Musaliar College of Engineering & Technology

MUSALIAR COLLEGE P.O., PATHANAMTHITTA - 689 653

..... -.....

LABORATORY RECORD

Certified that this is the Bonafide Record of the work done by

Sri/Smt.....

of.....Semester Class of .....(Roll No .....)

of .....Branch

in the .....Laboratory

during the academic year 2022 - 2024

Name of Examination

Reg. No. External Examiner Staff in- charge

DEPARTMENT OF COMPUTER APPLICATIONS

VISION

"To produce competent and dynamic professionals in the field of Computer Applications to thrive and cater the changing needs of the society through research and education".

MISSION

To impart high quality technical education and knowledge in Computer Applications. To introduce moral, ethical and social values to Computer Application students.To establish industry institute interaction to enhance the skills of Computer Application students. To promote research aimed towards betterment of society.

INDEX

EXPERIMENT NO. 1

CREATION OF A DATABASE USING DDL COMMANDS

AIM

To Creation of a database using DDL commands including integrity constraints.

1) Create an INVOICE database and show list of databases:

OUTPUT

• SQL> create database INVOICE;

Query OK, 1 row affected (0.40 sec)

• SQL> use INVOICE;

Database changed

• SQL> show databases;

2) Create a table STUDENT with following schema:

(id, name,age,city) and describe the table.

OUTPUT

• SQL> create table student (id number(5), name varchar(20), age number(3),city varchar(30));

Table created.

• SQL> desc student;

```
SQL> desc student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
 ID                                                 NUMBER(5)
 NAME                                               VARCHAR2(20)
 AGE                                                NUMBER(3)
 CITY                                               VARCHAR2(30)
```

3) Add a new column; dob to the existing relation student.

OUTPUT

• SQL> alter table student add dob date;

Table altered.

• SQL> desc student;

```
SQL> desc student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 ID                                                 NUMBER(5)
 NAME                                               VARCHAR2(20)
 AGE                                                NUMBER(3)
 CITY                                               VARCHAR2(30)
 DOB                                                DATE
```

4) Modify the datatype of age from number to int.

OUTPUT

• SQL> alter table student modify age int;

Table altered.

5) Delete the column city from the relation student.

OUTPUT

• SQL> alter table student drop column city;

Table altered.

SQL> desc student;

```
SQL> desc student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 ID                                                 NUMBER(5)
 NAME                                               VARCHAR2(20)
 AGE                                                NUMBER(38)
 DOB                                                DATE
```

6) Create table CUSTOMER {cust_id, address, city, ph_no}

OUTPUT

• SQL> CREATE TABLE CUSTOMER (cust_id NUMBER PRIMARY KEY,address VARCHAR2(30), city VARCHAR2(20), ph_no NUMBER);

Table created.

• SQL> desc CUSTOMER;

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------
CUST_ID                                   NOT NULL NUMBER
ADDRESS                                            VARCHAR2(30)
CITY                                               VARCHAR2(20)
PH_NO                                              NUMBER
```

7) Add a column name in the CUSTOMER table

OUTPUT

• SQL> alter table CUSTOMER add name varchar(15);

Table altered.

• SQL> desc CUSTOMER;

```
SQL> desc CUSTOMER;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 CUST_ID                                            NUMBER(38)
 ADDRESS                                            VARCHAR2(30)
 CITY                                               VARCHAR2(20)
 PH_NO                                              NUMBER(38)
 NAME                                               VARCHAR2(15)
```

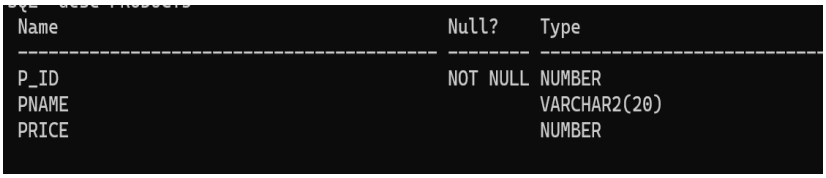
8) Create table PRODUCT { p_id, pname, price (default 0}

OUTPUT

• SQL> CREATE TABLE PRODUCT3 (p_id NUMBER PRIMARY KEY, pname VARCHAR2(20),price NUMBER DEFAULT 0);

Table created.

• SQL> desc PRODUCT;



Name	Null?	Type
P_ID	NOT NULL	NUMBER
PNAME		VARCHAR2(20)
PRICE		NUMBER

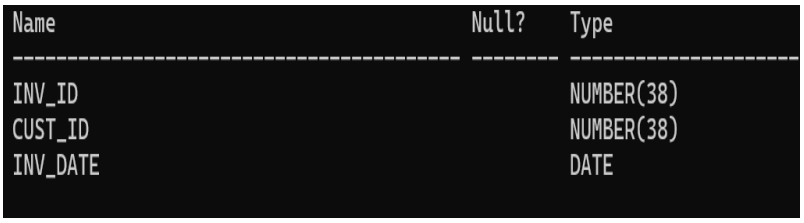
9) Create table INVOICE_MASTER { inv_id, cust_id(FK), inv_date}

OUTPUT

• SQL> CREATE TABLE INVOICE_MASTER (inv_id INT,cust_id INT, inv_date DATE,CONSTRAINT fk_cust_id FOREIGN KEY (cust_id) REFERENCES CUSTOMER(cust_id));

Table created.

• SQL> desc INVOICE_MASTER;



Name	Null?	Type
INV_ID		NUMBER(38)
CUST_ID		NUMBER(38)
INV_DATE		DATE

10) Create table INVOICE_ITEM {inv_id(FK), p_id (FK),quantity}

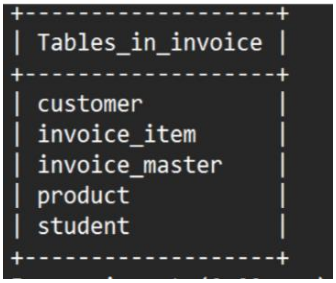
OUTPUT

• SQL> create table INVOICE_ITEM (inv_id int, constraint inv_id foreign key(inv_id) references INVOICE_MASTER(inv_id), p_id int, constraint p_id foreign key(p_id) references PRODUCT(p_id), quantity int);

11) Write SQL command to show the created tables.

OUTPUT

• SQL> show tables;



```
+------------------+
| Tables_in_invoice |
+------------------+
| customer         |
| invoice_item     |
| invoice_master   |
| product          |
| student          |
+------------------+
```

12) Write SQL command to delete table content without deleting the structure of the student relation.

• SQL> TRUNCATE table Student;

13) Write SQL command to delete the relation student.

• SQL> drop table student;

EXPERIMENT NO. 2

IMPLEMENTING DATA MANIPULATION LANGUAGE (DML) COMMANDS OF SQL

AIM

Create an application to apply Data Manipulation Language (DML) commands to modify the database.

1) Use INVOICE database and perform DML queries to the tables in the database.

OUTPUT

• SQL> use INVOICE;

2) Insert aleast 5 rows in the CUSTOMER.

OUTPUT

• SQL> insert into CUSTOMER values(1,'Anu','elattuparambil','ekm',8945678976);

1 row created.

• SQL> insert into CUSTOMER values(2,'abu','abc house','tvm',8945600976);

1 row created.

• SQL> insert into CUSTOMER values(3,'sreehari','rose house','tvm',NULL);

1 row created.

• SQL> insert into CUSTOMER values(4,'veena','abcd house','tvm',8945654511);

1 row created.

• SQL> insert into CUSTOMER values(5,'binu','white house','tvm',9945654511);

1 row created.

• SQL> insert into CUSTOMER values(6,'Anju','rose villa','calicut',NULL);

1 row created.

3) Display all the information of CUSTOMER table.

OUTPUT

• SQL> select * from CUSTOMER;

cust_id	name	address	city	ph_no
1	Anu	elattuparambil	ekm	8945678976
2	abu	abc house	tvm	8945600976
3	sreehari	rose house	tvm	NULL
4	veena	abcd house	tvm	8945654511
5	binu	white house	tvm	9945654511
6	Anju	rose villa	calicut	NULL

4) Insert aleast 5 rows in the PRODUCT table.

OUTPUT

• SQL> insert into PRODUCT values(101,'pen',10);

1 row created.

• SQL> insert into PRODUCT values(102,'soap',65);

1 row created.

• SQL> insert into PRODUCT values(103,'pencil',10);

1 row created.

• SQL> insert into PRODUCT values(104,'powder',100);

1 row created.

• SQL> insert into PRODUCT values(105,'shampoo',110);

1 row created.

• SQL> insert into PRODUCT values(106,'handwash',60);

1 row created.

5) Display all the information of PRODUCT table.

OUTPUT

• SQL> select * from PRODUCT;

p_id	pname	price
101	pen	10
102	soap	65
103	pencil	10
104	powder	100
105	shampoo	110
106	handwash	60

6) Update the city calicut with current city as kozhikode.

OUTPUT

• SQL> update CUSTOMER set city='kozhikode' where city='calicut';

1 rows updated.

• SQL> select * from CUSTOMER;

cust_id	name	address	city	ph_no
1	Anu	elattuparambil	ekm	8945678976
2	abu	abc house	tvm	8945600976
3	sreehari	rose house	tvm	NULL
4	veena	abcd house	tvm	8945654511
5	binu	white house	tvm	9945654511
6	Anju	rose villa	kozhikode	NULL

7) Update the phone number of cust_id = 2 with 9996667890.

OUTPUT

• SQL> update CUSTOMER set ph_no=9996667890 where cust_id=2 ;

1 rows updated.

SQL> select * from CUSTOMER;

cust_id	name	address	city	ph_no
1	Anu	elattuparambil	ekm	8945678976
2	abu	abc house	tvm	9996667890
3	sreehari	rose house	tvm	NULL
4	veena	abcd house	tvm	8945654511
5	binu	white house	tvm	9945654511
6	Anju	rose villa	kozhikode	NULL

8) Delete a PRODUCT having p_id '101' from PRODUCT table.

OUTPUT

• SQL> delete from PRODUCT where p_id=101;

1 row deleted.

• SQL> select * from PRODUCT;

