DBMS MANUAL ANSWERS

S.KARTHICK,
2ND YEAR CSE,
BATCH 2025

CONTENTS

1.DDL COMMANDS

```
1)create table department(depno number(2)
primary key,depname varchar2(15),deplocation
varchar(10));

2)create table employee(empno number(5) primary
key,empname varchar(20),designation
varchar(10),date_of_join Date NOT NULL,salary
decimal(9,2),depno references
department(depno));

3)create table course(coursecode number(2)
primary key,coursename varchar(15));
                    (or)
create table course1(coursecode
number(2),coursename varchar(15),constraint
coursecode1 primary key(COURSECODE));


4)create table student(rollno number(5) primary
key,name varchar(15),coursecode references
course(coursecode),mark1 number(3)
check(mark1>=0 and mark1<=100),mark2 number(3)
check(mark2>=0 and mark2<=100));
                      (or)
create table student2(rollno
number(15),constraint depno1 depno references
department(depno),constraint rollno1 primary
key(rollno));


5)desc employee;
desc student;
desc department;
```

```
desc course;

6)alter table employee add grade varchar(1);
alter table employee add phoneno decimal(10,0);
                (or)
alter table employee add(phoneno
decimal(10,0),grade varchar(1));

7)alter table department modify deplocation
varchar(15);

8)alter table student modify rollno
varchar(20);

9)alter table student drop column mark2;

10)select *from tab;

11)truncate table student;

12)drop table student;

13)alter table employee rename to emp;

14)create table student(name varchar(15) not
null);

15)alter table emp add work_location
varchar(15) default 'loc';
```

2.DML COMMANDS

```
1) SQL> insert into department
values(01,'sales','block a');

SQL> insert into department
values(02,'PURCHASE','BLOCK D');

1 row created.

SQL> insert into department
values(03,'PRODUCTION','BLOCK B');

1 row created.

SQL> insert into department
values(04,'MARKETING','BLOCK A');

1 row created.

SQL> insert into department
values(05,'ACCOUNTS','BLOCK C');

1 row created.

SQL> insert into department
values(06,'SOFTWARE','BLOCK E');

1 row created.


2) SQL> insert into
employee(empno,empname,designation,date_of_join,salar
y,depno,grade) values(1,'Siva','manager','05-Oct-
1987',15000.00,05,'A');

1 row created.
```

```
SQL> insert into employee
values(2,'Mani','salesman','12-Apr-
1987',5000.75,01,'F');

1 row created.

SQL> insert into employee
values(3,'Raju','Clerk','30-Nov-
1989',7000.00,02,'E');

1 row created.

SQL> insert into employee
values(4,'Babu','Clerk','04-Jan-
1995',5000.00,03,'E');

1 row created.

SQL> insert into employee
values(5,'Ram','salesman','08-Dec-
2000',3000.25,01,'F');

1 row created.

SQL> insert into employee
values(6,'Velu','programmer','24-Feb-
2002',10000.50,06,'D');

1 row created.

SQL> insert into employee
values(7,'Ravi','accountant','12-Sep-
1991',8000.25,05,'G');

1 row created.

SQL> insert into employee
values(8,'Balan','manager','07-Jun-
1993',12000.75,03,'A');

1 row created.
```

```
SQL> insert into employee
values(9,'Mahesh','Officer','18-Mar-
1997',10000.50,02,'B');

1 row created.

SQL> insert into employee
values(10,'Kumar','Analyst','15-Jan-
1995',14500.00,06,'C');

1 row created.
```

3) SQL> select * from department;

DEPNO	DEPNAME	DEPLOCATION
10	15	20
1	sales	block a
2	PURCHASE	BLOCK D
3	PRODUCTION	BLOCK B
4	MARKETING	BLOCK A
5	ACCOUNTS	BLOCK C
6	SOFTWARE	BLOCK E

7 rows selected.

SQL> select * from employee;

EMPNO	EMPNAME	DESIGNATIO	DATE_OF_J	SALARY	DEPNO	GRADE
1	Siva	manager	05-OCT-87	15000	5	A
2	Mani	salesman	12-APR-87	5000.75	1	F
3	Raju	Clerk	30-NOV-89	7000	2	E
4	Babu	Clerk	04-JAN-95	5000	3	E
5	Ram	salesman	08-DEC-00	3000.25	1	F
6	Velu	programmer	24-FEB-02	10000.5	6	D
7	Ravi	accountant	12-SEP-91	8000.25	5	G
8	Balan	manager	07-JUN-93	12000.75	3	A
9	Mahesh	Officer	18-MAR-97	10000.5	2	B
10	Kumar	Analyst	15-JAN-95	14500	6	C

10 rows selected.

Note: For proper display of the tables use the below command.

```
SQL> set linesize 200;
```

```
4) SQL> update employee set phoneno=9988776655 where
empno=1;

1 row updated.
SQL> update employee set phoneno=9786756453 where
empno=2;

1 row updated.

SQL> update employee set phoneno=8967453212 where
empno=3;

1 row updated.

SQL> update employee set phoneno=9563423788 where
empno=4;

1 row updated.

SQL> update employee set phoneno=9453423122 where
empno=5;

1 row updated.

SQL> update employee set phoneno=7866734109 where
empno=6;

1 row updated.

SQL> update employee set phoneno=9094534321 where
empno=7;

1 row updated.

SQL> update employee set phoneno=9006744309 where
empno=8;

1 row updated.

SQL> update employee set phoneno=7098967545 where
empno=9;
```

```
1 row updated.

SQL> update employee set phoneno=9923456786 where
empno=10;

1 row updated.
SQL> select empno,phoneno from employee;

     EMPNO    PHONENO
---------- ----------
         1 9988776655
         2 9786756453
         3 8967453212
         4 9563423788
         5 9453423122
         6 7866734109
         7 9094534321
         8 9006744309
         9 7098967545
        10 9923456786

10 rows selected.

5)SQL> create table emp as select * from employee;

Table created.

6)selectempname,designation,trunc(months_betwee
n(sysdate,date_of_join)/12) as year_of_service
from employee;

EMPNAME              DESIGNATION          YEAR_OF_SERVICE
-------------------- -------------------- ---------------
siva                 manager                           35
mani                 salesman                          35
raju                 clerk                             33
babu                 clerk                             28
ram                  salesman                          22
velu                 programmer                        20
ravi                 accountant                        31
balan                manager                           29
mahesh               officer                           25
kumar                analyst                           28
```

10 rows selected.

7) create view emp_info as select empname,designation,depno from employee;

View created.

SQL> select *from emp_info;

EMPNAME	DESIGNATION	DEPNO
siva	manager	5
mani	salesman	1
raju	clerk	2
babu	clerk	3
ram	salesman	1
velu	programmer	6
ravi	accountant	5
balan	manager	3
mahesh	officer	2
kumar	analyst	6

10 rows selected.

