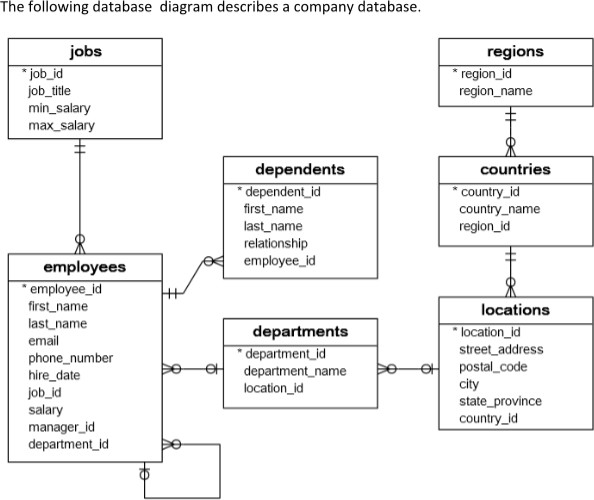
**COMPANY DATABASE**



The company database has seven tables:

1. The employees table stores the data of employees.
2. The jobs table stores the job data including job title and salary range.
3. The departments table stores department data.
4. The dependents table stores the employee’s dependents.
5. The locations table stores the location of the departments of the company.

6.The countries table stores the data of countries where the company is doing business.

7.The regions table stores the data of regions such as Asia, Europe, America, and the Middle East and Africa. The countries are grouped into regions.

ANSWER

* create table jobs(job\_id number(5) primary key,job\_title varchar2(25),min\_salary number(7,2),max\_salary number(7,2));

SQL> desc jobs;

Name Null? Type

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

JOB\_ID NOT NULL NUMBER(5)

JOB\_TITLE VARCHAR2(25)

MIN\_SALARY NUMBER(7,2)

MAX\_SALARY NUMBER(7,2)

* create table departments(department\_id number(5) primary key,department\_name

varchar2(20));

SQL> desc departments;

Name Null? Type

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

DEPARTMENT\_ID NOT NULL NUMBER(5)

DEPARTMENT\_NAME VARCHAR2(20)

* create table employees(employee\_id number(5) primary key,first\_name varchar2(25),last\_name varchar2(25),email varchar2(30),phone\_number number(10),hire\_date date,job\_id number(5) REFERENCES jobs,salary number(7,2),manager\_id number(5),department\_id number(5) REFERENCES departments);

SQL> desc employees;

Name Null? Type

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

EMPLOYEE\_ID NOT NULL NUMBER(5)

FIRST\_NAME VARCHAR2(25)

LAST\_NAME VARCHAR2(25)

EMAIL VARCHAR2(30)

PHONE\_NUMBER NUMBER(10)

HIRE\_DATE DATE

JOB\_ID NUMBER(5)

SALARY NUMBER(7,2)

MANAGER\_ID NUMBER(5)

DEPARTMENT\_ID NUMBER(5)

* create table regions(region\_id number(5) primary key,region\_name varchar2(30));

SQL> desc regions;

Name Null? Type

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

REGION\_ID NOT NULL NUMBER(5)

REGION\_NAME VARCHAR2(30)

* create table countries(country\_id number(5) primary key,country\_name varchar2(25),region\_id number(5) REFERENCES regions);

SQL> desc countries;

Name Null? Type

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

COUNTRY\_ID NOT NULL NUMBER(5)

COUNTRY\_NAME VARCHAR2(25)

REGION\_ID NUMBER(5)

* create table locations(location\_id number(5) primary key,street\_address varchar2(35),postal\_code number(6),city varchar2(20),state\_province varchar2(20),country\_id number(5) REFERENCES countries);

SQL> desc locations;

Name Null? Type

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

LOCATION\_ID NOT NULL NUMBER(5)

STREET\_ADDRESS VARCHAR2(35)

POSTAL\_CODE NUMBER(6)

CITY VARCHAR2(20)

STATE\_PROVINCE VARCHAR2(20)

COUNTRY\_ID NUMBER(5)

* create table dependants(dependant\_id NUMBER(5) PRIMARY KEY,first\_name VARCHAR2(25),last\_name VARCHAR2(25),relationship VARCHAR2(15),employee\_id NUMBER(5) REFERENCES employees);

SQL> desc dependants;

Name Null? Type

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

DEPENDANT\_ID NOT NULL NUMBER(5)

FIRST\_NAME VARCHAR2(25)

LAST\_NAME VARCHAR2(25)

RELATIONSHIP VARCHAR2(15)

EMPLOYEE\_ID NUMBER(5)

**SAILOR DATABASE**

Create the following tables and insert values accordingly

* SAILORS (SID:INTEGER, SNAME:STRING, RATING:INTEGER, AGE:REAL)
* BOATS (BID:INTEGER, BNAME:STRING, COLOR:STRING)

.• RESERVES (SID:INTEGER, BID:INTEGER, DAY:DATE) **SAILORS**

SID SNAME RATING AGE



| 22 Dustin | 7 |  | 45 |
| --- | --- | --- | --- |
| 29 Brutus | 1 |  | 33 |
| 31 Lubber | 8 |  | 55.5 |
| 32 Andy | 8 |  | 25.5 |
| 58 Rusty | 10 |  | 35 |
| 64 Horataio | 7 |  | 35 |
| 71 Zorba | 10 |  | 16 |
| 74 Horataio | 9 |  | 35 |
| 85 Art | 3 |  | 25.5 |
| 95 Bob **BOATS**  BID BNAME | 3 | COLOR | 63.5 |



| 101 | Interlake | blue |
| --- | --- | --- |
| 102 | Interlake | red |
| 103 | Clipper | green |
| 104 | Marine | red |

**RESERVES**

SID BID DAY



| 22 | 101 | 10-OCT-98 |
| --- | --- | --- |
| 22 | 102 | 10-OCT-98 |
| 22 | 103 | 08-OCT-98 |
| 22 | 104 | 07-OCT-98 |
| 31 | 102 | 0-NOV-98 |
| 31 | 103 | 06-NOV-98 |
| 31 | 104 | 12-NOV-98 |
| 64 | 101 | 05-SEP-98 |
| 64 | 102 | 08-SEP-98 |
| 74 | 103 | 08-SEP-98 |
|  |  |  |

DML



1. Show all records of Sailors

SQL>> SELECT \* FROM SAILORS;

SID SNAME RATING AGE

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

22 Dustin 7 45

29 Brutus 1 33

31 Lubber 8 55.5

32 Andy 8 25.5

58 Rusty 10 35

64 Horataio 7 35

71 Zorba 16 10

76 Bob 64 9

8 rows selected.

