_NO                                   NOT NULL NUMBER (38)


SQL> SELECT * FROM name, submission WHERE name.roll_no=submission.roll_no;

ROLL_NO     NAME                     SR_NO       ASSGN_ID    ROLL_NO
```

```
----------- ---------------------- ----------- ----------- ----------
         37 INDRANEEL              1           YYYY        37
         37 INDRANEEL              2           YXYY        37
         38 SHUBHAM                3           YXYY        38
         39 AKSHAY                 4           YXYY        39
```

```
SQL> SELECT * FROM name JOIN submission ON name.roll_no=submission.roll_no;

   ROLL_NO  NAME                    SR_NO       ASSGN_ID    ROLL_NO

----------- ----------------------- ----------- ----------- ----------

         37 INDRANEEL               1           YYYY        37

         37 INDRANEEL               2           YXYY        37

         38 SHUBHAM                 3           YXYY        38

         39 AKSHAY                  4           YXYY        39


SQL> SELECT name.roll_no, name, assgn_id FROM name INNER JOIN submission ON
name.roll_no = submission.roll_no;


   ROLL_NO NAME                           ASSGN_ID

---------- ------------------------------ ------------------------------

        37 INDRANEEL                      YYYY

        37 INDRANEEL                      YXYY

        38 SHUBHAM                        YXYY

        39 AKSHAY                         YXYY


SQL> SELECT * FROM name LEFT JOIN submission ON
name.roll_no=submission.roll_no;


   ROLL_NO NAME                                SR_NO

---------- ------------------------------ ----------

ASSGN_ID                          ROLL_NO

------------------------------ ----------

        37 INDRANEEL                               1

YYYY                                   37


        37 INDRANEEL                               2

YXYY                                   37


        38 SHUBHAM                                 3

YXYY                                   38
```

```
   ROLL_NO NAME                                SR_NO
---------- ------------------------------ ----------
ASSGN_ID                          ROLL_NO
------------------------------ ----------
        39 AKSHAY                                  4
YXYY                                   39


        40 SAKSHI



        41 KETAN




6 rows selected.



SQL> SELECT name.roll_no, submission.assgn_id
  2  FROM name
  3  LEFT JOIN submission ON name.roll_no = submission.roll_no;

   ROLL_NO ASSGN_ID
---------- ------------------------------
        37 YYYY
        37 YXYY
        38 YXYY
        39 YXYY
        40
        41

6 rows selected.

SQL>
SQL> SELECT * FROM name RIGHT JOIN submission ON
name.roll_no=submission.roll_no;
```

```
   ROLL_NO NAME                                SR_NO

---------- ------------------------------ ----------

ASSGN_ID                          ROLL_NO

------------------------------ ----------

        37 INDRANEEL                               1

YYYY                                   37


        37 INDRANEEL                               2

YXYY                                   37


        38 SHUBHAM                                 3

YXYY                                   38



   ROLL_NO NAME                                SR_NO

---------- ------------------------------ ----------

ASSGN_ID                          ROLL_NO

------------------------------ ----------

        39 AKSHAY                                  4

YXYY                                   39


SQL> CREATE TABLE a3_info(roll_no INT NOT NULL, name VARCHAR(30),cs_lang
VARCHAR(30),PRIMARY KEY (roll_no));

Table created.


SQL> INSERT INTO a3_info VALUES(37,'INDRANEEL','SQL');

1 row created.


SQL> INSERT INTO a3_info VALUES(40,'SAKSHI','C++');

1 row created.


SQL> INSERT INTO a3_info VALUES(38,'SHUBHAM','PYTHON');

1 row created.
```

```
SQL> INSERT INTO a3_info VALUES(39,'AKSHAY','JAVA');
1 row created.

SQL> INSERT INTO a3_info VALUES(41,'KETAN','REACT');
1 row created.




SQL> SELECT * FROM a3_info;
   ROLL_NO NAME                           CS_LANG
---------- ------------------------------ ------------------------------
        37 INDRANEEL                      SQL
        40 SAKSHI                         C++
        38 SHUBHAM                        PYTHON
        39 AKSHAY                         JAVA
        41 KETAN                          REACT

SQL> CREATE VIEW temp AS SELECT roll_no , cs_lang FROM a3_info;
View created.

SQL> SELECT * FROM temp;

   ROLL_NO CS_LANG
---------- ------------------------------
        37 SQL
        40 C++
        38 PYTHON
        39 JAVA
        41 REACT
SQL> UPDATE temp SET cs_lang='ANGULAR' WHERE roll_no=41;
1 row updated.
```

PLSQL-Assignment 4

Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory. Write a PL/SQL block of code for the following requirements: - Schema: 1. Borrower (Rollin, Name, DateofIssue, NameofBook, Status) 2. Fine (Roll_no, Date, Amt) • Accept roll_no & name of book from user. • Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day. • If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day. • After submitting the book, status will change from I to R. • If condition of fine is true, then details will be stored into fine table. Frame the problem statement for writing PL/SQL block inline with above statement.

```
SQL> CREATE TABLE borrower(roll_no NUMBER , name VARCHAR2(25), dateofissue
DATE, name_of_book VARCHAR2(25), status VARCHAR2(20));

Table created.


SQL> INSERT INTO borrower VALUES(45,'ASHUTOSH',TO_DATE('01-08-2022','DD-MM-
YYYY'),'HARRY POTTER','PENDING');

1 row created.


SQL> INSERT INTO borrower VALUES(46,'ARYAN',TO_DATE('15-08-2022','DD-MM-
YYYY'),'DARK MATTER','PENDING');

1 row created.


