Vel Tech
Rangarajan Dr. Sagunthala
R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)

School of Computing

Department of CSE (Data Science)

ACADEMIC YEAR 2025 – 2026 (Summer Semester)

BONAFIDE CERTIFICATE

NAME: G. HARI KRISHNA

VTU NO: 30631

REG.NO: 24UESS0019

BRANCH: CSE (Data science)

YEAR/SEM: 2nd / III

SLOT: S15 L12

Certified that this is a bonafide record of work done by above student in the **"10211DS207 – Database Management Systems"** during the year 2025-2026.

SIGNATURE OF FACULTY INCHARGE **SIGNATURE OF INCHARGE**

Submitted for the Semester Model Examination held on_____at Vel Tech Rangarajan Dr. Sagunthala R & D Institute of Science and Technology.

EXAMINER 1 **EXAMINER 2**

1. what is data?

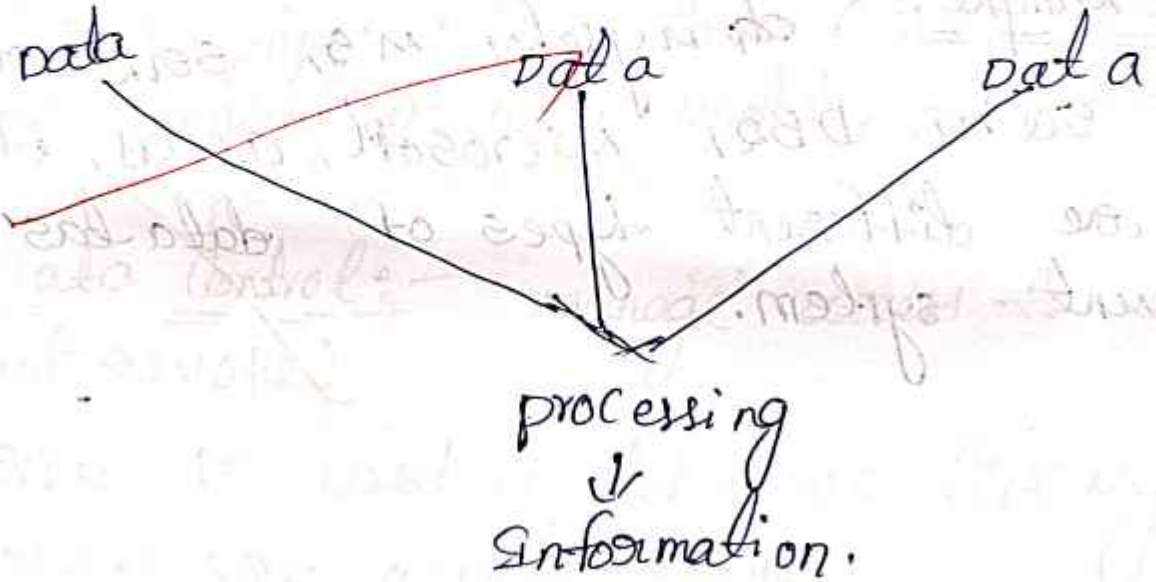
Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or eletronic memory etc.

Example:— Ankit, Delhi, 12, 80.

2. what is information?

when data are processed orginzed structure and interpreted in a given context, so, as to make them useful and meaningful, they are called information.

Ex:— Name — Ankit, city - delhi
class - 12 marks - 80



3. what is data base.

the data base is a collection of interrelated data which is used to retrieve, insert and delete the data efficiency. It is also used to organize the data in the form of a table, schema views and reports etc.

ex:— the college, data base organizes the data about the admin, staff study faculty etc.

4. what is DBMS?

DBMS stands for database mangement system which is software for creating, managing and manipulating data base. It ensures data is stored securly, interview efficiently and maintained consistently. key features include data storage, integral security, backup, recovery, and support for multiple users.

Example:— mysql, ms, sql server, oracle SQL, DB2, microsoft, access, etc. are different types of data base mangement system.

5.

6.

5. what is ER model?

An ER (Entity - relationship) model is a conceptual data model used in data bases design to represent data and its relationship visually it includes:

* Entity:- objects or things (E.g customer, order)
* Attributes:- properties of entities. (E.g. customer ID Name)
* Relationship: connections between entites (eg. customer place (order)

ER.. diagrams use rectangles for entities ovals for attributes, and diamonds, for relationship the model help in organizing and structuring for data bases.

6. what is SQL?—

SQL (structured query language) is a programming language used to manage and manipulate relation data bases.

- Data querying: retrive data (SELECT).
- Data manipulation: Add, update, and delete data (insert, update, delete).
- Data Control:- manage access to data (grant revoke).

SQL is used in data base like mysql postgre SQL oracle server.

Task 1:— Conceptual design after FIR using basic database design methodology and ER modeler design entity relationship. Diagram by satisfying the following sub-tasks:

1. a Identifying the entities
2. b Identifying the attributes.
3. c Identificating of relationship cardinality, types of relationship.
4. d reframing the relations with key constraints
5. e develop ER diagram for started case-task

Aim :—

To draw conceptual design through FIR using drawio tool.

PROCEDURE:—

a. Identifying the entities for bank management system sample output

b. Identifying the attributes for bank management system.

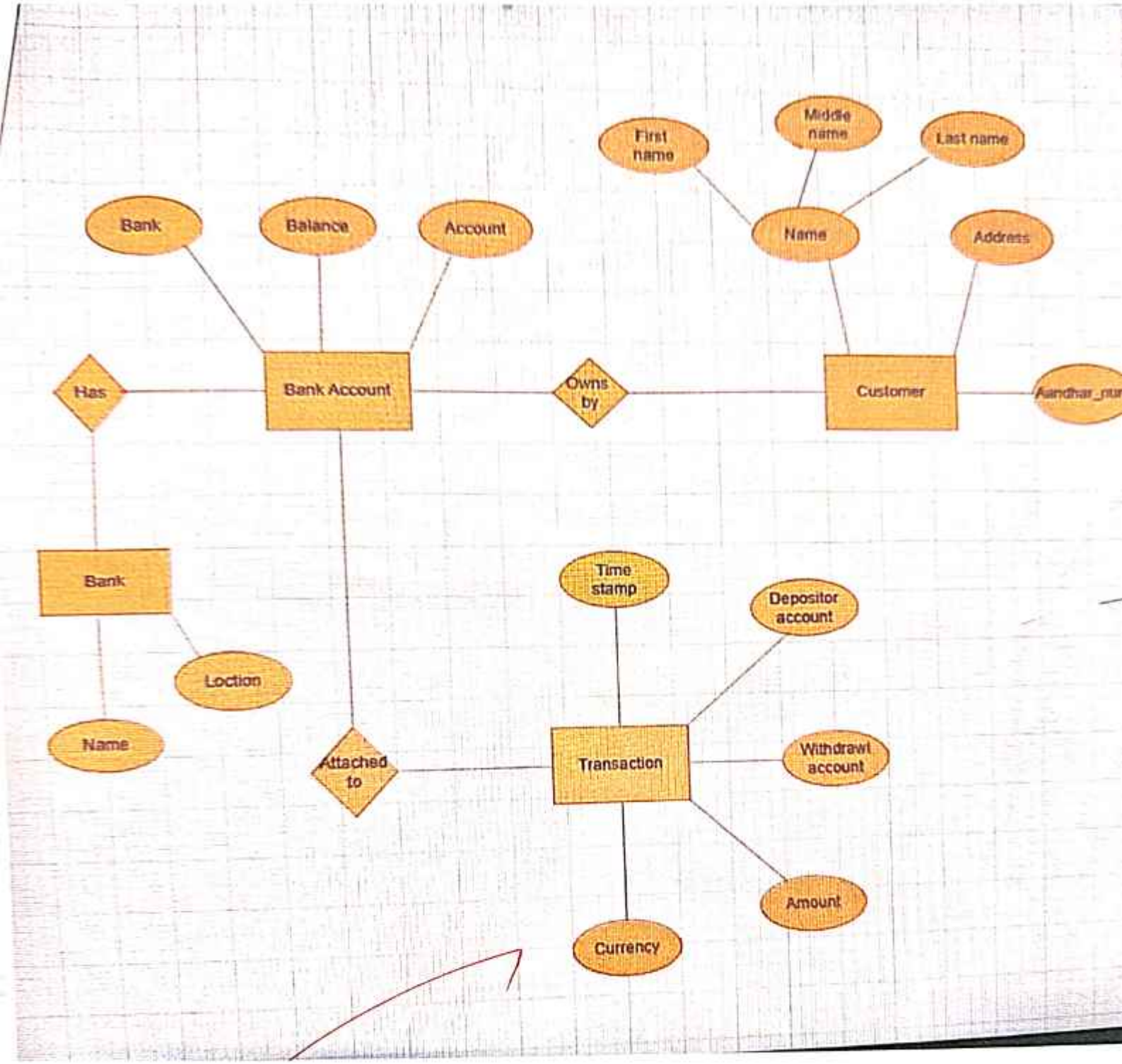
Sample output :—

c. Identification of relationships, cardinality type of relationship for bank management system.

d. reframing the relations with keys and constraints for bank management system.

Result :—

thus, drawing conceptual design through FIR using draw.io was executed successfully


First name
Middle name
Last name
Bank
Balance
Account
Name
Address
Has
Bank Account
Owns by
Customer
Aandhar_nu
Bank
Time stamp
Depositor account
Loction
Name
Attached to
Transaction
Withdrawl account
Amount
Currency

VEL TECH	
EX NO.	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

Result:- Thus, drawing conceptual design through FOR using draw, io exited successfully.

Task-2:-

Generating design of other traditional data base model.

Implementation of DDL, DML, DCL, and commands of SQL

Aim:- Implementation of DDL, DML, DCL and TCL commands of SQL with suitable examples

Objective:

* To understand the different issues involved in the design and implementation of a data base system.
* To understand and use data definition language to write query for a database

Theory:-

oracle has many tools such as SQL* plus oracle forms, oracle report writer, oracle graphics etc.

SQL* plus:- the SQL* plus tool is made up of two distinct parts there are

Interactive SQL: Interaction SQL is designed for create access and manipulate data stuctures like tabels and indexes.