```
SQL> select * from PRODUCT;

      P_ID PNAME                    PRICE
---------- -------------------- ----------
       102 soap                         65
       103 pencil                       10
       104 powder                      100
       105 shampoo                     110
```

EXPERIMENT NO. 3

ACCESSING DATABASE USING DQL COMMAND OF SQL

AIM

Create an application to retrieve data from databases using select, views

1) LIST ALL CUSTOMER DETAILS.

• SQL> select * from CUSTOMER;

cust_id	name	address	city	ph_no
1	Anu	elattuparambil	ekm	8945678976
2	abu	abc house	tvm	9996667890
3	sreehari	rose house	tvm	NULL
4	veena	abcd house	tvm	8945654511
5	binu	white house	tvm	9945654511
6	Anju	rose villa	kozhikode	NULL

2) List pname,price of all PRODUCTs.

• SQL> select pname,price from PRODUCT;

```
SQL> select * from PRODUCT;

      P_ID PNAME                     PRICE
---------- -------------------- ----------
       102 soap                         65
       103 pencil                       10
       104 powder                      100
       105 shampoo                     110
       106 shampoo                     110
```

3) List all PRODUCTs avoid duplicates.

• SQL> select distinct pname from PRODUCT;

```
SQL> select distinct pname from PRODUCT;

PNAME
--------------------
powder
pencil
shampoo
soap
```

4) List PRODUCT price between 50rs and 150rs.

• SQL>

```
SQL> select price from PRODUCT where price between 50 and 150;

     PRICE
----------
        65
       100
       110
       110
```

5) List the CUSTOMER details who lives in 'ekm','tvm'.

• SQL> select * from CUSTOMER where city in('ekm','tvm');

cust_id	name	address	city	ph_no
1	Anu	elattuparambil	ekm	8945678976
2	abu	abc house	tvm	9996667890
3	sreehari	rose house	tvm	NULL
4	veena	abcd house	tvm	8945654511
5	binu	white house	tvm	9945654511

6) List the invoice details from 1st january 2021 to 31st march 2021.

• SQL> select * from invoice_master where inv_date between '2021-01-01' and '2021-03-

31';

inv_id	cust_id	inv_date
112	2	2021-03-11

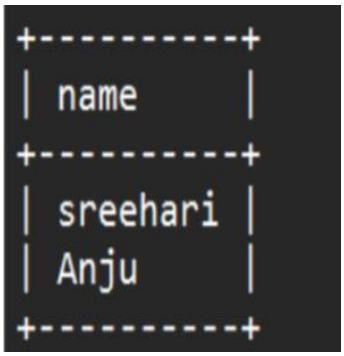
7) List the CUSTOMER details who not live in 'tvm'.

• SQL> select * from CUSTOMER where city not in('tvm');

cust_id	name	address	city	ph_no
1	Anu	elattuparambil	ekm	8945678976
6	Anju	rose villa	kozhikode	NULL

8) List the CUSTOMERs who have no phone number.

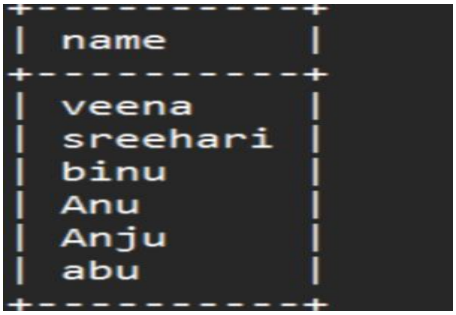
• SQL> select name from CUSTOMER where ph_no IS NULL;



name
sreehari
Anju

9) Display CUSTOMER names in descending order.

• SQL> select name from CUSTOMER order by name desc;



name
veena
sreehari
binu
Anu
Anju
abu

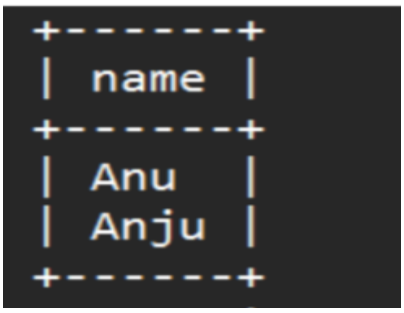
10) Display the CUSTOMER details sorted in ascending by city and descending by names.

• SQL> select * from CUSTOMER order by city asc,name desc;

cust_id	name	address	city	ph_no
1	Anu	elattuparambil	ekm	8945678976
6	Anju	rose villa	kozhikode	NULL
4	veena	abcd house	tvm	8945654511
3	sreehari	rose house	tvm	NULL
5	binu	white house	tvm	9945654511
2	abu	abc house	tvm	9996667890

11) Find the CUSTOMER names start with 'A'.

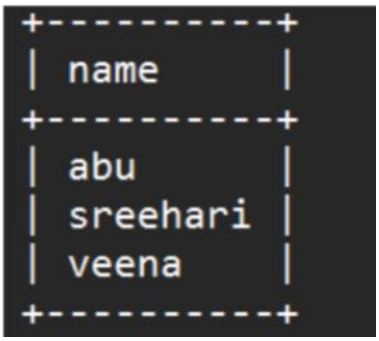
• SQL> select name from CUSTOMER where name LIKE 'A%';



name
Anu
Anju

12) Find the CUSTOMER names who have 'e' or 'a' in them.

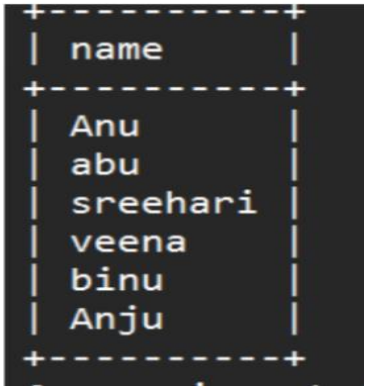
• SQL> select name from CUSTOMER where name LIKE '%e%' or name LIKE '%a%';



name
abu
sreehari
veena

13) Find the CUSTOMER names who have at least 3 characters.

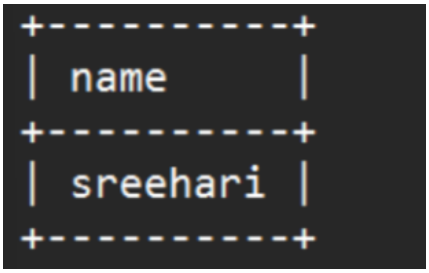
• SQL> select name from CUSTOMER where name like '___%';



name
Anu
abu
sreehari
veena
binu
Anju

14) List the names of CUSTOMERs their names have 'r' in second posititon.

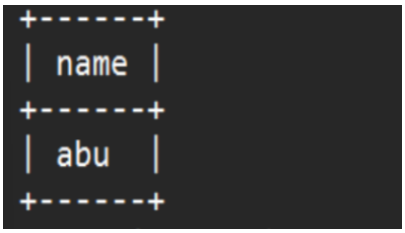
• SQL> select name from CUSTOMER where name like '_r%';



```
+----------+
| name     |
+----------+
| sreehari |
+----------+
```

15) Find the CUSTOMER names start with 'a' and are at least 3 characters in len

• SQL> select name from CUSTOMER where name like 'a__%';



```
+------+
| name |
+------+
| abu  |
+------+
```

EXPERIMENT NO. 4

ACCESSING DATABASE USING DQL COMMAND OF SQL

(GROUP BY,AGGREGATE FUNCTIONS)

AIM

Create an application to retrieve data from databases using select, views

1. List PRODUCT name, price except soap on price order.

• SQL> select pname,price from PRODUCT where pname != 'soap';