8) SQL> select empname from employee where salary=10000.50;

EMPNAME

Velu
Mahesh

9) SQL> select empname from employee where depno<>01;

EMPNAME

Siva
Raju
Babu
Velu
Ravi
Balan
Mahesh

```
Kumar

8 rows selected.
```

10)SQL> select empname from employee where date_of_join>'04-Apr-1995';

```
EMPNAME
--------------------
Ram
Velu
Mahesh
```

11)select empname from employee where depno in (2,3,5);

```
EMPNAME
--------------------
Siva
Raju
Babu
Ravi
Balan
Mahesh

6 rows selected.
```

12)SQL> select empname from employee where date_of_join between '01-Jan-1988' and '01-Jan-1998';

```
EMPNAME
--------------------
Raju
Babu
Ravi
Balan
Mahesh
Kumar

6 rows selected.
```

13)SQL> select * from employee where empname like 'R%';

```
     EMPNO EMPNAME               DESIGNATIO
DATE_OF_J     SALARY      DEPNO GRADE
---------- -------------------- ---------- --------
- ---------- ---------- -----
   PHONENO YEAR_OF_S
---------- ---------
         3 Raju                 Clerk      30-NOV-
89       7000          2 E
8967453212

         5 Ram                  salesman   08-DEC-
00    3000.25          1 F
9453423122

         7 Ravi                 accountant 12-SEP-
91    8000.25          5 G
9094534321
```

14) select * from employee where phoneno IS NULL;

no rows selected

15) select empname from employee where depno=01 and salary=5000.75;

```
EMPNAME
--------------------
Mani
```

16) SQL> select empname from employee where depno=01 or salary=5000.75;

```
EMPNAME
--------------------
Mani
Ram
```

17) SQL> select * from employee where not designation='manager' and not designation='Officer';

```
     EMPNO EMPNAME               DESIGNATIO
DATE_OF_J     SALARY       DEPNO GRADE
---------- -------------------- ---------- --------
- ---------- ---------- -----
   PHONENO YEAR_OF_S
---------- ---------
         2 Mani                 salesman   12-APR-
87    5000.75          1 F
9786756453

         3 Raju                 Clerk      30-NOV-
89       7000          2 E
8967453212

         4 Babu                 Clerk      04-JAN-
95       5000          3 E
9563423788


     EMPNO EMPNAME               DESIGNATIO
DATE_OF_J     SALARY       DEPNO GRADE
---------- -------------------- ---------- --------
- ---------- ---------- -----
   PHONENO YEAR_OF_S
---------- ---------
         5 Ram                  salesman   08-DEC-
00    3000.25          1 F
9453423122

         6 Velu                 programmer 24-FEB-
02    10000.5          6 D
7866734109

         7 Ravi                 accountant 12-SEP-
91    8000.25          5 G
9094534321
```

```
     EMPNO EMPNAME               DESIGNATIO
DATE_OF_J     SALARY       DEPNO GRADE
---------- -------------------- ---------- --------
- ---------- ---------- -----
   PHONENO YEAR_OF_S
---------- ---------
        10 Kumar                Analyst    15-JAN-
95      14500          6 C
9923456786
```

18) SQL> select depname from department order by depname desc;

```
DEPNAME
---------------
sales
SOFTWARE
PURCHASE
PRODUCTION
MARKETING
ACCOUNTS

7 rows selected.
```

19)SQL> delete from employee where date_of_join < '01-Jan-1989';

```
2 rows deleted.
```

3.BUID IN FUNCTIONS

1)SQL> select designation, length(designation) as length from employee;

```
DESIGNATIO     LENGTH
---------- ----------
officer             7
salesman            8
Clerk               5
Analyst             7
manager             7
accountant         10
programmer         10
salesman            8
Clerk               5
manager             7

10 rows selected.
```

2) SQL> select substr(depname ,1,3) from department;

```
SUBSTR(DEPNA
------------
sal
PUR
PRO
MAR
ACC
```

```
SOF

6 rows selected.


3) SQL> select replace(deplocation
,'BLOCK A','block x') from department;

REPLACE(DEPLOCATION,'BLOCKA','BLOCKX')
----------------------------------------
----------------------------------------
block a
BLOCK D
BLOCK B
block x
BLOCK C
BLOCK E

6 rows selected.


4) SQL> select replace('oooabc','o', '')
from dual;

REP
---
Abc


5) SQL> select distinct designation from
employee;

DESIGNATIO
----------
```

```
Clerk
Analyst
officer
salesman
programmer
accountant
manager

7 rows selected.

6) SQL> select lower(depname) from
department;

LOWER(DEPNAME)
---------------
sales
purchase
production
marketing
accounts
software

6 rows selected.

7) SQL> select upper(depname) from
department;

UPPER(DEPNAME)
---------------
SALES
PURCHASE
PRODUCTION
MARKETING
ACCOUNTS
```

SOFTWARE

6 rows selected.

8) SQL> select
translate(designation,grade,designation) from empl;

```
TRANSLATE(
----------
salesman
manager
Clerk
Clerk
salesman
programmer
accountant
manager
Officer
Analyst

10 rows selected.
```

9)SQL> select count(empname) from
employee;

```
COUNT(EMPNAME)
--------------
            10
```

10)SQL> select depno,min(salary) as
minimum,max(salary) as maximum from
employee group by depno order by depno;

```
     DEPNO    MINIMUM    MAXIMUM
---------- ---------- ----------
         1    3000.25    5000.75
```

```
         2       7000    10000.5
         3       5000   12000.75
         5       5000    8000.25
         6    10000.5      14500
```

11)SQL> select sum(salary) from employee;

```
SUM(SALARY)
-----------

      79503
```

12) SQL> select avg(salary)from employee;

```
AVG(SALARY)
-----------
     7950.3
```

13)SQL> select round(salary)from
employee;

```
ROUND(SALARY)
-------------
        10001
         5001
         7000
        14500
        12001
         8000
        10001
         3000
```

```
          5000
          5000

10 rows selected.
```

14)SQL> select trunc(salary)from employee;

```
TRUNC(SALARY)
-------------
         10000
          5000
          7000
         14500
         12000
          8000
         10000
          3000
          5000
          5000

10 rows selected.
```

15)SQL> select TO_CHAR(SYSDATE,'DDth Month YYYY')todays_date from dual;

```
TODAYS_DATE
-------------------------------------------
-----
14TH March     2023
```

```
16)SQL> select TO_CHAR(TO_DATE('20
september 1996','DD Month YYYY'),'DD-MON-
YYY')todays_date  from dual;

TODAYS_DAT
----------
20-SEP-996



17)SQL> select
empname,add_months(date_of_join,2) as
after_2months from emp;

EMPNAME              AFTER_2MO
-------------------- ---------
mahesh               18-MAY-97
Mani                 12-JUN-87
Raju                 31-JAN-90
Kumar                15-MAR-95
Balan                07-AUG-93
Ravi                 12-NOV-91
Velu                 24-APR-02
Ram                  08-FEB-01
Babu                 04-MAR-95
siva                 04-DEC-87

10 rows selected.

18)SQL>
selectlast_day(date_of_join),empname from
emp;

LAST_DAY( EMPNAME
```

```
--------- --------------------
31-MAR-97 mahesh
30-APR-87 Mani
30-NOV-89 Raju
31-JAN-95 Kumar
30-JUN-93 Balan
30-SEP-91 Ravi
28-FEB-02 Velu
31-DEC-00 Ram
31-JAN-95 Babu
31-OCT-87 siva

10 rows selected.
```