PL/SQL:- can be used for different applications.

Oracle Forms:- this total allows you to create a data entry screen along with the suitable menu objects thus it is the oracle forms tool that handles data gathering and data validation in a commericial application.

Reportwriter:- report writer allows programs to prepare innovative reports using data from the oracle strutures like tables, views. this report writer total that handles the reporting sation of commerial application.

Oracle graphics:- Some of the data can be better represented in the form of pictures the oracle graphics tool allows programmers of prepare graphs using data from oracle structure like tables, views etc.

SQL (structured query language):

structured query language is a database computer language designed for managing data in relational data base mangement system (RDBMS), and originally based upon realtion algebra its slope includes data query and update schema creation and modification and data access control.

SQL was one of the Edgar F Codd's relational model and belong the most widely used language for relational databases.

IBM developed SQL in mid of 1970's:-
oracle introduced in the year 1979 SQL used by IBM/DB2 and DS Database system's

SQL adopted as standard language for RDBS by ASNI in 1989.

Data types:-

1. CHAR(size): this data type is used to store character strings values of fixed length the size in brackets determines the number of characters the cell can hold the maximum number of character is 255 character.

2. VAR CHAR (size):-
this data type is used to store variable length alphanumeric data the maximum char can hold is 2000 character.

3. Number (p, s):- the number data type is used to store number of virtual any magnitude may be stored up to digits of precision number as larger as $9.90 * 10^{124}$ the precision the number of place to the right of the decimal. If scale is omited than the default is zero If precision is value are stored with their original precision.

up to the maximum of 88 digits.

4. Date:— this data type is used to represent data and time the standared format is DD-MM-YY as in 17-SEP-2009, To enter dates other than the standard format use the appropriate functions date time stores data in the 24-hours format By default the time in a date field is 12:00:00 am, If no time position is specified the defuelt date for a date feild is the first day the current month.

5. LONG:— this data type is used to store variable length character strings containing up to 2GB. long data can be used to store arrays of binary data in ASCII format long values cannot be indexed and the normal character functions such as SUBSTR cannot be applied.

6. RAW:— the raw data type is used to store binary data such as digitized picture or image data, as located into column's of these data types are stored with out any further conversation. RAW data type can have a maximum length of 255 byts long raw data type can contain up to 2GB.

DATA DEFINITION LANGUAGES: (for shopping management system):

A Data definition language (DDL) statements are used to define the database mangement structured or schema.

Create:- To make a new database table index, or stored query. A create statement in SQL. Created an object inside of a relational database mangement system (RDBMS).

Syntax:-

CREATE TABLE table_name
(Column_name1 datatype ([size]),
Column_name 2 datatype ([size]),
Column_name_n data_type ([size]);

Alter:- To modify an existing database object. Alter the structure of database

* To add a column in a table:

Syntax:-

ALTER TABLE table_name ADD
Column_name datatype ([size]).
alter table shopping customer address
ver Char [20];

* To delete column in a table

Syntax:-

ALTER TABLE table_name DROP_Column
Column_name;

* To modify a column in a table

Syntax:-

ALTER TABLE table_name modify column column_name data_type ((New_size));

Rename:-

Syntax:-

ALTER TABLE table_name RENAME column old_column_name to new_column_name;

Describe:-

Syntax:-

DESC table_name.

TRUNCATE TABLE:-

remove all records from a table, including all space allocated for the records are removed

Syntax:-

TRUNCATE TABLE table_name

DATA MANIPULATION LANGUAGE

Data manipulation language allows the users to query and manipulate data in existing schema in object It allows following data to input, delete, update and retrieve data in schema object.

Insert:-

value can be inserted into table using insert commands. there are two types insert commands - they are multiple value insert commands single value insert command.

syntax:-

INSERT INTO table-name VALUE (value1, value2, value3 - - -);

(OR)

INSERT INTO table-name (column1, column2, column3, - - -) VALUES (value1, value 2, value3

update:- this allows the user to create the particular column value using the where clause condition.

syntax:-

UPDATE <table-name> SET WHERE
<col1 = value>
<column = value>;

Delete:- this allows you to delete that particular Column value using where clause Condition.

Syntax:-

DELETE FROM <table_name> WHERE <Condition>;

select:- the select statement is used to query a database. this statement is used to retrive the information from the database the SELECT statement can be used in many ways they are:

1. select some column:-

To select specified number of colums from the table the following Command is used;

Syntax:-

SELECT Column_name FROM table_name;

2. select using distinct:-

the DISTINCT key word is used to return only different values (i.e) this Commands does not select the duplicate values from the table.

Syntax:-

SELECT DISTINCT Column name(s) from table_name;

3. select using between
BETWEEN can be used to get those
that fall within a range

Syntax:-
SELECT ~~DISTINCT~~
column name FROM table-name
WHERE
Column name BETWEEN value 1 and value

4. Renaming:-
the select statement can be used
to rename either a column or the entire table
Syntax:-
SELECT Column name AS new name FROM
table-name;

5. Sorting:-
the select statement with the order by
clause is used to sort the contents.
Syntax:-
SELECT Column name FROM table-name
WHERE
Condition
ORDER BY Column name ASC/DE

6. 'LO select by matching some patterns: –
the select statement along with like clause is used to math strings the like condition is used to specify a search pattern in a column.

Syntax: –
SELECT Column name FROM table_name WHERE Column name LIKE '%' or '_';
% : matches any sub string
_ : matches a single charater.

7. select using AND, OR, NOT: –
we can combine one (or) more conditions in a select statement using the logical operators AND, OR, NOT.

Syntax: –
SELECT column name FROM table_name WHERE Condition1 LOGICAL OPERATOR Condition 2.

DATA Control Languages: –

1. Create: –
* create user, kamal identified by kunal;
user created.

2. Grant :-

* grant all privilges to kamal;

Grant succeded.

3. Revoke :-

* Revoke all privileges from kamal;

Revoke succeeded.

TRANSACTION CONTROL LANGUAGES :-

1. Commit :-

* Commit;

Commit Complete

2. Save point :-

* Save point k1;

Save point Created

SQL> Connect
Enter user-name: system
Enter password:
Conneted.
SQL> Create table hariCname var char (12), ph_no number (10), addhar_no number (12);
~~porche~~
Create table hari (hari varChar(12), ph_no number(10), addhar_no number (12))

SQL> Create table student 11 (name varchar (12), ph_no number(10), addhar_no number (12));
Create table student11 (name varchar(12), ph_no number(10), addhar_no number (12))

SQL> Create table student 12 (name varchar(12), ph_no number (10), addhar_no number (12));
Table Created.
SQL> desc student12;

Name	NULL?	Type
NAME		VARCHAR(12)
PH_NO		NUMBER(10)
ADDHAR_NO		NUMBER(12).

SQL> alter ~~table~~ student 12 add gmail varchar(18
~~Table Created~~ alter

SQL> desc student12;

Name	Type
name	VARCHAR2(20)
ph_no	NUMBER(10)
addhar_no	NUMBER(12)
gmail	VERCHAR(18)

SQL> alter table student12 drop column gmail;
Table altered.

SQL> desc student12

name	null ?	Type
name		VARCHAR(20)
ph_no		number(10)
addhar_no		number(12)

SQL> alter table student12 modify addhar_no number(14);
Table altered

SQL> desc student12;

Name	NULL ?	Type
name		VARCHAR(20)
ph_no		number(10)
addhar_no		number(14)

SQL>

Name	NULL?	Type
name		VARchar2(12)
ph_no		number (10)
ID-no		number (14)

SQL> insert into student12 values ('hari', 9999, 123456);
1 row created.

SQL> insert into student12 values ('guru', 7777, 123456);
1 row created

SQL> insert into student12 values ('naveen', 8881, 7676);
1 row created

SQL> select * from student12;

Name	ph_no	ID-no
hari	628196	123123
guru	901010	132456
naveen	888888	456789

SQL> Commit;
Commit Complete.

SQL> Save point s1;
Save point Created.

VELTECH	
EX No.	2
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD (5)	4
TOTAL (20)	18
SIGN WITH DATE	

Task : 3 :- Developing queries with DML single - row function and operators.

Aim:- To perform the query processing on data bases for different retrival results of queries using DML, DRL single - raw operations using aggregate data, string, ident functions set clauses and operators.

Procedure:- Create table for employee schema and insert around 10 retail - store retait store - employes data in this realation perform multi raw functions.

Note :- these queries assume a sample database with an retail - store - employee table containg colums like retail store employee "name", "salary", and department", store - phone number", "employee - phone number".

1. Cant:- Count following by a column name returns the count of tuple in that column. If district key word is used then it will return only the count of unique tuple in the column otherwise it will return count of all the tuple.

Count (*) indicates
column.
syntax:- count (column name)
Ex:- select count (*) from retail_store
-employee.
2. sum:- sum followed by a column name
returns the sum of all the value in
that column.
syntax:- sum (column name)
Ex:- select sum (salary) from retail.
empolyees.
3. avg; avg followed by a column name
returns the avarage value of that
column value
syntax:- avg (n1, n2)
Example:- select avg. (salary) from retail
store_employees;
4. MAX: max followed by a column name
return the maximum value of that column
syntax:- MAX (column name).
Ex:- select MAX (salary) from retail
_store_empolyees
soln> select retail store employee name,
max (salary) from retail_store_employee
group by retail store_empolyee

SQL string function:-

string function are used to perform an operation input string and return an output string. following are the string function defined in SQL.

1. upper()

Query:-

select upper (retail store_employee_name) from retail_store_employee WHERE retail store_employee_id = 1;

2. lower()

Query:-

select lower (retails store_employee name) from retail_store where retail store employee_id = 1;

3. length()

Query:-

select length (retai store_employee_name) from retail_store_employee WHERE retail store_employee_id = 1

4. substr()

Query:-

select substr (retail store_employee_name, 1) from retail_store_employee WHERE retail store_employee_id = 1;

5. Concat ( ) :-

Query :-

Concat (retail store - employee - name, reporting
from retail - store - employee WHERE retail
store - employee - id = 1;

SQL Data and time functions :-

the data time function are built - in
function in the SQL these functions can
be used in SQL queries to perform
such as filtering and formatting data
for display purpose for storing a date
and time value in a data base
MYSQL offers the following date type
DATE format data types.