2. Show all records of Boats

SQL >> SELECT \* FROM BOATS;

BID BNAME COLOR

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

101 InterLake blue

102 InterLake red

103 Clipper green

104 Marine red

3. Show all records of Reserves

SQL >> SELECT \* FROM RESERVES;

SID BID DAY

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

22 101 10-OCT-98

22 102 10-OCT-98

22 103 08-OCT-98

22 104 07-OCT-98

31 102 10-NOV-98

31 103 06-NOV-98

31 104 12-NOV-98

64 101 05-SEP-98

64 102 08-SEP-98

76 103 08-SEP-98

10 rows selected.

4. Find the names and ages of all sailors.

SQL >> SELECT SNAME AS NAME,AGE FROM SAILORS;

NAME AGE

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

Dustin 45

Brutus 33

Lubber 55.5

Andy 25.5

Rusty 35

Horataio 35

Zorba 10

Bob 9

8 rows selected.

5. Find all sailors with a rating above 8.

SQL >> SELECT \* FROM SAILORS WHERE RATING > 8;

SID SNAME RATING AGE

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

58 Rusty 10 35

71 Zorba 16 10

76 Bob 64 9

6. Find sailors name with a rating above 7 & age above 25.

SQL >> SELECT SNAME AS "NAME" FROM SAILORS WHERE RATING > 7 AND AGE > 25;

NAME

----------

Lubber

Andy

Rusty

7. Display all the names & colors of the boats.

SQL >> SELECT BNAME,COLOR FROM BOATS;

BNAME COLOR

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

InterLake blue

InterLake red

Clipper green

Marine red

8. Find all the boats with Red color.

SQL >> SELECT \* FROM BOATS WHERE COLOR = 'red';

BID BNAME COLOR

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

102 InterLake red

104 Marine red

9. Find the names of sailors who have reserved boat number 103.

SQL >> SELECT S.SNAME FROM SAILORS S,RESERVES R WHERE S.SID=R.SID AND R.BID=103;

SNAME

----------

Dustin

Lubber

Bob

10. Find the names of sailors who have reserved Red boat.

SQL >> SELECT DISTINCT S.SNAME FROM SAILORS S,RESERVES R,BOATS B WHERE S.SID=R.SID AND R.BID=B.BID AND B.COLOR='red';

SNAME

----------

Lubber

Dustin

Horataio

11. Find the colors of boats reserved by Lubber.

SQL >> SELECT B.COLOR FROM SAILORS S,RESERVES R,BOATS B WHERE S.SID=R.SID AND R.BID=B.BID AND S.SNAME='Lubber';

COLOR

----------

red

green

red

12. Find names of the sailors who have reserved at least one boat.

SQL >> SELECT S.SNAME FROM SAILORS S,RESERVES R WHERE S.SID=R.SID;

SNAME

----------

Dustin

Dustin

Dustin

Dustin

Lubber

Lubber

Lubber

Horataio

Horataio

Bob

10 rows selected.

13. Find names of the sailors who have reserved two different boats.

SQL >> SELECT S.SNAME FROM SAILORS S WHERE S.SID IN (SELECT S.SID FROM SAILORS S, RESERVES R,BOATS B WHERE S.SID=R.SID AND B.BID =R.BID GROUP BY S.SID HAVING COUNT (B.BID) >1 );

SNAME

----------

Dustin

Lubber

Horataio

**Aggregate Functions**

1. To count number SIDs of sailors in Sailors table.

SQL >> SELECT COUNT(SID) FROM SAILORS;

COUNT(SID)

----------

8

1. To count numbers of boats booked in Reserves table.

SQL >> SELECT COUNT( DISTINCT BID) FROM RESERVES;

COUNT(DISTINCTBID)

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

4

1. To count number of Boats in Boats table.

SQL >> SELECT COUNT(BID) FROM BOATS;

COUNT(BID)

----------

4

1. To find sum of rating from Sailor.

SQL >> SELECT SUM(RATING) FROM SAILORS;

SUM(RATING)

-----------

121

1. To find sum of distinct age of Sailors.

SQL >> SELECT SUM(DISTINCT AGE) FROM SAILORS;

SUM(DISTINCTAGE)

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

213

6.To find average of distinct age of Sailors.

SQL >> SELECT AVG(DISTINCT AGE) FROM SAILORS;

AVG(DISTINCTAGE)

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

30.4285714

7.To find age of Oldest Sailor.

SQL >> SELECT MAX(AGE) FROM SAILORS;

MAX(AGE)

----------

55.5

8.To find age of Youngest Sailor.

SQL >> SELECT MIN(AGE) FROM SAILORS;

MIN(AGE)

----------

9

9.Find the average age of sailors with a rating of 10.

SQL >> SELECT AVG(AGE) FROM SAILORS WHERE RATING = 10;

AVG(AGE)

----------

35

10.Count the number of different sailor names.

SQL >> SELECT COUNT(DISTINCT SNAME) FROM SAILORS;

COUNT(DISTINCTSNAME)

-----------

8

11.Find the name and age of the oldest sailor.

SQL>> SELECT SNAME,AGE FROM SAILORS WHERE AGE=(SELECT MAX(AGE) FROM

SAILORS);

SNAME AGE

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

Lubber 55.5

12.Count the number of Sailors.

SQL>> SELECT COUNT(\*) AS "NO\_SAILORS" FROM SAILORS;

NO\_SAILORS

----------

8

13.Find the names of sailors who are older than the oldest sailor with a rating of 10.

SQL >> SELECT SNAME FROM SAILORS WHERE AGE>(SELECT MAX(AGE) FROM SAILORS WHERE RATING = 10);

SNAME

----------

Dustin

Lubber

**ORDER BY**

1.Display the details of sailors in alphabetical order .

