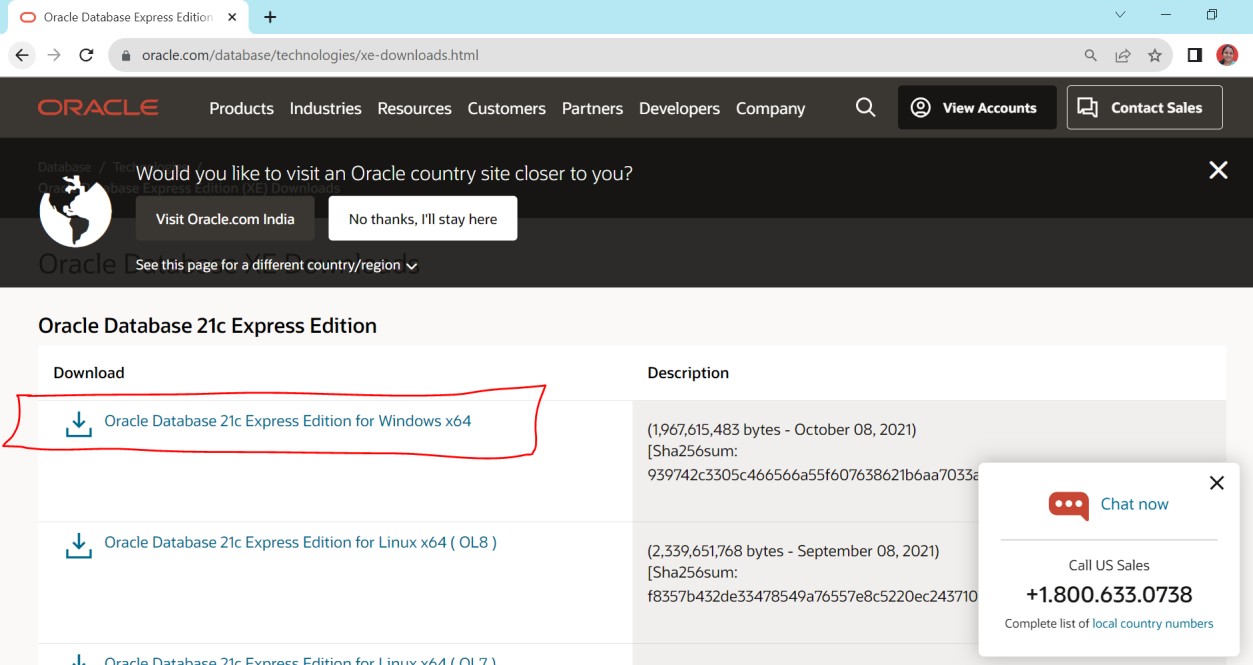


**DEPARTMENT OF COMPUTER ENGINEERING**

DBMS LAB REPORT

By Mrs. Vaishali Jorwekar

Download Oracle setup from below link: <https://www.oracle.com/database/technologies/xe-downloads.html>



By Mrs. Vaishali Jorwekar

**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 COST\_PRICE | DESCRIPTION | PROFIT\_PER | UNIT\_MEASU | QUANTITY | REORDER | SELL\_PRICE |
| 001 | shampoo | 1 | one | 4 | 2 | 10 |
| 15 |  |  |  |  |  |  |
| 002 | oil | 13 | one | 4 | 2 | 11 |
| 16 |  |  |  |  |  |  |

SQL> CREATE TABLE auto (

1. roll\_no NUMBER GENERATED ALWAYS AS IDENTITY NOT NULL,
2. name VARCHAR2(20),
3. 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  1 row created. | name | VALUES | (38 | ,'SHUBHAM'); |
| SQL> INSERT INTO  1 row created. | name | VALUES | (39 | ,'AKSHAY'); |
| SQL> INSERT INTO  1 row created. | name | VALUES | (40 | ,'SAKSHI'); |
| SQL> INSERT INTO  1 row created. | name | VALUES | (41 | ,'KETAN'); |

SQL> SELECT \* FROM name; ROLL\_NO NAME

1. INDRANEEL
2. SHUBHAM
3. AKSHAY
4. SAKSHI
5. 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

1. INDRANEEL YXYY
2. SHUBHAM YXYY
3. 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

1. INDRANEEL 2

YXYY 37

1. SHUBHAM 3

YXYY 38

ROLL\_NO NAME SR\_NO

ASSGN\_ID ROLL\_NO

1. AKSHAY 4

YXYY 39

1. SAKSHI
2. KETAN

6 rows selected.

