Practical No 1

Aim: Demonstrate the Upsampling and downsampling inImage:

Code:

```
import cv2
import matplotlib.pyplot as plt
img = cv2.imread(r'C:\Users\yashvarik\OneDrive\Pictures\Saved Pictures\triumph-speed-400-01.webp')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
downsampled = cv2.pyrDown(img)
upsampled = cv2.pyrUp(img)
plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
plt.title("Original")
plt.imshow(img)
plt.axis("off")
plt.subplot(1, 3, 2)
plt.title("Downsampled")
plt.imshow(downsampled)
plt.axis("off")
plt.subplot(1, 3, 3)
plt.title("Upsampled")
plt.imshow(upsampled)
plt.axis("off")

plt.show()
```

Output:

Practical No 2

Aim: Demonstrate the following aspects of signal and image processing

1 convolution on grey and color image

2 templet macthing in python

Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img_color = cv2.imread(r"C:\Users\yashvarik\OneDrive\Pictures\Saved Pictures\triumph-speed-400-01.webp")
img_color = cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB)
img_gray = cv2.cvtColor(img_color, cv2.COLOR_RGB2GRAY)
kernel = np.array([[0, -1, 0],
                   [-1, 5, -1],
                   [0, -1, 0]])
convolved_gray = cv2.filter2D(img_gray, -1, kernel)
convolved_color = cv2.filter2D(img_color, -1, kernel)
plt.figure(figsize=(10,6))
plt.subplot(2,2,1); plt.title("Original Gray")
plt.imshow(img_gray, cmap="gray"); plt.axis("off")
plt.subplot(2,2,2); plt.title("Convolved Gray")
plt.imshow(convolved_gray, cmap="gray"); plt.axis("off")
plt.subplot(2,2,3); plt.title("Original Color")
plt.imshow(img_color); plt.axis("off")
plt.subplot(2,2,4); plt.title("Convolved Color")
plt.imshow(convolved_color); plt.axis("off")
plt.show()
```

Output:

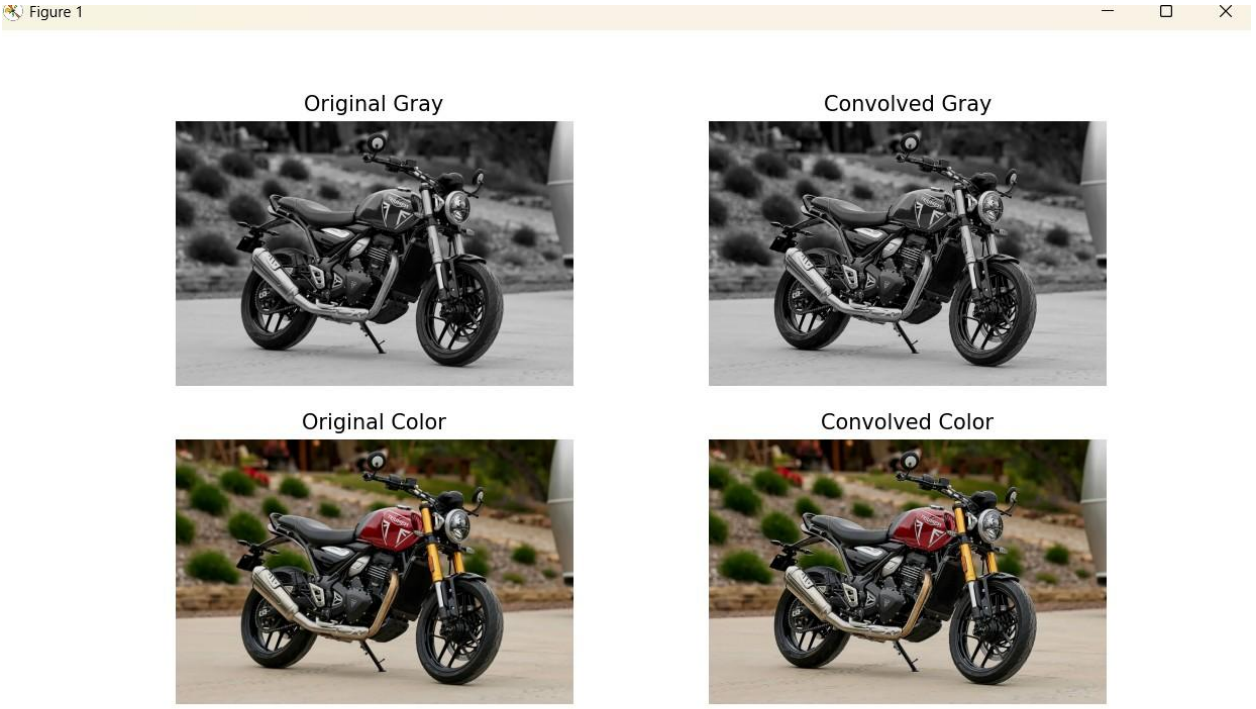


Figure 1
Original Gray
Convolved Gray
Original Color
Convolved Color

Practical No 3

Aim: Intensity Transformations using

1. Image Negatives (Linear)
2. Log Transformations
3. Power-Law (Gamma) Transformations
4. Piecewise-Linear Transformation Functions in python without comments

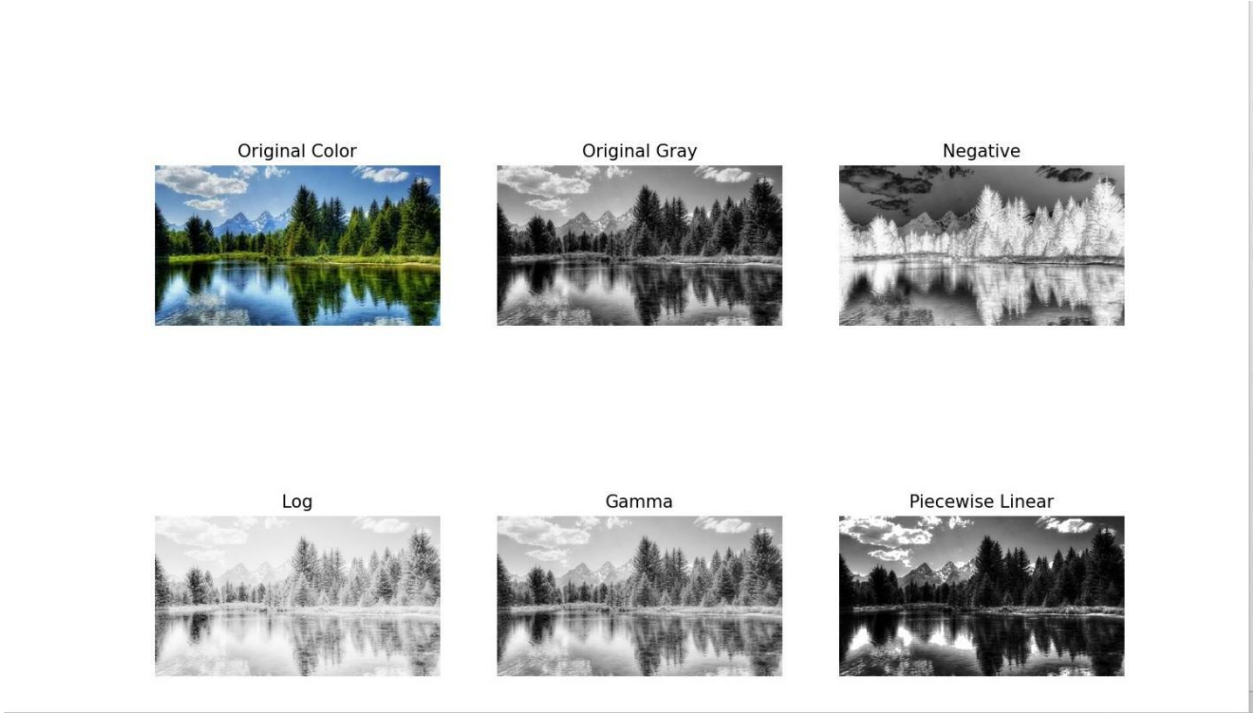
Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img_color = cv2.imread(r"C:\Users\yashvarik\OneDrive\Pictures\Saved Pictures\asip stuff.jpg")
img_color = cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB)
img_gray = cv2.cvtColor(img_color, cv2.COLOR_RGB2GRAY)
negative = 255 - img_gray
img_float = img_gray.astype(np.float32)
c = 255 / np.log(1 + np.max(img_float))
log_trans = c * np.log(1 + img_float)
log_trans = np.array(log_trans, dtype=np.uint8)
gamma = 0.5
gamma_trans = np.array(255 * (img_float / 255) ** gamma, dtype=np.uint8)
def piecewise_linear(x):
    if x < 100:
        return 0.5 * x
    elif x < 200:
        return 1.5 * (x - 100) + 50
    else:
        return 255
piecewise = np.array([piecewise_linear(i) for i in range(256)], dtype=np.uint8)
piecewise_img = cv2.LUT(img_gray, piecewise)


plt.figure(figsize=(12,8))
```

```
plt.subplot(2,3,1); plt.title("Original Color"); plt.imshow(img_color); plt.axis("off")
plt.subplot(2,3,2); plt.title("Original Gray"); plt.imshow(img_gray, cmap="gray"); plt.axis("off")
plt.subplot(2,3,3); plt.title("Negative"); plt.imshow(negative, cmap="gray"); plt.axis("off")
plt.subplot(2,3,4); plt.title("Log"); plt.imshow(log_trans, cmap="gray"); plt.axis("off")
plt.subplot(2,3,5); plt.title("Gamma"); plt.imshow(gamma_trans, cmap="gray"); plt.axis("off")
plt.subplot(2,3,6); plt.title("Piecewise Linear"); plt.imshow(piecewise_img, cmap="gray"); plt.axis("off")
plt.show()
```