SQL> INSERT INTO borrower VALUES(47,'ROHAN',TO_DATE('24-08-2022','DD-MM-
YYYY'),'SILENT HILL','PENDING');

1 row created.


SQL> INSERT INTO borrower VALUES(48,'SANKET',TO_DATE('26-08-2022','DD-MM-
YYYY'),'GOD OF WAR','PENDING');

1 row created.


SQL> INSERT INTO borrower VALUES(49,'SARTHAK',TO_DATE('09-09-2022','DD-MM-
YYYY'),'SPIDER-MAN','PENDING');

1 row created.


SQL> CREATE TABLE fine (

  2    roll_no NUMBER,

  3    return_date DATE,

  4    fine NUMBER
  5  );

Table created.


SQL> DECLARE

  2   i_roll_no NUMBER;

  3   name_of_book VARCHAR2(25);

  4   no_of_days NUMBER;

  5     return_date DATE := TO_DATE(SYSDATE,'DD-MM-YYYY');

  6   temp NUMBER;

  7   doi DATE;

  8   fine NUMBER;

  9   BEGIN

 10    i_roll_no := &i_roll_no;

 11    name_of_book := '&nameofbook';

 12   dbms_output.put_line(return_date);

 13   SELECT to_date(borrower.dateofissue,'DD-MM-YYYY') INTO doi FROM
borrower WHERE
      borrower.roll_no = i_roll_no AND borrower.name_of_book =
name_of_book;

 14   no_of_days := return_date-doi;

 15   dbms_output.put_line(no_of_days);

 16    IF (no_of_days >15 AND no_of_days <=30) THEN

 17    fine := 5*no_of_days;

 18    ELSIF (no_of_days>30 ) THEN

 19    temp := no_of_days-30;

 20    fine := 150 + temp*50;

 21    END IF;

 22    dbms_output.put_line(fine);

 23    INSERT INTO fine VALUES(i_roll_no,return_date,fine);

 24     UPDATE borrower SET status = 'RETURNED' WHERE borrower.roll_no =
i_roll_no;

 25     END;

 26     /

Enter value for i_roll_no: 46

Enter value for nameofbook: DARK MATTER

02-OCT-23
```

```
413

19300

PL/SQL procedure successfully completed.


SQL> select * from BORROWER;


   ROLL_NO NAME                      DATEOFISS NAME_OF_BOOK

---------- ------------------------- --------- -------------------------

STATUS

--------------------

        45 ASHUTOSH                  01-AUG-22 HARRY POTTER

PENDING


        46 ARYAN                     15-AUG-22 DARK MATTER

RETURNED


        47 ROHAN                     24-AUG-22 SILENT HILL

PENDING




   ROLL_NO NAME                      DATEOFISS NAME_OF_BOOK

---------- ------------------------- --------- -------------------------

STATUS

--------------------

        48 SANKET                    26-AUG-22 GOD OF WAR

PENDING


        49 SARTHAK                   09-SEP-22 SPIDER-MAN

PENDING

SQL> select * from FINE;


   ROLL_NO RETURN_DA       FINE

---------- --------- ----------

        46 02-OCT-23      19300
```

PL/SQL-Assignment 5

Stored Procedure and Stored Function. Write a Stored Procedure namely proc_Grade 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. Write a PL/SQL block for using procedure created with above requirement. Stud_Marks(name, total_marks) Result(Roll,Name, Class) Frame the separate problem statement for writing PL/SQL Stored Procedure and function, inline with above statement. The problem statement should clearly state the requirements.

```
SQL> CREATE TABLE stud_marks(name VARCHAR2(25),total_marks NUMBER);


Table created.


SQL> CREATE TABLE result(roll_number NUMBER , name VARCHAR2(25), class
VARCHAR2(30));


Table created.


SQL> CREATE OR REPLACE FUNCTION func_1(r IN NUMBER, n IN VARCHAR2,m IN
NUMBER) RETURN VARCHAR2 AS

  2     BEGIN

  3     procedure_1(r,n,m);

  4     return 'SUCCESSFULL';

  5     END;

  6     /


Function created.


SQL> CREATE OR REPLACE PROCEDURE procedure_1 ( roll_no IN NUMBER, name IN
VARCHAR2 ,marks IN NUMBER) AS

  2     BEGIN

  3      IF (marks<=1500 and marks>=990) THEN

  4    DBMS_OUTPUT.PUT_LINE ('DISTINCTION');

  5      INSERT INTO result VALUES (roll_no,name,'DISTINCTION');

  6     ELSIF (marks<=989 and marks>=900) THEN

  7      DBMS_OUTPUT.PUT_LINE ('FIRST CLASS');
  8      INSERT INTO result VALUES (roll_no,name,'FIRST CLASS');
  9      ELSIF (marks<=899 and marKs>825) THEN
 10     DBMS_OUTPUT.PUT_LINE('HIGHER SECOND CLASS');
 11     INSERT INTO result VALUES (roll_no,name,'HIGHER SECOND CLASS');
 12     ELSE
 13    DBMS_OUTPUT.PUT_LINE ('FAIL');
 14     INSERT INTO result VALUES (roll_no,name,'FAIL');
 15
 16        END IF;
 17         INSERT INTO stud_marks VALUES (name,marks);
 18         END procedure_1;
 19       /

Procedure created.