SQL >> SELECT \* FROM SAILORS ORDER BY SNAME;

SID SNAME RATING AGE

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

32 Andy 8 25.5

76 Bob 64 9

29 Brutus 1 33

22 Dustin 7 45

64 Horataio 7 35

31 Lubber 8 55.5

58 Rusty 10 35

71 Zorba 16 10

8 rows selected.

2.Display the details of sailors in de alphabetical order.

SQL >> SELECT \* FROM SAILORS ORDER BY SNAME DESC;

SID SNAME RATING AGE

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

71 Zorba 16 10

58 Rusty 10 35

31 Lubber 8 55.5

64 Horataio 7 35

22 Dustin 7 45

29 Brutus 1 33

76 Bob 64 9

32 Andy 8 25.5

8 rows selected.

3.Display the details of sailors in the form of ratings (highest to lowest).

SQL >> SELECT \* FROM SAILORS ORDER BY RATING DESC;

SID SNAME RATING AGE

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

76 Bob 64 9

71 Zorba 16 10

58 Rusty 10 35

31 Lubber 8 55.5

32 Andy 8 25.5

64 Horataio 7 35

22 Dustin 7 45

29 Brutus 1 33

8 rows selected.

4.Display all the sailors according to their ages.

SQL >>SELECT \* FROM SAILORS ORDER BY AGE;

SID SNAME RATING AGE

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

76 Bob 64 9

71 Zorba 16 10

32 Andy 8 25.5

29 Brutus 1 33

58 Rusty 10 35

64 Horataio 7 35

22 Dustin 7 45

31 Lubber 8 55.5

8 rows selected.

5.Displays all the sailors according to rating, if rating is same then sort according to age.

SQL >> SELECT \* FROM SAILORS ORDER BY RATING,AGE;

SID SNAME RATING AGE

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

29 Brutus 1 33

64 Horataio 7 35

22 Dustin 7 45

32 Andy 8 25.5

31 Lubber 8 55.5

58 Rusty 10 35

71 Zorba 16 10

76 Bob 64 9

8 rows selected.

6.Displays all the sailors according to rating (Lower Rating First), if rating is same then sort according to age (Younger First).

SQL >> SELECT \* FROM SAILORS ORDER BY RATING,AGE;

SID SNAME RATING AGE

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

29 Brutus 1 33

64 Horataio 7 35

22 Dustin 7 45

32 Andy 8 25.5

31 Lubber 8 55.5

58 Rusty 10 35

71 Zorba 16 10

76 Bob 64 9

8 rows selected.

**GROUP BY**

1. Find the age of the youngest sailor for each rating level.

SQL >> SELECT RATING,MIN(AGE) FROM SAILORS GROUP BY RATING;

RATING MIN(AGE)

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

1 33

8 25.5

7 35

64 9

10 35

16 10

6 rows selected.

**HAVING CLAUSE**

1.Find the age of youngest sailor with age >= 18 for each rating with at least 2 such sailors.

SQL >> SELECT RATING,MIN(AGE) FROM SAILORS WHERE AGE >=18 GROUP BY RATING HAVING COUNT(\*)>1;

RATING MIN(AGE)

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

8 25.5

7 35

2.For each red boat; find the number of reservations for this boat.

SQL >> SELECT B.BID,COUNT(\*) AS "RESERVATION" FROM BOATS B,RESERVES R WHERE B.BID=R.BID AND B.COLOR = 'red' GROUP BY B.BID;

BID RESERVATION

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

102 3

104 2

3.Find the average age of sailors for each rating level that has at least two sailors.

SQL >> SELECT RATING,AVG(AGE) FROM SAILORS GROUP BY RATING HAVING COUNT(\*)>1;

RATING AVG(AGE)

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

8 40.5

7 40

4:Find those ratings for which the average age of sailors is the minimum overall ratings.

SELECT S.rating, AVG(S.age) AS avgage FROM Sailors S where s.age = (SELECT MIN (age) FROM Sailors) GROUP BY S.rating;

RATING AVGAGE

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

64 9

## PL / SQL PROGRAMMING

1.Check whether number is prime or not.

set serveroutput on;

declare

i number:=2; f number:=1;

n number:=&n; begin

for i in 2..n/2 loop

if n mod i=0 then

f:=0;

exit; end if;

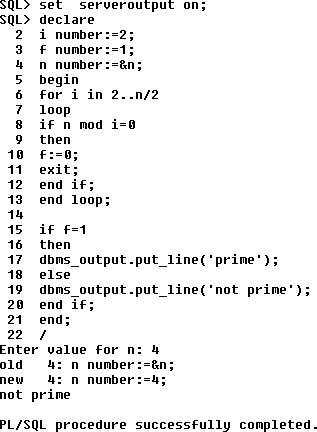
end loop;

if f=1 then

dbms\_output.put\_line('prime'); else

dbms\_output.put\_line('not prime'); end if;

end;

/

2. Find the Factorial of a number.

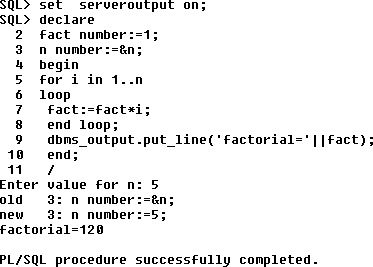
set serveroutput on; declare

fact number:=1; n number:=&n; begin

for i in 1..n loop fact:=fact\*i; end loop;

dbms\_output.put\_line('factorial='||fact); end;

/



### FUNCTION

1. Write a Function to get the sum of 2 numbers.

create or replace function sum1(a in number,b in number) return number

as

s number;

begin

s:=a+b;

return s;

end;

/

declare

a number;

b number;

c number;

begin

a:=&a;

b:=&b;

c:=sum1(a,b);

dbms\_output.put\_line(chr(10)||'Sum is '||c);

end;

/

a=5

b=6

sum is 11

2.Write a function to print Weekday in letters.

CREATE OR REPLACE FUNCTION get\_weekday\_name(p\_date IN DATE)

RETURN VARCHAR2

IS

v\_weekday VARCHAR2(20);

BEGIN

-- Use TO\_CHAR to get the weekday name

SELECT TO\_CHAR(p\_date, 'Day', 'NLS\_DATE\_LANGUAGE=English') INTO v\_weekday FROM DUAL;

RETURN v\_weekday;

EXCEPTION

WHEN OTHERS THEN

-- Handle exceptions, if necessary

RAISE;

END;

/

### PROCEDURE

1.Develop a procedure to perform Banking operations.