DATE format YYYY - MM - DD.

DATETIME format : YYYY - MM - DD HH:MM:SS

TIME STAMP formed : YYYY - MM - DD

YEAR format : YYYY or YY.

Curdate ( )

Query : SELECT current date from dual

Curtime ( ) :-

Query :- SELECT current - time ( )
from dual;

ADD Date (date, DAYS)

SQL > select add Date (2018-08-01, 31)

ADD time (exp1, exp2)

SQL > select ADD time (2018-08-01 23:59:59:99999, '1:1' 00:27);

day of month (date)

SQL > select day of month (2018-02-15)

day of week (date):-

SQL > select day of week (2018-02-15)

day of year (date):-

SQL > select day of year (2018-02-15)

month (date):-

SQL > select month (2018-08-01)

Time (expr)

SQL > select time (2018-08-01) 1:33:25);

sysdate:-

SQL > select sysdate from dual;

next day:-

SQL > select

next-day (sysDate, Day) from dual

to-char:-

SQL > select to-char (sys date dd) from dual

to-date:-

SQL> select to date (sysdate, dd/mm/yy)
from dual;

CHARACTER FUNCTION:-

init cap char: select init cap (hello) from dual;

lower(char): select lower (HELLO) from dual

upper(char): select upper ("hello") from dual

trim (char, [set]): select trim (sct cse) from dual;

rtrim (char, set): select rtrim (cscrat) from dual;

replace (char, search): select replace Jack and June; from dual

string functions:-

Concat:- Concat returns char concatnd with char2, Both char1 and char2 can be any of the data types.

SQL> select
Concat ("oracle", 'corporation') from dual;
oracle corporation.

Lpad:- LPAD returns expr, left-padded to length n characters with the sequence of characters in expr2.

SQL> SELECT
LPAD('oracle', 15, '*') from dual;

Rpad:- RPAD returns expr, right-padded to length n character with expr2 replicated as many time as necessary

SQL> select
("oracle", " '*') from dual;

Ltrim:- return a character expression after removing leading blanks.

LTRIM ('SSMITHSS' 'S') from dual;

Rtrim:- returns a character string after truncating all trailing blanks

SQL> select
RTRIM ('SMITHSS' 'S') from dual;

Lower:- return a character expression after converting upper case character data to lowercase

SQL> select lower (DBMS) from dual;

upper:- returns a character expression with lower character data converted to upper case

SQL> select upper (dbms) from dual;

length:- return the number of charaters rather then the number of bytes of the given string expr excluding trailing blanks.

SQL> SELECT length (database) from dual;

substr:- returns part of a charater binary text, or image expression

SQL > select substr ('ABCDEF GHI' : 3, 4) from dual;

select substr (retail store - employee - name 1, 4, from retail - store - employee;

Instr:- the INSTR functions search string for substring the function returns for integer indicating string that first charater of this a correnle

SQL> select INSTR ('Corporate FLOOR', 'OR', 3, 2) from dual;

output:-

Create table retail_store_employees
(retail_store_employee_id number, store_id
number, retail_store_employee varchar(34), salary
number, department varchar(23), store_phone
no number)
Table created

desc retail_store_employees;

Name	NULL?	Type
retail_store_employee_id		number
store_id		number
retail_store_employee		varchar(34)
salary		number
department		varchar(23)
store_phone_no		number

insert into retail_store_employees values
(25780, 12345, 'hari', 2900, 'harisoy' 74166159);
1 row created

insert into retail_store employees values.
(56787, 66778, 'hari', 68000, 'ravi' 906300797)
1 row created.

insert int retail_store_employee values
(89809, 56765, "shiva" 'ravi', 7696557716)
1 row created

salary department

25780 vasu 7466159

56787 vishu 9063007

89809 shiva 7696574

select count (*) from retail_store_emp

count (*)

3

select count sum (salary) from retail_st employees;

sum (salarly)

153788

select avg (salary) from retail_store_emp

Avg (salary)

51267.6667

select min (salary) from retail_store_e

min (salary)

51262.6667

29000

select max (salary) from retail_store_employees:

max (salary)

68000

select upper (retail_store_employee name) from, retail_store_employees, retail_store_ employee_id = 2580

upper (retail_store

hari

select lower (retail_store_employeename) from retail_store_employees where retail_store_employee_id = 56487;

lower (retail_store_employee name)

---

select sysdata from dual;

sys data.

21-Aug-25

select next_day (sys date, mud) from dual;

next day.

27-Aug-25

select add_months (sys date, 2) from dual.

add_month.

21-oct-25

select last_day (sys date) from dual;

last_day.

31-Aug-25

select least 10-feb-07, 12-oct-07 from dual. least

10-feb-17

select greatest (10-feb-7, 12-oct-10) from dual;

greatest

12-oct-07

select trunc (sys)

TRUNC (sys)

24-AUG-25

select round (sys date, day) from dual;

ROUND (sys)

31-AUG-25

select to char (sys date, dd/mm/yy) fr
dual;

TO_CHAR (

28/8/25

select to_date (sys date, dd/mm/yy) fr
dual;

TO date (

28-AUG-25

select concat ('oracle', corporation') fro
dual;

concat ('ORACLE')

oracle corporation

select lpad ('oracle' 15 '*') from dual;

LPAD ('ORACLE')

* * * * * oracle

select ltrim ('smith ss', 's') from dual

LTRIM

mith ss

select rtrim ('smith ss' 's') from dual;
RTRIM('
---
smith.

select lower ('DBMS') from dual;
LOWE
---
dbms

select upper ('DBMS') from dual;
UPPE

DBMS.

select length ('DATABASE') from dual;
length ('DATABASE')

9

select substr ('dbms' 1,3) from dual;
sub
---
dbms

select instr ('corporate floor', 'or', 3,2) from dual;
INSTR ('CORPORATE FLOOR', 'OR' 3,2)

14

select * from retail_store_employees where salary is null;
no rows selected

select * from retail_store_employees where salary is not null;

RETAIL - STORE - employee - sd

store - sd retail - rd

29000	ravi	27780	12345
68000	ravi	56787	66778
56788	ravi	89809	86765

6 rows selected

VEL TECH	
EX NO.	3
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	4
TOTAL (20)	19
SIGN WITH DATE	

Result:— thus, developing queries with single row function and operation is executed.

Task 4:— developing queries with DML multi-row function and operators.
perform the advanced query processing (multirow perform.
abilities) and test its heuristics using the designing of optimal correlated and nested subqueries such as finding summary statistics.
Consider the schema for
employees (emp-no, emp-name-name, department dept no, salary, age)
orders (emp-no, order-id, price, qty-ord, qty-hand)
Item file (item id, item name, qty-ord, qty-hand item mate)
queries using union, Intersect, MINUS.
union:— the union operator returns all distinct rows selected by two or more queries.
SQL> select emp-no from employees;
SQL> select emp-no from orders.
SQL> select emp-no from employees union select emp-no from orders;
union all:—
SQL> select emp-no from employees union all select emp-no from orders;

Intersect :-
SQL> select emp-no from employees union all select emp-no from orders;

Minus :-
SQL> select emp-no from employees minus start emp-no from orders;

Queries using group by, having clause and order clause.

group by : this query is used to group to all the records in a relation together for each and every value of a specific key(s) and then relation.

SQL> select deptno, count (*) from employees group by dept no;

group by having :- the having clause was added to SQL because the WHERE keywords not be used with aggregate function the having clause must follow the group by clause in a query must also precede the order by clause if used.

SQL> select deptno, count (*) from employees group by dept no having deptno is not null;

order by:- this query is used to display. is used to display a selected set of feilds from a relation in an ordered - manner base. on some feild

Syntax:-

select < column (s) > from < table name > where [condition (s)] [order by < column name > asc/desc];

SQL> select empno, ename, salary from employees order by salary;

SQL> select empno, emp-name, salary from employees order by salary desc

SQL* plus having following operators.

SQL> select salary + comm from emp-master

salary + comm

SQL> select salary + comm net-sal from emp-master;

SQL> select 12* (salary + comn) ennual-netsal from emp master;

subqueries:-

SQL> select * from employees.

SQL> insert into employees * from employee where emp-id in (select emp-id from emplid

SQL> update employees set salary = salary *10 where department in (select deportment from employee's where department = "sales";

In:-

query: SELECT * from employees where department In (sales 'marketing')

NOT In:-

query: SELECT * from employee where department NOT IN ("sales", marketing")

Exits:-

query: SELECT * from employees where exists (select * from order's where order: emp_no = (link unvailable);

NOT Exist:-

query: select * from employees where salary > all (select salary from employees WHERE department = 'sales');

ANY:-

query: select * from where salary > any (select salary from employees where department = 'sales');

— 11 Task-4

SQL> connect

Enter user-name: system

Enter password:

Connected.

SQL> desc student;

Name	Null?	Type
S_ID		NUMBER
S_NAME		VARCHAR2(10)
S_ADDRESS		VARCHAR2(12)
PH_NO		NUMBER

SQL> desc course;

Name	Null?	Type
C_ID		NUMBER
C_NAME		VARCHAR2(11)

SQL> desc student_course;

Name	Null?	Type
S_ID		NUMBER
C_ID		NUMBER

SQL> select*from student;

insert into student values(7,'hari','gggggggg',444444);

row created.

SQL> select*from student;

S_ID	S_NAME	S_ADDRESS	PH_NO
12	vasu	xxxxxxx	9797979797
10	ram	yyyyyy	888888888
11	hey	kkkkkkk	8999998
1	sid	cccccccc	222222222
2	hari	gggggggg	444444

SQL> insert into course values(122,ait');

1 row created.

SQL> insert into course values(152,'dbms');

1 row created.

SQL> insert into course values(112,'ds');

1 row created.

SQL> insert into course values(192,'fcn');

1 row created.

SQL> insert into course values(232,'dms');

1 row created.

SQL> select* from course;

C_ID C_NAME

---

122 alt

152 dbms

112 ds

192 fcn

232 dms

SQL> insert into student_course values(45,55);

1 row created.