```
SQL> select pname,price from PRODUCT where pname != 'soap';

PNAME                    PRICE
-------------------- ----------
pencil                       10
powder                      100
shampoo                     110
shampoo                     110
```

2. Find minimum, maximum and average price of PRODUCTs.

• SQL> select min(price),max(price),avg(price) from PRODUCT;

```
SQL> select min(price),max(price),avg(price) from PRODUCT;

MIN(PRICE) MAX(PRICE) AVG(PRICE)
---------- ---------- ----------
        10        110         79
```

3. Find number of CUSTOMERs in each city.

• SQL> select city,count(city) as count from CUSTOMER group by city;

```
SQL> select city,count(city) as count from CUSTOMER group by city;

CITY                    COUNT
-------------------- ----------
kollam                       1
kozhikode                    3
tvm                          2
```

4. Find number of CUSTOMERs in each city, only include city with more than or equal to 3 CUSTOMERs.

• SQL> select city,count(city) as count from CUSTOMER group by city having count(city) > = 3;

```
SQL> select city,count(city) as count from CUSTOMER group by city having count(city) >= 3;

CITY                 COUNT
-------------------- ----------
kozhikode                 3
```

5. Show data as displayed ‘who, where’ Eg: ‘CUSTOMER1 lives in ekm’.

• SQL> select CONCAT(name,' livesin ',city) AS 'whowhere' from CUSTOMER.

•SQL> CREATE TABLE EMP1(ENO NUMBER(4),ENAME VARCHAR(20),DGN CHAR(18),DOJ DATE,SAL NUMBER(6));
Table created.

•SQL> ALTER TABLE EMP1 MODIFY ENAME VARCHAR(25);
Table altered.

•SQL> ALTER TABLE EMP1 ADD PLACE VARCHAR(25);
Table altered.

•SQL> ALTER TABLE EMP1 DROP COLUMN PLACE ;
Table altered.

•SQL> INSERT INTO EMP1 VALUES(1,'ANU','MANAGER','02-JAN-2020',20000);
1 row created.

•SQL> INSERT INTO EMP1 VALUES(2,'EBY','ACCOUNTANT','03-FEB-2023',26000);
1 row created.

•SQL> INSERT INTO EMP1 VALUES(3,'ROSY','MANAGER','31-DEC-2019',30000);
1 row created.

•SQL> SELECT * FROM EMP1;

ENO	ENAME	DGN	DOJ	SAL
1	ANU	MANAGER	02-JAN-20	2000
2	EBY	ACCOUNTANT	03-FEB-23	26000
3	ROSY	MANAGER	31-DEC-19	30000

•SQL> CREATE TABLE STUD1(SNO NUMBER(4),SNAME VARCHAR(20),DEPT CHAR(18),ADMISSION NUMBER(6));
Table created.

•SQL> INSERT INTO STUD1 VALUES(1,'ANIE','MCA',22011);
1 row created.

•SQL> INSERT INTO STUD1 VALUES(2,'EBY','BCA',23111);
1 row created.

•SQL> INSERT INTO STUD1 VALUES(3,'CHRIS','BA',12345);
1 row created.

•SQL> SELECT * FROM STUD1;

SNO	SNAME	DEPT	ADMISSION
1	ANIE	MCA	22011
2	EBY	BCA	23111
3	CHRIS	BA	12345

•SQL> SELECT * FROM STUD1 ORDER BY SNAME;

SNO	SNAME	DEPT	ADMISSION
1	ANIE	MCA	22011
3	CHRIS	BA	12345
2	EBY	BCA	23111

•SQL> SELECT * FROM EMP1 ORDER BY ENAME;

ENO	ENAME	DGN	DOJ	SAL
1	ANU	MANAGER	02-JAN-20	20000
2	EBY	ACCOUNTANT	03-FEB-23	26000

•SQL> SELECT * FROM EMP1 ORDER BY ENAME DESC;

ENO	ENAME	DGN	DOJ	SAL
2	EBY	ACCOUNTANT	03-FEB-23	26000
1	ANU	MANAGER	02-JAN-20	20000

•SQL> SELECT * FROM STUD1 ORDER BY SNAME DESC;

SNO	SNAME	DEPT	ADMISSION
2	EBY	BCA	23111
3	CHRIS	BA	12345
1	ANIE	MCA	22011

•SQL> SELECT MAX (SAL) FROM EMP1;

MAX(SAL)
26000

•SQL> SELECT MIN (SAL) FROM EMP1;

```
 MIN(SAL)
 ----------
   20000
```

•SQL> SELECT SUM (SAL) FROM EMP1;

```
 SUM(SAL)
  ----------
   46000
```

•SQL> SELECT AVG (SAL) FROM EMP1;

```
 AVG(SAL)
 ----------
   23000
```

•SQL> SELECT COUNT (ENAME) FROM EMP1;

```
COUNT(ENAME)
  ------------
        2
```

•SQL> SELECT COUNT (*) FROM EMP1;

```
 COUNT(*)
  ----------
      2
```

•SQL> SELECT MAX (ADMISSION) FROM STUD1;

MAX(ADMISSION)

23111

•SQL> SELECT MIN (ADMISSION) FROM STUD1;

MIN(ADMISSION)

12345

•SQL> SELECT SUM (ADMISSION) FROM STUD1;

SUM(ADMISSION)

57467

•SQL> SELECT AVG (ADMISSION) FROM STUD1;

AVG(ADMISSION)

19155.6667

•SQL> SELECT COUNT (SNAME) FROM STUD1;

COUNT(SNAME)

3

•SQL> SELECT COUNT (*) FROM STUD1;

COUNT(*)

3

• SQL> SELECT DEPT, COUNT(*) FROM STUD1 GROUP BY DEPT;

```
DEPT                COUNT(*)
------------------  ----------
BCA                          1
BA                           1
MCA                          1
```

•SQL> SELECT DGN, COUNT(*) FROM EMP1 GROUP BY DGN;

```
DGN                 COUNT(*)
---------------             ----------
ACCOUNTANT                   1
MANAGER                      1
```

SQL> SELECT DGN, AVG(SAL) FROM EMP1 GROUP BY DGN;

```
DGN                 AVG(SAL)
------------------     ----------
ACCOUNTANT          26000
MANAGER                 20000
```

SQL> SELECT DGN, COUNT (*) FROM EMP1 GROUP BY DGN HAVING COUNT (*) > 5;

no rows selected

SQL> SELECT DGN, COUNT (*) FROM EMP1 GROUP BY DGN HAVING COUNT (*) < 5;

```
DGN                 COUNT(*)
------------------ ----------
ACCOUNTANT                   1
MANAGER                      1
```

EXPERIMENT NO. 5

IMPLEMENTING DCL AND TCL COMMANDS OF SQL

AIM

To apply DCL and TCL commands to impose restrictions on databases.

a) DCL COMMANDS

1. Create two users user1 and user2

• SQL> CREATE USER user1 IDENTIFIED BY password1;

User created.

• SQL> CREATE USER user2 IDENTIFIED BY password2;

User created.

2. Write SQL query to give insert and select privilege to user 1 on customer table

• SQL> GRANT INSERT, SELECT ON CUSTOMER TO user1;

Grant succeeded.

• SQL> SELECT * FROM USER_TAB_PRIVS WHERE GRANTEE = 'USER1';

GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRA HIE
USER1	SYSTEM	CUSTOMER	SYSTEM	INSERT	NO NO

3. Write SQL query to give update and delete privilege to user 2 on all tables

• SQL> GRANT CONNECT, RESOURCE TO user2;

Grant succeeded.

• SQL> GRANT UPDATE, DELETE ON CUSTOMER TO user2;

Grant succeeded.

4. Write SQL query to remove delete privilege of user 2.

• SQL> SELECT * FROM USER_TAB_PRIVS WHERE GRANTEE = 'USER2';

GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRA	HIE
USER2	SYSTEM	CUSTOMER	SYSTEM	DELETE	NO	NO

b) TCL COMMANDS

1. Create another database school and create a table student with attributes roll_no, name and department.

• SQL> create database SCHOOL;

• SQL> CREATE TABLE student (

roll_no number(3),

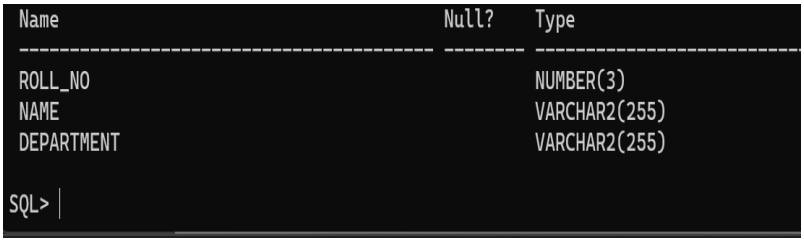
name VARCHAR(255),

department VARCHAR(255)

);

Table created.

• SQL> desc student;



2. Insert aleast 5 rows in the Student table.

• SQL> insert into student values(1,'abhilash','mca');

1 row created.

• SQL> insert into student values(2,'nayana','mca');

1 row created.

• SQL> insert into student values(3,'ammu','mca');

1 row created.

• SQL> insert into student values(4,'anu','mba');

1 row created.

3. List all the rows in the Student table.

• SQL> select * from student;