SQL> DECLARE
  2    name_1 VARCHAR2(25);
  3     roll_no_1 NUMBER;
  4     marks_1 NUMBER;
  5     class VARCHAR2(25);
  6     BEGIN
  7     roll_no_1:=&roll_no_1;
  8     name_1:='&name_1';
  9     marks_1:=&marks_1;
 10    class := func_1(roll_no_1,name_1,marks_1);
 11     dbms_output.put_line(class);
 12    END;
 13     /
Enter value for roll_no_1: 2
Enter value for name_1: Ram
Enter value for marks_1: 1500
DISTINCTION
SUCCESSFULL
PL/SQL procedure successfully completed.
```

PLSQL-Assignment 6

Write a PL/SQL block of code using parameterized cursor that will merge the data available in newly created table N_RollCall with the data available in the O_RollCall. If the data in the first table already exists in the second table then that data should be skipped.

```
SQL> create table new_roll(roll int,name varchar(10));

Table created.


SQL> create table old_roll(roll int,name varchar(10));

Table created.


SQL> insert into new_roll values(2,'b');

1 row created.


SQL> insert into old_roll values(4,'d');

1 row created.


SQL>  insert into old_roll values(3,'bcd');

1 row created.


SQL> insert into old_roll values(1,'bc');

1 row created.


SQL> insert into old_roll values(5,'bch');

1 row created.


SQL> insert into new_roll values(5,'bch');

1 row created.


SQL> insert into new_roll values(1,'bc');

1 row created.


SQL> select * from new_roll;

      ROLL NAME
```

```
---------- ----------
         2 b
         5 bch
         1 bc

SQL> select * from old_roll;
      ROLL NAME
---------- ----------
         4 d
         3 bcd
         1 bc
         5 bch

SQL> CREATE OR REPLACE PROCEDURE roll1_list AS
  a INT;
  a1 VARCHAR2(10);
  b INT;
  b1 VARCHAR2(10);
  CURSOR c1 IS SELECT roll, name FROM old_roll;
  CURSOR c2 IS SELECT roll, name FROM new_roll;
 BEGIN
  OPEN c1;
  OPEN c2;

  LOOP
    FETCH c1 INTO a, a1;
    EXIT WHEN c1%NOTFOUND;

    -- Check if a record with the same 'roll' exists in new_roll
    BEGIN
      SELECT roll INTO b FROM new_roll WHERE roll = a;
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        -- If no matching record found, insert into new_roll
        INSERT INTO new_roll (roll, name) VALUES (a, a1);
    END;
  END LOOP;

  CLOSE c1;
  CLOSE c2;

  -- Commit the transaction to save changes permanently
  COMMIT;
END;
/
Procedure created.

SQL> call roll1_list();
Call completed.

SQL> select * from new_roll;
      ROLL NAME
---------- ----------
         2 b
         5 bch
         1 bc
         4 d
         3 bcd
```

PLSQL-Assignment 7

Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers). Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table. Frame the problem statement for writing Database Triggers of all types, in-line with above statement. The problem statement should clearly state the requirements.

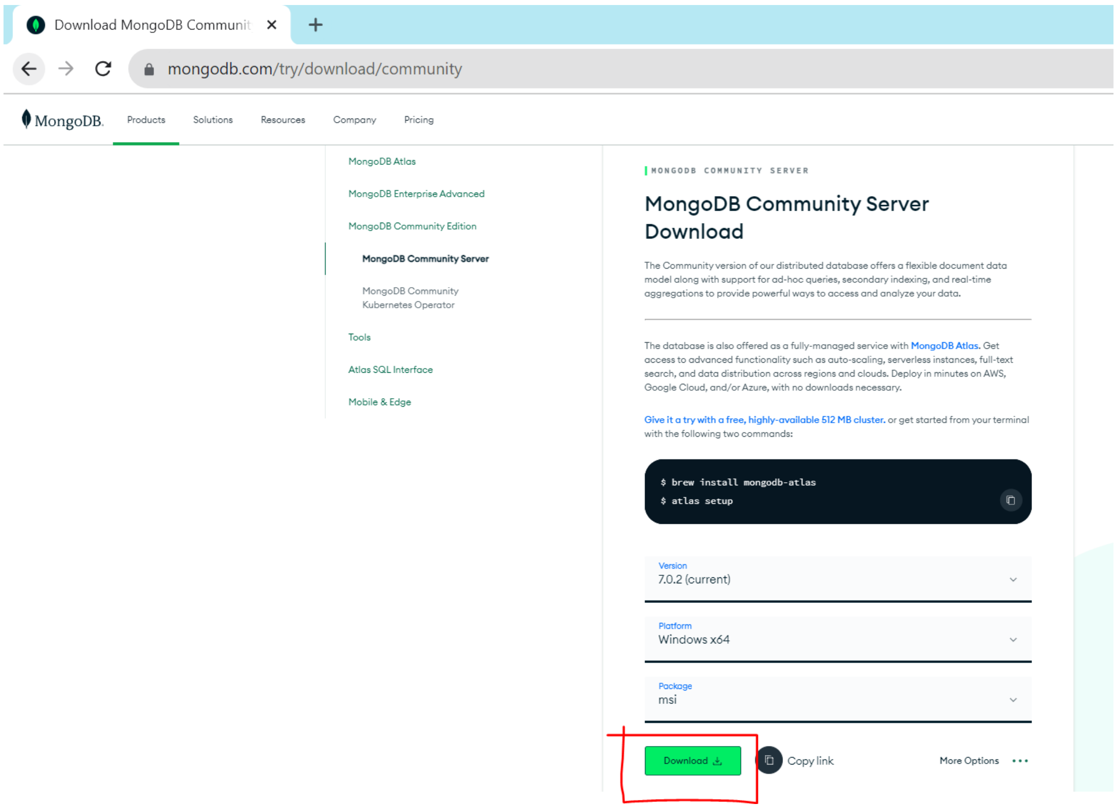
```
SQL> CREATE TABLE lib_tab2(book_name VARCHAR2(25),status VARCHAR2(15));

Table created.