SQL> SELECT name.roll\_no, submission.assgn\_id

1. FROM name
2. LEFT JOIN submission ON name.roll\_no = submission.roll\_no;

ROLL\_NO ASSGN\_ID

37 YYYY

1. YXYY
2. YXYY
3. 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

1. INDRANEEL 2

YXYY 37

1. SHUBHAM 3

YXYY 38

ROLL\_NO NAME SR\_NO

ASSGN\_ID ROLL\_NO

1. 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++

1. SHUBHAM PYTHON
2. 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

1. SQL 40 C++
2. PYTHON
3. 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 (

1. roll\_no NUMBER,
2. return\_date DATE,
3. fine NUMBER

5 );

Table created.

SQL> DECLARE

1. i\_roll\_no NUMBER;
2. name\_of\_book VARCHAR2(25);
3. no\_of\_days NUMBER;
4. return\_date DATE := TO\_DATE(SYSDATE,'DD-MM-YYYY');
5. temp NUMBER;
6. doi DATE;
7. fine NUMBER;
8. BEGIN
9. i\_roll\_no := &i\_roll\_no;
10. name\_of\_book := '&nameofbook';
11. dbms\_output.put\_line(return\_date);
12. 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;

1. no\_of\_days := return\_date-doi;
2. dbms\_output.put\_line(no\_of\_days);
3. IF (no\_of\_days >15 AND no\_of\_days <=30) THEN
4. fine := 5\*no\_of\_days;
5. ELSIF (no\_of\_days>30 ) THEN
6. temp := no\_of\_days-30;

20 fine := 150 + temp\*50;

1. END IF;
2. dbms\_output.put\_line(fine);
3. INSERT INTO fine VALUES(i\_roll\_no,return\_date,fine);
4. UPDATE borrower SET status = 'RETURNED' WHERE borrower.roll\_no = i\_roll\_no;
5. 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

* 1. ASHUTOSH 01-AUG-22 HARRY POTTER

PENDING

* 1. ARYAN 15-AUG-22 DARK MATTER RETURNED
  2. ROHAN 24-AUG-22 SILENT HILL

PENDING

ROLL\_NO NAME DATEOFISS NAME\_OF\_BOOK

STATUS

* 1. SANKET 26-AUG-22 GOD OF WAR

PENDING

* 1. 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

1. BEGIN
2. procedure\_1(r,n,m);
3. return 'SUCCESSFULL';
4. END;

6 /

Function created.

SQL> CREATE OR REPLACE PROCEDURE procedure\_1 ( roll\_no IN NUMBER, name IN VARCHAR2 ,marks IN NUMBER) AS

1. BEGIN
2. IF (marks<=1500 and marks>=990) THEN
3. DBMS\_OUTPUT.PUT\_LINE ('DISTINCTION');
4. INSERT INTO result VALUES (roll\_no,name,'DISTINCTION');
5. ELSIF (marks<=989 and marks>=900) THEN
6. DBMS\_OUTPUT.PUT\_LINE ('FIRST CLASS');
7. INSERT INTO result VALUES (roll\_no,name,'FIRST CLASS');
8. ELSIF (marks<=899 and marKs>825) THEN
9. DBMS\_OUTPUT.PUT\_LINE('HIGHER SECOND CLASS');
10. INSERT INTO result VALUES (roll\_no,name,'HIGHER SECOND CLASS');
11. ELSE
12. DBMS\_OUTPUT.PUT\_LINE ('FAIL');
13. INSERT INTO result VALUES (roll\_no,name,'FAIL'); 15
14. END IF;
15. INSERT INTO stud\_marks VALUES (name,marks);
16. END procedure\_1; 19 /

Procedure created.

SQL> DECLARE

* 1. name\_1 VARCHAR2(25);
  2. roll\_no\_1 NUMBER;
  3. marks\_1 NUMBER;
  4. class VARCHAR2(25);
  5. BEGIN
  6. roll\_no\_1:=&roll\_no\_1;

8 name\_1:='&name\_1';

9 marks\_1:=&marks\_1;

1. class := func\_1(roll\_no\_1,name\_1,marks\_1);
2. dbms\_output.put\_line(class);
3. 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

1. a INT;
2. a1 VARCHAR2(10);
3. b INT;
4. b1 VARCHAR2(10);
5. CURSOR c1 IS SELECT roll, name FROM old\_roll;
6. CURSOR c2 IS SELECT roll, name FROM new\_roll;
7. BEGIN
8. OPEN c1;
9. OPEN c2;