```
 ROLL_NO NAME     DEPARTME
-------- -------- --------
       1 abhilash mca
       2 nayana   mca
       3 ammu     mca
       4 anu      mba
```

4. Insert another row in the table and roll back it

• SQL> insert into student2 values(6,'anju','bca');

1 row created.

• SQL> rollback;

Rollback complete.

5. Insert another row in the table and save it

• SQL> insert into student2 values(6,'anju','bca');

1 row created.

• SQL> savepoint t1;

Savepoint created.

• SQL> insert into student2 values(7,'varun','bca');

1 row created.

6. List all the rows in the Student table.

• mysql> select * from student;

```
   ROLL_NO NAME     DEPARTME
---------- -------- --------
         6 anju     bca
         7 varun    bca
```

• mysql> rollback to t1;

Rollback complete.

• mysql> select * from student;

```
   ROLL_NO NAME     DEPARTME
---------- -------- --------
         6 anju     bca
         7 varun    bca
```

EXPERIMENT NO. 6

OPTIMIZING DATABASES

AIM

To Optimizing databases (Join, Aggregate & Set operations, Other operator like arithmetic, logical, special etc.)

SUBQUERIES

•SQL> SELECT * FROM AUTHOR;

```
TITLE              AUTHOR          PUBLISHER    CATEGORY                YEAR
------------------ --------------- ------------ ----------------- ----------
     PRICE
----------
PHYSICAL EDUCATION BINDU B         A R COMPANY  PHYSICAL STUDIES        2009
       350

ORGANIC CHEMISTRY  JAMES BOND      BOND COMPANY CHEMICAL STUDIES        2000
       500
```

•SQL> CREATE TABLE PUBLISHER(PUBID NUMBER(4),DISTRIBUTER VARCHAR(10),CITY VARCHAR(10),DISCOUNT NUMBER(3),TIME NUMBER(4),CREDIT VARCHAR(5));

Table created.

•SQL> DESC PUBLISHER;

```
SQL> desc publisher;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PUBID                                              NUMBER(4)
 DISTRIBUTER                                        VARCHAR2(10)
 CITY                                               VARCHAR2(10)
 DISCOUNT                                           NUMBER(3)
 TIME                                               NUMBER(4)
 CREDIT                                             VARCHAR2(5)
```

•SQL> INSERT INTO PUBLISHER VALUES(01,'L L BOOKS','KLM',20,9,'CRDIT');

1 row created.

•SQL> INSERT INTO PUBLISHER VALUES(02,'L B BOOKS','KTM',10,10,'CRDT');

1 row created.

•SQL> SELECT * FROM PUBLISHER;

```
    PUBID DISTRIBUTE CITY         DISCOUNT       TIME CREDI
--------- ---------- ---------- ---------- ---------- -----
        1 L L BOOKS  KLM                20          9 CRDIT
        2 L B BOOKS  KTM                10         10 CRDT
```

•SQL> CREATE TABLE ORDERS(ORDERNO NUMBER(4),TITLE VARCHAR(10),PUBID NUMBER(2),QTY NUMBER(5));

Table created.

•SQL> INSERT INTO ORDERS VALUES(23,'PHYSIC EDU',01,345);

1 row created.

•SQL> INSERT INTO ORDERS VALUES(24,'ORGANIC',02,546);

1 row created.

•SQL> SELECT * FROM ORDERS;

ORDERNO	TITLE	PUBID	QTY
23	PHYSIC EDU	1	345
24	ORGANIC	2	546

•SQL> SELECT TITLE FROM AUTHOR WHERE PRICE>(SELECT AVG(PRICE)FROM AUTHOR);

```
SQL> SELECT TITLE FROM AUTHOR WHERE PRICE>(SELECT AVG(PRICE)FROM AUTHOR);

TITLE
------------------
ORGANIC CHEMISTRY
```

•SQL> SELECT TITLE FROM AUTHOR WHERE PRICE<(SELECT AVG(PRICE)FROM AUTHOR WHERE CATEGORY='PHYSICAL STUDIES');

no rows selected

•SQL> SELECT DISTRIBUTER FROM PUBLISHER WHERE PUBID IN(SELECT PUBID FROM AUTHOR WHERE TITLE='ORGANIC CHEMISTRY');

```
SQL> SELECT DISTRIBUTER FROM PUBLISHER WHERE PUBID IN(SELECT PUBID FROM AUTHOR WHERE TITLE='ORGANIC CHEMISTRY');

DISTRIBUTE
----------
L L BOOKS
L B BOOKS
```

•SQL> SELECT * FROM PUBLISHER WHERE DISCOUNT>(SELECT AVG(DISCOUNT)FROM PUBLISHER);