SQL> insert into student_course values(25,67);

row created.

SQL> insert intostudent_course values(75,87);

1 row created.

SQL> insert into student_course values(85,90);

1 row created.

SQL> insert into student_course values(5,10);

1 row created.

SQL> select*from student_course1;

S_ID	C_ID
45	55
25	67
75	87
85	90
5	10

SQL> select s_id from student;

```
S_ID
----
  12
  10
  11
   1
   2

> select c_id from course;

C_ID
----
 122
 152
 112
 152
 232

> select c_id from student_course;

C_ID
----
 55
 57
```

```
SQL> select s_id from student union select c_id from course;

S_ID
----------
  20
  21
  22
  23
  24
  25
  26
  27
  28
  29
  30

20 rows selected.

SQL> select s_id from student union all select c_id from course;

S_ID
----------
```

```
SQL> select s_id from student intersect select c_id from course;

no rows selected

SQL> select s_id from student minus select c_id from course;

      S_ID
      ----
         1
         2
```

12

SQL> select s_id,count(*)from student group by s_id;

S_ID	COUNT(*)
1	1
11	1
2	1
12	1
10	1

SQL> select c_id,count(*)from course group by c_id having c_id is not null;

C_ID	COUNT(*)
152	1
112	1
232	1
192	1
122	1

SQL> select s_id,s_name,ph_no from student order by s_id;

S_ID	S_NAME	PH_NO

90876555

```
SQL> insert into vasu select*from vasu where s_id in(select s_id from vasu);

5 rows created.

SQL> select*from student where s_id in(12,10);

    S_ID S_NAME    S_ADDRESS       PH_NO
--------- --------- ----------- ----------
      12 vasu     xxxxxxxx    9797979797
      10 ram      yyyyyyy     888888888
      12 vasu     xxxxxxxx    9797979797
      10 ram      yyyyyyy     888888888

SQL> select*from naveen where s_id not in(12,10);

    S_ID S_NAME    S_ADDRESS       PH_NO
--------- --------- ----------- ----------
      11 hey      kkkkkkk     123458
       1 sid      cccccccc    222222222
       2 hari     ggggggggg   444444
      11 hey      kkkkkkk     123458
       1 sid      cccccccc    222222222
```

VEL TECH	
[illegible]	
PERFORMANCE (5)	

sql> select * from order_master where order_no = select order_no from order)

sql> select * from order_master where order in (select order_no from order_detail)

sql> select * from order_master where order_in (select order_no from order_details);

sql> select * from order_details where qty = all (select qty_hand from item file where item rate = 250);

~~Thus~~ —

Result: — thus the developing sql with DML multi-row transform and operations.

VEL TECH	
EX NO.	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

Task-5 :- writing join queries, equivalent and/OR and/OR RECUSIVE queries.

Title:- Implementation of different types of joins and recursive queries.

- A sql join combines records from two tables.
- A join locates related column values in the two tables.
- A query can contain zero one, or multiple join operations.
- inner join is the same as join the keyword inner is optinal.

Theory:-

the sql joins clauses is used to combine records from two or more tables in a data base. a join means for combining feilds from two tables by using values common to each. the join is actually performed where clauses which combines specified of tables

Syntax:-

SELECT column 1, column 2, column 3 -- from table_name 1, table_name 2 WHERE table_name 1 column name = table name 2 column name;

Types of joins :-

1. simple join
2. self join
3. outer join.

It is most common type of join. it retrives the rows from 2 tables having a common column and is further classified into equi-join and/OR

A join which is based on equalites is called equi-join.

example:-

select * from item cust where item.id = cost_id;

in the above statement item_id = cust_id perform join statement retrives rows from both the tables provided they both have the same id as specified by the where clauses since the where the comparison operator (=) to perform a join, it is said to be it combines the matched rows of tables can be used as follows.

- To insert records in the target table.
- to create tables and insert records in this table..
- To update records in the target table
- To create views.

select * from item, cust where item.id = cust id;

Table aliases.

Table aliases are used to make multiple table queries. shorted and more readable we given an alias name in the from class and use it instead of the name through the query.

self join:
joining of a table to itself is known as
-join one row in a table to another.
it can compare each row of the table to itself
and also with other rows of the same table.

example:—
select * from emp x, emp y where x.salary >=
(select avg (salary) from x. emp where x.
deptno = y. deptno);

outer join:—
it extends the result of simple join as outer
join return all the rows returned by sample
join well those rows from one table do not
match any row from the table. the symbol (+)
represent outer join.

here are the different types of joins in SQL;
(inner) join : returns record that having match
values in both tables.
select column-name(s) from table1 INNER
join table2 on table. column-name = table
2 column-name;
LEFT (outer) join : return all records from
left and the matched records from the right
table
SELECT column name(s) from table left join
table 2 on table 1. column-name = table2
column-name;

RIGHT (outer) join : return all return from
right table. and the matched records the
left table.

join [illegible] on table 1 FROM table 1 right column_name; column_name = table2 and/OR

FULL OUTER join table1. column_name = table 2 column_name;

LEFT join

Right join.

Inner join

Full outer join

Consider the following two tables - number and borrowed.

inner join query: —

SELECT borrowed. membno. number. NAME. FROM member.
Inner join borrowed.

Left join query: —

SELECT member name. borrowed membnor.
LEFT join borrowed on borrowed membno = member., membno;

Task-5:— writing Join queries, equivlent and/OR recursive queries.

```
SQL*Plus: Release 11.2.0.2.0 Production on Thu Sep 18 13:59:53 2025

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> create table hari (member_no number,name varchar(10));
create table hari (member_no number,name varchar(10))
            *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL> create table krishna (member_no number,name varchar(17));

Table created.

SQL> desc krishna
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MEMBER_NO                                          NUMBER
 NAME                                               VARCHAR2(17)

SQL> insert into krishna values(1,'krishna');

1 row created.

SQL> insert into krishna values(1,'guru');

1 row created.

SQL> insert into krishna values(1,'ashok');

1 row created.

SQL> insert into krishna values(1,'royal');

1 row created.

SQL> select*from krishna;

 MEMBER_NO NAME
---------- ----------
         1 krishna
         1 guru
         1 ashok
         1 royal

SQL> create table royal(member_no number,royal_id varchar(14));

Table created.

SQL> desc royal;
 Name                                      Null?    Type
```

```
---------------------------------- -------- ---------------------------
MEMBER_NO                                   NUMBER
ROYAL_ID                                    VARCHAR2(14)

SQL> insert into royal values(2,'b101');

1 row created.

SQL> insert into royal values(3,'b102');

1 row created.

SQL> insert into royal values(4,'b103');

1 row created.

SQL> insert into royal values(5,'b104');

1 row created.

SQL> select*from royal;

 MEMBER_NO ROYAL_ID
---------- --------------
         2 b101
         3 b102
         4 b103
         5 b104


SQL> select royal.member_no,krishna.name from krishna inner join royal on
krishna.member_no=royal.member_no;

no rows selected

SQL> select royal.member_no,krishna.name from krishna right outer join royal on
krishna.member_no=royal.member_no;

 MEMBER_NO NAME
---------- ----------------
         5
         4
         3
         2

SQL> select royal.member_no,krishna.name from krishna left outer join royal on
krishna.member_no=royal.member_no;

 MEMBER_NO NAME
---------- ----------------
           royal
           ashok
           guru
           krishna
```

```
SQL> select royal.member_no,krishna.name from krishna full outer join royal on
krishna.member_no=royal.member_no;

MEMBER_NO NAME
--------- ----------
        2
        3
        4
        5
          royal
          ashok
          guru
          krishna

8 rows selected.

SQL> create table
  2
SQL> create table ashok(emp_id number, emp_name varchar(17));

Table created.

SQL> insert into ashok values(7,'hari');
insert into ashok values(7,'hari')
                         *
ERROR at line 1:
ORA-00917: missing comma


SQL> insert into ashok values(7,'hari');

1 row created.

SQL> insert into ashok values(3,'guru');

1 row created.

SQL> insert into ashok values(4,'vasu');

1 row created.

SQL> insert into ashok values(5,'naveen');

1 row created.

SQL> select*from ashok;

   EMP_ID EMP_NAME
--------- ----------
        7 hari
        3 guru
        4 vasu
        5 naveen

SQL>
```

VEL TECH	
EX NO.	1
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

RESULT:- thus, implementation inner join, inner outer join right outer join, full outer join is executed successfully

Retrieve total key value:-

select member.name borrowed member.mem_no from member

select join borrowed on borrowed.mem_no = member.mem_no;

Retrieve outer join record:-

Output: outer join and full join are the same

select member.name, borrowed.mem_no from member, member.mem_no;

VEL TECH	
EX NO.	
PERFORMANCE (5)	
RESULT AND ANALYSIS (5)	
VIVA VOCE (5)	
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	

Result:- Thus the implementation of SQL Commands using joins and recursive queries are executed successfully.

TASK-5:- PL/SQL procedures functions, loops.

Aim:- to implement PL/SQL procedures, functions and loops on number theory and business scenarios.

procedure:-

PL/SQL is combination SQL along with procedural feautres of programming language it was developed by oracle in the early 90's to enhance the capablites of SQL. PL/SQL is on the three key programming language embedded in the oracle along with SQL it self and java.