Output:



Practical No 4

Aim: Write a program to apply edge detection techniques such as sobel and canny to extract meaningful information from the given image samples.

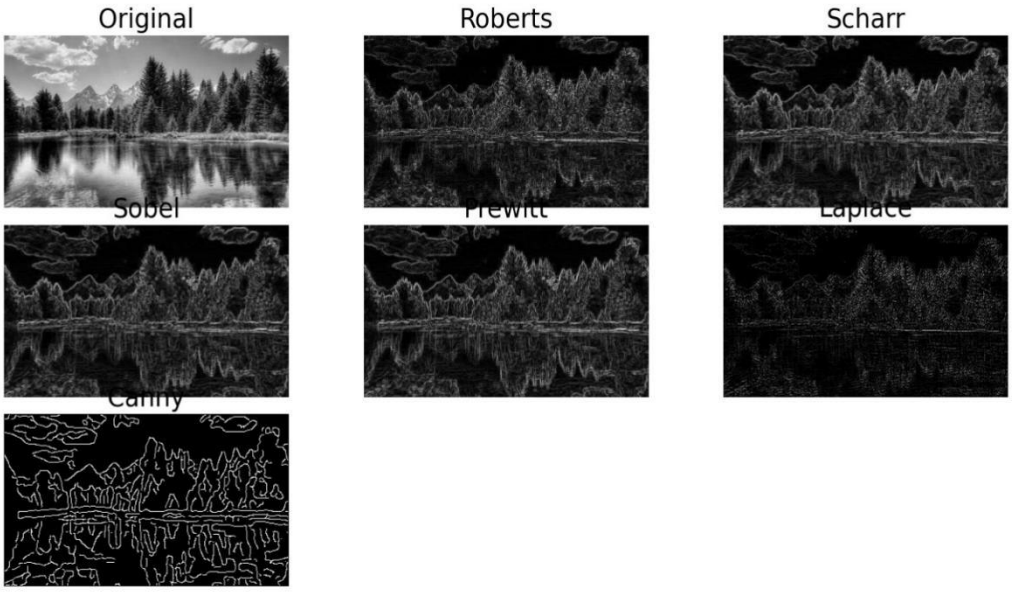
Code:

```
import numpy as np
from skimage.io import imread
from skimage import filters, feature
from skimage.color import rgb2gray
import matplotlib.pylab as pylab
def plot_image(image, title=""):
    pylab.title(title, size=20)
    pylab.imshow(image, cmap='gray')
    pylab.axis('off')
im = imread(r"C:\Users\yashvarik\OneDrive\Pictures\Saved Pictures\asip stuff.jpg")
im = rgb2gray(im)
pylab.figure(figsize=(18, 15))
pylab.subplot(3, 3, 1)
plot_image(im, 'Original')
pylab.subplot(3, 3, 2)
plot_image(filters.roberts(im), 'Roberts')
pylab.subplot(3, 3, 3)
plot_image(filters.scharr(im), 'Scharr')
pylab.subplot(3, 3, 4)
plot_image(filters.sobel(im), 'Sobel')
pylab.subplot(3, 3, 5)
plot_image(filters.prewitt(im), 'Prewitt')
pylab.subplot(3, 3, 6)
laplace_edges = np.clip(filters.laplace(im), 0, 1)
plot_image(laplace_edges, 'Laplace')
pylab.subplot(3, 3, 7)
plot_image(feature.canny(im, sigma=2), 'Canny')
```

```
pylab.subplots_adjust(wspace=0.1, hspace=0.1)

pylab.show()
```

Output:



Practical No 1

AIM: Distributed Databases Perform Horizontal Fragmentation
Create a global conceptual schema Emp(Eno;Ename;Address;Email;Salary) and insert 10 records. Divide Emp into horizontal fragments using the condition that Emp1 contains the tuples with salary <= 10,000 and Emp2 with 10,000<salary<=20,000 on twodifferent nodes.

Fire the following queries:
(i) Find the salary of all employees.
(ii) Find the Email of all employees where salary=12,000.
(iii) Find the employee name and Email where employee number is known.
(iv) Find the employee name where salary>20,000.

Open the sql with system/root user.Check global database as:
select * from global_name;

Connect to global database
first.SQL>Connect
system@xe
Enter password:
Connected.

Alter session to get the access to create users
SQL> alter session set "_ORACLE_SCRIPT"=true;Session altered.

Create different users and grant them various privileges.SQL> create USER user1 IDENTIFIED by u1;
User created.

SQL> Grant create session,create table,create sequence,create view,create procedure,connect,resource, create database link, unlimited tablespace to user1;Grant succeeded.

```
SQL> alter session set "_ORACLE_SCRIPT"=true;

Session altered.

SQL> create USER user1 IDENTIFIED by u1;

User created.

SQL> Grant create session,create table,create sequence,create view,create procedure,connect,resource, create database link, unlimited tablespace to user1;

Grant succeeded.
```

SQL> create USER user2 IDENTIFIED by u2;User created.

SQL> Grant create session,create table,create sequence,create view,create procedure,connect,resource, create database link, unlimited tablespace to user2;

Grant succeeded.

```
SQL> create USER user2 IDENTIFIED by u2;

User created.

SQL> Grant create session,create table,create sequence,create view,create procedure,connect,resource, create database link, unlimited tablespace to user2;

Grant succeeded.
```

SQL> create USER user3 IDENTIFIED by
u3;User created.