19)select trunc(Month_Between(sysdate,date_of_join)) from employee;

20)SQL> select next_day(sysdate,'friday') as d from dual;

```
D
---------
17-MAR-23
```

21)SQL> select to_char(date '2000-01-01','day')from dual;

```
TO_CHAR(D
---------
saturday
```

4.JOINS,SETS AND SUBQUERIES

JOINS

```
1) SQL> select empl.empname,department.depname
from empl,department where
empl.depno=department.depno;

EMPNAME              DEPNAME
-------------------- ---------------
Mani                 sales
Siva                 ACCOUNTS
Raju                 PURCHASE
Babu                 PRODUCTION
Ram                  sales
Velu                 SOFTWARE
Ravi                 ACCOUNTS
Balan                PRODUCTION
Mahesh               PURCHASE
Kumar                SOFTWARE

10 rows selected.
                    (or)
SQL> select empname,depname from
empl,department where
empl.depno(+)=department.depno;

EMPNAME              DEPNAME
-------------------- ---------------
Mani                 sales
Ram                  sales
Raju                 PURCHASE
Mahesh               PURCHASE
Babu                 PRODUCTION
Balan                PRODUCTION
                     MARKETING
Ravi                 ACCOUNTS
```

```
Siva                 ACCOUNTS
Kumar                SOFTWARE
Velu                 SOFTWARE

11 rows selected.
```

2) SQL> select x.empname,x.salary from empl x,empl y where y.empname='Raju' and x.salary>y.salary;

```
EMPNAME                  SALARY
-------------------- ----------
Siva                      15000
Velu                    10000.5
Ravi                    8000.25
Balan                  12000.75
Mahesh                  10000.5
Kumar                     14500

6 rows selected.
```

3)SQL> select e.empname,d.depname from empl e full outer join department d on e.depno=d.depno;

```
EMPNAME              DEPNAME
-------------------- --------------
Mani                 sales
Siva                 ACCOUNTS
Raju                 PURCHASE
Babu                 PRODUCTION
Ram                  sales
Velu                 SOFTWARE
Ravi                 ACCOUNTS
Balan                PRODUCTION
Mahesh               PURCHASE
Kumar                SOFTWARE
                     MARKETING
```

11 rows selected.

4) SQL> select emp.name,dept.dname from emp,dept where dept.deptno in (10,20,11);

```
NAME       DNAME
---------- ----------
Indira     Slaes
Indira     Security
Indira     Manager
Jumana     Slaes
Jumana     Security
Jumana     Manager
Murshi     Slaes
Murshi     Security
Murshi     Manager
Mahesh     Slaes
Mahesh     Security

NAME       DNAME
---------- ----------
Mahesh     Manager
Shobi      Slaes
Shobi      Security
Shobi      Manager
```

SET OPERATORS

1)SQL> select * from product1 union select * from product2;

```
ITEM_ ITEM_NAME
----- ----------
co1   kitkat
co2   perk
```

```
co3   fivestaar
co4   hajmola
co5   tublorone
go1   sonata
go2   titan
go3   fasttrack
go4   rolex
go5   noise
```

2)SQL> select * from product1 union all select * from product2;

```
ITEM_ ITEM_NAME
----- ----------
co1   kitkat
co2   perk
co3   fivestaar
co4   hajmola
co5   tublorone
go1   sonata
go2   titan
go3   fasttrack
go4   rolex
go5   noise
co1   kitkat
```

3)SQL> select item_name from product1 intersect select item_name from product2;

```
ITEM_NAME
----------
kitkat
```

4)SQL> select item_name,item_code from product1 minus select item_name,item_code from product2;

```
ITEM_NAME  ITEM_
---------- -----
fivestaar  co3
hajmola    co4
perk       co2
tublorone  co5
```

SUB QUERIES

1)SQL> select empname,salary from empl where salary=(select max(salary) from empl);

EMPNAME	SALARY
Siva	15000

2)SQL> select empname,salary from empl where salary>any(select avg(salary) from empl);

EMPNAME	SALARY
Siva	15000
Velu	10000.5
Balan	12000.75
Mahesh	10000.5
Kumar	14500

3)SQL> select empname,salary from empl where salary<all(select avg(salary) from empl);

EMPNAME	SALARY
Mani	5000.75
Raju	7000
Babu	5000
Ram	3000.25
Ravi	8000.25

```
4)SQL> select dname from dept where
exists(select deptno from emp where
dept.deptno=emp.deptno);

DNAME
----------
Slaes
Security
Manager
Delivery

5)SQL> select empname from employee where
exists(select depno from department where
department.depno=employee.depno and
department.depname='soft');

EMPNAME
----------
velu
kumar


6)SQL> select job,deptno,count(*) as empnum
from emp group by job,deptno having count(*)>2;
```

5.Views,synonyms and sequences

```
1)SQL> create view emp_desg as select
empname,designation,depname from
employee,department where
employee.depno=department.depno;

View created.
2)SQL> select *from emp_desg;

EMPNAME    DESIGNATIO DEPNAME
---------- ---------- --------------
Siva       manager    a
Mani       salesman   s
Raju       Clerk      pc
Babu       Clerk      p
Ram        salesman   s
Velu       programmer soft
Ravi       accountant a
Balan      manager    p
Mahesh     Officer    pc
Kumar      Analyst    soft

10 rows selected.
3)SQL> create sequence seq_emp start with
11 increment by 1 maxvalue 13 nocycle;

Sequence created.
4)SQL> insert into employee
values(seq_emp.nextval,'raghul','programm
er','20-JUN-1995',7500.05,03,'A');
```

```
1 row created.
5)SQL> create synonym s_emp for employee;

Synonym created.
6)SQL> select *from s_emp;

     EMPNO EMPNAME    DESIGNATIO
DATE_OF_J     SALARY       DEPNO G
---------- ---------- ---------- --------
- ---------- ---------- -
         1 Siva       manager    05-OCT-
87      15000          5 A
         2 Mani       salesman   12-APR-
87    5000.75          1 F
         3 Raju       Clerk      30-NOV-
89       7000          2 E
         4 Babu       Clerk      04-JAN-
95       5000          3 E
         5 Ram        salesman   08-DEC-
00    3000.25          1 F
         6 Velu       programmer 24-FEB-
02    10000.5          6 D
         7 Ravi       accountant 12-SEP-
91    8000.25          5 G
         8 Balan      manager    07-JUN-
93   12000.75          3 A
         9 Mahesh     Officer    18-MAR-
97    10000.5          2 B
        10 Kumar      Analyst    15-JAN-
95      14500          6 C
        13 raghul     programmer 20-JUN-
95    7500.05          3 A
```

```
11 rows selected.
7)SQL> update s_emp set phneno=987645534
where empname='Balan';

1 row updated.
8)SQL> create index using on
employee(empno,salary);

Index created.
```

Note: to delete the created index

```
SQL> drop index using;

Index dropped.



SQL> create index ind on
employee(empno,salary);

Index created.
```

6.TCL AND DCL COMMANDS

1)SQL> savepoint s3;

Savepoint created.