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE deposit(NO IN NUMBER,amount IN NUMBER(7,2)) IS

BAL NUMBER;

BEGIN

SELECT BALANCE INTO BAL FROM ACCOUNT WHERE ACCOUNT\_NO=NO;

UPDATE ACCOUNT SET BALANCE:=BAL+amount WHERE ACCOUNT\_NO=NO;

dbms\_output.put\_line('Amount deposited');

END;

CREATE OR REPLACE PROCEDURE withdrawal(NO IN NUMBER,amount IN NUMBER(7,2)) IS

BAL NUMBER;

BEGIN

SELECT BALANCE INTO BAL FROM ACCOUNT WHERE ACCOUNT\_NO=NO;

IF BAL<amount THEN

dbms\_output.put\_line('Insufficient balance!!');

dbms\_output.put\_line('Withdrawal not possible.');

ELSE

UPDATE ACCOUNT SET BALANCE:=BAL+amount WHERE ACCOUNT\_NO=NO;

dbms\_output.put\_line('Amount withdrawan');

END IF;

END;

CREATE OR REPLACE PROCEDURE display(NO IN NUMBER) IS

ACC\_NO NUMBER;

NAME VARCHAR2(20);

BAL NUMBER;

BEGIN

SELECT ACCOUNT\_NO INTO ACC\_NO,ACCOUNT\_HOLDER INTO NAME,BALANCE INTO BAL FROM ACCOUNT WHERE ACCOUNT\_NO=NO;

dbms\_output.put\_line('....Bank account details....');

dbms\_output.put\_line('Account No : '||ACC\_NO);

END;

BEGIN

WHILE choice<=3

LOOP

dbms\_output.put\_line('Banking');

dbms\_output.put\_line('1. Deposit');

dbms\_output.put\_line('2. Withdrawal');

dbms\_output.put\_line('3. Account details');

choice:=&choice;

IF choice=1 THEN

amount:=&amount;

deposit(amount);

ELSIF choice=2 THEN

amount:=&amount;

withdrawal(amount);

ELSIF choice=3 THEN

display();

ELSE

dbms\_output.put\_line('Invalid choice!!');

END IF

END LOOP

END;

/

### CURSOR

1.Create a cursor to Display sid and rating of all sailors.

DECLARE

v\_sid sailors.sid%TYPE;

v\_rating sailors.rating%TYPE;

CURSOR sailor\_cursor IS

SELECT sid, rating

FROM sailors;

BEGIN

-- Open the cursor

OPEN sailor\_cursor;

LOOP

FETCH sailor\_cursor INTO v\_sid, v\_rating;

EXIT WHEN sailor\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Sailor ID: ' || v\_sid || ', Rating: ' || v\_rating);

END LOOP;

CLOSE sailor\_cursor;

EXCEPTION

WHEN OTHERS THEN

-- Handle exceptions, if any

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/

### 

### TRIGGERS

1. Write a trigger to restrict deletion operation on Professor table.

SET SERVEROUTPUT ON

CREATE OR REPLACE TRIGGER restrict\_del BEFORE DELETE ON Professor

FOR EACH ROW

BEGIN

RAISE\_APPLICATION\_ERROR(-20002,'Deletion cannot be performed on Professor table.');

END;

/

Trigger created.

SQL > SELECT \* FROM Professor;

PID PNAME SALARY

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

111 ATHIRA 70000

2.Create a trigger to manage the counts of male and female students.

CREATE TABLE STUDENT\_C(ROLL NUMBER(5) PRIMARY KEY,NAME VARCHAR(20),GENDER CHAR(1));

Table created.

CREATE TABLE COUNT(FCOUNT INTEGER,MCOUNT INTEGER);

Table created.

SET SERVEROUTPUT ON

CREATE OR REPLACE TRIGGER count\_stud AFTER INSERT ON STUDENT\_C

FOR EACH ROW

DECLARE cnt int;

BEGIN

IF :NEW.GENDER='F' THEN

SELECT FCOUNT INTO cnt FROM COUNT;

IF cnt=0 THEN

INSERT INTO COUNT(FCOUNT) VALUES(1);

ELSE

INSERT INTO COUNT(FCOUNT) VALUES(cnt+1);

END IF;

END IF;

IF :NEW.GENDER='M' THEN

SELECT MCOUNT INTO cnt FROM COUNT;

IF cnt=0 THEN

INSERT INTO COUNT(MCOUNT) VALUES(1);

ELSE

INSERT INTO COUNT(MCOUNT) VALUES(cnt+1);

END IF;

END IF;

END;

/

Trigger created.

3.Create a trigger for gold price management.

Trigger : To create a trigger for gold price management

gold\_price : date , price

manage : gold\_value,date

create table g\_price(g\_date date,price number(9,2));

Table created.

create table g\_details(gold\_value number(3),g\_date date);

table created.

insert into g\_details values(50,SYSDATE);

1 row created.

insert into g\_price values(SYSDATE,5000);

1 row created.

CREATE TABLE price\_BACKLOG(g\_date date,old\_price number(9,2),new\_price number(9,2));

Table created.

SET SERVEROUTPUT ON

CREATE OR REPLACE TRIGGER gprice\_TR AFTER UPDATE ON g\_price

FOR EACH ROW

BEGIN

INSERT INTO price\_BACKLOG VALUES(:OLD.g\_date,:OLD.price,:NEW.price);

END;

/

Trigger created.

UPDATE g\_price SET price=1112 where g\_date >= '20-JUN-2023';

1 row updated.