SQL> Grant create session,create table,create sequence,create view,create procedure,connect,resource, create database link, unlimited tablespace to user3;Grant succeeded.

```
SQL> create USER user3 IDENTIFIED by u3;

User created.

SQL> Grant create session,create table,create sequence,create view,create procedure,connect,resource, create database link, unlimited tablespace to user3;

Grant succeeded.
```

Creating table and inserting 10 data's.
SQL> create table emp(eno number(3) primary key,ename varchar2(20),address varchar2(30),emailvarchar2(30),sal number(6));
Table created.

SQL> insert into emp
values(101,'ram','dadar','ram@gmail.com',10000);1 row created.

SQL> insert into emp
values(102,'tom','cst','tom@gmail.com',25000);1 row created.

SQL> insert into emp
values(103,'sam','wadala','sam@gmail.com',30000);1 row created.

SQL> insert into emp
values(104,'sonu','wadala','sonu@gmail.com',12000);1 row created.

SQL> insert into emp
values(105,'monu','matunga','monu@gmail.com',24000);1 row created.

SQL> insert into emp
values(106,'mona','sion','mona@gmail.com',35000);1 row created.

```
SQL> create table emp(eno number(3) primary key,ename varchar2(20),address varchar2(30),email varchar2(30),sal number(6));

Table created.

SQL> insert into emp values(101,'ram','dadar','ram@gmail.com',10000);

1 row created.

SQL> insert into emp values(102,'tom','cst','tom@gmail.com',25000);

1 row created.

SQL> insert into emp values(103,'sam','wadala','sam@gmail.com',30000);

1 row created.

SQL> insert into emp values(104,'sonu','wadala','sonu@gmail.com',12000);

1 row created.

SQL> insert into emp values(105,'monu','matunga','monu@gmail.com',24000);

1 row created.

SQL> insert into emp values(106,'mona','sion','mona@gmail.com',35000);

1 row created.
```

SQL> insert into emp
values(107,'sona','sion','sona@gmail.com',40000);1 row created.

SQL> insert into emp
values(108,'harry','kurla','harry@gmail.com',30000);1 row created.

SQL> insert into emp
values(109,'marry','kurla','marry@gmail.com',60000);1 row created.

SQL> insert into emp
values(110,'anni','kurla','anni@gmail.com',10000);1 row created.

```
SQL> insert into emp values(107,'sona','sion','sona@gmail.com',40000);

1 row created.

SQL> insert into emp values(108,'harry','kurla','harry@gmail.com',30000);

1 row created.

SQL> insert into emp values(109,'marry','kurla','marry@gmail.com',60000);

1 row created.

SQL> insert into emp values(110,'anni','kurla','anni@gmail.com',10000);

1 row created.
```

Perform the
commit.
SQL> commit;
Commit
complete.

```
SQL> commit;

Commit complete.
```

Now open another sqlplus and login with user1 id and password.SQL> connect user1@xe

Connected.

```
SQL> connect system@xe
Enter password:
Connected.
```

Create database link to global database in order to access the data of global database table.SQL> create database link l1 connect to system identified by system using 'xe';
Database link created.

```
SQL> create database link l1 connect to system identified by system using 'xe';

Database link created.
```

Fire select query to check if the link to databases is functioning well.SQL> select * from emp@l1;

ENO ENAME ADDRESS
---------- ---------- ----------

-----EMAIL SAL
---------- ----------

101 ram ram@gmail.c om
dadar
10000

102 tom tom@gmail.c om cst 25000

```
103         sam       wadala
 sam@gmail.com                  30000

      ENO ENAME           ADDRESS
 ---------- - - - - - - - - - - - ----------------------------

 -----EMAIL                  SAL
 -------------------------------- -----------

104         sonu      wadala
 sonu@gmail.com
            12000

105         monu      matunga
 monu@gmail.com
            24000

      106 mona             sion
 mona@gmail.com                 35000

      ENO ENAME           ADDRESS
 ---------- - - - - - - - - - - - ----------------------------

 -----EMAIL                  SAL

 ----------------------      -----------
      107 sona             sion
 sona@gmail.com                 40000
      108 harry            kurla
 harry@gmail.com                30000

      109 marry            kurla
 marry@gmail.com                60000

      ENO ENAME           ADDRESS


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


 -----EMAIL                  SAL
 -------------------------------- -----------

      110 anni             kurla
 anni@gmail.com                 10000

 10 rows selected.
```

```
SQL> select * from emp@l1;

       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       101 ram                  dadar
ram@gmail.com                      10000

       102 tom                  cst
tom@gmail.com                      25000

       103 sam                  wadala
sam@gmail.com                      30000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       104 sonu                 wadala
sonu@gmail.com                     12000

       105 monu                 matunga
monu@gmail.com                     24000

       106 mona                 sion
mona@gmail.com                     35000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       107 sona                 sion
sona@gmail.com                     40000

       108 harry                kurla
harry@gmail.com                    30000

       109 marry                kurla
marry@gmail.com                    60000


       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       110 anni                 kurla
anni@gmail.com                     10000
```

Create table and insert the data from global database table where all sal<=10000.SQL> create table emp1 as select * from emp@l1 where sal<=10000;
Table created.

```
SQL> create table emp1 as select * from emp@l1 where sal<=10000;

Table created.
```

View the inserted data into the created table.SQL> select * from emp1;

ENO ENAME ADDRESS
---------- ---------- ----------

-----EMAIL SAL

---------- ----------
101 ram dadar
ram@gmail.com 10000
110 anni kurla
anni@gmail.com 10000

```
SQL> select * from emp1;

       ENO ENAME                ADDRESS
---------- -------------------- ------------------------------
EMAIL                                 SAL
------------------------------ ----------
       101 ram                  dadar
ram@gmail.com                       10000

       110 anni                 kurla
anni@gmail.com                      10000
```

Perform the commit.SQL> commit; Commit complete.

```
SQL> commit;

Commit complete.
```

Now open another sqlplus and login with user2 id and password.SQL> connect user2@xe
Connected.

Create database link to global database in order to access the data of global database table.SQL> create database link l22 connect to system identified by system using 'xe';
Database link created.
Create table and insert the data from global database table where all 10000 < sal < = 20000.SQL> create table emp2 as select * from emp@l22 where sal>10000 and sal<=20000; Table created.

View the inserted data into the created table.SQL> select * from emp2;

```
       ENO ENAME           ADDRESS
---------- - - - - - - - - - - - ----------------------------

-----EMAIL                   SAL
 ---------------------------------- ------
                    ------


     104 sonu              Wadala
sonu@gmail.com                  12000
```

Perform the commit.
SQL> commit;
Commit complete.

Now let's create insert trigger in global database table in order to maintain the consistency of the data in all the distributed tables.

SQL> create database link l22 connect to user2 identified by u2 using 'xe';Database link created.

SQL> create database link l1 connect to user1 identified by u1 using 'xe';Database link created.

Creating insert trigger

```
SQL> create or replace trigger
 triginsertemp122 after insert on emp
 3 for each
 row4
 begin
 5 if :new.sal>10000 and :new.sal<=20000
 then6  insert into emp2@l22 values
 7  (:new.eno,:new.ename,:new.address,:new.email,:new.sal);
 8  else
 9  insert into emp1@l1 values
10  (:new.eno,:new.ename,:new.address,:new.email,:new.sal);
11  end if;
12  end;
13 /
```

Trigger created.