11

1. LOOP
2. FETCH c1 INTO a, a1;
3. EXIT WHEN c1%NOTFOUND; 15
4. -- Check if a record with the same 'roll' exists in new\_roll
5. BEGIN
6. SELECT roll INTO b FROM new\_roll WHERE roll = a;
7. EXCEPTION
8. WHEN NO\_DATA\_FOUND THEN
9. -- If no matching record found, insert into new\_roll
10. INSERT INTO new\_roll (roll, name) VALUES (a, a1);
11. END;
12. END LOOP;

25

1. CLOSE c1;
2. CLOSE c2;

28

1. -- Commit the transaction to save changes permanently
2. COMMIT;
3. END;

32 /

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

1. AFTER UPDATE OR DELETE OR INSERT ON lib\_tab FOR EACH ROW
2. ENABLE
3. BEGIN
4. IF UPDATING THEN
5. dbms\_output.put\_line(:OLD.status);
6. INSERT INTO library\_audit2 VALUES (SYSDATE,:OLD.book\_name,:OLD.status,:NEW.status,'UPDATE');
7. ELSIF INSERTING THEN
8. dbms\_output.put\_line(:NEW.status);
9. INSERT INTO library\_audit2 VALUES

(SYSDATE,:NEW.book\_name,:OLD.status,:NEW.status,'INSERT');

1. ELSE
2. dbms\_output.put\_line(:OLD.book\_name||'deleting');
3. INSERT INTO library\_audit2 VALUES(SYSDATE,:OLD.book\_name,:OLD.status,:NEW.status,'DELETE');
4. END IF;
5. 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