SQL> CREATE TABLE library_audit2(date_modified DATE, book_name
VARCHAR2(25),old_status VARCHAR(15),new_status VARCHAR2(15),action
VARCHAR2(25));

Table created.


SQL> INSERT INTO lib_tab2 VALUES('DARK MATTER','AVAILABLE');

1 row created.


SQL> INSERT INTO lib_tab2 VALUES('SILENT HILL','UNAVAILABLE');

1 row created.


SQL> INSERT INTO lib_tab2 VALUES('GOD OF WAR','AVAILABLE');

1 row created.


SQL> INSERT INTO lib_tab2 VALUES('SPIDER-MAN','UNAVAILABLE');

1 row created.


SQL> INSERT INTO lib_tab2 VALUES('UNCHARTED','AVAILABLE');

1 row created.


SQL> CREATE OR REPLACE TRIGGER trigger_3

  2  AFTER UPDATE OR DELETE OR INSERT ON lib_tab FOR EACH ROW

  3  ENABLE

  4  BEGIN
  5  IF UPDATING THEN

  6  dbms_output.put_line(:OLD.status);

  7  INSERT INTO library_audit2 VALUES
(SYSDATE,:OLD.book_name,:OLD.status,:NEW.status,'UPDATE');

  8                 ELSIF INSERTING THEN

  9                          dbms_output.put_line(:NEW.status);

 10                          INSERT INTO library_audit2 VALUES

(SYSDATE,:NEW.book_name,:OLD.status,:NEW.status,'INSERT');

 11  ELSE

 12  dbms_output.put_line(:OLD.book_name||'deleting');

 13  INSERT INTO library_audit2
VALUES(SYSDATE,:OLD.book_name,:OLD.status,:NEW.status,'DELETE');

 14  END IF;

 15  END;

 16  /

Trigger created.


SQL> DELETE FROM lib_tab2 WHERE book_name = 'SILENT HILL';

1 row deleted.


SQL> UPDATE lib_tab2 SET status = 'UNAVAILABLE' WHERE book_name =
'UNCHARTED';

1 row updated.


SQL> UPDATE lib_tab2 SET status = 'PRE-ORDER' WHERE book_name = 'GOD OF
WAR';

1 row updated.


SQL> Select * from library_audit2;

no rows selected


SQL> Select * from lib_tab2;

BOOK_NAME                 STATUS

------------------------- ---------------

DARK MATTER               AVAILABLE

GOD OF WAR                PRE-ORDER
```

SPIDER-MAN	UNAVAILABLE
UNCHARTED	UNAVAILABLE

MongoDB Practical

Download MongoDB Community Server:

https://www.mongodb.com/try/download/community


Download MongoDB Communit
mongodb.com/try/download/community
MongoDB.
Products
Solutions
Resources
Company
Pricing
MongoDB Atlas
MongoDB Enterprise Advanced
MongoDB Community Edition
MongoDB Community Server
MongoDB Community Kubernetes Operator
Tools
Atlas SQL Interface
Mobile & Edge

MONGODB COMMUNITY SERVER

MongoDB Community Server Download

The Community version of our distributed database offers a flexible document data model along with support for ad-hoc queries, secondary indexing, and real-time aggregations to provide powerful ways to access and analyze your data.

The database is also offered as a fully-managed service with MongoDB Atlas. Get access to advanced functionality such as auto-scaling, serverless instances, full-text search, and data distribution across regions and clouds. Deploy in minutes on AWS, Google Cloud, and/or Azure, with no downloads necessary.

Give it a try with a free, highly-available 512 MB cluster, or get started from your terminal with the following two commands:

```
$ brew install mongodb-atlas
$ atlas setup
```

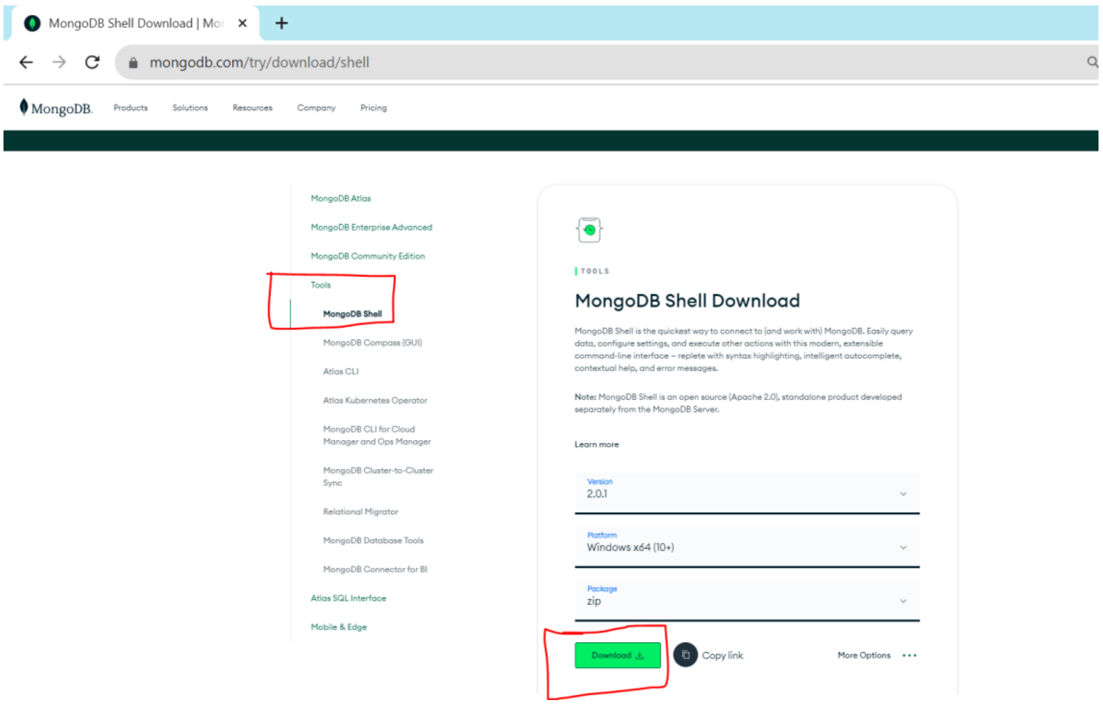
Version: 7.0.2 (current)

Platform: Windows x64

Package: msi

Download MongoDB Shell Download

https://www.mongodb.com/try/download/shell


MongoDB Shell Download | Mo
mongodb.com/try/download/shell
MongoDB.
Products
Solutions
Resources
Company
Pricing
MongoDB Atlas
MongoDB Enterprise Advanced
MongoDB Community Edition
Tools
MongoDB Shell
MongoDB Compass (GUI)
Atlas CLI
Atlas Kubernetes Operator
MongoDB CLI for Cloud Manager and Ops Manager
MongoDB Cluster-to-Cluster Sync
Relational Migrator
MongoDB Database Tools
MongoDB Connector for BI
Atlas SQL Interface
Mobile & Edge

TOOLS

MongoDB Shell Download

MongoDB Shell is the quickest way to connect to (and work with) MongoDB. Easily query data, configure settings, and execute other actions with this modern, extensible command-line interface – replete with syntax highlighting, intelligent autocomplete, contextual help, and error messages.

Note: MongoDB Shell is an open source (Apache 2.0), standalone product developed separately from the MongoDB Server.

Learn more

Version: 2.0.1

Platform: Windows x64 (10+)

Package: zip

MangoDB -Assignment 1

Design and Develop MangoDB queries using CRUD operation (Use CRUD operation Save Method and Logical Operator.

In earlier versions of MongoDB, there used to be a **save()** function in the MongoDB shell that could be used to save or update documents in a collection. However, this function is now deprecated in recent versions of MongoDB (starting with version 3.2) and has been removed in MongoDB 4.0 and later versions. Instead, it's recommended to use the more specific insert, update, and replace operations to work with documents in collections.

```
test> use ass10;

switched to db ass10

ass10>


ass10> db.createCollection("Library");

{ ok: 1 }


ass10> db.library.insert({"bid":1,"name":"C++"});

{

  acknowledged: true,

  insertedIds: { '0': ObjectId("65191fe647ff05b0c6ab5250") }

}


ass10> db.library.insert({"bid":2,"name":"java"});

{

  acknowledged: true,

  insertedIds: { '0': ObjectId("6519201947ff05b0c6ab5251") }

}


ass10> db.library.insert({"bid":3,"name":"Python"});

{

  acknowledged: true,

  insertedIds: { '0': ObjectId("6519204047ff05b0c6ab5252") }

}


ass10> db.library.find()
```

```
[
  { _id: ObjectId("65191fc147ff05b0c6ab524f") },
  { _id: ObjectId("65191fe647ff05b0c6ab5250"), bid: 1, name: 'C++' },
  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },
  { _id: ObjectId("6519204047ff05b0c6ab5252"), bid: 3, name: 'Python' }
]

ass10> db.library.find().pretty();
[
  { _id: ObjectId("65191fc147ff05b0c6ab524f") },
  { _id: ObjectId("65191fe647ff05b0c6ab5250"), bid: 1, name: 'C++' },
  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },
  { _id: ObjectId("6519204047ff05b0c6ab5252"), bid: 3, name: 'Python' }
]

ass10> db.library.update({"name":"Python"},{$set:{"name":"Python3.7"}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne,
updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

ass10> db.library.find()
[
  { _id: ObjectId("65191fc147ff05b0c6ab524f") },
  { _id: ObjectId("65191fe647ff05b0c6ab5250"), bid: 1, name: 'C++' },
  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },
  {
    _id: ObjectId("6519204047ff05b0c6ab5252"),
    bid: 3,
    name: 'Python3.7'
  }
]

ass10> db.library.remove({"bid":1});
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne,
deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }

ass10> db.library.find()
[
  { _id: ObjectId("65191fc147ff05b0c6ab524f") },
  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },
  {
    _id: ObjectId("6519204047ff05b0c6ab5252"),
    bid: 3,
    name: 'Python3.7'
  }
]

ass10> db.library.find({"name":"java"})
[ { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' } ]
ass10> db.library.insert({"bid":4,"name":"java","desc":"fake book"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6519224647ff05b0c6ab5253") }
}

ass10> db.library.find()
[
  { _id: ObjectId("65191fc147ff05b0c6ab524f") },
  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },
  {
    _id: ObjectId("6519204047ff05b0c6ab5252"),
    bid: 3,
    name: 'Python3.7'
  },
  {
    _id: ObjectId("6519224647ff05b0c6ab5253"),
    bid: 4,
    name: 'java',
    desc: 'fake book'
  }
]
ass10> db.library.find({"name":"java"});
[
  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },
  {
    _id: ObjectId("6519224647ff05b0c6ab5253"),
    bid: 4,
    name: 'java',
    desc: 'fake book'
  }
]

ass10> db.library.find({$and:[{"name":"java"},{"desc":"fake book"}]})
[
  {
    _id: ObjectId("6519224647ff05b0c6ab5253"),
    bid: 4,
    name: 'java',
    desc: 'fake book'
  }
]

ass10> db.library.find({$or:[{"name":"java"},{"desc":"fake book"}]})
[
  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },
  {
    _id: ObjectId("6519224647ff05b0c6ab5253"),
    bid: 4,

    name: 'java',

    desc: 'fake book'

  }

]


ass10> db.library.find({$or:[{"name":"java"},{"name":"Python3.7"}]})

[

  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },

  {

    _id: ObjectId("6519204047ff05b0c6ab5252"),

    bid: 3,

    name: 'Python3.7'

  },

  {

    _id: ObjectId("6519224647ff05b0c6ab5253"),

    bid: 4,

    name: 'java',

    desc: 'fake book'

  }

]


ass10> db.library.insert({"bid":4,"name":"my story","cost":500});

DeprecationWarning: Collection.insert() is deprecated. Use insertOne,
insertMany, or bulkWrite.