SELECT \* FROM price\_BACKLOG;

G\_DATE OLD\_PRICE NEW\_PRICE

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

20-JUN-23 88888 88888

20-JUN-23 88888 776677

20-JUN-23 776677 1111

### EXCEPTIONS

1. Create a plsql program to raise an exception on an exceptional case (Divide by zero Exception).

SET SERVEROUTPUT ON

DECLARE

A INTEGER:=&A;

B INTEGER:=&B;

C INTEGER;

BEGIN

IF(B=0)THEN

RAISE\_APPLICATION\_ERROR(-20001,'DIVISION BY ZERO');

ELSE

C:=A/B;

DBMS\_OUTPUT.PUT\_LINE('RESULT IS : '||C);

END IF;

END;

/

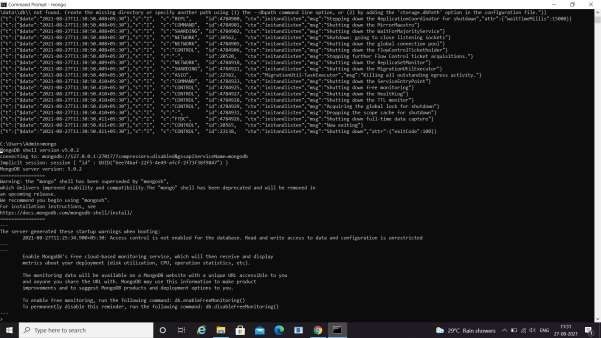
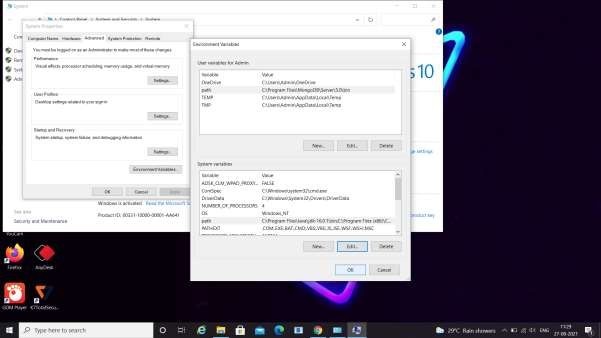
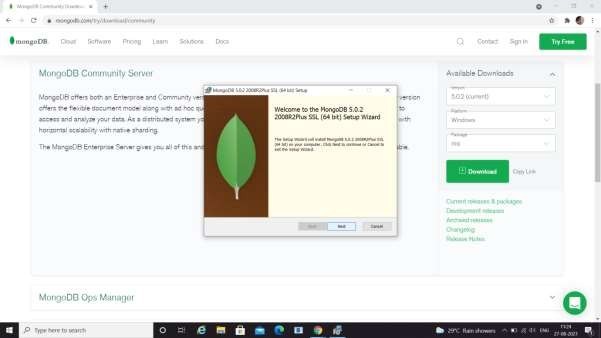
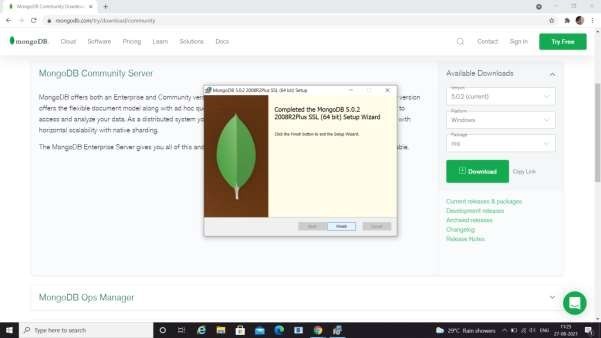
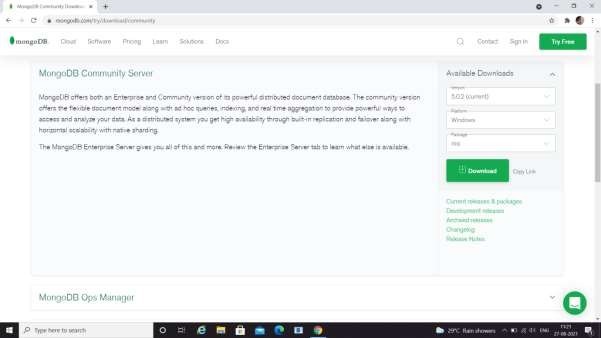
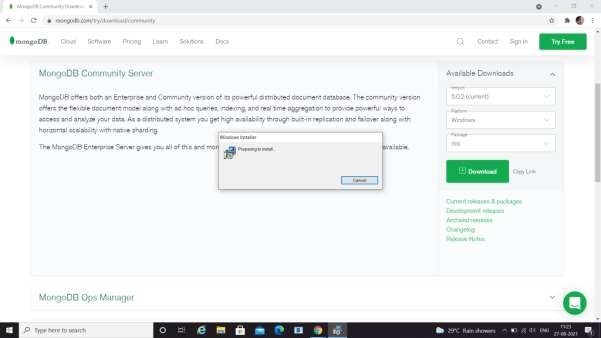
a=9

b=0

9/0

RAISE\_APPLICATION\_ERROR.

# MongoDB Installation:



**MONGODB PROGRAMS**

### Create a database named Employee. Create a collection named empDetails You can use any of the fields Name, Age ,e\_mail, phone,salary

1. **Insert 5 documents in it using the different insert() methods and**
   * use EMP

switched to db EMP

>db EMP

* + db.createCollection("empDetails")

{ "ok" : 1 }

* + show collections empDetails

>[db.empDetails.insert({Name:"Mohan",Age:30,Email:"mohan@gmail.com",Salary:5000](mailto:mohan@gmail.com)

})

WriteResult({ "nInserted" : 1 })

>db.empDetails.insert({Name:"Raju",Age:35,Email:"[raju@gmail.com](mailto:raju@gmail.com)",Salary:7000})

WriteResult({ "nInserted" : 1 })

>db.empDetails.insert({Name:"Bhuvan",Age:25,Email:"[bhuvan@gmail.com](mailto:bhuvan@gmail.com)",Salary:1000 0})

WriteResult({ "nInserted" : 1 })

>db.empDetails.insert({Name:"Meera",Age:27,Email:"[meera@gmail.com](mailto:meera@gmail.com)",salary:9000})

WriteResult({ "nInserted" : 1 })

>db.empDetails.insert({Name:"Maya",Age:28,Email:"[maya@gmail.com](mailto:maya@gmail.com)",salary:15000})

WriteResult({ "nInserted" : 1 })

* + db.empDetails.count()

5

### Find the details of employee whose name is mohan

* db.empDetails.findOne({Name:"Mohan"})

{

"\_id" : ObjectId("611283a5ba6fd56e242ddb3a"), "Name" : "Mohan",

"Age" : 30,

"Email" :["mohan@gmail.com",](mailto:mohan@gmail.com)

"Salary": 5000

}

### Fetch the documents of employees whose salary &gt;=5000

* db.empDetails.find({$and:[{salary:{$gte:5000}}]})