2)SQL> delete employee where empname like 'R%';

3 rows deleted.

3)SQL> select *from employee;

```
     EMPNO EMPNAME    DESIGNATIO
DATE_OF_J     SALARY       DEPNO G
PHNENO
---------- ---------- ---------
- ---------- ---------- - ----------
         1 Siva       manager    05-OCT-
87      15000          5 A
         2 Mani       salesman   12-APR-
87    5000.75          1 F
         4 Babu       Clerk      04-JAN-
95       5000          3 E
         6 Velu       programmer 24-FEB-
02    10000.5          6 D
         8 Balan      manager    07-JUN-
93   12000.75          3 A  987645534
         9 Mahesh     Officer    18-MAR-
97    10000.5          2 B
```

```
        10 Kumar       Analyst    15-JAN-
95       14500           6 C
        13 raghul      programmer 20-JUN-
95    7500.05           3 A

8 rows selected.


4) SQL> savepoint s4;

Savepoint created.

5) SQL> delete employee where empname
like 'S%';

1 row deleted.

6) SQL> select *from employee;

     EMPNO EMPNAME    DESIGNATIO
DATE_OF_J     SALARY       DEPNO G
PHNENO
---------- ---------- ---------- --------
- ---------- ---------- - ----------
         2 Mani        salesman   12-APR-
87    5000.75           1 F
         4 Babu        Clerk      04-JAN-
95       5000           3 E
         6 Velu        programmer 24-FEB-
02    10000.5           6 D
         8 Balan       manager    07-JUN-
93   12000.75           3 A  987645534
         9 Mahesh      Officer    18-MAR-
97    10000.5           2 B
```

```
        10 Kumar       Analyst    15-JAN-
95      14500           6 C
        13 raghul      programmer 20-JUN-
95    7500.05           3 A

7 rows selected.


7)SQL> rollback to savepoint s4;

Rollback complete.

8)SQL> select *from employee;

     EMPNO EMPNAME    DESIGNATIO
DATE_OF_J     SALARY       DEPNO G
PHNENO
---------- ---------- ---------- --------
- ---------- ---------- - ----------
         1 Siva       manager    05-OCT-
87      15000           5 A
         2 Mani       salesman   12-APR-
87    5000.75           1 F
         4 Babu       Clerk      04-JAN-
95       5000           3 E
         6 Velu       programmer 24-FEB-
02    10000.5           6 D
         8 Balan      manager    07-JUN-
93   12000.75           3 A  987645534
         9 Mahesh     Officer    18-MAR-
97    10000.5           2 B
        10 Kumar      Analyst    15-JAN-
95      14500           6 C
```

```
        13 raghul     programmer 20-JUN-
95    7500.05           3 A

8 rows selected.

9)SQL> commit;

Commit complete.
SQL> delete employee where empname='S%';

0 rows deleted.


10)SQL> create user user3 identified by
user3;

User created.

SQL> create user user4 identified by
user4;

User created.

11)SQL> grant create session to user3;

Grant succeeded.

SQL> grant create session to user4;

Grant succeeded.
12)SQL> grant create table to user3;

Grant succeeded.
```

```
SQL> grant insert any table to user3;

Grant succeeded.
SQL> create table stud(rollno
number(5),name varchar2(20));

Table created.

SQL> insert into stud values(1,'siva');

1 row created.

SQL> insert into stud values(2,'velu');

1 row created.


13)SQL> grant insert on stud to user3;

Grant succeeded.

SQL> grant select on stud to user3;

Grant succeeded.
   SQL> grant insert,select,update,delete
on stud to user4;

Grant succeeded.
    SQL> insert into student16.stud
values(3,'bala');

1 row created.
```

(Or)

```
SQL> connect student12
Enter password:
Connected.
10) SQL> grant dba to user1;

Grant succeeded.

SQL> connect user1;
Enter password:
Connected.
12) SQL> create table student (rollno
number(5),name varchar2(10));

Table created.

SQL> insert into student
values(1,'Suji');

1 row created.

SQL> insert into student
values(2,'Mala');

1 row created.

SQL> insert into student
values(3,'Devi');

1 row created.

SQL> select * from student;

    ROLLNO NAME
---------- ----------
```

```
         1 Suji
         2 Mala
         3 Devi

13) SQL> grant insert on student to
user2;

Grant succeeded.

SQL> grant select on student to user2;

Grant succeeded.

SQL> connect user1;
Enter password:
Connected.
SQL> insert into user1.student
values(4,'Viji');

1 row created.

SQL> select * from user1.student;

    ROLLNO NAME
---------- ----------
         1 Suji
         2 Mala
         3 Devi
         4 Viji

14) SQL> revoke insert on student from
user2;

Revoke succeeded.
```

```
SQL> revoke select on student from user2;

Revoke succeeded.

SQL> select * from user1.student;

    ROLLNO NAME
---------- ----------
         1 Suji
         2 Mala
         3 Devi
         4 Viji
```

7. SIMPLE PL/SQL PROGRAMS

```
SQL> set serveroutput on;
1) odd or even
  SQL> declare
  2  num number;
  3  begin
  4  num:=&num;
  5  if(mod(num,2)=0) then
  6  dbms_output.put_line(num||'is even');
  7  else
  8  dbms_output.put_line(num|| 'is odd');
  9  end if;
 10  end;
 11  /
Enter value for num: 68
old   4: num:=&num;
new   4: num:=68;
68is even

PL/SQL procedure successfully completed.

2) fibanocci series
  SQL> declare
  2  num number;
  3  first number;
  4  second number;
  5  temp  number;
  6  i number;
  7  begin
  8  i:=1;
  9  num:=&num;
 10  first:=0;
```

```
 11  second:=1;
 12  temp:=first+second;
 13  while i<=num loop
 14  dbms_output.put_line(first ||' ');
 15  first:=second;
 16  second:=temp;
 17  temp:=first+second;
 18  i:=i+1;
 19  end loop;
 20  end;
 21  /
Enter value for num: 5
old   9: num:=&num;
new   9: num:=5;
0
1
1
2
3

PL/SQL procedure successfully completed.
```

3) leap year or not

```
  SQL> declare
  2  year number(4);
  3  begin
  4  year:=&year;
  5  if(mod(year,4)=0) then
  6  dbms_output.put_line('the year is leap
year');
  7  else
  8  dbms_output.put_line('the year is not a
leap year');
  9  end if;
 10  end;
 11  /
Enter value for year: 2023
old   4: year:=&year;
```

```
new   4: year:=2023;
the year is not a leap year


Enter value for year: 2020
old   4: year:=&year;
new   4: year:=2020;
the year is leap year

PL/SQL procedure successfully completed.
```