```
   PUBID DISTRIBUTE CITY         DISCOUNT       TIME CREDI
-------- ---------- ---------- ---------- ---------- -----
       1 L L BOOKS  KLM                20          9 CRDIT
```

JOINS

•SQL> create table employees1(emp_id number(20),emp_name varchar(20),city varchar(20),salary number(19),age number(19));
Table created.

•SQL> create table projects1(project_no number(20),emp_id number(20),department varchar(20));
Table created.

•SQL> insert into employees1 values(1,'angeleena','chicago',200000,30);
1 row created.

•SQL> insert into employees1 values(2,'robert','austin',300000,26);
1 row created.

•SQL> insert into employees1 values(3,'cristian','denver',100000,42);
1 row created.

•SQL> insert into employees1 values(4,'krstan','washington',500000,29);
1 row created.

•SQL> insert into projects1 values(101,1,'testing');
1 row created.

•SQL> insert into projects1 values(102,2,'development');
1 row created.

•SQL> insert into projects1 values(103,3,'designing');
1 row created.

•SQL> insert into projects1 values(104,4,'development');
1 row created.

•SQL> select * from employees1;

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	angeleena	chicago	200000	30
2	robert	austin	300000	26
3	cristian	denver	100000	42
4	krstan	washington	500000	29

•SQL> select * from projects1;

PROJECT_NO	EMP_ID	DEPARTMENT
101	1	testing
102	2	development
103	3	designing
104	4	development

•SQL> select employees1.emp_name,projects1.department from employees1 inner join projects1 on projects1.emp_id = employees1.emp_id;

EMP_NAME	DEPARTMENT
angeleena	testing
robert	development
cristian	designing
krstan	development

•SQL> select employees1.emp_name,projects1.department from employees1 left join projects1 on projects1.emp_id = employees1.emp_id;

EMP_NAME	DEPARTMENT
angeleena	testing
robert	development
cristian	designing
krstan	development

•SQL> insert into employees1 values(5,'rusel','loss angels',10000,36);
1 row created.

•SQL> insert into employees1 values(6,'mary','canada',20000,48);
1 row created.

•SQL> select * from employees1;

EMP_ID	EMP_NAME	CITY	SALARY	AGE
1	angeleena	chicago	200000	30
2	robert	austin	300000	26
3	cristian	denver	100000	42
4	krstan	washington	500000	29
5	rusel	loss angels	10000	36
6	mary	canada	20000	48

6 rows selected.

•SQL> select employees1.emp_name,projects1.department from employees1 left join projects1 on projects1.emp_id = employees1.emp_id;

EMP_NAME	DEPARTMENT
angeleena	testing
robert	development
cristian	designing
krstan	development
mary	
rusel	

6 rows selected.

•SQL> select employees1.emp_name,projects1.department from employees1 right join projects1 on projects1.emp_id = employees1.emp_id;

EMP_NAME	DEPARTMENT
angeleena	testing
robert	development
cristian	designing
krstan	development

•SQL> select employees1.emp_name,projects1.department from employees1 full join projects1 ON projects1.emp_id = employees1.emp_id;

EMP_NAME	DEPARTMENT
angeleena	testing
robert	development
cristian	designing
krstan	development
mary	
rusel	

6 rows selected.

Set Operations

•SQL> CREATE TABLE EMP1(ENO NUMBER(4),ENAME VARCHAR(20),DGN CHAR(18),DOJ DATE,SAL NUMBER(6));

Table created.

•SQL> ALTER TABLE EMP1 MODIFY ENAME VARCHAR(25);

Table altered.

•SQL> ALTER TABLE EMP1 ADD PLACE VARCHAR(25);

Table altered.

•SQL> ALTER TABLE EMP1 DROP COLUMN PLACE ;

Table altered.

•SQL> INSERT INTO EMP1 VALUES(1,'ANU','MANAGER','02-JAN-2020',20000);

1 row created.

•SQL> INSERT INTO EMP1 VALUES(2,'EBY','ACCOUNTANT','03-FEB-2023',26000);

1 row created.

•SQL> INSERT INTO EMP1 VALUES(3,'ROSY','MANAGER','31-DEC-2019',30000);

1 row created.

•SQL> SELECT * FROM EMP1;

ENO	ENAME	DGN	DOJ	SAL
1	ANU	MANAGER	02-JAN-20	2000
2	EBY	ACCOUNTANT	03-FEB-23	26000
3	ROSY	MANAGER	31-DEC-19	30000

•SQL> CREATE TABLE STUD1(SNO NUMBER(4),SNAME VARCHAR(20),DEPT CHAR(18),ADMISSION NUMBER(6));

Table created.

•SQL> INSERT INTO STUD1 VALUES(1,'ANIE','MCA',22011);

1 row created.

•SQL> INSERT INTO STUD1 VALUES(2,'EBY','BCA',23111);

1 row created.

•SQL> INSERT INTO STUD1 VALUES(3,'CHRIS','BA',12345);

1 row created.

•SQL> SELECT * FROM STUD1;

SNO	SNAME	DEPT	ADMISSION
1	ANIE	MCA	22011
2	EBY	BCA	23111
3	CHRIS	BA	12345

•SQL> CREATE TABLE STUD2(SNO NUMBER(4),SNAME VARCHAR(20),DEPT CHAR(18),ADMISSION NUMBER(6));

Table created.

•SQL> INSERT INTO STUD2 VALUES(3,'ANCY','PHD',22012);

1 row created.

•SQL> INSERT INTO STUD2 VALUES(4,'ADORA','PHD',9002);

1 row created.

•SQL> INSERT INTO STUD2 VALUES(5,'AVHUJ','BVCA',45602);

1 row created.

•SQL> SELECT * FROM STUD1 UNION SELECT * FROM STUD2;

SNO	SNAME	DEPT	ADMISSION
1	ANIE	MCA	22011
2	EBY	BCA	23111
3	ANCY	PHD	22012
3	CHRIS	BA	12345
4	ADORA	PHD	9002
4	CMMU	BA ENGLISH	22014
5	ANU	BA ENGLISH	22222
5	AVHUJ	BVCA	45602

8 rows selected.

•SQL> SELECT * FROM STUD1 UNION ALL SELECT * FROM STUD2;

SNO	SNAME	DEPT	ADMISSION
1	ANIE	MCA	22011
2	EBY	BCA	23111
3	CHRIS	BA	12345
3	ANCY	PHD	22012
4	CMMU	BA ENGLISH	22014
5	ANU	BA ENGLISH	22222
3	ANCY	PHD	22012
4	ADORA	PHD	9002
5	AVHUJ	BVCA	45602

9 rows selected.

•SQL> SELECT * FROM STUD1 INTERSECT SELECT * FROM STUD2;

SNO	SNAME	DEPT	ADMISSION
3	ANCY	PHD	22012

•SQL> SELECT * FROM STUD1 MINUS SELECT * FROM STUD2;

SNO	SNAME	DEPT	ADMISSION
1	ANIE	MCA	22011
2	EBY	BCA	23111
3	CHRIS	BA	12345
4	CMMU	BA ENGLISH	22014
5	ANU	BA ENGLISH	22222

EXPERIMENT NO. 7

PL/SQL PROGRAMS - FAMILIARIZATION WITH TRIGGER

AIM

To have familiarize with trigger functions.

PROGRAM

•SQL> create table customers(id number(2),name varchar(7),age number(2),address varchar(9),salary number(5));

Table created.

•SQL> insert into customers values(1,'ramesh',32,'ahmedabad',2000);

1 row created.

•SQL> insert into customers values(2,'khilan',25,'delhi',1500);

1 row created.

•SQL> insert into customers values(3,'kaushik',23,'kota',2000);

1 row created.

•SQL> insert into customers values(4,'chaita',25,'mumbai',6500);

1 row created.

•SQL> insert into customers values(5,'hardik',27,'bhopal',8500);

1 row created.

•SQL> insert into customers values(6,'komal',22,'mp',4500);

1 row created.

•SQL> select * from customers;

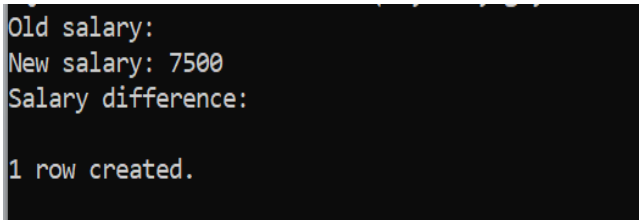
ID	NAME	AGE	ADDRESS	SALARY
1	ramesh	32	ahmedabad	2000
2	khilan	25	delhi	1500
3	kaushik	23	kota	2000
4	chaita	25	mumbai	6500
5	hardik	27	bhopal	8500
6	komal	22	mp	4500

•ed trigger.sql

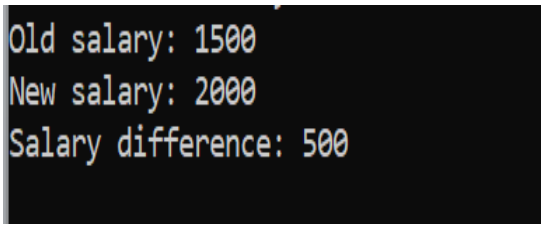
```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
  sal_diff number;
BEGIN
  sal_diff := :NEW.salary  - :OLD.salary;
  dbms_output.put_line('Old salary: ' || :OLD.salary);
  dbms_output.put_line('New salary: ' || :NEW.salary);
  dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

Trigger created.

•SQL> INSERT INTO customers (id,name,age,address,salary) values (7,'kriti',22,'hp',7500);



•SQL> UPDATE customers
2 SET salary = salary + 500
3 WHERE id = 2;



EXPERIMENT NO. 8

PL/SQL PROGRAMS - FAMILIARIZATION WITH STORED PROCEDURES

AIM

To have familiarize with stored procedure.

PROGRAM – 1

Aim : Create a procedure to display sum of two numbers .

PROGRAM

```
create or replace procedure pr1(a number,b number) is

c number;

begin

c:=a+b;

dbms_output.put_line('sum is'||c);

end;
```

OUTPUT

```
SQL> ed prosum.sql

SQL> @prosum.sql
  7  /

Procedure created.

SQL> EXEC pr1(12,34);
sum is46

PL/SQL procedure successfully completed.
```

PROGRAM – 2

Aim : Create a procedure to display commission of a employee.

PROGRAM

```
create or replace procedure pr2(n in number,comm out number)is
s employee2.sal%type;
begin
select sal into s from employee2 where eno=n;
comm:=s*.1;
end;
```

- Calling program

```
declare
n employee2.eno%type:=&n;
comm number;
begin
pr2(n,comm);
dbms_output.put_line('Commission is'||comm);
end;
```

OUTPUT

```
Enter value for n: 123
old   2: n employee2.eno%type:=&n;
new   2: n employee2.eno%type:=123;
Commission is123.3

PL/SQL procedure successfully completed.
```

PROGRAM – 3

Aim : Create a procedure to display currency.

PROGRAM

```
create or replace procedure pr3(currency in out number)is
begin
currency:=currency*82;
end;
```

- Calling program

```
declare
currency number:=&currency;
begin
pr3(currency);
dbms_output.put_line(currency);
end;
```

OUTPUT

```
Enter value for currency: 2009
old   2: currency number:=&currency;
new   2: currency number:=2009;
164738

PL/SQL procedure successfully completed.
```

PROGRAM – 4

Aim : Create a procedure to find greatest of 3 numbers.

```
CREATE OR REPLACE PROCEDURE FindGreatest(num1 IN NUMBER, num2 IN NUMBER,
num3 IN NUMBER, result OUT NUMBER) AS
BEGIN
   IF num1 >= num2 AND num1 >= num3 THEN
      result := num1;
   ELSIF num2 >= num1 AND num2 >= num3 THEN
      result := num2;
   ELSE
      result := num3;
   END IF;
END FindGreatest;
```

- Calling program

```
DECLARE
   greatest_num NUMBER;
BEGIN
   FindGreatest(10, 25, 15, greatest_num);
   DBMS_OUTPUT.PUT_LINE('The greatest number is: ' || greatest_num);
END;
```

Dropping a procedure

- SQL> DROP PROCEDURE pr1;
  Procedure dropped.