{ "\_id" : ObjectId("613726aa40cbedb84b494e0d"), "Name" : "Mohan", "Age" : 30, "Email":["moham@gmail.com",](mailto:moham@gmail.com) "salary" : 5000 }

{ "\_id" : ObjectId("613726d340cbedb84b494e0e"), "Name" : "Raju", "Age" : 35,

"Email" : ["raju@gmail.com",](mailto:raju@gmail.com) "salary" : 7000 }

{ "\_id" : ObjectId("613726fc40cbedb84b494e0f"), "Name" : "Bhuvan", "Age" : 25,

"Email" : ["bhuvan@gmail.com",](mailto:bhuvan@gmail.com) "salary" : 10000 }

{ "\_id" : ObjectId("6137272f40cbedb84b494e10"), "Name" : "Meera", "Age" : 27,

"Email" : ["meera@gmail.com",](mailto:meera@gmail.com) "salary" : 9000 }

{ "\_id" : ObjectId("6137275040cbedb84b494e11"), "Name" : "Maya", "Age" :28, "Email" : ["maya@gmail.com", "salary"](mailto:maya@gmail.com) : 15000 }

### Find the documents of employees whose name starts with letter r

**>**db.empDetails.find({$or:[{"Name":/^R/}]});

{ "\_id" : ObjectId("613726d340cbedb84b494e0e"), "Name" : "Raju", "Age" : 35,

"Email" : ["raju@gmail.com",](mailto:raju@gmail.com) "salary" : 7000 }

### Find the documents of employees whose name is not in mohan,raju,bhuvan

>db.empDetails.find({"Name":{$not:{$in:["Mohan","Raju","Bhuvan"]}}}).pretty()

{

"\_id" : ObjectId("6137272f40cbedb84b494e10"), "Name" : "Meera",

"Age" : 27,

"Email":["meera@gmail.com",](mailto:meera@gmail.com)"salary" : 9000

}

{

"\_id" :ObjectId("6137275040cbedb84b494e11"), "Name" : "Maya",

"Age" : 28,

"Email":["maya@gmail.com",](mailto:maya@gmail.com)"salary" : 15000

}

### Find the documents of employees whose names are mohan , raju, bhuvan

>db.empDetails.find({"Name":{$in:["Mohan","Raju","Bhuvan"]}}).pretty()

{

"\_id" : ObjectId("613726aa40cbedb84b494e0d"),

"Name": "Mohan",

"Age" : 30,

"Email":["moham@gmail.com",](mailto:moham@gmail.com)"salary" :5000

}

{

"\_id" : ObjectId("613726d340cbedb84b494e0e"), "Name" : "Raju",

"Age" : 35,

"Email":["raju@gmail.com",](mailto:raju@gmail.com)

"salary" : 7000

}

{

"\_id" : ObjectId("613726fc40cbedb84b494e0f"), "Name" : "Bhuvan",

"Age" : 25,

"Email" : ["bhuvan@gmail.com",](mailto:bhuvan@gmail.com)

"salary" : 10000

}

### Retrieve the details of employees whose age is less than 30. Display only the fieldsname, salary

* + db.empDetails.find({Age:{$lt:30}},{Name:1,salary:1})

{ "\_id" : ObjectId("613726fc40cbedb84b494e0f"), "Name" : "Bhuvan", "salary" : 10000 }

{ "\_id" : ObjectId("6137272f40cbedb84b494e10"), "Name" : "Meera", "salary" : 9000 }

{ "\_id" : ObjectId("6137275040cbedb84b494e11"), "Name" : "Maya", "salary" : 15000 }

### Find the details of employees whose salary is grea5000 and age is &lt; 30

* + db.empDetails.find({$and:[{salary:{$gt:5000}},{Age:{$lt:30}}]})

{ "\_id" : ObjectId("613726fc40cbedb84b494e0f"), "Name" : "Bhuvan", "Age" : 25, "Email"

:["bhuvan@gmail.com",](mailto:bhuvan@gmail.com) "salary" : 10000 }

{ "\_id" : ObjectId("6137272f40cbedb84b494e10"), "Name" : "Meera", "Age" : 27, "Email"

:["meera@gmail.com",](mailto:meera@gmail.com) "salary" : 9000 }

{ "\_id" : ObjectId("6137275040cbedb84b494e11"), "Name" : "Maya", "Age" : 28, "Email" : ["maya@gmail.com",](mailto:maya@gmail.com) "salary" : 15000 }

### Update the e-mail of employee whose name is mohan // findOneAndUpdate()

>db.empDetails.updateOne({Name:'Mohan'},{$set:{Email:'mohan12345@gmail.com'}}

)

| { "acknowledged" : | true, | "matchedCount" : | 1, |
| --- | --- | --- | --- |
| "modifiedCount" : 1 | } |  |  |

>db.empDetails.findOneAndUpdate({Name:'Mohan'},{$set:{Email:'mohan12345@gma il.com'}})

{

"\_id" : ObjectId("613726aa40cbedb84b494e0d"),"Name"

: "Mohan",

"Age" : 30,

"Email" : ["mohan12345@gmail.com",](mailto:mohan12345@gmail.com)"salary" : 5000

}

### Delete all the documents of employees whose age>56

* + db.empDetails.find({Age:{$gt:56}})
  + db.empDetails.deleteMany({Age:{$gt:56}})

{ "acknowledged" : true, "deletedCount" : 0}

# MONGOBD EMPLOYEE DATABASE

**1.In database Employee.** use db\_name switched to db db\_name

> use emp

switched to db emp

>show dbs admin 0.000GB config 0.000GB emp 0.000GB harsha 0.000GB local 0.000GB med\_data 0.000GB test

>show collections empDetails

>db.empDetails.find()

{ "\_id" : ObjectId("613726aa40cbedb84b494e0d"), "Name" : "Mohan", "Age" : 30,"Email" : ["mohan12345@gmail.com", "salary"](mailto:mohan12345@gmail.com) : 5000 }

{ "\_id" : ObjectId("613726d340cbedb84b494e0e"), "Name" : "Raju", "Age" : 35, "Email" : ["raju@gmail.com",](mailto:raju@gmail.com) "salary" : 7000 }