4) factorial of a number

```
SQL> declare
  2  n number;
  3  s number;
  4  begin
  5  s:=1;
  6  n:=&n;
  7  for i in 1..n loop
  8  s:=s*i;
  9  end loop;
 10  dbms_output.put_line('the factorial is
'||s);
 11  end;
 12
 13  /
Enter value for n: 5
old   6: n:=&n;
new   6: n:=5;

the factorial is 120

PL/SQL procedure successfully completed.
```

5) sum of n natural numbers

```
 SQL>  declare
  2     n number;
```

```
  3     s number;
  4     begin
  5     s:=0;
  6     n:=&n;
  7     for i in 1..n loop
  8     s:=s+i;
  9     end loop;
 10    dbms_output.put_line('the sum of natural
numbers is '||s);
 11    end;
 12
 13  /
Enter value for n: 5
old   6:    n:=&n;
new   6:    n:=5;
the sum of natural numbers is 15

PL/SQL procedure successfully completed.
6)SQL> declare
  2  n number:='&number';
  3  s number:=0;
  4  r number;
  5  len number;
  6  m number;
  7  t number;
  8
  9  begin
 10  while n>0
 11  loop
 12  s:=0;
 13  t:=n;
 14  m := t;
 15
 16  len := length(to_char(t));
 17  while t>0
 18  loop
 19  r := mod(t , 10);
 20  s := s + power(r , len);
```

```
 21  t := trunc(t / 10);
 22  end loop;
 23  if m = s
 24  then
 25  dbms_output.put_line(m||' ');
 26  end if;
 27  n:=n-1;
 28  end loop;
 29  end;
 30  /
Enter value for number: 200
old   2: n number:='&number';
new   2: n number:='200';
153
9
8
7
6
5
4
3
2
1

PL/SQL procedure successfully completed.
```

```
7)SQL> declare
  2  s VARCHAR2(10) := '&abccba';
  3      l VARCHAR2(20);
  4      t VARCHAR2(10);
  5  BEGIN
  6      FOR i IN REVERSE 1..Length(s) LOOP
  7          l := Substr(s, i, 1);
  8
  9
```

```
 10          t := t ||''||l;
 11      END LOOP;
 12
 13      IF t = s THEN
 14        dbms_output.Put_line(t ||''  ||' is
palindrome');
 15      ELSE
 16        dbms_output.Put_line(t ||'' ||' is
not palindrome');
 17      END IF;
 18  END;
 19  /
Enter value for abccba: abccba
old   2: s VARCHAR2(10) := '&abccba';
new   2: s VARCHAR2(10) := 'abccba';
abccba is palindrome

PL/SQL procedure successfully completed.
```

note: to display all table names

```
SQL> select table_name from user_tables;

TABLE_NAME
------------------------------
IGSTABL
DATABASE
DEPARTMENT
EM
EMPLOYEE
COURSE
STUDENT
EMP1
PRODUCT2
PRODUCT1

10 rows selected.
```

8 . PROCEDURAL FUNCTIONS

```
1)create or replace procedure getdetail(dno in
employee.depno %type) as
name employee.empname %type;
sal employee.salary %type;
cursor cur_emp is
select empname,salary from employee where
depno=dno;
begin
open cur_emp;
loop
 fetch cur_emp into name,sal;
exit when cur_emp %notfound;
dbms_output.put_line('employee name'||name);
dbms_output.put_line('salary'||sal);
dbms_output.put_line('departmnent'||dno);
end loop;
close cur_emp;
end;

SQL> set serveroutput on;
SQL> create or replace procedure getdetail(dno
in employee.depno %type) as
  2  name employee.empname %type;
  3  sal employee.salary %type;
  4  cursor cur_emp is
  5  select empname,salary from employee where
depno=dno;
  6  begin
  7  open cur_emp;
  8  loop
  9   fetch cur_emp into name,sal;
 10  exit when cur_emp %notfound;
 11  dbms_output.put_line('employee
name'||name);
 12  dbms_output.put_line('salary'||sal);
```

```
 13  dbms_output.put_line('departmnent'||dno);
 14  end loop;
 15  close cur_emp;
 16  end;
 17  /

Procedure created.

SQL> exec getdetail(2)
employee nameRaju
salary7000
departmnent2
employee nameMahesh
salary10000.5
departmnent2

PL/SQL procedure successfully completed.



2)create or replace procedure detail(eno in
employee.empno %type) as
name employee.empname %type;
begin
select empname into name from employee where
empno=eno;
dbms_output.put_line('employee name'||name);
end;

SQL> create or replace procedure detail(eno in
employee.empno %type) as
  2  name employee.empname %type;
  3  begin
  4  select empname into name from employee
where empno=eno;
  5  dbms_output.put_line('employee
name'||name);
  6  end;
```

```
  7  /

Procedure created.

SQL> exec detail(6)
employee nameVelu

PL/SQL procedure successfully completed.


3)create or replace procedure detail(eno in
employee.empno %type) as
name employee.empname %type;
sal employee.salary %type;

begin
select empname,salary into name,sal from
employee where empno=eno;
dbms_output.put_line('employee name'||name);
dbms_output.put_line('salary'||sal);

end;


SQL> create or replace procedure detail(eno in
employee.empno %type) as
  2  name employee.empname %type;
  3  sal employee.salary %type;
  4
  5  begin
  6  select empname,salary into name,sal from
employee where empno=eno;
  7  dbms_output.put_line('employee
name'||name);
  8  dbms_output.put_line('salary'||sal);
  9
 10  end;
 11  /
```

```
Procedure created.

SQL> exec detail(6)
employee nameVelu
salary10000.5

PL/SQL procedure successfully completed.
```

4)create table vendor (vendno number(5),vendname varchar2(5));

```
Table created.
create table vendor (vendno number(5),vendname
varchar2(5));

Table created.
create or replace procedure detail(vno in
vendor.vendno %type) as
name vendor.vendname %type;
begin
select vendname into name from vendor where
vendno=vno;
dbms_output.put_line('vendor name'||name);
end;

SQL> create or replace procedure detail(vno in
vendor.vendno %type) as
  2  name vendor.vendname %type;
  3  begin
  4  select vendname into name from vendor
where vendno=vno;
  5  dbms_output.put_line('vendor name'||name);
  6  end;
  7
  8  /
```

```
Procedure created.

SQL> exec detail(1)
vendor namesiva

PL/SQL procedure successfully completed.
```

```
5)create or replace procedure detail(rollno in
student.rollno %type,m1 in number,m2 in
number,m3 in number) as
tot number(9);
begin
tot:=m1+m2+m3;
update student set total=tot where
rollno=rollno;
commit;
dbms_output.put_line('total'||tot);
exception
when others then

dbms_output.put_line('error');

end;
```

```
SQL> create or replace procedure detail(rollno
in student.rollno %type,m1 in number,m2 in
number,m3 in number) as
  2  tot number(9);
  3  begin
  4  tot:=m1+m2+m3;
  5  update student set total=tot where
rollno=rollno;
  6  commit;
```

```
  7  dbms_output.put_line('total'||tot);
  8  exception
  9  when others then
 10
 11  dbms_output.put_line('error');
 12
 13  end;
 14  /

Procedure created.

SQL> exec detail(2,100,98,100)
total298

PL/SQL procedure successfully completed.
```