{

  acknowledged: true,

  insertedIds: { '0': ObjectId("65192b1e1890f8d9ebe31230") }

}

ass10> db.library.find();

[

  { _id: ObjectId("65191fc147ff05b0c6ab524f") },

  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },

  {

    _id: ObjectId("6519204047ff05b0c6ab5252"),
    bid: 3,

    name: 'Python3.7'

  },

  {

    _id: ObjectId("6519224647ff05b0c6ab5253"),

    bid: 4,

    name: 'java',

    desc: 'fake book'

  },

  {

    _id: ObjectId("65192b1e1890f8d9ebe31230"),

    bid: 4,

    name: 'my story',

    cost: 500

  }

]


ass10> db.library.insert({"bid":4,"name":"my story","cost":800});

{

  acknowledged: true,

  insertedIds: { '0': ObjectId("65192b3c1890f8d9ebe31231") }

}

ass10> db.library.insert({"bid":4,"name":"my story2.0","cost":800});

{

  acknowledged: true,

  insertedIds: { '0': ObjectId("65192b4a1890f8d9ebe31232") }

}


ass10> db.library.insert({"bid":4,"name":"my story beta
version","cost":800});

{

  acknowledged: true,

  insertedIds: { '0': ObjectId("65192b551890f8d9ebe31233") }

}
```

```
ass10> db.library.find();
[
  { _id: ObjectId("65191fc147ff05b0c6ab524f") },
  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },
  {
    _id: ObjectId("6519204047ff05b0c6ab5252"),
    bid: 3,
    name: 'Python3.7'
  },
  {
    _id: ObjectId("6519224647ff05b0c6ab5253"),
    bid: 4,
    name: 'java',
    desc: 'fake book'
  },
  {
    _id: ObjectId("65192b1e1890f8d9ebe31230"),
    bid: 4,
    name: 'my story',
    cost: 500
  },
  {
    _id: ObjectId("65192b3c1890f8d9ebe31231"),
    bid: 4,
    name: 'my story',
    cost: 800
  },
  {
    _id: ObjectId("65192b4a1890f8d9ebe31232"),
    bid: 4,
    name: 'my story2.0',
    cost: 800
  },
  {
    _id: ObjectId("65192b551890f8d9ebe31233"),
    bid: 4,
    name: 'my story beta version',
    cost: 800
  }
]
ass10> db.library.find({"cost":{$gte:100}})
[
  {
    _id: ObjectId("65192b1e1890f8d9ebe31230"),
    bid: 4,
    name: 'my story',
    cost: 500
  },
  {
    _id: ObjectId("65192b3c1890f8d9ebe31231"),
    bid: 4,
    name: 'my story',
    cost: 800
  },
  {
    _id: ObjectId("65192b4a1890f8d9ebe31232"),
    bid: 4,
    name: 'my story2.0',
    cost: 800
  },
  {
    _id: ObjectId("65192b551890f8d9ebe31233"),
    bid: 4,
    name: 'my story beta version',
    cost: 800
  }
]
ass10> db.library.find({"cost":{$gte:500}})
```

```
[
  {
    _id: ObjectId("65192b1e1890f8d9ebe31230"),
    bid: 4,
    name: 'my story',
    cost: 500
  },
  {
    _id: ObjectId("65192b3c1890f8d9ebe31231"),
    bid: 4,
    name: 'my story',
    cost: 800
  },
  {
    _id: ObjectId("65192b4a1890f8d9ebe31232"),
    bid: 4,
    name: 'my story2.0',
    cost: 800
  },
  {
    _id: ObjectId("65192b551890f8d9ebe31233"),
    bid: 4,
    name: 'my story beta version',
    cost: 800
  }
]
```

```
ass10> db.library.find({"cost":{$gte:600}})
[
  {
    _id: ObjectId("65192b3c1890f8d9ebe31231"),
    bid: 4,
    name: 'my story',
    cost: 800
  },
  {
    _id: ObjectId("65192b4a1890f8d9ebe31232"),
    bid: 4,
    name: 'my story2.0',
    cost: 800
  },
  {
    _id: ObjectId("65192b551890f8d9ebe31233"),
    bid: 4,
    name: 'my story beta version',
    cost: 800
  }
]
```

```
ass10> db.library.find({"cost":{$in:[100,200,500]}})
[
  {
    _id: ObjectId("65192b1e1890f8d9ebe31230"),
    bid: 4,
    name: 'my story',
    cost: 500
  }
]
```