EXPERIMENT NO. 9

PL/SQL PROGRAMS - FAMILIARIZATION WITH FUNCTIONS

AIM

To have familiarize with FUNCTIONS.

PROGRAM – 1

Aim : Create a function to display sum of two numbers .

PROGRAM

```
create or replace function fun1(a number,b number)
return number is
c number;
begin
c:=a+b;
return c;
end;
```

- Calling program

```
declare
x number:=&x;
y number:=&y;
z number;
begin
z:=fun1(x,y);
dbms_output.put_line('Sum is'||z);
end;
```

OUTPUT

```
Function created.

SQL> @prgm17calling.sql
  9  /
Enter value for x: 10
old   2: x number:=&x;
new   2: x number:=10;
Enter value for y: 20
old   3: y number:=&y;
new   3: y number:=20;
Sum is30
```

PROGRAM – 2

Aim : Create a function to display commission of a employee .

PROGRAM

```
create or replace function fun2(n number) return
number is
comm number;
s employee2.sal%type;
begin
select sal into s from employee2 where eno=n;
comm:=s*.1;
return comm;
end;
```

- Calling program

```
declare
n employee2.eno%type:=&n;
comm number;
begin
comm:=fun2(n);
dbms_output.put_line('Commission is'||comm);
end;
```

OUTPUT

```
Function created.

SQL> @prgm18calling.sql
  8  /
Enter value for n: 123
old   2: n employee2.eno%type:=&n;
new   2: n employee2.eno%type:=123;
Commission is123.3

PL/SQL procedure successfully completed.
```

EXPERIMENT NO. 10

INSTALLATION AND CONFIGURATION OF NOSQL DATABASES -MONGODB

AIM

To familiarize the installation and configuration of NoSQL database MONFODB.

How to Download & Install MongoDB on Windows :

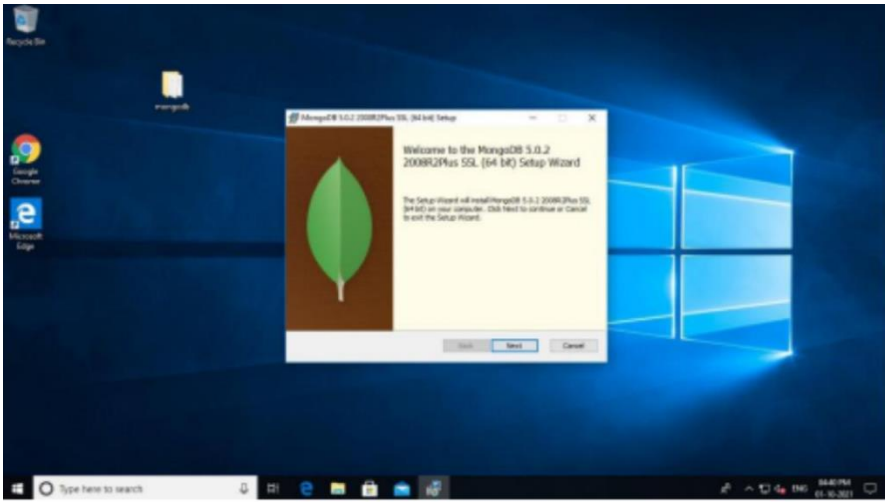
The installers for MongoDB are available in both the 32-bit and 64-bit format. The 32-bit installers are good for development and test environments. But for production environments you should use the 64-bit installers. Otherwise, you can be limited to the amount of data that can be stored within MongoDB.

Download & Install MongoDB on Windows:

The following steps can be used to install MongoDB on Windows 10:

Step 1:

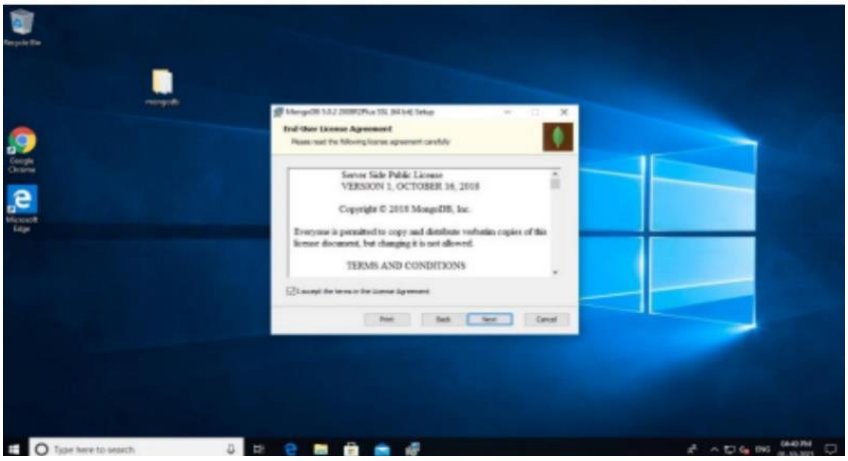
After downloading MongoDB, open the msi file and click next.



Step 2:

a. Accept the End-User License Agreement

b. Click Next

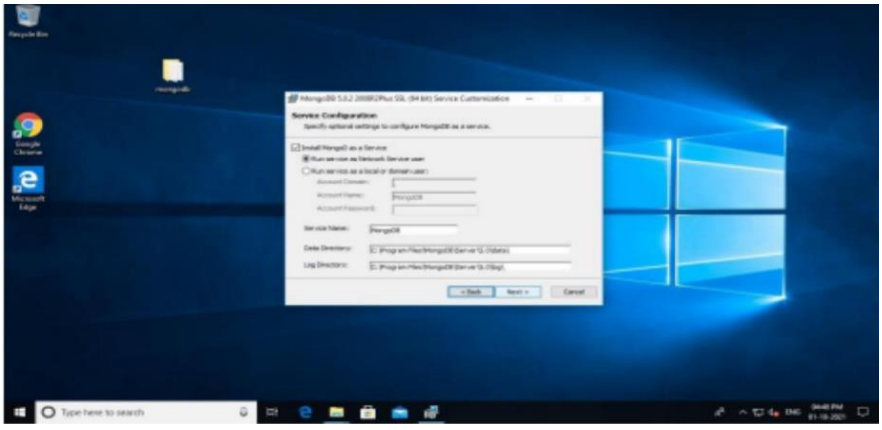


Step 3:

Click on the "complete" button to install all of the components. The custom option can be used to install selective components or if you want to change the location of the installation.

1. Select “Run service as Network Service user”. Make a note of the data directory, we’ll need this later.

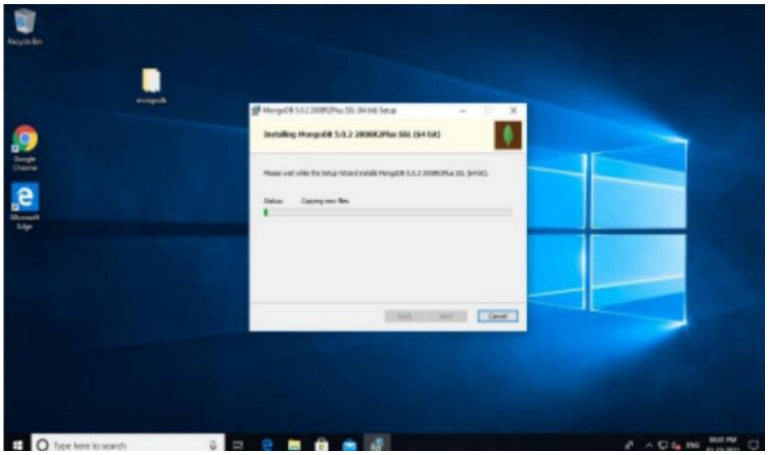
2. Click Next

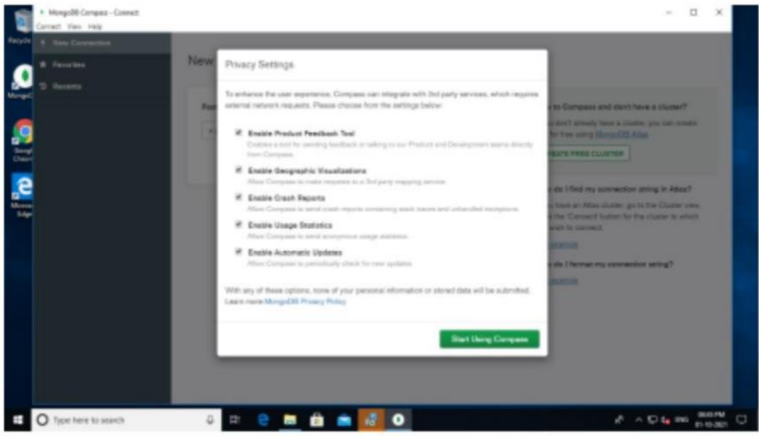


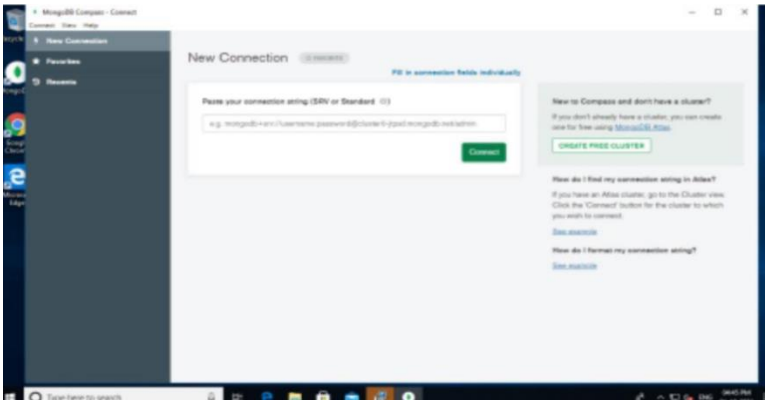
Step 4:

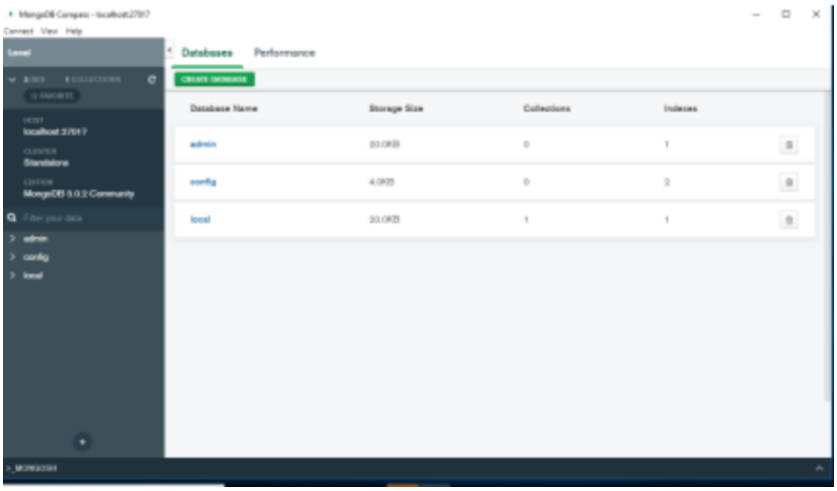
Click on the Install button to start the installation. Installation begins. Click next once completed.

Click on the Finish button to complete the installation.




Privacy Settings
Start Using Compass


New Connection
Connect


Databases
Performance
Database Name
Storage Size
Collections
Indexes
admin
config
local

EXPERIMENT NO. 11

QUERY PROCESSING

(a) Performing CRUD operations

1. Write a MongoDB query to create an INVOICE database and collections (CUSTOMER,PRODUCT) and its corresponding documents.

• use INVOICE

Output :

switched to db INVOICE

• db.createCollection("CUSTOMER")

Output :

{ "ok" : 1 }

•db.createCollection("PRODUCT")

Output :

{ "ok" : 1 }

• show collections

Output :

CUSTOMER

PRODUCT

• db.CUSTOMER.insertOne({name: "Anju",address: "Anju Bhavan",city: "Malapuram",age:"40"
,phone_no: "12356789"})

Output :

{ acknowledged: true,
insertedId: ObjectId("62a5f16bb99c322459d42f24") }

• db.CUSTOMER.insertMany([{name: "Sarath",address: "Sarath Nivas",city: "Calicut",age:"45"
,phone_no: "9886409789"},{name: "Nayana",address: "Nayans",city: "Kozhikode",age:"50"
,phone_no: "97890642"},{name: "jimin",address: "abcd",city: "kozhikode",age:"34",phone_no:
"9863665432"},{name: "Surya",address: "white house",city: "wandoor",age:"43",phone_no:
"9048857036"}])

Output :

{ acknowledged: true,
insertedIds:
{ '0': ObjectId("62a5f3d7b99c322459d42f25"),
'1': ObjectId("62a5f3d7b99c322459d42f26"),
'2': ObjectId("62a5f3d7b99c322459d42f27"),
'3': ObjectId("62a5f3d7b99c322459d42f28") } }

• db.PRODUCT.insertMany([{productName:"pen",price:"10",quantity:"10"},{productName:"pe
ncil",price:"10",quantity:"3"},{productName:"sharpner",price:"5",quantity:"4"},{productName
:"eraser",price:"8",quantity:"7"}])

Output :

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("613648ef86c70cdaad83dd7b"),

ObjectId("613648ef86c70cdaad83dd7c"),

ObjectId("613648ef86c70cdaad83dd7d"),

ObjectId("613648ef86c70cdaad83dd7e")

]

}

2.Write a MongoDB query to display all documents from the collection CUSTOMER

• db.CUSTOMER.find()

Output :

{ _id: ObjectId("62a5f16bb99c322459d42f24"),name: 'Anju', address: 'Anju Bhavan',

city: 'Malapuram', age: '40', phone_no: '12356789' }{ _id:

ObjectId("62a5f3d7b99c322459d42f25"), name: 'Sarath',address: 'Sarath Nivas', city: 'Calicut',age:

'45',phone_no: '9886409789' }{ _id: ObjectId("62a5f3d7b99c322459d42f26"),name: 'Nayana',

address: 'Nayans', city: 'Kozhikode',age: '50' phone_no: '97890642' }{ _id:

ObjectId("62a5f3d7b99c322459d42f27"), name: 'jimin',address: 'abcd', city: 'kozhikode', age: '34',

phone_no: '9863665432' }{ _id: ObjectId("62a5f3d7b99c322459d42f28"),name: 'Surya', address:

'white house', city: 'wandoor',age: '43', phone_no: '9048857036' }

3.Write a MongoDB query to show all documents from the collection CUSTOMER in

formatted way

• db.CUSTOMER.find().pretty()

Output :

{ _id: ObjectId("62a5f16bb99c322459d42f24"),

name: 'Anju',

address: 'Anju Bhavan',

city: 'Malapuram',

age: '40',

phone_no: '12356789' }

{ _id: ObjectId("62a5f3d7b99c322459d42f25"),

name: 'Sarath',

address: 'Sarath Nivas',

city: 'Calicut',

age: '45',

phone_no: '9886409789' }

{ _id: ObjectId("62a5f3d7b99c322459d42f26"),

name: 'Nayana',

address: 'Nayans',

city: 'Kozhikode',

age: '50',

phone_no: '97890642' }

{ _id: ObjectId("62a5f3d7b99c322459d42f27"),

name: 'jimin',

address: 'abcd',

city: 'kozhikode',

age: '34',

phone_no: '9863665432' }

{ _id: ObjectId("62a5f3d7b99c322459d42f28"),

name: 'Surya',

address: 'white house',

city: 'wandoor',

age: '43',

phone_no: '9048857036' }

4.Write a MongoDB query to show all documents from the collection PRODUCT

• db.PRODUCT.find().pretty()

Output :

{ _id: ObjectId("62a5f472b99c322459d42f29"),

productName: 'pen',

price: '10',

quantity: '10' }

{ _id: ObjectId("62a5f472b99c322459d42f2a"),

productName: 'pencil',

price: '10',

quantity: '3' }

{ _id: ObjectId("62a5f472b99c322459d42f2b"),

productName: 'sharpner',

price: '5',

quantity: '4' }

{ _id: ObjectId("62a5f472b99c322459d42f2c"),

productName: 'eraser',

price: '8',

quantity: '7' }

5.Write a MongoDB query to update the city wandoor to kochi in the collection CUSTOMER and display the updated value .

• db.CUSTOMER.updateOne({"city":"wandoor"},{$set:{"city":"kochi"}})

Output :

{ acknowledged: true,

insertedId: null,

matchedCount: 0,

modifiedCount: 0,

upsertedCount: 0 }

• db.CUSTOMER.find({"city":"kochi"}).pretty()

Output :

{ _id: ObjectId("62a5f3d7b99c322459d42f28"),

name: 'Surya',

address: 'white house',

city: 'kochi',

age: '43',

phone_no: '9048857036' }

6.Write a MongoDB query to delete the first matched document whose city is “Calicut” and display the documents

• db.CUSTOMER.deleteOne({city:"Calicut"})

Output :

{ "acknowledged" : true, "deletedCount" : 1 }

• db.CUSTOMER.find().pretty()

Output :

{ _id: ObjectId("62a5f16bb99c322459d42f24"),

name: 'Anju',

address: 'Anju Bhavan',

city: 'Malapuram',

age: '40',

phone_no: '12356789' }

{ _id: ObjectId("62a5f3d7b99c322459d42f26"),

name: 'Nayana',

address: 'Nayans',

city: 'Kozhikode',

age: '50',

phone_no: '97890642' }

{ _id: ObjectId("62a5f3d7b99c322459d42f27"),

name: 'jimin',

address: 'abcd',

city: 'kozhikode',

age: '34',

phone_no: '9863665432' }

{ _id: ObjectId("62a5f3d7b99c322459d42f28"),

EXPERIMENT NO. 12

NoSQL – RETRIEVING DATA

1.Write a MongoDB query to display the customer documents having city ‘kozhikode’ or ‘Malappuram’.