S.NO	sections & description.
1	Decalaration. this section start with the key word Declare it an optinal section and defines all variable cursors, subprograms, and other element to be used in the program
2.	Executable Commands. this section is enclosed between the key words Begin and end and it is a mandatory section. Consist the excutable PL/SQL statement of the program. it should have at least one exatable of code, which may be just a NULL command to indicate that nothing should be executed
3.	Exception handling this section starts with the keyword exception this optinal section contains exptions(s) [illegible] program

simple program to print a sentence.

syntax:-

```
DECLARE
  < declaration section >
BEGIN
  < executable commands(s)>
EXCEPTION
  < exception handling >
END;
```

program:-

```
DECLARE
Message varchar2(20):='booking closed";
BEGIN
dbms_output_line (message);
END;
```

static input:-

```
sql> set serveroutput on
sql> declare.
  2  x number <5>;
  3  y number <5>;
  4  z number<9>;
  5  begin
  6  x:=10;
  7  y:=12;
  8  z= x+y;
  9  dbms_output.put_line ('sum is':: z);
  10 end;
  11 /
Sum is 22
PL/SQL procedure successfully completed.
```

Syntax:-

```
SQL> declare
  2  var1 integer;
  3  var2 integer;
  4  var3 integer;
  5  begin
  6  var1:= &var1;
  7  var2:=&var2;
  8  var3:= var1+var2;
  9  dbms_output.put_line(var3);
  10 end;
  11 /
Enter value for var1: 20
old   6: var1:=&var1;
new   6: var1:=20;
Enter value for var2: 30
old   7: var2:=&var2;
new   7: var2:=30;
50
PL/SQL procedure successfully completed.
```

DECLARE
hid number(3):=100;
BEGIN
IF(hid=10)THEN
dbms_output.put_line('value of hid is 10');
ELSIF (hid=20)THEN
dbms_output.put_line('value of hid is 20');
ELSIF (hid=30) then
dbms_output.put_line(value of hid 30);
else
dbms_output.put_line("None of the value matches

END;
dbms_output.put_line('Exact table of
hid is;
END;
/
Procedure the table is matching
exact table of hid is : 1
PL/SQL procedure successfully completed.

```
DECLARE
hid number(1);
oid number(1);
BEGIN
<<outer_loop>>
FOR hid IN 1..3 LOOP
<<inner_loop>>
FOR oid IN 1..3 LOOP
dbms_output.put_line('hid is:'||hid||' and oid is:'||oid);
END LOOP inner_loop;
END LOOP outer_loop;
END;
/
hid is: 1 and oid is: 1
hid is: 1 and oid is: 2
hid is: 1 and oid is: 3
hid is: 2 and oid is: 1
hid is: 2 and oid is: 2
hid is: 2 and oid is: 3
hid is: 3 and oid is: 1
hid is: 3 and oid is: 2
hid is: 3 and oid is: 3
```

Task 5:- pl/sql procedure, function

```
SQL*Plus: Release 11.2.0.2.0 Production on Thu Sep 25 14:41:38 2025

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

SQL> connect
Enter user-name: system
Enter password:
ERROR:
ORA-01017: invalid username/password; logon denied


SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> set serveroutput on
SQL> declare
  2  x number(5);
  3  y number(5);
  4  z number(9);
  5  begin
  6  x:=10;
  7  y:=12;
  8  z:=x+y;
  9  dbms_output.put_line('sum is'|| z);
 10  end;
 11  /
sum is22

PL/SQL procedure successfully completed.

SQL> declare
  2  var1 integer;
  3  var2 integer;
  4  var3 integer;
  5  begin
  6  var1:=&var1;
  7  var2:=&var2;
  8  var3:=var1+var2;
  9  dbms_output.put_line(var3);
 10  end;
 11  /
```

```
Enter value for var1: 20
old   6: var1:=&var1;
new   6: var1:=20;
Enter value for var2: 30
old   7: var2:=&var2;
new   7: var2:=30;
50

PL/SQL procedure successfully completed.

SQL> create or replace procedure csinformation
  2  (c_id in number,c_name in varchar2)
  3  is
  4  begin
  5  dbms_output.put_line('ID:' || c_id);
  6  dbms_output.put_line('name:' || c_name);
  7  end;
  8  /

Procedure created.

SQL> exec csinformation(101,'raam');
ID:101
name:raam

PL/SQL procedure successfully completed.

SQL> set serveroutput on;
SQL> exec csinformation(101,'raam');
ID:101
name:raam

PL/SQL procedure successfully completed.
```

VEL TECH	
EX NO.	6
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	19
SIGN WITH DATE	

Result:— pl/sql procedures, functions loop executed successfully.

on created

```
> declare
2 mesg varchar2 <200>;
3 begin.
4 mesg := cs information2 <102, 'raman'>;
5 dbms_output.put_line <mesg>;
6 end;
7 /
```

vehicle available.

```
SQL> declare
2 mesg varchar2 <200>;
3 begin
4. mesg := cs information 2 <206, 'raman'>;
5 dbms_output.put_line <mesg>;
6 end;
7 /
```

ANO vehicle avaliable

pl/SQL procedure successfully completed.

VEL TECH	
EX NO.	6
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	5
VIVA VOCE (5)	5
RECORD (5)	4
TOTAL (20)	19
SIGNATURE WITH DATE	

Result:-

PL/SQL procedures, functions loops executed successfully

TASK-7 PL/SQL procedure for loops:

Aim:- to write PL/SQL programs using loops for printing prime number customer ID's and for demonstrating loop control in different scenarios.

procedure:-

1. start a PL/SQL block or procedure.
2. use a cursor (if required) to fech customer ID from a table.
3. for each ID, check whether it is prime number using a loop.
4. use for loop/ while loop to demonstrate prime checking
5. print the result using DBMS_output.put_line
6. end the block.

Example 1: using while loop with cursor.

prime check using WHILE loop.

```
Create or replace procedure print_prime customer
cursor cust_cur is
  select customer_id from customer;
  v_id number;
  v_is_prime BOOLEAN;
  v_i number;
begin
  open cust_cur;
  loop
    fetch cust_cur INTO v_id;
```

```
exit when cust_cur%NOTFOUND;
-- prime check using while loop
if v_id < 2 then
  v_is_prime := FALSE;
ELSE
    v_is_prime := true;
    v_i := 2;
  while v_i <= TRUNC (SQRT (v_id)) Loop
   if MOD (v_id, v_i) = 0 then
        v_is_prime := false;
        exit;

      v_i := v_i + 1;
    End loop;
End if;
If v_isprime then
DBMS_output.put_line (prime customer ID' || v_id);
End if;
end loop;
close cust_cur;
END;
```

this procedure checks all customer ID's in the table and prints the prime ones using while loop.

<u>Example:</u> — 2 using FOR loop for first N prime numbers.

TASK7:- PL/SQL procedure for loops

```
SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> declare
  2     lo number(3);
  3     hi number(3);
  4      n number(2);
  5      m number(2);
  6      c number(20);
  7  begin
  8      dbms_output.put_line('enter the customer id from to limit:');
  9      lo:=&lo;
 10      hi:=&hi;
 11      for n in lo.. hi
 12       loop
 13      c:=0;
 14       for m in 1..n
 15       loop
 16       if mod(n,m)=0 then
 17      c:=c+1;
 18      end if;
 19     end loop;
 20     if c<=2 then
 21    dbms_output.put_line(n||'\n');
 22    end if;
 23    end loop;
 24    end;
 25    /
Enter value for lo: 101
old  9:      lo:=&lo;
new  9:      lo:=101;
Enter value for hi: 120
old 10:      hi:=&hi;
new 10:      hi:=120;

PL/SQL procedure successfully completed.

SQL> declare
  2   bk number(10):=&bk;
  3   s number(20):=0;
  4   r number(20);
  5   m number(20):=bk;
  6   len number(20);
  7  begin
  8   len:=trunc(log(10,bk))+1;
  9   while bk>0
 10   loop
 11     r:=mod(bk,10);
 12     s:=s+power(r,len);
 13     bk:=trunc(bk/10);
 14   end loop;
 15   if m=s
 16   then
 17     dbms_output.put_line('given number is armstrong');
 18   else
```

```
19     dbms_output.put_line('given number is not an armstrong');
20   end if;
21 end;
22 /
Enter value for bk: 2634
old   2:   bk number(32):=&bk;
new   2:   bk number(32):=2634;

PL/SQL procedure successfully completed.

SQL> CREATE OR REPLACE PROCEDURE print_prime_customers IS
  2   CURSOR cust_cur IS
  3     SELECT customer_id FROM customers;  -- Table with customer IDs
  4   v_id NUMBER;
  5   v_is_prime BOOLEAN;
  6   v_i NUMBER;
  7 BEGIN
  8   OPEN cust_cur;
  9   LOOP
 10     FETCH cust_cur INTO v_id;
 11     EXIT WHEN cust_cur%NOTFOUND;
 12
 13     -- Prime check using WHILE loop
 14     IF v_id < 2 THEN
 15       v_is_prime := FALSE;
 16     ELSE
 17       v_is_prime := TRUE;
 18       v_i := 2;
 19       WHILE v_i <= TRUNC(SQRT(v_id)) LOOP
 20         IF MOD(v_id, v_i) = 0 THEN
 21           v_is_prime := FALSE;
 22           EXIT;
 23         END IF;
 24         v_i := v_i + 1;
 25       END LOOP;
 26     END IF;
 27
 28     IF v_is_prime THEN
 29       DBMS_OUTPUT.PUT_LINE('Prime Customer ID: ' || v_id);
 30     END IF;
 31   END LOOP;
 32   CLOSE cust_cur;
 33 END;
 34 /

Warning: Procedure created with compilation errors.

SQL> CREATE OR REPLACE PROCEDURE print_first_n_primes(n NUMBER) IS
  2     v_num     NUMBER := 2;     -- number to check for prime
  3     v_count   NUMBER := 0;     -- how many primes found so far
  4     v_is_prime BOOLEAN;
  5 BEGIN
  6     WHILE v_count < n LOOP
  7       v_is_prime := TRUE;
  8
  9       -- Prime check using FOR LOOP
 10       FOR i IN 2 .. TRUNC(SQRT(v_num)) LOOP
```

```
      IF MOD(v_num, i) = 0 THEN
        v_is_prime := FALSE;
        EXIT;
      END IF;
    END LOOP;

    IF v_is_prime THEN
      DBMS_OUTPUT.PUT_LINE('prime: ' || v_num);
      v_count := v_count + 1;
    END IF;

    v_num := v_num + 1;
  END LOOP;
END;
/
```

VEL TECH	
EX NO.	7
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	4/9/25

RESULT:- To write pl/sql program using loops for printing prime number customer RDS and for demostracting loop control in different scenarious executed successfully

```
procedure print_first_n_prime
(n number) is
num number := 2;
count number := 0;
is_prime BOOLEAN;
BEGIN
while v_count < n loop
    v_is_prime := true
  -- prime check using for loop ---
for i in 2 . trunc (sqrt (v_num)) loop
    if MOD(v_num, i) = 0 then.
        v_is_prime := false;
    exit;
end if;
end loop;
if v_is_prime then.
DBMS_output_Line (prime:' || v_num);
  v_count := v_count + 1;
end if;
  v_num := v_num + 1;
end loop;
end;

BEGIN
    print_first_n_prime(10)); -- print first
                                10 prime numbers --
END;
```

VEL TECH	
EX NO.	7/5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	4/11/24

result: — PL/SQL procedure for loops executed successfully

Task 3: Normalising database using functional dependency up to BCNF.