{ "\_id" : ObjectId("613726fc40cbedb84b494e0f"), "Name" : "Bhuvan", "Age" : 25, "Email" :["bhuvan@gmail.com",](mailto:bhuvan@gmail.com) "salary" : 10000 }

{ "\_id" : ObjectId("6137272f40cbedb84b494e10"), "Name" : "Meera", "Age" : 27,

"Email": ["meera@gmail.com",](mailto:meera@gmail.com) "salary" : 9000 }

{ "\_id" : ObjectId("6137275040cbedb84b494e11"), "Name" : "Maya", "Age" : 28, "Email" :["maya@gmail.com",](mailto:maya@gmail.com) "salary" : 15000 }

db.empDetails.updateMany({Name :”Mohan”},{Dept : “Designer”}})

{ “acknolegement” :true, “matchedCount :1, “modified Count” : 1 }

db.empDetails.updateMany({Name :”Raju”},{Dept : “Tester”}})

{ “acknolegement” :true, “matchedCount :1, “modified Count” : 1 }

db.empDetails.updateMany({Name :”Bhuvan”},{Dept : “Developer”}}) {“acknolegement” :true, “matchedCount :1, “modified Count” : 1 }

db.empDetails.updateMany({Name :”Meera”},{Dept : “Designer”}})

{ “acknolegement” :true, “matchedCount :1, “modified Count” : 1 }

db.empDetails.updateMany({Name :”Maya”},{Dept : “Tester”}})

{ “acknolegement” :true, “matchedCount :1, “modified Count” : 1 }

* db.empDetails.find()

{ "\_id" : ObjectId("613726aa40cbedb84b494e0d"), "Name" : "Mohan", "Age" : 30, "Email"

:["mohan12345@gmail.com",](mailto:mohan12345@gmail.com) "salary" : 5000 ,”Dept “ : “Designer”}

{ "\_id" : ObjectId("613726d340cbedb84b494e0e"), "Name" : "Raju", "Age" : 35, "Email" : ["raju@gmail.com",](mailto:raju@gmail.com) "salary" : 7000,”Dept “ : “Tester” }

{ "\_id" : ObjectId("613726fc40cbedb84b494e0f"), "Name" : "Bhuvan", "Age" : 25, "Email" :["bhuvan@gmail.com",](mailto:bhuvan@gmail.com) "salary" : 10000,”Dept “ : “Developer”}

{ "\_id" : ObjectId("6137272f40cbedb84b494e10"), "Name" : "Meera", "Age" : 27, "Email"

:["meera@gmail.com",](mailto:meera@gmail.com) "salary" : 9000 ,”Dept “ : “Designer”}

{ "\_id" : ObjectId("6137275040cbedb84b494e11"), "Name" : "Maya", "Age" : 28, "Email" : ["maya@gmail.com",](mailto:maya@gmail.com) "salary" : 15000, ,”Dept “ : “Tester”}

### find the average salary of each dept.

db.empDetails.aggregate([{$group:{\_id:"$Dept","Avg\_Salary":{$avg:"$Salary"}}}])

{ "\_id" : "Designer", "Salary" : 10000 }

{ "\_id" : "Developer", "Salary" : 10000 }

{ "\_id" : "Tester", "Salary" : 8000 }

### 2.find the minimum salary of each dept.

db.empDetails.aggregate([{$group:{\_id:"$Dept","Min\_Salary":{$min:"$Salary"}}}])

{ "\_id" : "Designer", "Salary" : 5000 }

{ "\_id" : "Developer", "Salary" : 10000 }

{ "\_id" : "Tester", "Salary" : 7000 }

### 3.find the maximum salary of each dept.

* db.empDetails.aggregate([{$group:{\_id:"$Dept","Max\_Salary":{$max:"$Salary"}}}])

{ "\_id" : "Tester", "Max\_Salary" : 9000 }

{ "\_id" : "Designer", "Max\_Salary" : 15000 }

{ "\_id" : "Developer", "Max\_Salary" : 10000 }

### 4.find the no.of employees of each dept.

db.empDetails.aggregate([{$group:{id:”Dept”,”No of employees”:{$sum:1}}}])

{“id”:”Dept”,”No of employees” : 5 }

### 5.sort the collection empDetails in descending order of name

* db.empDetails.find().sort({"Name": -1})

{ "\_id" : ObjectId("613726d340cbedb84b494e0e"), "Name" : "Raju", "Age" : 35, "Email" :["raju@gmail.com",](mailto:raju@gmail.com) "salary" : 7000 }

{ "\_id" : ObjectId("613726aa40cbedb84b494e0d"), "Name" : "Mohan", "Age" : 30, "Email"

:["mohan12345@gmail.com",](mailto:mohan12345@gmail.com) "salary" : 5000 }

{ "\_id" : ObjectId("6137272f40cbedb84b494e10"), "Name" : "Meera", "Age" : 27, "Email"

:["meera@gmail.com",](mailto:meera@gmail.com) "salary" : 9000 }

{ "\_id" : ObjectId("6137275040cbedb84b494e11"), "Name" : "Maya", "Age" : 28, "Email" : ["maya@gmail.com",](mailto:maya@gmail.com) "salary" : 15000 }

{ "\_id" : ObjectId("613726fc40cbedb84b494e0f"), "Name" : "Bhuvan", "Age" : 25, "Email" :["bhuvan@gmail.com",](mailto:bhuvan@gmail.com) "salary" : 10000 }

### 6.Create a text index for ‘name’ and search for names mohan and bhuvan

db.empDetails.createIndex({Name : “text”})

{

“numIndexesBefore” : 1,

“numIndexesAfter” : 2, “createdCollectionAutomatically” : false, “ok” : 1

}

* db.empDetails.find()

{ "\_id" : ObjectId("6141db0be3945e33a7255b56"), "Name" : "Mohan", "Age" : 30, "Email"

:["mohan@gmail.com",](mailto:mohan@gmail.com) "Salary" : 5000 }

{ "\_id" : ObjectId("6141db14e3945e33a7255b57"), "Name" : "Raju", "Age" : 35, "Email" :["raju@gmail.com",](mailto:raju@gmail.com) "Salary" : 7000 }

{ "\_id" : ObjectId("6141db20e3945e33a7255b58"), "Name" : "Bhuvan", "Age" : 25, "Email" : ["bhuvan@gmail.com",](mailto:bhuvan@gmail.com) "Salary" : 10000 }

* db.empDetails.find({$text:{$search:"Mohan Bhuvan"}})

{ "\_id" : ObjectId("6141db20e3945e33a7255b58"), "Name" : "Bhuvan", "Age" : 25, "Email" :["bhuvan@gmail.com",](mailto:bhuvan@gmail.com) "Salary" : 10000 }

{ "\_id" : ObjectId("6141db0be3945e33a7255b56"), "Name" : "Mohan", "Age" : 30, "Email" : ["mohan@gmail.com", "Salary"](mailto:mohan@gmail.com) : 5000 }

### 7. create a database Inventory and create an orders collection.Apply Map Reduce operation for finding the total purchase of each customer.