```
ass10> db.library.find({"cost":{$nin:[100,200,500]}})
[
  { _id: ObjectId("65191fc147ff05b0c6ab524f") },
  { _id: ObjectId("6519201947ff05b0c6ab5251"), bid: 2, name: 'java' },
  {
    _id: ObjectId("6519204047ff05b0c6ab5252"),
    bid: 3,
    name: 'Python3.7'
  },
  {
    _id: ObjectId("6519224647ff05b0c6ab5253"),
    bid: 4,
    name: 'java',
    desc: 'fake book'
  },
  {
    _id: ObjectId("65192b3c1890f8d9ebe31231"),
    bid: 4,
    name: 'my story',
    cost: 800
  },
  {
    _id: ObjectId("65192b4a1890f8d9ebe31232"),
    bid: 4,
    name: 'my story2.0',
    cost: 800
  },
  {
    _id: ObjectId("65192b551890f8d9ebe31233"),
    bid: 4,
    name: 'my story beta version',
    cost: 800
  }
]
```

MangoDB Assignment 2

Implement aggregation and indexing with suitable example using MongoDB

```
//USE DATABASE

> use comp;

switched to db comp

//CREATE COLLECTION WEBSITE

> db.createCollection('website');

{ "ok" : 1 }

//INSERT VALUES IN WEBSITE

> db.website.insert({'roll':'1','name':'harsh','amount':1000,'ur
l':'www.yahoo.com'});
WriteResult({ "nInserted" : 1 })

>db.website.insert({'roll':'2','name':'jitesh','amount':2000,'url':'www.yah
oo.com '});
WriteResult({ "nInserted" : 1 })

>db.website.insert({'roll':'3','name':'rina','amount':3000,'url':'www.googl
e.com' });
WriteResult({ "nInserted" : 1 })

>db.website.insert({'roll':'4','name':'ash','amount':4000,'url':'www.gmail.
com'}) ;
WriteResult({ "nInserted" : 1 })

>db.website.insert({'roll':'5','name':'ash','amount':1000,'url':'www.pvg.co
m'});
WriteResult({ "nInserted" : 1 })

//SUM AGGREGATE
> db.website.aggregate({$group:{_id:"$name","total":{$sum:"$amount"}}});
{ "_id" : "ash", "total" : 5000 }
{ "_id" : "rina", "total" : 3000 }
{ "_id" : "jitesh", "total" : 2000 }
{ "_id" : "harsh", "total" : 2000 }

//AVG AGGREGATE
> db.website.aggregate({$group:{_id:"$name","total": {$avg:"$amount"}}});
{ "_id" : "ash", "total" : 2500 }
{ "_id" : "rina", "total" : 3000 }
{ "_id" : "jitesh", "total" : 2000 }
{ "_id" : "harsh", "total" : 1000 }

 //MIN AGGREGATION
> db.website.aggregate({$group:{_id:"$name","total":{$min:"$amount"}}});
{ "_id" : "ash", "total" : 1000 }
{ "_id" : "rina", "total" : 3000 }
{ "_id" : "jitesh", "total" : 2000 }
{ "_id" : "harsh", "total" : 1000 }

//MAX AGGREGATION
> db.website.aggregate({$group:{_id:"$name","total":{$max:"$amount"}}});
```

```
{ "_id" : "ash", "total" : 4000 }
{ "_id" : "rina", "total" : 3000 }
{ "_id" : "jitesh", "total" : 2000 }
{ "_id" : "harsh", "total" : 1000 }


//FIRST AGGREGATION

> db.website.aggregate({$group:{_id:"$name","total":{$first:"$amount"}}});
{ "_id" : "ash", "total" : 4000 }
{ "_id" : "rina", "total" : 3000 }
{ "_id" : "jitesh", "total" : 2000 }
 { "_id" : "harsh", "total" : 1000 }

 //LAST AGGREGATION
> db.website.aggregate({$group:{_id:"$name","total":{$last:"$amount"}}});
{ "_id" : "ash", "total" : 1000 }
{ "_id" : "rina", "total" : 3000 }
{ "_id" : "jitesh", "total" : 2000 }
{ "_id" : "harsh", "total" : 1000 }

 //PUSH AGGREGATION
> db.website.aggregate({$group:{_id:"$name","total": {$push:"$amount"}}});
{ "_id" : "ash", "total" : [ 4000, 1000 ] }
{ "_id" : "rina", "total" : [ 3000 ] }
{ "_id" : "jitesh", "total" : [ 2000 ] }
{ "_id" : "harsh", "total" : [ 1000, 1000 ] }

//COUNT AGGREGATION
> db.website.aggregate({$group:{_id:"$name","total": {$sum:1}}});
{ "_id" : "ash", "total" : 2 }
{ "_id" : "rina", "total" : 1 }
{ "_id" : "jitesh", "total" : 1 }
{ "_id" : "harsh", "total" : 2 }

//ADDTOSET AGGREGATE
> db.website.aggregate({$group:
{_id:"$name","total"{$addToSet:"$amount"}}});
{ "_id" : "ash", "total" : [ 1000, 4000 ] }
{ "_id" : "rina", "total" : [ 3000 ] }
{ "_id" : "jitesh", "total" : [ 2000 ] }
{ "_id" : "harsh", "total" : [ 1000 ] }

//INDEXING
> db.createCollection('website1'); { "ok" : 1 }
> db.website1.insert({'r':1,'name':'harsh'});
WriteResult({ "nInserted" : 1 })


> db.website1.find().pretty()
{ "_id" : ObjectId("5ba3509a444926329738012d"), "roll" : 1, "name" :
"harsh" } { "_id" : ObjectId("5ba3529344492632973801 2e"), "roll" : 1,
"name" : "harsh" }


> db.website1.createIndex({'name':1})
{ "numIndexesBefore" : 2, "note" : "all indexes already exist", "ok" : 1 }
```

```
//CREATE INDEXING
> db.website1.createIndex({'name':-1})
{ "createdCollectionAutomatically" : false, "numIndexesBefore" : 2,
"numIndexesAfter" : 3, "ok" : 1 }