Let's insert one row to the global table and check whether it is being available in the distributed tableor not.

SQL> insert into emp
values(111,'ratan','bandra','ratan@gmail.com',14500);1 row created.

Perform the commit.
SQL> commit;
Commit

Check in table emp2 if this data is being added or not, since the sal>14500 it should be added inemp2.

```
SQL> select * from emp2;

       ENO ENAME           ADDRESS
---------- - - - - - - - - - - - ----------------------------

-----EMAIL                   SAL
 ---------------------------------- ------
                    ------


     104 sonu              wadala
 sonu@gmail.com                  12000


      111 ratan             bandra
  ratan@gmail.com                14500
```

Now open another sqlplus and login with user3 id and password.SQL> connect user3@xe
Connected.

Create database link for all the three tables here in order to access various types of data based on givequestion
SQL> create database link l0 connect to system identified by system using 'xe';Database link created.

SQL> create database link l22 connect to user2 identified by u2 using 'xe';Database link created.

SQL> create database link l1 connect to user1 identified by u1 using 'xe';Database link created.

(i) Find the salary of all employees.SQL> select sal from emp@l0;

```
SAL
-----------

     10000
     25000
     30000
     12000
     24000
     35000
     40000
     30000
     60000
     10000
     14500
```

11 rows selected.

(ii) Find the Email of all employees where salary=12,000.SQL> select email from emp2@l22 where sal=12000;

EMAIL

---

sonu@gmail.com

(iii) Find the employee name and Email where employee number is known.SQL> select ename,email from emp@l0 where eno=109;

ENAME	EMAIL
marry	marry@gmail.com

(iv) Find the employee name whose salary >20000 SQL> select ename from emp@l0 where sal>20000;

ENAME

---

to
m
sa
m
m
o
n
u
m
o
na
so
na
ha
rr
y
m
ar
ry

7 rows selected.

AIM: Distributed Databases Perform Vertical Fragmentation

Create a global conceptual schema Emp(Eno;Ename;Address;email;Salary) and insert 10 records. Divide Emp into vertical fragments Emp1(Eno;Ename;Address) andEmp2(Eno;Email;Salary) on two different nodes.

Fire the following queries:
a) Find the salary of an employee where employee number is known.
b) Find the Email where the employee name is known.
c) Find the employee name and email where employee number is known.
d) Find the employee name whose salary is > 2000.

Open sqlplus with system user and connect to the global database.SQL> connect system@xe
Connected.

Check if the table already exist. In our case the table already exist.SQL> select * from emp;

```
    ENO ENAME           ADDRESS
---------- ------------ ----------------------------

-----EMAIL                   SAL
-------------------------------- -----------

     101 ram ram@gmail.com
                                  dadar
                                        10000



     102 tom tom@gmail.com
                                        cst
                                             25000
```

103 sam wadala
sam@gmail.com 30000

ENO ENAME ADDRESS
---------- - - - - - - - - - - - - ----------

-----EMAIL SAL
---------- ----------

104 sonu wadala
sonu@gmail.com
12000

105 monu matunga
monu@gmail.com 24000
106 mona sion
mona@gmail.com 35000

ENO ENAME ADDRESS
---------- - - - - - - - - - - - ----------

-----EMAIL SAL

---
107 sona sion
sona@gmail.com 40000

108 harry kurla
harry@gmail.com 30000

109 marry kurla
marry@gmail.com 60000

ENO ENAME ADDRESS
---------- - - - - - - - - - - - ----------

-----EMAIL SAL
---------- ----------

110 anni kurla
anni@gmail.com 10000

111 ratan bandra
ratan@gmail.com 14500

11 rows selected.

Now open another sqlplus and login with user1 id and password.SQL> connect user1@xe
Connected.

Create database link to global database in order to access the data of global database table.SQL> create database link l1 connect to system identified by system using 'xe';
Database link created.

Create table and insert the data from global database table
SQL> create table e1 as select eno,ename,address from emp@l1;Table created.

View the inserted data into the created table.SQL> select * from e1;

ENO	ENAME	ADDRESS
101	ram	dadar
102	tom	cst
103	sam	wadala
104	sonu	wadala
105	monu	matunga
106	mona	sion
107	sona	sion
108	harry	kurla
109	marry	kurla
110	anni	kurla
111	ratan	bandra

11 rows selected.

Perform the
commit.
SQL> commit;
Commit
complete.

Now open another sqlplus and login with user2 id and
password.SQL> connect user2@xe
Connected.

Create database link to global database in order to access the data of global database table.SQL> create database link l22 connect to system identified by system using'xe';
Database link created.

Create table and insert the data from global database
table SQL> create table e2 as select eno,email,sal from
emp@l22;Table created.

View the inserted data into the created
table.SQL> select * from e2;

ENO	EMAIL	SAL
101	ram@gmail.com	10000
102	tom@gmail.com	25000
103	sam@gmail.com	30000
104	sonu@gmail.com	12000
105	monu@gmail.com	24000
106	mona@gmail.com	35000
107	sona@gmail.com	40000
108	harry@gmail.com	30000
109	marry@gmail.com	60000
110	anni@gmail.com	10000
111	ratan@gmail.com	14500

11 rows selected.

Perform the
commit.
SQL> commit;
Commit
complete.

Now open another sqlplus and login with user3 id and
password.SQL> connect user3@xe
Connected.

Create database link for tables here in order to access various types of data based on give

questionSQL> create database link l11 connect to user1 identified by u1 using'xe';
Database link created

SQL> create database link l22 connect to user2 identified by u2 using
'xe';Database link created

(i) Find the salary of an employee where employee number is known.SQL> select sal from e2@l22 where eno=&eno;
Enter value for eno: 105
old 1: select sal from e2@l22 where eno=&enonew 1: select sal from e2@l22 where eno=105

SAL

24000

(ii) Find the Email where the employee name is known.
SQL> select email from e2@l22 where eno=(select eno from e1@l11 where ename='&ename');
Enter value for ename: mona
old 1: select email from e2@l22 where eno=(select eno from e1@l11 where ename='&ename')new 1: select email from e2@l22 where eno=(select eno from e1@l11 where ename='mona')

EMAI
L
---

mona@gmail.com

(iii) Find the employee name and email where employee number is known.
SQL> select e11.ename,e22.email from e1@l11 e11,e2@l22 e22 where e11.eno in (select e22.enofrom e2@l22 where e22.eno=&eno);
Enter value for eno: 110
old 1: select e11.ename,e22.email from e1@l11 e11,e2@l22 e22 where e11.eno in (select e22.enofrom e2@l22 where e22.eno=&eno)
new 1: select e11.ename,e22.email from e1@l11 e11,e2@l22 e22 where e11.eno in (select e22.enofrom e2@l22 where e22.eno=110)

ENAME EMAIL
---

anni anni@gmail.com

(iv) Find the employee name whose salary is > 2000.
SQL> select ename from e1@l11 where eno in (select eno from e2@l22 where sal>2000);

ENAME
---
ra m to m sa so
n u m o n u m o na so na ha rr y m ar ry an ni ra ta n

11 rows selected.

Practical No 2

Demonstrate Object-Oriented Databases

Q. Create the following types using OODBMS:
PersonType (Per_id, Per_name, DOB)
DOB is the date of birth of a person. Include a method to find the age of aperson.
BankType (account_no, per_id, balance)

Create the appropriate tables.Insert 5 tuples and fire the following queries:

(i) Find name and age of a person
(ii)Find all names with account number.
(iii)Find the names with balance more than 12,000

SQL> create or replace type persontype as object(personid number(3), person_name varchar2(20), dob date, member function age return number);

Type created.