6)

```
create or replace procedure detail as

begin
for employee in (select *from employee where
designation='manager') loop
dbms_output.put_line('employee
name'||employee.empname);
end loop;
end;
```

```
SQL> create or replace procedure detail as
  2
  3  begin
  4  for employee in (select *from employee
where designation='manager') loop
  5  dbms_output.put_line('employee
name'||employee.empname);
  6  end loop;
```

```
  7  end;
  8  /

Procedure created.

SQL> exec detail
employee nameBalan
employee namesiva

PL/SQL procedure successfully completed.
```

7)

```
SQL> create or replace procedure Get(dno in
emp.depno%TYPE) as
  2  name emp.empname%TYPE;
  3  sal emp.salary%TYPE;
  4  NewSal number(10);
  5  begin
  6  for e in (select *  from emp where
depno=dno) loop
  7  NewSal:=e.salary+(e.salary*0.1);
  8  name:=e.empname;
  9  update emp set salary=NewSal where
depno=dno;
 10  commit;
 11  dbms_output.put_line('Employee
Name'||name);
 12  dbms_output.put_line('Update Salary
'||NewSal);
 13  end loop;
 14  Exception
 15  when others then
 16   dbms_output.put_line('Error');
 17  end;
 18  /

Procedure created.
```

```
SQL> exec Get(1)
Employee NameMani
Update Salary  5501
Employee NameRam
Update Salary  3300

PL/SQL procedure successfully completed.
```

8)
```
SQL> create or replace procedure Get  as
  2  name employee.empname%TYPE;
  3  sal employee.salary%TYPE;
  4  grade varchar2(2);
  5  begin
  6  dbms_output.put_line('Employee Name'||'
Salary'||'  Grade');
  7  for e in (select *  from employee)loop
  8  sal:=e.salary;
  9  if  sal < 10000 then
 10   grade:='D';
 11  elsif sal < 15000 then
 12  grade:='C';
 13  elsif sal < 20000 then
 14  grade:='B';
 15  else
 16  grade:='A';
 17  end if;
 18  name:=e.empname;
 19  dbms_output.put_line(name ||'   '||sal||'
'|| grade);
 20  end loop;
 21  Exception
 22  when others then
 23   dbms_output.put_line('Error');
 24  end;
 25  /
```

```
Procedure created.
```

NOTE: To see errors occurred

```
SQL> show error
No errors.


SQL> exec Get
Employee Name  Salary  Grade
siva   15000   B
Mani   3300   D
Raju   7000   D
Babu   5000   D
Ram   3300   D
Velu   10000.5   C
Ravi   8000.25   D
Balan   12000.75   C
Mahesh   10000.5   C
Kumar   14500   C

PL/SQL procedure successfully completed.
```

```
9)SQL> create or replace function get1(dno in
employee.depno%TYPE) return number as
  2  cnt number(2);
  3  begin
  4     select count(*) into cnt from employee
where depno=dno;
  5   return cnt;
  6  end;
  7  /
```

```
Function created.

SQL> select get1(1) from dual;

   GET1(1)
----------
         2

10)SQL> create or replace function empsal(eno
in employee.empno%TYPE) return number as
  2  sal number(8);
  3  begin
  4     select salary into sal from employee
where empno=eno;
  5   return sal;
  6  end;
  7  /

Function created.

SQL> select empsal(1)
  2  from dual;

 EMPSAL(1)
----------
     15000




11)SQL> create or replace function depnam(dno
in depart.depno%TYPE) return varchar as
  2  name varchar2(20);
  3  begin
  4     select depname into name from depart
where depno=dno;
```

```
  5   return name;
  6  end;
  7  /

Function created.


SQL> select depnam(1) from dual;

DEPNAM(1)
--------------------------------------------------
--------------------------------
Sales




12)SQL> create or replace function totSal(dno
in employee.depno%TYPE) return number as
  2  tot number(10):=0;
  3  begin
  4  for e in (select salary from employee
where depno=dno) loop
  5  tot:=tot+e.salary;
  6  end loop;
  7  return tot;
  8  end;
  9  /

Function created.

SQL> select totSal(1) from dual;

 TOTSAL(1)
----------
      6600
```

NOTE: user defined exception created is:

```
Exception when others then
    dbms_output.put_line('error occured');
```

9.TRIGGERS

```
1)create or replace trigger inSal before insert
or update on emp
 for each row
 begin
if :new.salary>10000 then
     raise_application_error(-20001,'Salary
must less than 10000');
end if;
 end;
/


2)create or replace trigger deldata before
delete on dep
 for each row
 begin
  delete from emp where depno=:OLD.depno;
 end;
  /

3)CREATE OR REPLACE TRIGGER display_tot
AFTER INSERT ON employee
DECLARE
  total_records NUMBER;
BEGIN
  SELECT COUNT(*) INTO total_records FROM
employee;
  DBMS_OUTPUT.PUT_LINE('Total number of
records: ' || total_records);
END;

4)CREATE OR REPLACE TRIGGER availability1
BEFORE INSERT OR UPDATE ON department
```

```
FOR EACH ROW
DECLARE
  dept_exists NUMBER;
BEGIN
  SELECT COUNT(*) INTO dept_exists FROM
department WHERE depno = :NEW.depno;
  IF dept_exists = 0 THEN
     dbms_output.put_line('not have');
 else
     RAISE_APPLICATION_ERROR(-20001,
'Department number already exist');
  END IF;
END;
/
```

```
5)CREATE OR REPLACE TRIGGER check_salary
AFTER INSERT ON e1
FOR EACH ROW
WHEN (NEW.designation = 'manager' AND
NEW.salary BETWEEN 4000 AND 6000)
BEGIN
RAISE_APPLICATION_ERROR(-20001, 'Invalid job
title or salary range');
END;
/
```

```
7)alter trigger availability disable;
 alter trigger availability enable;

8)drop trigger availability;
```

```
9)ALTER TABLE d1 DISABLE ALL TRIGGERS;
ALTER TABLE d1 ENABLE ALL TRIGGERS;
```

10)CREATE OR REPLACE TRIGGER no_friday
BEFORE INSERT OR UPDATE OR DELETE ON emp
FOR EACH ROW
WHEN (TO_CHAR(sysdate, 'Dy') = 'Fri')
BEGIN
  raise_application_error(-20001, 'Cannot
perform this operation on Fridays');
END;
/

11)CREATE OR REPLACE TRIGGER check_pk
BEFORE INSERT OR UPDATE ON emp
FOR EACH ROW
DECLARE
  emp_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO emp_count FROM emp WHERE
empno = :NEW.empno;
DBMS_OUTPUT.PUT_LINE(emp_count);
  IF emp_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'empno
already exists in employee table');
  END IF;
END;
/