Given relational tables created task-2 perform normalisation up to BCNF based on given dependencies as follows for the same relation specified below.

1. Identify employee attributes: employee_ID, name, dept, job_title, manager_ID, hire_date, salary.
2. Define relational schema: employee (emp_ID, name, dept, job_title, manager_ID, hire_date, salary)
3. Determine functional dependencies (FDs) between attributes

- Employee_ID → name, department, job_title, manager_ID, hire_date, salary.
- department → manager_ID
- manager_ID → name.

Step 2:— Convert to 1NF.

1. Eliminate repeating groups or arrays (none in this example)
2. Create separate tables for each repeating group.

Step 3:— Convert to 2NF

1. Ensure each non-key attribute depends on the entire key.
2. Move non-key attributes to separate tables if they depends on only part of the primary key.

- Create department table
- Create employee table.

Step 4:— Convert to 3NF

1. Ensure there are no transitive dependencies.
2. Move non-key attributes to separate tables if they depend on another key attribute
- Create manager table
- Update department table.

FUNCTIONAL DEPENDENCY :

Attributes in Table

[illegible]

employee_id name department job_title hire_date manager_id hire_date salary

Functional Dependencies

employee_id → name department job_title hire_date manager_id salary

Generate

Add Invalid Dependency

NORMAL FORM :

Check Normal Form

2NF
The table is in 2NF

3NF
The table is in 3NF

BCNF
The table is in BCNF

Show Steps

2NF

Find all candidate keys. The candidate keys are { employee_id }. The set of key attributes = { employee_id }
for each non-trivial FD, check whether the LHS is a proper subset of any candidate key or a proper subset of any key attribute
checking FD: employee_id --> name department job_title hire_date manager_id salary

3NF

Find all candidate keys. The candidate keys are { employee_id }. The set of key attributes = { employee_id }
for each FD, check whether the LHS is superkey or the RHS is a key attribute
checking functional dependency employee_id --> name department job_title hire_date manager_id salary

BCNF

A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.

CONVERT 2NF :

Normalize to 2NF

Attributes

employee_id | name | department | job_title | manager_id | hire_date | salary

Functional Dependencies

employee_id --> name department job_title hire_date manager_id salary

Show Steps

```
First, find the minimal cover of the FDs, which includes the FDs
employee_id --> name
employee_id --> department
employee_id --> job_title
employee_id --> hire_date
employee_id --> manager_id
employee_id --> salary
Initialy rel[1] is the original table:
Round1: checking table rel[1]
***** The table is in 2NF already, send it to output *****
```

CONVERT 3NF :

1NF to 3NF

Attributes

employee_id | name | department | job_title | manager_id | hire_date | salary

Functional Dependencies

employee_id --> name
employee_id --> department
employee_id --> job_title
employee_id --> hire_date
employee_id --> manager_id
employee_id --> salary

Show Steps

```
Table already in 3NF
```

VERT BCNF :

malize to BCNF

butes

oyee_id | name | department | job_title | manager_id | hire_date | salary

ctional Dependencies

oyee_id → name department job_title hire_date manager_id salary

ow Steps

as already in BCNF, return itself.

eps:— Convert to BCNF.
ensure every determinant is a candidate key.
check for overlapping candidate key
decompose relations to eliminate redundancy
no further decomposition needed.

sing Griffith tool:—
. Input relations schema functional dependencies
. Griffith tool generates a dependency graph
. Analyze the graph to identify normalization issues.
. Apply normalization rules to transform the schema
. Verify the resulting schema meets BCNF criteria

Griffith tool steps:—
1. create a new project in griffith.
2. define the relational schema and FD's
3. run the dependency graph tool.
4. Analyze the graph for normalization issues.
5. Apply transformation using the normalize tool.
6. verify BCNF compliance using the BCNF check "tool".

VEL TECH	
EX NO.	8
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	25
MARK DATE	

Normalized schema:—
1. employee (employee_ID, ~~name~~, department_ID, job title, hire_date, salary)
2. department (department_ID, manager_ID).
3. manager (manager_ID, Name

Task 4:- Backing up and recovery in database

perform following backup and recovery scenarios.

a) recovering a NONARCHIVELOG database with incremental backup.

b) restoring the server parameter file.

c) performing recovery with a backup control file.

scenario 1: recovering a nonarchivelog database with increment backups.

step 1: backup database.

BACKUP DATABASE [database_name] TO DISK = "backup_file.bak' WITH NOFORMAT, NOINIT, NAME = NAME = 'FULL Database backup', SKIP, SKIP, REWIND NOUNLOAD STATS = 10.

step 2:- Create incremental backup:-

BACKUP DATABASE [database_name] TO DISK = 'increment_backup.bak' WITH DIFFERENTIAL, NOFORMAT. NAME = incremental database backup, SKIP, REWIND, NOUNLOAD, STATS = 10.

step 3:- Simulate data loss

- Intentionally delete or modify data

step 4:- restore database

RESTORE DATABASE [database_name] FROM DISK = "backup_file.bak' WITH REPLACE.

step 5:- Apply incremental backup.

RESTORE DATABASE [database_name] FORM DISK = 'increment backup.bak' WITH REPLACE

step 6:- recover database.

RECOVER DATABASE [database_name]

step 7:- open database

ALTER DATABASE [database_name] SET ONLINE

Scenario 2:- restoring the server parameter file (SPFILE)

step 1:- BACKUP SPFILE.

BACKUP SERVER PARAMETER FILE TO FILE
= spfile.bak';
- step2:- simulate spfile loss.
- delete or modify spfile
step3:- restore spfile
STARTUP MOUNT
RESTORE SERVER PARAMETER FILE FROM
FILE = 'spfile.bak.
SHUTDOWN
STARTUP.

Scenario3:- performing recovery with backup control file
step1:- backup control file
BACKUP CONTROLFILE TO FILE = 'controlfile.bak';
- step2:- simulate control file loss.
- delete or modify control file
- step3:- restore control file
STARTUP MOUNT
RESTORE CONTROLFILE FROM FILE = 'controlfile.bak'
ALTER CONTROLFILE REUSE;
step4:- Recover database.
RECOVER DATABASE USING BACKUP CONTROLFILE
step5:- open database.
ALTER DATABASE OPEN RESETLOGS;

Saw server Commands:-
- BACKUP DATABASE
- RESTORE DATABASE.
- RECOVER DATABASE
- ALTER DATABASE.
- BACKUP SERVER PARAMETER FILE.
- RESTORE SERVER PARAMETER FILE.
- BACKUP CONTROLFILE
- RESTORE CONTROLFILE.

VEL TECH	
EX NO.	9
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

Result:- Backing up and recovery in data base
Perform following backup and recovery executed successfully

25-10-22

TASK 10 - CRUD OPERATIONS IN DOCUMENT DATABASES.

Aim: —

to perfom mangose using NPM design on Mongo DB designing document database and perfoming CRUD operations like creating inserting like creating, querying, finding and remove operations

STEPS: —

step1) install mango using following link.
https // www mangodb com/try/download/community

step2) install mongosh using the below link.
https// w.w.w mangobb.com/mongbb-shell #download-and-install-mongosh.

step3) to add the mongobd shell binary's location to your path environment variable.

open the control panel.

In the system and security category click system click advanced system system properties model display click Enviroment variables.

In the system variables selection path and click edit. enviroment variables model displays.

click new add the filepath to your mongosh binary.

click to conform your changes on each other model click ok to confirm your changes.

To conform that your path enviorment variable correctly configured to find mongosh open command help - command if your path is correctly list valid commands display.

step4) open mango shell 4.0 from
c:\program files\mangodb\server\bin\mangod.exe
step5) type the CRUD (CREATE READ UPDATE DELETE) COMMANDS GIVEN IN TEXT FILE.

CRUD OPERATIONS :—

```
db.create Collection("mylab")
{"ok":1}
> db.mylab.insert one {item "Canvas", qty: 100 tags:
[cotton] size {h: 28, w: 35.5, unom: "cm"}}
{
"acknowledged": true,
"insertedId": objectId("627d113aee73990c74637
}
> db.mylab.find ({item: "Canvas"})
{"id": objectId("627d13aee73990c74637c") item Can
"qty": 100 "tags": ["cotton"], "size": {"h": 28 "w" 35.5
: "cm"}}
> db.mylab.insertmany {item s, journal qty 25 tags ["blue
"red"] size { h: 14 w: 21 unom: "cm"} {item may qty
tags ["gray] size {h 27, w 35.5 uom 35.5 unom cm}
{item "mouse pad" qty: 25 tags ["gel "blue"], size:
unom: "cm"}}]
{
"acknowledged": true
"insertedIds": [
    objectId("627d15598c73990c074639
    objectId("627d15598c73990c074e637e")
    objectId("627d15598c73990c074e639f
]
```