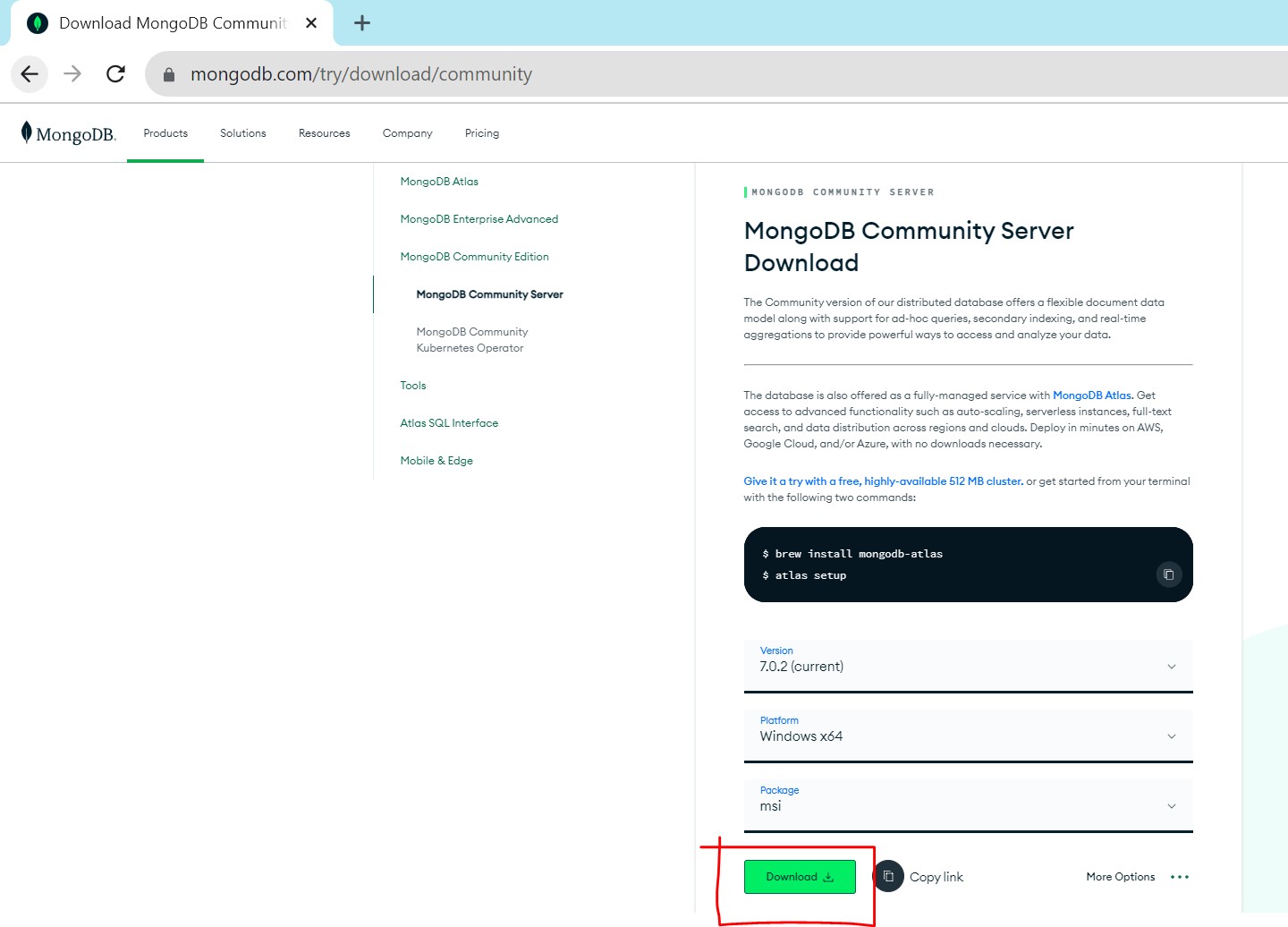
GOD OF WAR PRE-ORDER

SPIDER-MAN UNAVAILABLE

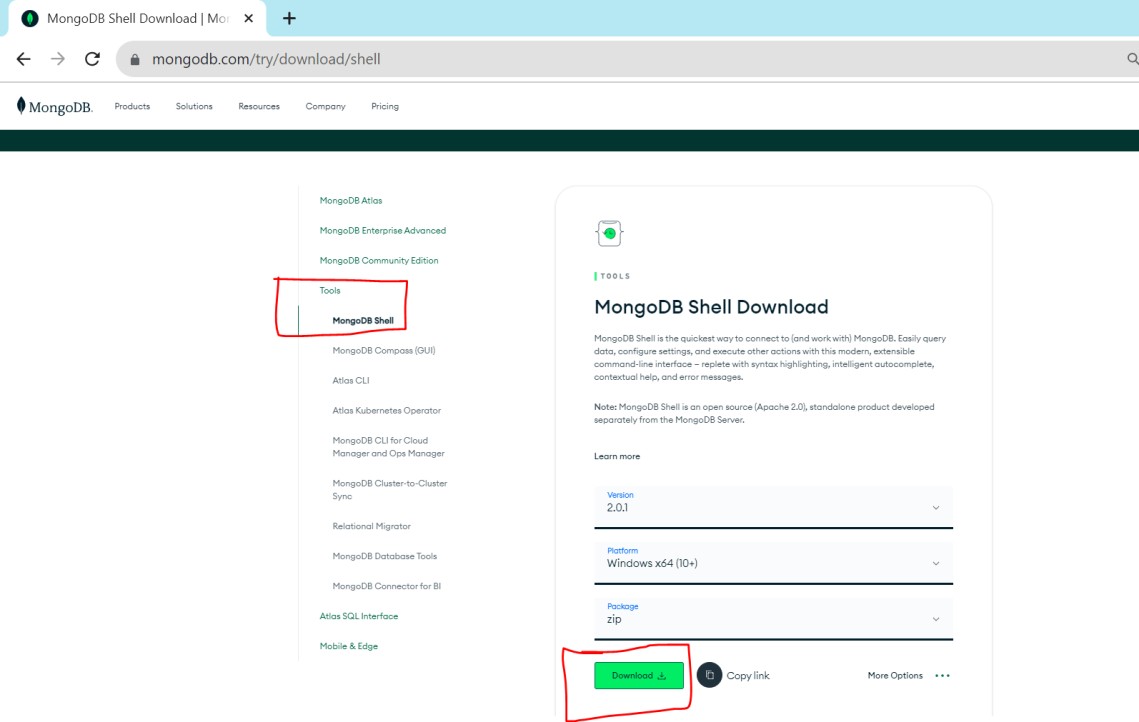
UNCHARTED UNAVAILABLE

**MongoDB Practical**

**Download MongoDB Community Server:** [**https://www.mongodb.com/try/download/community**](https://www.mongodb.com/try/download/community)



**Download MongoDB Shell Download** [**https://www.mongodb.com/try/download/shell**](https://www.mongodb.com/try/download/shell)



**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 function in the MongoDB

**save()**

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','am[ount':1000,'url':'www.pvg.co](http://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("5ba35293444926329738012e"), "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("5ba35293444926329738012e"), "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 }