SQL> create or replace type body persontype as member function age return number2 as
3 begin
4 return(round((sysdate-dob)/365));
5 end age;
6 end;
7 /

Type body created.

SQL> create or replace type banktype as object(account_no number(20),person refpersontype,balance number(20,2)); 2 /

Type created.

```
SQL> create or replace type persontype as object(personid number(3), person_name varchar2(20), dob date, member function age return number);
  2  /

Type created.

SQL> create or replace type body persontype as member function age return number
  2  as
  3  begin
  4  return
  5  (round((sysdate-dob)/365));
  6  end age;
  7  end;
  8  /

Type body created.

SQL> create or replace type banktype as object(account_no number(20),person ref persontype,balance number(20,2));
  2  /

Type created.
```

SQL> create table person of persontype;

Table created.

SQL> insert into person values(persontype(1,'sam','24- apr- 1993'));1

row created.

SQL> insert into person values(persontype(2,'anni','24- aug- 1992'));1

row created.

SQL> insert into person values(persontype(3,'marry','11- apr- 1992'));1
row created.

SQL> insert into person values(persontype(4,'harry','5-oct- 1992'));1 row created.

SQL> insert into person values(persontype(5,'sunny','14- jun- 1993'));1

row created.

```
SQL> create table person of persontype;

Table created.

SQL> insert into person values(persontype(1,'sam','24-apr-1993'));

1 row created.

SQL> insert into person values(persontype(2,'anni','24-aug-1992'));

1 row created.

SQL> insert into person values(persontype(3,'marry','11-apr-1992'));

1 row created.

SQL> insert into person values(persontype(4,'harry','5-oct-1992'));

1 row created.

SQL> insert into person values(persontype(5,'sunny','14-jun-1993'));

1 row created.
```

SQL> select * from person; PERSONID

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

PERSON_NAME

DOB1

sam 24-APR-93

2 anni 24-AUG-92
3 marry 11-APR-92
4 harry 05-OCT-92
5 Sunny 14-JUN-9

```
SQL> select * from person;

  PERSONID PERSON_NAME          DOB
---------- -------------------- ---------
         1 sam                  24-APR-93
         2 anni                 24-AUG-92
         3 marry                11-APR-92
         4 harry                05-OCT-92
         5 sunny                14-JUN-93
```

SQL>select p.*,p.age() "Age" from person p;

PERSONID PERSON_NAME DOB Age

1	sam	24-APR-93	21
2	anni	24-AUG-92	21
3	marry	11-APR-92	22
4	harry	05-OCT-92	21
5	Sunny	14-JUN-93	21

```
SQL> select p.*,p.age() "Age" from person p;

  PERSONID PERSON_NAME          DOB              Age
---------- -------------------- --------- ----------
         1 sam                  24-APR-93         31
         2 anni                 24-AUG-92         32
         3 marry                11-APR-92         33
         4 harry                05-OCT-92         32
         5 sunny                14-JUN-93         31
```

(i) Find name and age of a person

SQL> select p.person_name,p.age() "Age" from person p;

PERSON_NAME	Age
sam	21
anni	21
marry	22
harry	21
Sunny	21

```
SQL> select p.person_name,p.age() "Age" from person p;

PERSON_NAME                    Age
-------------------- ----------
sam                            31
anni                           32
marry                          33
harry                          32
sunny                          31
```

SQL> create table banktable of banktype;

Table created.

SQL> insert into banktable select 101,ref(p),10000 from person p wherep.person_name='sam';
1 row created.

SQL> insert into banktable select 102,ref(p),20000 from person p wherep.person_name='anni';

1 row created.

SQL> insert into banktable select 103,ref(p),30000 from person p wherep.person_name='marry';

1 row created.

SQL> insert into banktable select 104,ref(p),40000 from person p wherep.person_name='harry';

1 row created.

SQL> insert into banktable select 105,ref(p),50000 from person p wherep.person_name='sunny';

1 row created.

```
SQL> create table banktable of banktype;

Table created.

SQL> insert into banktable select 101,ref(p),10000 from person p where p.person_name='sam';

1 row created.

SQL> insert into banktable select 102,ref(p),20000 from person p where p.person_name='anni';

1 row created.

SQL> insert into banktable select 103,ref(p),30000 from person p where p.person_name='marry';

1 row created.

SQL> insert into banktable select 104,ref(p),40000 from person p where p.person_name='harry';

1 row created.

SQL> insert into banktable select 105,ref(p),50000 from person p where p.person_name='sunny';

1 row created.
```

```
SQL> select account_no,deref(person), balance from

    banktable;ACCOUNT_NO
-----------
    DEREF(PERSON)(PERSONID, PERSON_NAME,
--------------------------------------------------------------------------------
      DOB)BALANCE
-----------

         101
    PERSONTYPE(1,
       'sam', '24- APR-
       93')10000

         102
    PERSONTYPE(2, 'anni',
       '24- AUG-92')20000

    ACCOUNT_NO
-----------
    DEREF(PERSON)(PERSONID,
--------------------------------------------------------------------------------

      PERSON_NAME, DOB)BALANCE
-----------


         103
    PERSONTYPE(3, 'marry',
       '11- APR-92')30000
         104
    PERSONTYPE(4, 'harry',

    '05- OCT-
-----------
    92')ACCOUNT_NO

    DEREF(PERSON)(PERSONID, PERSON_NAME, DOB)
```

BALA NCE 400

00

105
PERSONTYPE(5, 'Sunny', '14-JUN-93')50000

```
SQL> select account_no,deref(person), balance from banktable;

ACCOUNT_NO
----------
DEREF(PERSON)(PERSONID, PERSON_NAME, DOB)
--------------------------------------------------------------------------------
   BALANCE
----------
       101
PERSONTYPE(1, 'sam', '24-APR-93')
     10000

       102
PERSONTYPE(2, 'anni', '24-AUG-92')
     20000

ACCOUNT_NO
----------
DEREF(PERSON)(PERSONID, PERSON_NAME, DOB)
--------------------------------------------------------------------------------
   BALANCE
----------

       103
PERSONTYPE(3, 'marry', '11-APR-92')
     30000

       104
PERSONTYPE(4, 'harry', '05-OCT-92')

ACCOUNT_NO
----------
DEREF(PERSON)(PERSONID, PERSON_NAME, DOB)
--------------------------------------------------------------------------------
   BALANCE
----------
     40000

       105
PERSONTYPE(5, 'sunny', '14-JUN-93')
     50000
```

ii) Find all names with account number

SQL> select p.person_name,b.account_no from banktable b,person p
wherep.personid=b.person.personid;

PERSON_NAME	ACCOUNT_NO
sam	101
anni	102
marry	103

harry 104
sunny 105

```
SQL> select p.person_name,b.account_no from banktable b,person p where p.personid=b.person.personid;

PERSON_NAME          ACCOUNT_NO
-------------------- ----------
sam                         101
anni                        102
marry                       103
harry                       104
sunny                       105
```

iii) Find the names with balance more than 12,000

SQL> select p.person_name, b.balance from banktable b , person p
wherep.personid=b.person.personid
and b.balance>12000;

PERSON_NAME

---

anni 20000BAL
A
marry 30000
harNryCE 40000
Sunny 50000

```
SQL> select p.person_name, b.balance from banktable b , person p where p.personid=b.person.personid
  2  and b.balance>12000;

PERSON_NAME          BALANCE
-------------------- ----------
anni                    20000
marry                   30000
harry                   40000
sunny                   50000
```