> db.myc[illegible] find ({}, {item, qty, _id})
{ "_id": ObjectID("627d13aee739074e639c"), item
canvas "qty": 100 }
{ "_id": ObjectID("627d1598c739074e74637d"), item
journal "qty": 25 }.
{ "_id": ObjectID("627d1598c734637e"), item "mat"
qty 85 }
{
"_id": ObjectID("627d13aee4e6397c"),
"item": "Canvas
qty": 100
}
{
"_id" ObjectID("627d1598c739900c4e6397d"),
item: journal.
qty: 25
}
{ "_id": ObjectID("627d1598c739900c74e6391"),
{
"id": ObjectID("627d1598c739900074e6391f"),
"item": mus pad",
"qty": 25.
"_id" ObjectID("627d13aee739900074 6397c"),
item": "Canvas",
"qty": 100,
"tags": [
"Cotton"
],
"size": {
"h": 28
"w": 35.5,
"uom": "cm"
} }

```
db.createCollection("mylab")
db.mylab.insertOne({item:"canvas",qty:100,tags:["cotton"],size:{h:28,w:35.5,uom:"cm"}})
db.mylab.find({item:"canvas"})
db.mylab.insertMany([{item:"journal",qty:25,tags:["blank","red"],size:{h:14,w:21,uom:"cm"}}
,
{item:"mat",qty:85,tags:["gray"],size:{h:27.9,w:35.5,uom:"cm"}},
{item:"mousepad",qty:25,tags:["gel","blue"],size:{h:19,w:22.85,uom:"cm"}}])
db.mylab.find({},{item:1,qty:1})
db.mylab.find({},{item:1,qty:1}).pretty()
db.mylab.find({item:"canvas"}).pretty().sort({item:-1})
db.mylab.deleteOne({item:"journal"}
db.mylab.find({},{item:1,qty:1}).pretty()
```

output:

Output:

```
{ "ok" : 1 }
{
        "acknowledged" : true,
        "insertedId" : ObjectId("68efcd6e4b9c34878362b38e")
}
{ "_id" : ObjectId("68efcd6e4b9c34878362b38e"), "item" : "canvas", "qty" : 100, "tags" : [ "cotton" ], "size" : { "h" : 28, "w" : 35.5, "uom" : "cm" } }
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("68efcd6e4b9c34878362b38f"),
                ObjectId("68efcd6e4b9c34878362b390"),
                ObjectId("68efcd6e4b9c34878362b391")
        ]
}
{ "_id" : ObjectId("68efcd6e4b9c34878362b38e"), "item" : "canvas", "qty" : 100 }
{ "_id" : ObjectId("68efcd6e4b9c34878362b38f"), "item" : "journal", "qty" : 25 }
{ "_id" : ObjectId("68efcd6e4b9c34878362b390"), "item" : "mat", "qty" : 85 }
{ "_id" : ObjectId("68efcd6e4b9c34878362b391"), "item" : "mousepad", "qty" : 25 }
{
        "_id" : ObjectId("68efcd6e4b9c34878362b38e"),
        "item" : "canvas",
        "qty" : 100
,
```

"_id" : ObjectId("68efcd6e4b9c34878362b38f"),
"item" : "journal",
"qty" : 25

" : ObjectId("68efcd6e4b9c34878362b390"), "item" : "mat", "qty" : 85 }

"_id" : ObjectId("68efcd6e4b9c34878362b391"),
"item" : "mousepad",
"qty" : 25

"_id" : ObjectId("68efcd6e4b9c34878362b38e"),
"item" : "canvas",
"qty" : 100,
"tags" : [
"cotton"
],
"size" : {
"h" : 28,
"w" : 35.5,
"uom" : "cm"
}

"_id" : ObjectId("627d13acc73990c074e6397c"),
"item" : "canvas",
"qty" : 100
}
{
"_id" : ObjectId("627d1598c73990c074e6397d"),
"item" : "journal",
"qty" : 25
}
{ "_id" : ObjectId("627d1598c73990c074e6397e"), "item" : "mat", "qty" : 85 }
{
"_id" : ObjectId("627d1598c73990c074e6397f"),"item" : "mousepad","qty" : 25}

objectid("627d13a0e739900c74e637c"),
: "Canvas"
: 25

d": objectid("627d1598c739900c74e637e")
n": mat": qty: 85}

": objectid("627d1987074639 7f") : "marker
qty": 25}

VEL TECH	
EX NO.	10
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	20
TOTAL (20)	
SIGN WITH DATE	

Result:— the implementation of CRUD operations like creating inserting, finding and removeing operations using mangoDB is successfully executed.

TASK 11 - CRUD OPERATIONS IN GRAPH DATA BASES.

Aim: — to perform crud operations like creating inserting, querfing, finding, delete operations on graph spaces

* Create node with properties: —
properties are the key-value pairs using which a node stores data. You can create a node with properties using the Create clause. you need specify these properties separated by commans "flower braces" { }.

Syntax: —
Following is the syntax to create node with properties.
CREATE (node. label {key: value. key2 value---}

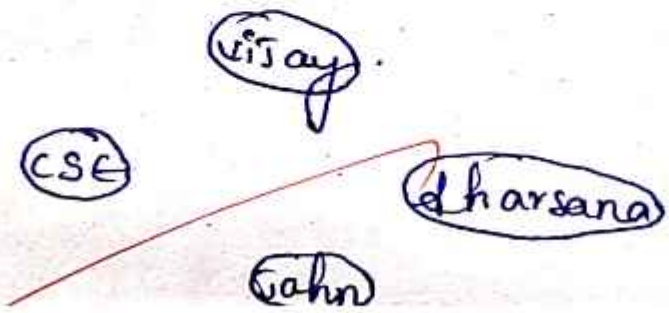
* Verifying the creating node: —
To verify the creation of the node type of execute the following query in the dollar prompt
MATCH (n) RETURN n

* Creating relationships: —
we can create a relationship using the CREATE clause. we will specify relationship with the square braces "[ ]" depending on the direction of the relationship it is placed between hyphen "-" and arrow "→" as shown the following syntax

Syntax: —
Following is the syntax to create a relationship using the CREATE clause.
CREATE (node1)-[Relationship type]->(node2)

create (n:student { sid : "VTU 30721",
sname : "John"
dept name "CSE" }

output :—
added 1 label, created 1 node, set 3 properties
completed after 16 ms.
create (n, student { sid : "VTU 3063",
sname "vijay",
dept name : "CSE"
})
output :—
added 1 label, created 1 node, set 3 properties
completed after 12 ms.
create (n.dept { deptname "cse", dpt id : "doo1" })
output :—
added 1 label, created 1 node, set 2 propties
completed after 72 ms.
select all nodes in your database using
command.
* match (n) return (n)
output :—

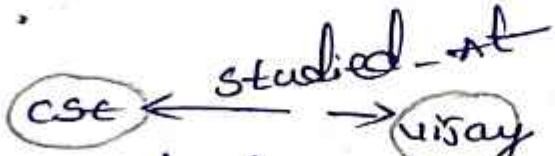


output:—

a) create relationship between student and cse.

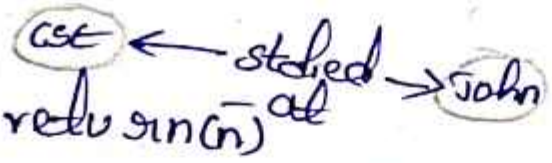
MATCH (S.student) (d.dept) WHERE S name. vijay AND d. deptname = 'cse'
Create (S)-[St: STUDIED-AT] -> (d)

output:—

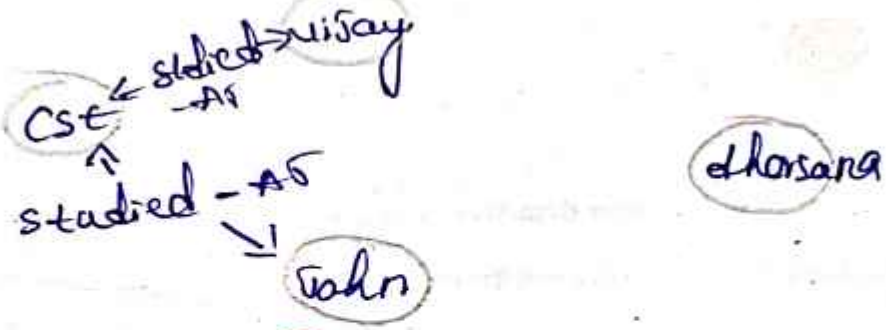


MATCH (S. student) (d. dept) . WHERE S. name='
AND d. deptname = 'CSE' CREATE (S)-(St. STUDIED
-AT] -> (d)

output:—



match (n) return (n)



b) delete a node from student

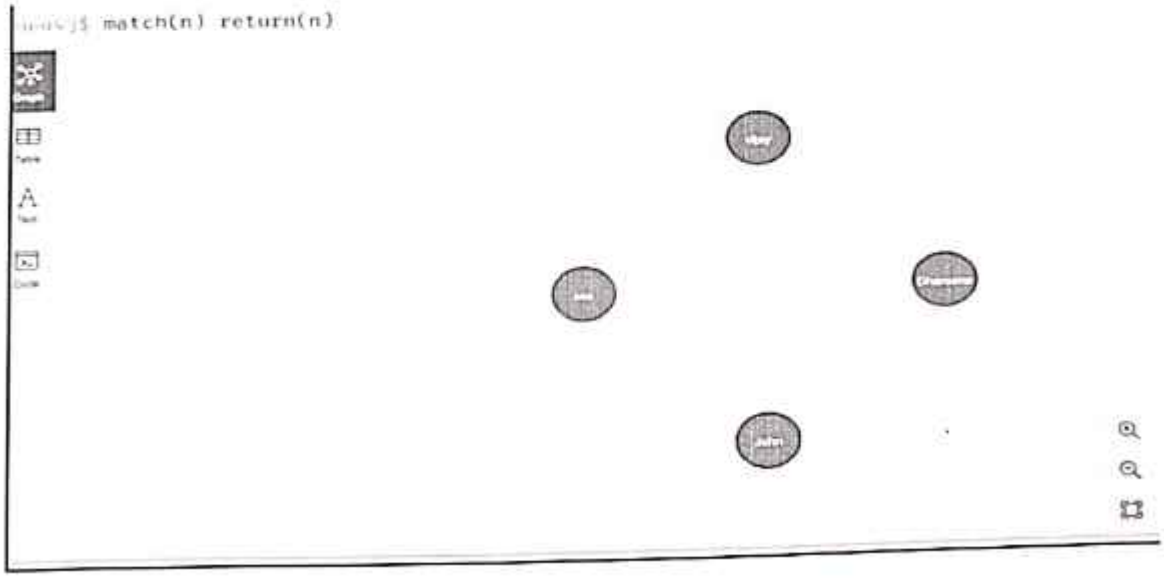
match (n: student {sname : Dharsna'}) DELETE

output:—

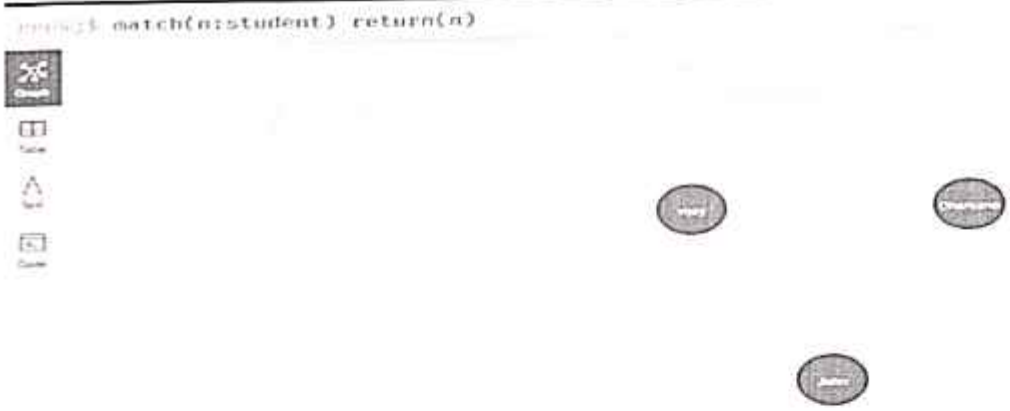
Deleted 1 node, completed after 10834 ms.