12)CREATE OR REPLACE TRIGGER no_dml
BEFORE INSERT OR UPDATE OR DELETE ON employee
BEGIN
  RAISE_APPLICATION_ERROR(-20001, 'DML
operations are not allowed on employee table');
END;
/

10 . CURSORS

3)DECLARE

```
  CURSOR emp_cursor IS
    select d.depno,d.deplocation,d.depname,e.salary from depart
d,employee e where d.depno=e.depno;
  emp_record emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
   DBMS_OUTPUT.PUT_LINE('Depno Depname deplocation salary');
  LOOP
    FETCH emp_cursor INTO emp_record;
    EXIT WHEN emp_cursor%NOTFOUND;
     DBMS_OUTPUT.PUT_LINE( emp_record.depno||'
'||emp_record.deplocation||'  '||emp_record.depname||'
'||emp_record.salary);
    DBMS_OUTPUT.PUT_LINE('-----------------------');
  END LOOP;
  CLOSE emp_cursor;
END;
/
```

5)DECLARE

```
  CURSOR emp_cursor IS
    SELECT empno, empname, designation, salary
    FROM e1;
BEGIN
  FOR emp_record IN emp_cursor
  LOOP
    IF emp_record.designation= 'manager' THEN
      UPDATE e1
      SET salary = salary + 500
      WHERE empno = emp_record.empno;
    ELSIF emp_record.designation = 'Clerk' THEN
      UPDATE e1
      SET salary = salary + 200
      WHERE empno = emp_record.empno;
    END IF;
  END LOOP;
  COMMIT;
END;
/
```

```
8)SQL> CREATE TABLE Ord (
  2    order_id INT PRIMARY KEY,
  3    order_date DATE,
  4    delivery_date DATE,
  5     status varchar2(3));

Table created.



insert into ord values(1,sysdate,sysdate,'p');
insert into ord values(2,'20-May-23',sysdate,'d');

DECLARE
  -- Declare variables for cursor and order details
  CURSOR order_cursor IS
    SELECT order_id, order_date, delivery_date,status
    FROM Ord;

  v_order_id Ord.order_id%TYPE;
  v_order_date Ord.order_date%TYPE;
  v_delivery_date Ord.delivery_date%TYPE;
 sta ord.status%TYPE;
BEGIN
  -- Open the cursor
```

```
  OPEN order_cursor;

  -- Fetch and display order details
  LOOP
    FETCH order_cursor INTO v_order_id, v_order_date,
v_delivery_date,sta;
    EXIT WHEN order_cursor%NOTFOUND;

    -- Display order details
    DBMS_OUTPUT.PUT_LINE('Order ID: ' || v_order_id);
    DBMS_OUTPUT.PUT_LINE('Order Date: ' || v_order_date);
    DBMS_OUTPUT.PUT_LINE('Delivery Date: ' || v_delivery_date);
DBMS_OUTPUT.PUT_LINE('status: ' || sta);
    DBMS_OUTPUT.PUT_LINE('------------------------');
  END LOOP;

  -- Close the cursor
  CLOSE order_cursor;
END;
/
```

9)DECLARE

```
  -- Declare variables for cursor and order details
  CURSOR order_cursor IS
    SELECT order_id, order_date, delivery_date, status
    FROM ord
    FOR UPDATE;

  v_order_id ord.order_id%TYPE;
  v_order_date ord.order_date%TYPE;
  v_delivery_date ord.delivery_date%TYPE;
  v_status ord.status%TYPE;
BEGIN
  -- Open the cursor
  OPEN order_cursor;

  -- Update order status if order date is less than delivery date
  LOOP
    FETCH order_cursor INTO v_order_id, v_order_date,
v_delivery_date, v_status;
    EXIT WHEN order_cursor%NOTFOUND;

    IF v_order_date < v_delivery_date THEN
      -- Update order status to 'p'
```

```
    UPDATE ord
    SET status = 'p'
    WHERE CURRENT OF order_cursor;

    -- Display the updated order details
    DBMS_OUTPUT.PUT_LINE('Order ID: ' || v_order_id);
    DBMS_OUTPUT.PUT_LINE('Order Date: ' || v_order_date);
    DBMS_OUTPUT.PUT_LINE('Delivery Date: ' || v_delivery_date);
    DBMS_OUTPUT.PUT_LINE('Status: ' || 'p');
    DBMS_OUTPUT.PUT_LINE('-----------------------');
   END IF;
  END LOOP;

  -- Close the cursor
  CLOSE order_cursor;
END;
```

11.XML

XML stands for Extensible Markup Language. It is a widely used markup language designed to store, transport, and structure data in a human-readable and machine-readable format. XML doesn't describe how data should be displayed or styled like HTML does; instead, it focuses on representing the structure and content of the data.

Key features of XML:

Tags and Elements: XML uses tags to define elements, which can be nested to form a hierarchical structure. Elements can contain data, other elements, or both.

Attributes: Elements can have attributes that provide additional information about the element. Attributes are specified within the opening tag of an element.

Well-Formedness: XML documents must follow specific syntax rules to be considered "well-formed." This includes properly nested elements, closing tags, and correctly quoted attribute values.

Document Object Model (DOM): XML documents can be represented in a tree-like structure called the Document Object Model. This allows programs to manipulate XML data programmatically.

Namespaces: XML namespaces are used to avoid naming conflicts when different XML vocabularies are combined in a single document.

Validation: XML can be validated against an XML Schema Definition (XSD) or Document Type Definition (DTD) to ensure its structure adheres to predefined rules.

Extensibility: The "Extensible" in XML's name reflects its ability to define custom elements and structures, making it versatile for various applications.

Human and Machine-Readable: While designed for machines to process, XML is also human-readable, making it useful for configuration files, data interchange, and more.

XML is commonly used for:

- Data Interchange: XML is often used to exchange data between different systems, regardless of their underlying technologies.

- Configuration Files: Many software applications use XML to store configuration settings due to its human-readable format.

- Web Services: XML is a foundation of many web service protocols like SOAP and XML-RPC.

- RDF and Semantic Web: XML is used to represent RDF (Resource Description Framework) data in the Semantic Web.

- Document Formats: Some document formats, like Microsoft Office's Word (.docx) and Excel (.xlsx), use XML as their base structure.

- It's important to note that while XML has been widely adopted, other formats like JSON have gained popularity for their simplicity and efficiency in representing structured data. The choice between XML and other formats depends on the specific needs of a given project.

11.Create an XML database and validate it using XML schema.