Practical No 3

CouchDB CURD Operation

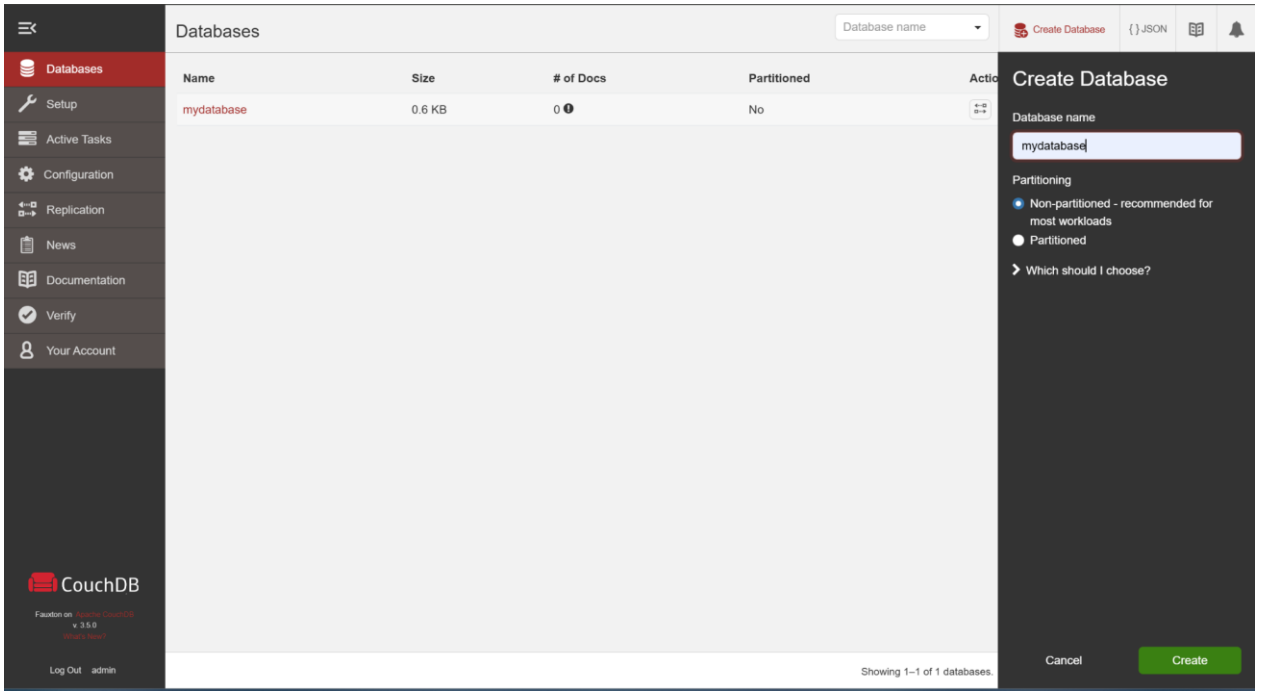
Step 1:

🛠️ Step 1: Install & Setup CouchDB

1. **Download CouchDB**
   - Apache CouchDB Download ↗
   - Install according to your OS (Windows, Linux, macOS).
2. **Run CouchDB**
   - Once installed, start CouchDB service.
   - Default URL: http://127.0.0.1:5984/_utils/ ↗ → This is the **Fauxton dashboard**.

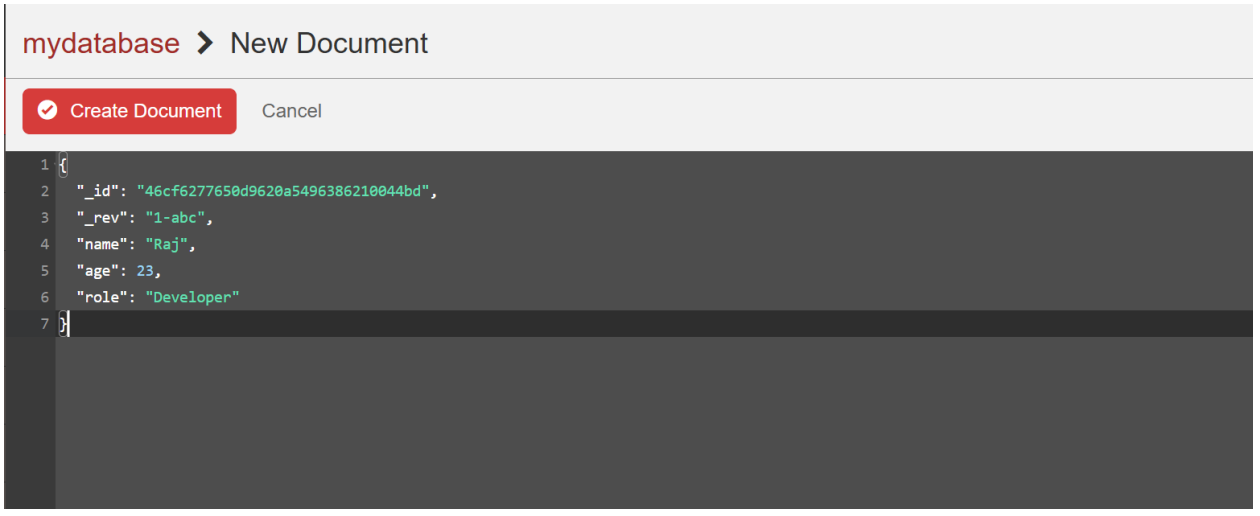
Step 2:

Create Database



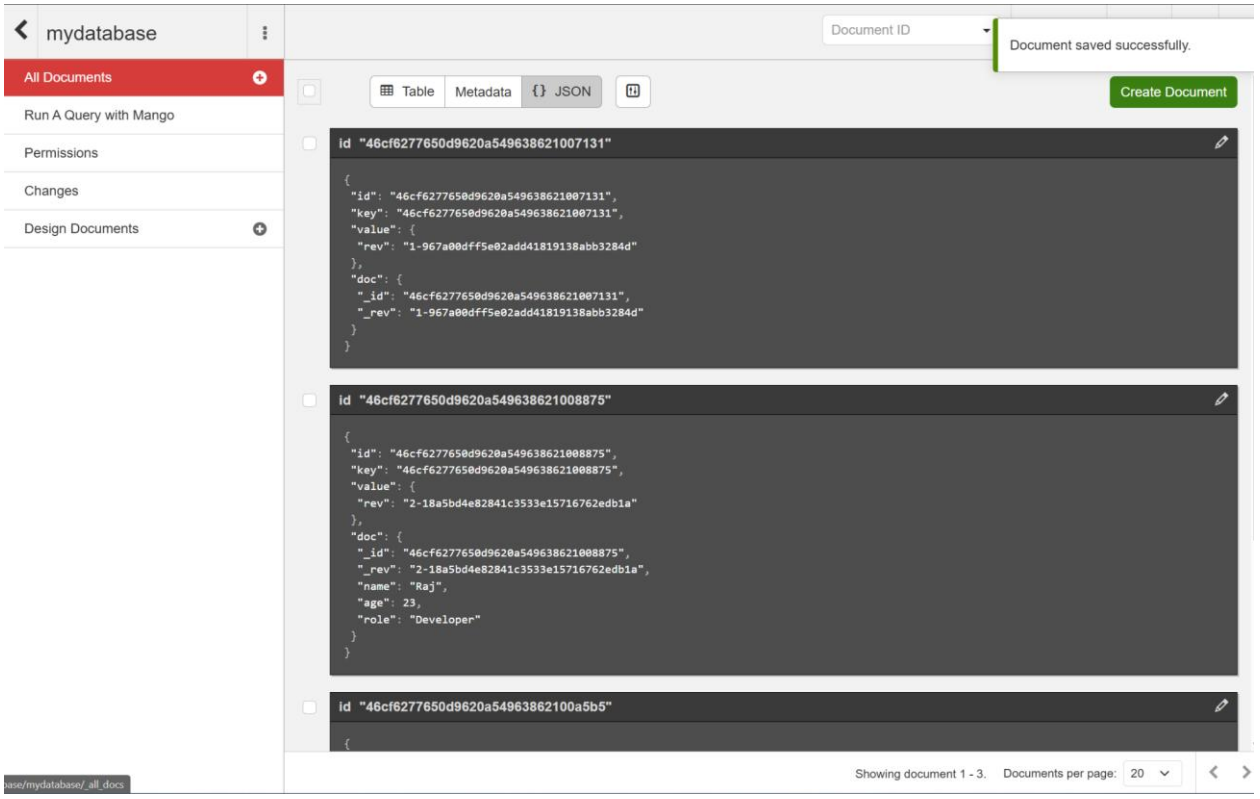
Step 3:

CREATE data in JSON format



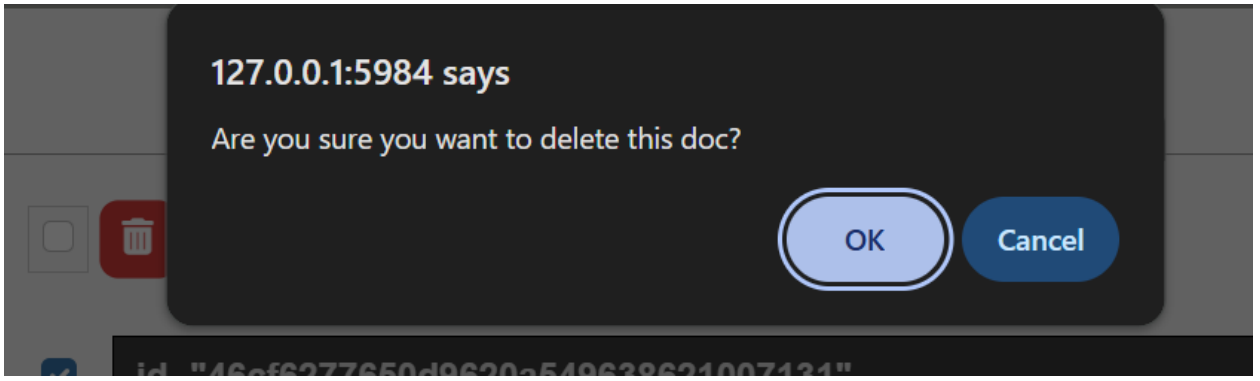
Step 4:

READ the data

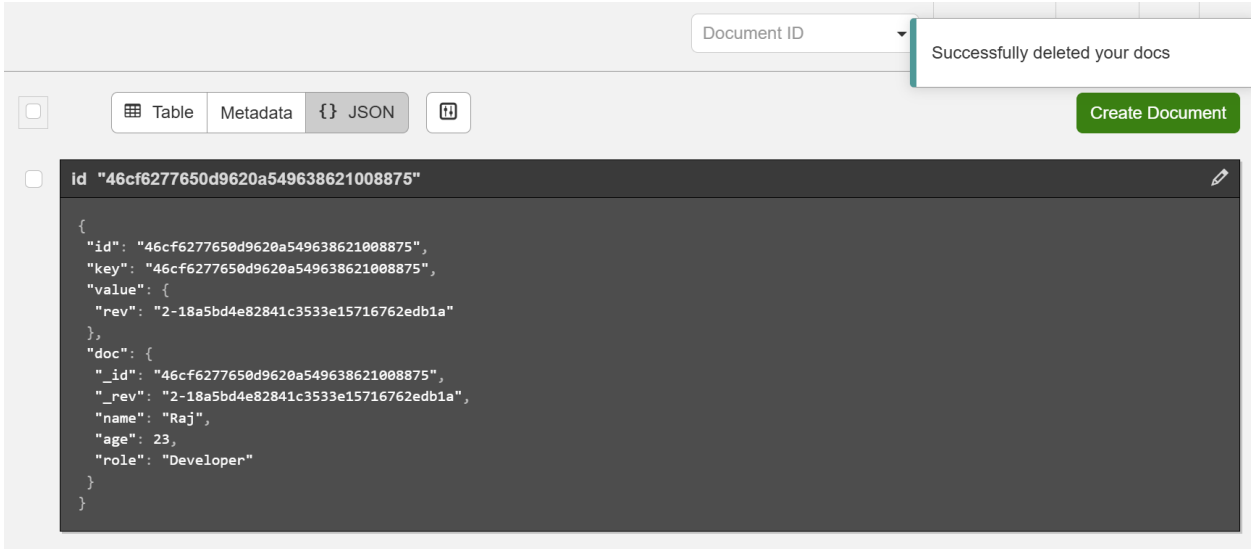


Step 5:

DELETE the data from database

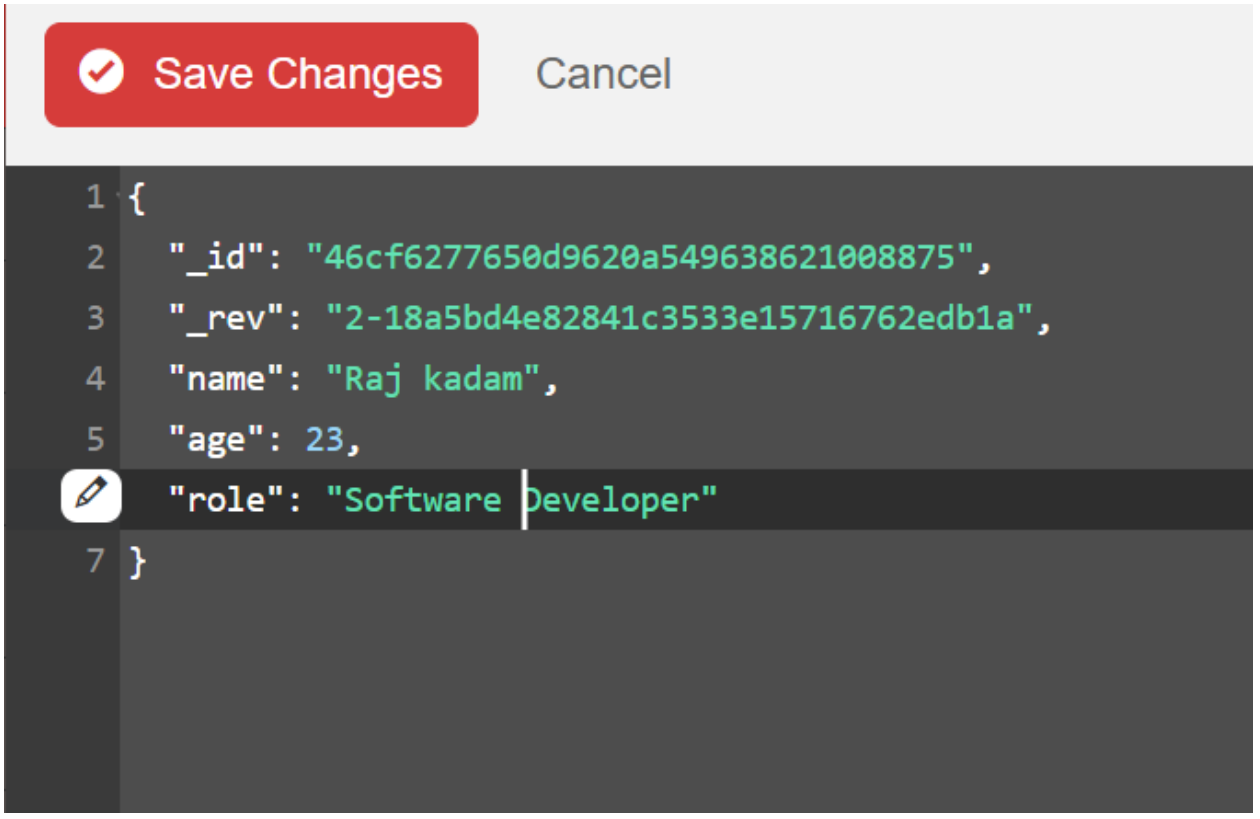


In database data has been deleted

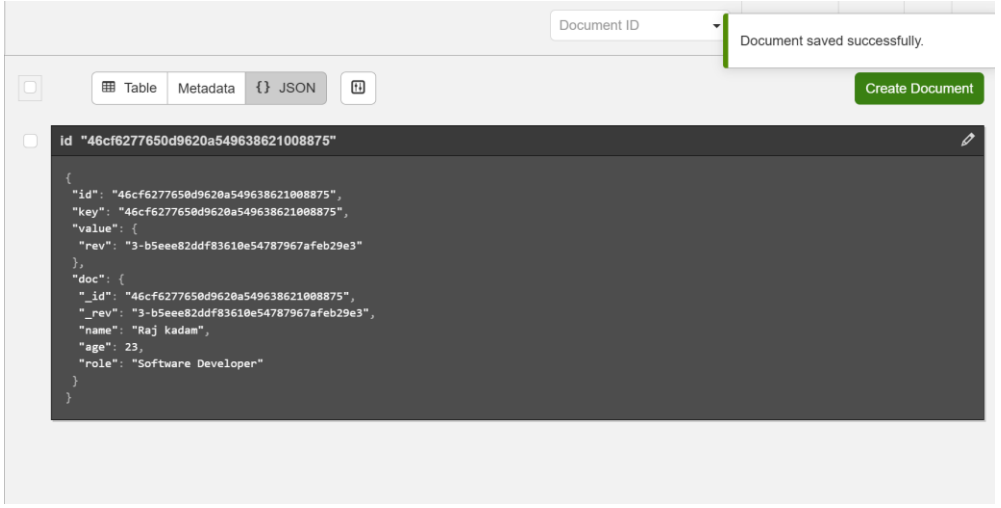


Step 6:

UPDATE the data



In database data has been updated



MongoDB CURD Operation

This document explains how to perform basic CRUD (Create, Read, Update, Delete) operations in MongoDB using the MongoDB shell (mongosh).

Step 1: Start MongoDB

1. Run `mongod` to start the MongoDB server.
2. Open `mongosh` to access the MongoDB shell.

Step 2: Create Database & Collection

Switch to or create a database:

use mydb

```
mongosh> use mydb
switched to db mydb
```

Create a collection:

db.createCollection("users")

```
mydb> db.createCollection("users")
{ ok: 1 }
```

Step 3: CRUD Operations

1. Create (Insert)

Insert one document:

db.users.insertOne({ name: "Raj", age: 23, role: "Developer" })

```
mydb> db.users.insertOne({ name: "Raj", age: 23, role: "Developer" })
{
  acknowledged: true,
  insertedId: ObjectId('68c40bfb449cbae38c228fb5')
}
```

Insert many documents:

db.users.insertMany([{ name: "Alice", age: 25 },{ name: "Bob", age: 30 }])

```
mydb> db.users.insertMany([
...   { name: "Alice", age: 25 },
...   { name: "Bob", age: 30 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68c40c01449cbae38c228fb6'),
    '1': ObjectId('68c40c01449cbae38c228fb7')
  }
}
```

2. Read (Find)

Find all documents:

db.users.find()

```
mydb> db.users.find()
[
  {
    _id: ObjectId('68c40bfb449cbae38c228fb5'),
    name: 'Raj',
    age: 23,
    role: 'Developer'
  },
  { _id: ObjectId('68c40c01449cbae38c228fb6'), name: 'Alice', age: 25 },
  { _id: ObjectId('68c40c01449cbae38c228fb7'), name: 'Bob', age: 30 }
]
```

Find one document:

db.users.findOne({ name: "Raj" })

Find with condition:

db.users.find({ age: { $gt: 24 } })

```
mydb> db.users.findOne({ name: "Raj" })
{
  _id: ObjectId('68c40bfb449cbae38c228fb5'),
  name: 'Raj',
  age: 23,
  role: 'Developer'
}
mydb> db.users.find({ age: { $gt: 24 } })
[
  { _id: ObjectId('68c40c01449cbae38c228fb6'), name: 'Alice', age: 25 },
  { _id: ObjectId('68c40c01449cbae38c228fb7'), name: 'Bob', age: 30 }
]
```

3. Update

Update one document:

db.users.updateOne({ name: "Raj" },{ $set: { role: "Senior Developer", age: 24 } })

```
mydb> db.users.updateOne(
...   { name: "Raj" },
...   { $set: { role: "Senior Developer", age: 24 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Update many documents:

db.users.updateMany({ age: { $gt: 25 } },{ $set: { status: "Experienced" } })

```
mydb> db.users.updateMany(
...   { age: { $gt: 25 } },
...   { $set: { status: "Experienced" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

4. Delete

Delete one document:

db.users.deleteOne({ name: "Bob" })

```
mydb> db.users.deleteOne({ name: "Bob" })
{ acknowledged: true, deletedCount: 1 }
```

Delete many documents:

db.users.deleteMany({ age: { $lt: 24 } })

Step 4: Verify Data

Drop collection:

db.users.drop()

```
mydb> db.users.drop()
true
```

Drop database:

db.dropDatabase()

```
mydb> db.dropDatabase()
{ ok: 1, dropped: 'mydb' }
mydb>
```

Practical No 4

Aim: CRUD Operations using Redis and Apache Cassandra

1. Redis CRUD Operations

- Create & Read:

```
$ redis-cli
127.0.0.1:6379> SET user:1 "Alice"
OK
127.0.0.1:6379> GET user:1
"Alice"
```

- Update:

```
127.0.0.1:6379> SET user:1 "Alice Johnson"
OK
127.0.0.1:6379> GET user:1
"Alice Johnson"
```

- Delete:

```
127.0.0.1:6379> DEL user:1
(integer) 1
127.0.0.1:6379> GET user:1
(nil)
```

2. Apache Cassandra CRUD Operations

- Create & Insert:

```
$ cqlsh
cqlsh> CREATE KEYSPACE demo WITH replication = {'class':'SimpleStrategy','replication_factor':1};
cqlsh> USE demo;
cqlsh:demo> CREATE TABLE users(id int PRIMARY KEY, name text);
cqlsh:demo> INSERT INTO users(id, name) VALUES(1, 'Alice');
```

- Read:

```
cqlsh:demo> SELECT * FROM users;
 id | name
----+-------
  1 | Alice
```

- Update:

```
cqlsh:demo> UPDATE users SET name = 'Alice Johnson' WHERE id = 1;
cqlsh:demo> SELECT * FROM users;
 id | name
----+--------------
  1 | Alice Johnson
```

- Delete:

```
cqlsh:demo> DELETE FROM users WHERE id = 1;
cqlsh:demo> SELECT * FROM users;
 id | name
----+------
```