• use INVOICE

switched to db INVOICE

• db.CUSTOMER.find({$or:[{"city":"Kozhikode"},{"city":"Malapuram"}]}).pretty()

Output :

{ _id: ObjectId("62a5f16bb99c322459d42f24"),

name: 'Anju',

address: 'Anju Bhavan',

city: 'Malapuram',

age: '40',

phone_no: '12356789' }

{ _id: ObjectId("62a5f3d7b99c322459d42f26"),

name: 'Nayana',

address: 'Nayans',

city: 'Kozhikode',

age: '50',

phone_no: '97890642' }

2.Write a MongoDB query to display all customer documents who have age <50.

• db.CUSTOMER.find({"age":{$lt:"50"}}).pretty()

Output :

{ _id: ObjectId("62a5f16bb99c322459d42f24"),

name: 'Anju',

address: 'Anju Bhavan',

city: 'Malapuram',

age: '40',

phone_no: '12356789' }

{ _id: ObjectId("62a5f3d7b99c322459d42f27"),

name: 'jimin',

address: 'abcd',

city: 'kozhikode',

age: '34',

phone_no: '9863665432' }

{ _id: ObjectId("62a5f3d7b99c322459d42f28"),

name: 'Surya',

address: 'white house',

city: 'kochi',

age: '43',

phone_no: '9048857036' }

3.Write a MongoDB query to display all customer documents who have age >=40.

• db.CUSTOMER.find({"age":{$gte:"40"}}).pretty()

Output :

{ _id: ObjectId("62a5f16bb99c322459d42f24"),

name: 'Anju',

address: 'Anju Bhavan',

city: 'Malapuram',

age: '40',

phone_no: '12356789' }

{ _id: ObjectId("62a5f3d7b99c322459d42f26"),

name: 'Nayana',

address: 'Nayans',

city: 'Kozhikode',

age: '50',

phone_no: '97890642' }

{ _id: ObjectId("62a5f3d7b99c322459d42f28"),

name: 'Surya',

address: 'white house',

city: 'kochi',

age: '43',

phone_no: '9048857036' }

4.Write a MongoDB query to find customer documents who not live in ‘kozhikode’.

• db.CUSTOMER.find({"city":{$ne:"kozhikode"}}).pretty()

Output :

{ _id: ObjectId("62a5f16bb99c322459d42f24"),

name: 'Anju',

address: 'Anju Bhavan',

city: 'Malapuram',

age: '40',

phone_no: '12356789' }

{ _id: ObjectId("62a5f3d7b99c322459d42f26"),

name: 'Nayana',

address: 'Nayans',

city: 'Kozhikode',

age: '50',

phone_no: '97890642' }

{ _id: ObjectId("62a5f3d7b99c322459d42f28"),

name: 'Surya',

address: 'white house',

city: 'kochi',

age: '43',

phone_no: '9048857036' }

5.Write a MongoDB query to display the customer documents having neither city ‘kochi’ nor ‘Malappuram’.

• db.CUSTOMER.find({$nor:[{"city":"kochi"},{"city":"Malapuram"}]}).pretty()

Output :

{ _id: ObjectId("62a5f3d7b99c322459d42f26"),

name: 'Nayana',

address: 'Nayans',

city: 'Kozhikode',

age: '50',

phone_no: '97890642' }

{ _id: ObjectId("62a5f3d7b99c322459d42f27"),

name: 'jimin',

address: 'abcd',

city: 'kozhikode',

age: '34',

phone_no: '9863665432' }

6.Write a MongoDB query to display the distinct name of the product in PRODUCTION collection

• db.PRODUCT.distinct("productName")

Output :

['eraser', 'pen', 'pencil', 'sharpner'

EXPERIMENT NO. 13

MONGODB-AGGREGATE FUNCTIONS

1.Write a MongoDB query to sort customer details in ascending order of their name.

• db.CUSTOMER.find().sort({name:1}).pretty()

Output :

{ _id: ObjectId("62a5f16bb99c322459d42f24"),

name: 'Anju',

address: 'Anju Bhavan',

city: 'Malapuram',

age: '40',

phone_no: '12356789' }

{ _id: ObjectId("62a5f3d7b99c322459d42f27"),

name: 'Jimin',

address: 'abcd',

city: 'kozhikode',

age: '34',

phone_no: '9863665432' }

{ _id: ObjectId("62a5f3d7b99c322459d42f26"),

name: 'Nayana',

address: 'Nayans',

city: 'Kozhikode',

age: '50',

phone_no: '97890642' }

{ _id: ObjectId("62a5f3d7b99c322459d42f28"),

name: 'Surya',

address: 'white house',

city: 'kochi',

age: '43',

phone_no: '9048857036' }

2.Write a MongoDB query to count number of city in CUSTOMER.

• db.CUSTOMER.aggregate([{ $count: "city" }])

Output :

{ city: 4 }

3.Write a MongoDB query to count number of customers in each city.

•db.CUSTOMER.aggregate([{$group:{_id: "$city",total:{$sum:1}}}])

Output :

{ _id: 'Kozhikode', total: 1 }

{ _id: 'kochi', total: 1 }

{ _id: 'Malapuram', total: 1 }

{ _id: 'kozhikode', total: 1 }

4.Write a MongoDB query to find minimum and maximum on CUSTOMER collection.

• db.CUSTOMER.aggregate([{$group:{_id:"",min_age:{$min:"$age"},max_age:{$max:"$age"}

}}])

Output :

{ _id: '', min_age: '34', max_age: '50' }

5.Write a MongoDB query to find minimum and maximum age in each city.

• db.CUSTOMER.aggregate([{$group:{_id:"$city",min_age:{$min:"$age"},max_age:{$max:"$

age"}}}])

{ _id: 'kozhikode', min_age: '34', max_age: '34' }

{ _id: 'kochi', min_age: '43', max_age: '43' }

{ _id: 'Kozhikode', min_age: '50', max_age: '50' }

{ _id: 'Malapuram', min_age: '40', max_age: '40' }

EXPERIMENT NO. 14

MONGODB-CREATE USERS AND ROLES

1.Write a MongoDB query to creates a user named reportsUser in the admin database but does not yet assign roles:.

• use student

• db.createUser({ user: "reportsUser", pwd: passwordPrompt(), roles: [ ] }

{ ok: 1 }

2.Write a MongoDB query to creates a user named appAdmin in the student database and gives the user readWrite access to the config database.

• db.createUser({ user: "appAdmin",pwd: passwordPrompt(), roles:[{ role: "readWrite", db: "config" }]})

{ ok: 1 }

3.Write a MongoDB query to creates a user named restricted in the student database. This user may only authenticate if connecting from IP address 192.0.2.0 to IP address 198.51.100.0

• db.createUser({user: "restricted", pwd: passwordPrompt(), roles: [{ role: "readWrite", db: "reporting" }],authenticationRestrictions: [{clientSource: ["192.0.2.0"], serverAddress: ["198.51.100.0"]}]})

{ ok: 1 }

4.Write a MongoDB query to drops the reportUser user on the student database.

• db.dropUser("reportsUser", {w: "majority", wtimeout: 5000})

{ ok: 1 }

5.Write a MongoDB query to operations return information about an example appAdmin user in an student database.

• db.getUser("appAdmin")

{

_id: 'admin.appAdmin',

userId: UUID("60cfa9b0-d876-44e4-be1f-928a75532865"),

user: 'appAdmin',

db: 'admin',

roles: [

{ role: 'clusterAdmin', db: 'admin' },

{ role: 'readWrite', db: 'config' }

],

mechanisms: ['SCRAM-SHA-1', 'SCRAM-SHA-256']

}

6.Write a MongoDB query to gives appAdmin the readWrite role on the products database and the read role on the student database.

• db.grantRolesToUser("appAdmin",["readWrite" , { role: "read", db: "student" }],{ w: "majority" , wtimeout: 4000 })

{ ok: 1 }

7.Write a MongoDB query to method removes the user's roles: the read role and the readWrite role on the student database.

• db.revokeRolesFromUser("appAdmin",[{ role: "read", db: "student" }, "readWrite"],{ w: "majority" })

{ ok: 1 }