Let's assume we want to create a simple XML database to store information about books. Here's an example of how the XML data might look:

```
<!-- books.xml -->
<library>
  <book>
    <title>Introduction to Programming</title>
    <author>John Smith</author>
    <year>2022</year>
  </book>
  <book>
    <title>Data Science Basics</title>
    <author>Jane Doe</author>
    <year>2021</year>
  </book>
</library>
```

Create XML Schema (XSD):

An XML schema defines the structure, data types, and constraints for your XML data. Here's an example XSD for the above XML:

```
<!-- books.xsd -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="library">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="book" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="title" type="xs:string"/>
              <xs:element name="author" type="xs:string"/>
              <xs:element name="year" type="xs:integer"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Validate XML against XSD:

You can use various tools and libraries to validate your XML against the XSD schema. One common tool is xmllint, which comes with many Unix-like systems.

Open your terminal and run the following command to validate the XML against the XSD:

```
xmllint --schema books.xsd books.xml
```

If the XML is valid according to the schema, you'll see a message indicating that no errors were found. If there are errors, xmllint will point them out.

12.NOSQL

NoSQL (Not Only SQL) is a class of database systems designed to handle large volumes of unstructured, semi-structured, or structured data in a more flexible and scalable way compared to traditional relational databases. Unlike relational databases, NoSQL databases do not use the traditional tabular schema with fixed columns and rows. Instead, they provide various data models that suit different types of applications and data storage needs.

Key characteristics of NoSQL databases:

- Schema Flexibility: NoSQL databases offer flexible schemas, allowing you to store data without a predefined structure. This is particularly useful for applications where the data can change or evolve over time.

- Horizontal Scalability: Many NoSQL databases are designed to scale horizontally, meaning they can handle increased workloads by adding more machines or nodes to the database cluster.

- Data Models: NoSQL databases come in different data models, including document-oriented, key-value, column-family, and graph databases. Each model is optimized for specific types of data and use cases.

- High Availability: Many NoSQL databases prioritize high availability and fault tolerance. They often use replication and sharding to ensure that data is available even if some nodes fail.

- Low Latency: NoSQL databases are designed for quick data retrieval, making them suitable for applications that require low-latency access to large volumes of data.

- Use Cases: NoSQL databases are commonly used for web applications, real-time analytics, Internet of Things (IoT) platforms, social media, mobile applications, and more.

Types of NoSQL databases:

- Document-Oriented Databases: Examples include MongoDB, Couchbase. They store data in documents similar to JSON or XML, allowing for flexible schemas and nested structures.

- Key-Value Stores: Examples include Redis, Amazon DynamoDB. Data is stored as key-value pairs, where keys are unique identifiers and values can be simple data or more complex structures.

- Column-Family Stores: Examples include Apache Cassandra, HBase. Data is organized into columns rather than rows,

making them suitable for handling large volumes of sparse data.

- Graph Databases: Examples include Neo4j, Amazon Neptune. These databases focus on representing and traversing relationships between data points, making them ideal for connected data scenarios.

- Wide-Column Stores: Examples include Google Bigtable. Similar to column-family stores, but optimized for high scalability and performance.

NoSQL databases are not a one-size-fits-all solution; each type of NoSQL database has its own strengths and limitations. The choice of a NoSQL database depends on the specific requirements of your application, the nature of your data, and the scalability needs you have.

12. Create Document, column and graph based data using NOSQL database tools.

DOCUMENTS:

Certainly, here's an example of how you can create document-based data using a NoSQL database like MongoDB:

- Install and set up MongoDB.
- Create a database and collection.
- Insert documents into the collection.

Example using MongoDB's shell:

```
// Connect to MongoDB
use MyDatabase

// Create a collection named "Products"
db.createCollection("Products")

// Insert documents representing products
db.Products.insertOne({
  name: "Product A",
  price: 29.99,
  category: "Electronics",
  description: "Smartphone with advanced features."
})

db.Products.insertOne({
  name: "Product B",
  price: 49.99,
  category: "Clothing",
```

```
  description: "Premium cotton t-shirt."
})
```

In this example, we've created a collection named "Products" and inserted documents representing different products.

Each document can have different fields, and the schema can be flexible. MongoDB allows you to store data without a rigid structure commonly found in relational databases.

Remember to plan your document structure based on your application's requirements, considering factors like the types of queries you'll perform and the data relationships you need to represent.

COLUMNS:

Certainly, column-based data storage is commonly associated with column-family NoSQL databases like Apache Cassandra. Here's an example of how you can create column-based data using Cassandra:

- Install and set up Apache Cassandra.
- Create a keyspace and table with columns.
- Insert data into the table.

Example using Cassandra's CQL (Cassandra Query Language):

```
-- Create a keyspace named "MyKeyspace"
CREATE KEYSPACE MyKeyspace WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};

-- Use the keyspace
USE MyKeyspace;

-- Create a table named "UserData"
CREATE TABLE UserData (
  user_id UUID PRIMARY KEY,
  username TEXT,
  email TEXT,
```

```
  age INT,
  address TEXT
);
```

After creating the table, you can insert data into it:

```
-- Insert data into the "UserData" table
INSERT INTO UserData (user_id, username, email, age, address)
VALUES (
  uuid(),
  'john_doe',
  'john@example.com',
  30,
  '123 Main St, City'
);
```

In this example, we've created a table named "UserData" with columns for user information like username, email, age, and address. The data is organized in a columnar format, which can provide benefits for specific types of queries, especially when querying large datasets.

Column-family databases like Cassandra are particularly well-suited for scenarios where you have massive amounts of data and need to scale horizontally across multiple nodes. Just keep in mind that schema design is crucial for optimizing performance and query efficiency in a column-family database.

GRAPHS:

Certainly, you can create graph-based data using graph database tools like Neo4j. Here's an example of how you can create a simple graph using Neo4j:

- Install and set up Neo4j.
- Create nodes for entities and relationships to connect them.

Example Cypher queries for Neo4j:

```
-- Create nodes for users
CREATE (:User {name: 'Alice', age: 30})
CREATE (:User {name: 'Bob', age: 25})
CREATE (:User {name: 'Charlie', age: 28})

-- Create relationships (e.g., friendships)
MATCH (alice:User), (bob:User)
```

```
WHERE alice.name = 'Alice' AND bob.name = 'Bob'
CREATE (alice)-[:FRIEND]->(bob)

MATCH (bob:User), (charlie:User)
WHERE bob.name = 'Bob' AND charlie.name = 'Charlie'
CREATE (bob)-[:FRIEND]->(charlie)
```

In this example, we've created nodes representing users and established FRIEND relationships between them.

You can then use Cypher queries to traverse and analyze the graph:

```
-- Find friends of Alice's friends (2nd-degree connections)
MATCH (alice:User)-[:FRIEND]->()-[:FRIEND]->(foaf:User)
WHERE alice.name = 'Alice'
RETURN foaf.name

-- Find common friends between Bob and Charlie
MATCH (bob:User)-[:FRIEND]->(common:User)<-[:FRIEND]-(charlie:User)
WHERE bob.name = 'Bob' AND charlie.name = 'Charlie'
RETURN common.name
```

Neo4j offers a powerful way to work with graph data, allowing you to model and query complex relationships

easily. Designing your graph schema thoughtfully based on your use case is important for optimal performance and efficient querying.