> db.website1.getIndexses()
2018-09-20T13:28:09.628+0530 TypeError: Property 'getIndexses' of object
om.website is not a function


> db.website1.getIndexes()

[ {"v" : 1, "key" : { "_id" : 1 }, "name" : "_id_", "ns" : "harsh.website1"
},
{ "v" : 1, "key" : { "name" : 1 }, "name" : "name_1", "ns" :
"harsh.website1" },
{ "v" : 1, "key" : { "name" : -1 }, "name" : "name_-1", "ns" :
"harsh.website1" } ]


> db.website1.createIndex({'name':-1})
{ "numIndexesBefore" : 3, "note" : "all indexes already exist", "ok" : 1 }

//DROP INDEX
> db.website.dropIndex({'name':-1})
{ "nIndexesWas" : 3, "ok" : 1 }> db.website1.dropIndex({'name':1})
{ "nIndexesWas" : 2, "ok" : 1 }> db.website1.dropIndex({'name':1})
{ "nIndexesWas" : 1, "ok" : 0, "errmsg" : "can't find index with key:{
name: 1.0 }" }

//GET INDEXING
> db.website1.getIndexes() [ { "v" : 1, "key" : { "_id" : 1 }, "name" :
"_id_", "ns" : "harsh.website1" } ]

> db.website1.find().pretty()
{ "_id" : ObjectId("5ba3509a444926329738012d"), "roll" : 1, "name" :
"harsh" }
{ "_id" : ObjectId("5ba352934449263297380l2e"), "roll" : 1, "name" :
"harsh" }

> db.website1.createIndex({'name':1})
{ "createdCollectionAutomatically" : false, "numIndexesBefore" : 1,
"numIndexesAfter" : 2, "ok" : 1 }

> db.website1.getIndexes()
[ { "v" : 1, "key" : { "_id" : 1 }, "name" : "_id_", "ns" :
"harsh.website1" },
{ "v" : 1, "key" : {"name" : 1 }, "name" : "name_1", "ns" :
"harsh.website1" } ]

> db.website1.dropIndex({'name':1})
{ "nIndexesWas" : 2, "ok" : 1 }

> db.website1.getIndexes()
 [ { "v" : 1, "key" : { "_id" : 1 }, "name" : "_id_", "ns" :
"harsh.website1" } ]

> db.website1.createIndex({'name':1,'r':-1})
{"createdCollectionAutomatically" : false, "numIndexesBefore" : 1,
"numIndexesAfter" : 2, "ok" : 1 }
```

```
> db.website1.getIndexes()
 [ { "v" : 1, "key" : { "_id" : 1 }, "name" : "_id_", "ns" :
"harsh.website1" },
{ "v" : 1, "key" : { "name" : 1, "r" : -1 }, "name" : "name_1_r_-1", "ns" :
"harsh.website1" } ] (i-
search)`db.website1.insert({'roll':1,'name':'harsh'});':
```

MangoDB Assignment 3

Implement Map reduces operation with suitable example using MongoDB

```
> db.createCollection('Journal');
{ "ok" : 1 }

>db.Journal.insert({'book_id':1,'book_name':'JavacdOOP','amt':500,'status':
'A vailable'}); WriteResult({ "nInserted" : 1 })

>
db.Journal.insert({'book_id':1,'book_name':'JavaOOP','amt':400,'status':'No
t Available'}); WriteResult({ "nInserted" : 1 })

>db.Journal.insert({'book_id':1,'book_name':'Java','amt':300,'s tatus':'Not
Available'});
WriteResult({ "nInserted" : 1 })

>db.Journal.insert({'book_id':2,'book_name':'Java','amt':300,'s
tatus':'Available'});
WriteResult({ "nInserted" : 1 })

>db.Journal.insert({'book_id':2,'book_name':'OPP','amt':200,'st
atus':'Available'});
WriteResult({ "nInserted" : 1 })

>db.Journal.insert({'book_id':2,'book_name':'C+','amt':200,'status':'Availa
ble'} );
WriteResult({ "nInserted" : 1 })

>db.Journal.insert({'book_id':3,'book_name':'C+','amt':150,'status':'Availa
ble'} );
WriteResult({ "nInserted" : 1 })

> db.Journal.insert({'book_id':3,'book_name':'C+ +','amt':200,'status':'Not
Available'});
WriteResult({ "nInserted" : 1 })

> db.Journal.insert({'book_id':4,'book_name':'OPP C+
+','amt':300,'status':'Not Available'}); WriteResult({ "nInserted" : 1 })

> db.Journal.insert({'book_id':5,'book_name':'OPP C+
+','amt':400,'status':'Available'});
WriteResult({ "nInserted" : 1 })

> db.Journal.insert({'book_id':5,'book_name':'C+
+','amt':400,'status':'Available'});
WriteResult({ "nInserted" : 1 })

> db.Journal.insert({'book_id':5,'book_name':'C++
Java','amt':400,'status':'Not Available'}); WriteResult({ "nInserted" : 1
})

> var mapfunction=function(){ emit(this.book_id,this.amt)};

> var reducefunction=function(key,value){return Array.sum(value);};

> db.Journal.mapReduce(mapfunction,reducefunction, {'out':'new'});
{ "result" : "new",
"timeMillis" : 49,"counts" : {
"input" : 12,
"emit" : 12,
"reduce" : 4,
"output" : 5
}, "ok" : 1 }

> db.new.find().pretty();
{ "_id" : 1, "value" : 1200 }
{ "_id" : 2, "value" : 700 }
{ "_id" : 3, "value" : 350 }
{ "_id" : 4, "value" : 300 }
{ "_id" : 5, "value" : 1200 }
```