>use inventory

>switched to db inventory

* db.createCollection("orders")

{ "ok" : 1 }

>db.orders.insert({custid:200,name:"maya",item:"rice",price:340}) WriteResult({ "nInserted" : 1 })

>db.orders.insert({custid:201,name:"anu",item:"rice",price:340}) WriteResult({ "nInserted" : 1 })

>db.orders.insert({custid:202,name:"Meera",item:"sugar",price:35}) WriteResult({ "nInserted" : 1 })

* db.orders.insert({custid:201,name:"anu",item:"wheat",price:40}

WriteResult({ "nInserted" : 1 })

>db.orders.insert({custid:202,name:"Meera",item:"teapower",price:250}) WriteResult({ "nInserted" : 1 })

>db.orders.insert({custid:203,name:"navya",item:"wheat",price:40}) WriteResult({ "nInserted" : 1 })

* db.orders.find()

{ "\_id" : ObjectId("614251b0ec9cd36cd1db1f46"), "custid" : 200, "name" : "maya", "item"

:"rice", "price" : 340 }

{ "\_id" : ObjectId("614251c0ec9cd36cd1db1f47"), "custid" : 201, "name" : "anu", "item"

:"rice", "price" : 340 }

{ "\_id" : ObjectId("614251dcec9cd36cd1db1f48"), "custid" : 202, "name" : "Meera", "item"

:"sugar", "price" : 35 }

{ "\_id" : ObjectId("614251f2ec9cd36cd1db1f49"), "custid" : 201, "name" : "anu", "item"

:"wheat", "price" : 40 }

{ "\_id" : ObjectId("6142520dec9cd36cd1db1f4a"), "custid" : 202, "name" : "Meera", "item" : "teapower", "price" : 250 }

{ "\_id" : ObjectId("61425230ec9cd36cd1db1f4b"), "custid" : 203, "name" : "navya", "item"

:"wheat", "price" : 40 }

>var mapFunction1=function(){emit(this.custid,this.price);};

>varreduceFunction1=function(keyCustId,valuesPrices){return Array.sum(valuesPrices);};

>db.orders.mapReduce(mapFunction1,reduceFunction1,{out: "map\_example"}) { "result" : "map\_example", "ok" : 1 }

* db.map\_example.find()

{ "\_id" : 201, "value" : 380 }

{ "\_id" : 200, "value" : 340 }

{ "\_id" : 203, "value" : 40 }

{ "\_id" : 202, "value" : 285 }

* db.orders.find()

{ "\_id" : ObjectId("61425321ec9cd36cd1db1f4c"), "custid" : 200, "name" : "maya", "item"

:"rice", "price" : 340 }

{ "\_id" : ObjectId("61425323ec9cd36cd1db1f4d"), "custid" : 201, "name" : "anu", "item" : "rice", "price" : 340 }

{ "\_id" : ObjectId("61425325ec9cd36cd1db1f4e"), "custid" : 202, "name" : "Meera", "item" : "sugar", "price" : 35 }

{ "\_id" : ObjectId("61425326ec9cd36cd1db1f4f"), "custid" : 201, "name" : "anu", "item"

:"wheat", "price" : 40 }

{ "\_id" : ObjectId("61425328ec9cd36cd1db1f50"), "custid" : 202, "name" : "Meera", "item"

:"teapower", "price" : 250 }

{ "\_id" : ObjectId("6142532bec9cd36cd1db1f51"), "custid" : 203, "name" : "navya", "item" : "wheat", "price" : 40 }

# MONGODB COLLECTION & INSERT

**1. Python program to create a collection & insert and print the Student details**

# importing Mongoclient

from pymongo import MongoClient

# Making Connection

myclient = MongoClient("mongodb://localhost:27017/")

# database

db =myclient["Studentnew"]

# Created or Switched to collection mycollection = db["student"]

# Creating Dictionary of records to be # inserted

record = { "\_id": 12,

"name": "Cam",

"Roll No": "1011",

"Branch": "CA"}

# Inserting the record in the collection # by using collection.insert\_one()

mycollection.insert\_one(record)

#selecting all the records and printing x = mycollection.find\_one()

for x in mycollection.find():

print(x)

**OUTPUT**

{'\_id': 12, 'name': 'Cam', 'Roll No': '1011', 'Branch': 'CA'}

**2. Python program to create a collection & insert ‘n’ STUDENTS details using 'insert many.**

# importing Mongoclient from pymongo

from pymongo import MongoClient

# Making Connection

myclient = MongoClient("mongodb://localhost:27017/")

# database

db =myclient["Student"]

# Created or Switched to collection mycollection = db["student"]

# Creating Dictionary of records to be

# inserted

record =[ { "\_id": 13,

"name": "Lee",

"Roll No": "1012",

"Branch": "MBA"},

{ "\_id": 14,

"name": "Tom",

"Roll No": "1013",

"Branch": "BBA"},

{ "\_id": 15,

"name": "Jhon",

"Roll No": "1013",

"Branch": "MCA"}]

# Inserting the record in the collection # by using collection.insert\_many()

mycollection.insert\_many(record)

#selecting all the records and printing x = mycollection.find\_one ()

for x in mycollection.find():

print(x)

}

## Output

{'\_id': 13, 'name': 'Lee', 'Roll No': '1012', 'Branch': 'MBA'}

{'\_id': 14, 'name': 'Tom', 'Roll No': '1013', 'Branch': 'BBA'}

{'\_id': 15, 'name': 'Jhon', 'Roll No': '1013', 'Branch': 'MCA'}