Select all the nodes in your database using match command.

match(n) return(n)



match(n:student) return(n)



a) Create relationship between student and cse .

MATCH(s:student),(d:dept) WHERE s.Sname ='vijay' AND d.deptname='cse'

CREATE(s)-[st:STUDIED_AT]->(d)

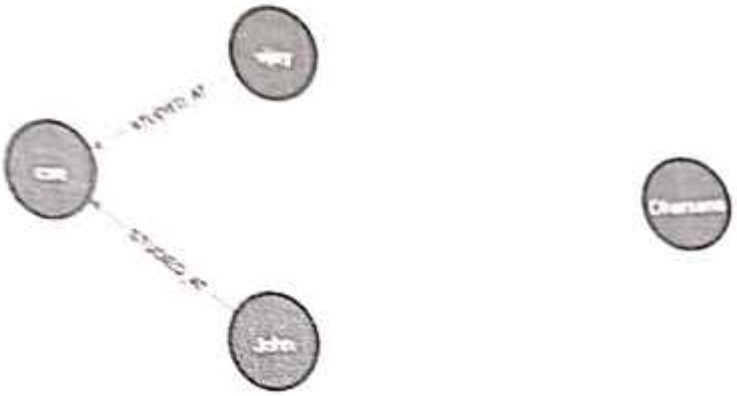
return s,d

MATCH(s:student),(d:dept) WHERE s.Sname = 'Vijay' AND d.Dnamme = 'CSE'

CREATE(s)-[st:STUDIED_AT]->(d)

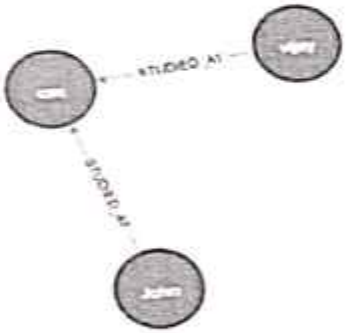
return s,d

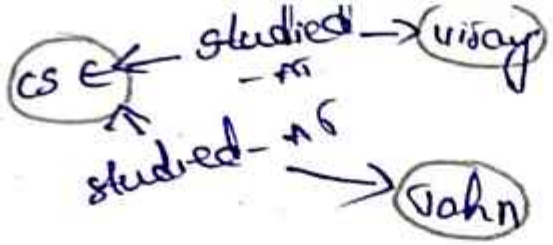
```
match(n) return(n)
neo4j$ match(n) return(n)
```



b) Delete a node from student

```
match(n:student{Sname:'Dharsana'}) Delete(n)
neo4j$ match(n)return(n)
```





VEL TECH	
EX NO.	11
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	(20)
SIGN WITH DATE	9/10/25

Result:— the, implementation of crud operations like creating, inserting, finding and remove operations using graph DB, is successfully executed.

DATABASE MANAGEMENT SYSTEM

(10211DS207)

TASK NO: 12

ORPHANAGE MANAGEMENT SYSTEM

TEAM DETAILS:

Team Leader: PRONAY MONDAL(VTU 30566)

Reg No: 24UEDC0054

Team Members:

Names:	VTU:	Registration No:
Talakonda Gopalaswamy	VTU30[illegible]0	24UEDC0005
Gujjneni Harikrishna	VTU3063[illegible]	24UESS0019
Annem Naga Gotham Reddy	VTU30576	24UEDC0064
Katreddy Gurunath Reddy	VTU33242	

16-10-25

DATABASE MANAGEMENT SYSTEM

(10211DS207)

TASK NO: 12

ORPHANAGE MANAGEMENT SYSTEM

TEAM DETAILS:

Team Leader: PRONAY MONDAL(VTU 30566)

Reg No: 24UEDC0054

Team Members:

Names:	VTU:	Registration No:
Talakonda Gopalaswamy	VTU30540	24UEDC0005
Gujjneni Harikrishna	VTU30631	24UESS0019
Annem Naga Gotham Reddy	VTU30576	24UEDC0064
Katreddy Gurunath Reddy	VTU33242	

Aim:
To develop a microproject on Orphanage Management System.

1.ER Diagram

The ER diagram models the data structure of an orphanage management system. It helps visualize how different entities—such as children, staff, donors, and rooms—are related, enabling efficient database design and streamlined operations like tracking admissions, donations, events, and medical records.

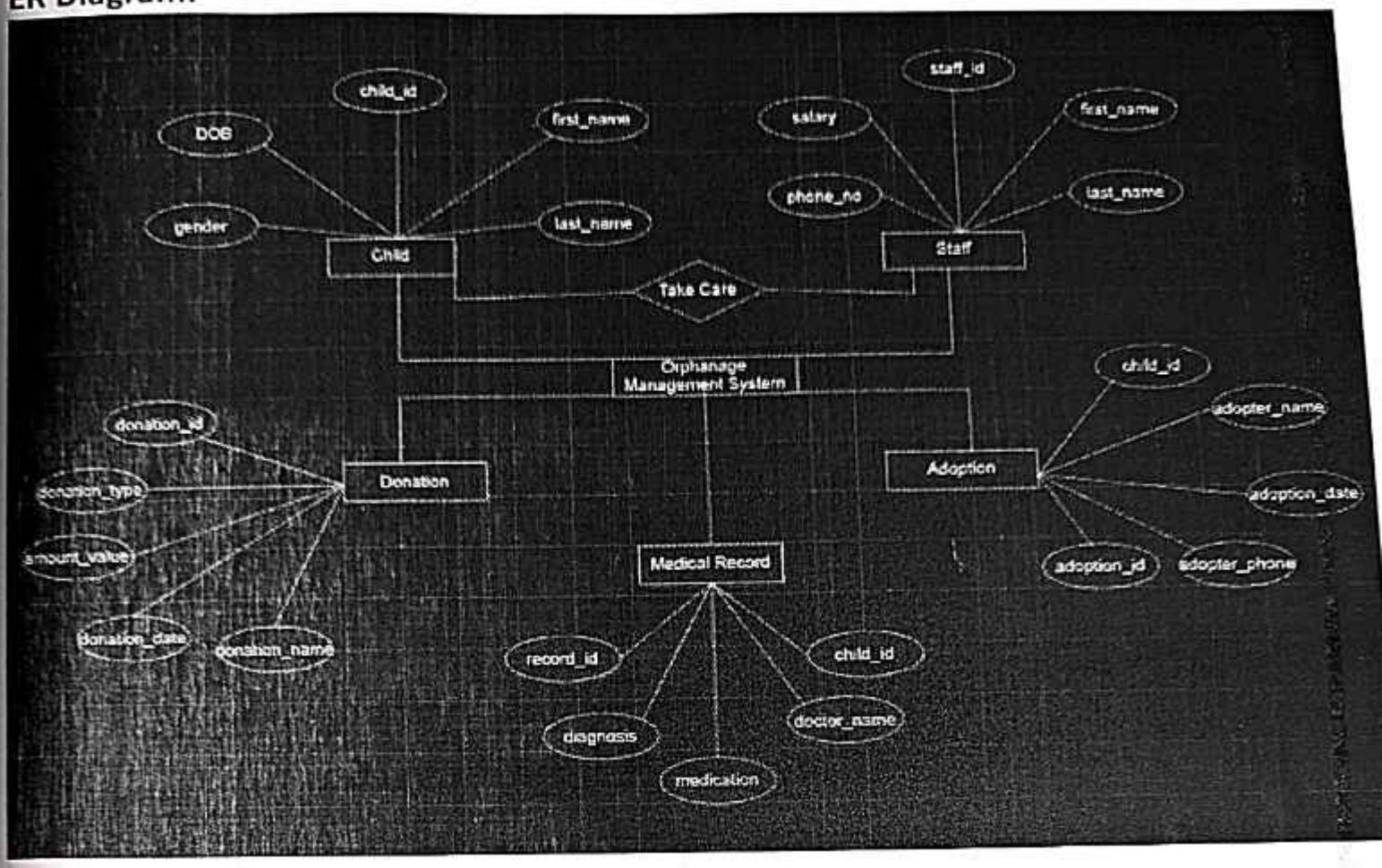
Entities:

1. Child
   -Attributes: Child_ID(PK), First name, Last Name, Gender.
2. Staff
   -Attributes: Staff_ID(PK), First name, Last Name, Phone_no.
3. Adoption
   -Attributes: Adopter_name, Adopter_date, Adopter_phone, Adoption_ID.
4. Medical Record
   -Attributes: Medication, Doctor_name, Diagnosis, Record_ID.
5. Donation
   -Attributes: Donation_ID, Donation_type, Amount_value, Donation_date.

Relationships:

- One Staff member (Caretaker) can care for many Children, but each Child is assigned to one Caretaker.
- One Child has many Medical Records over time.
- One Sponsor can make multiple Donations.
- A Child can only be involved in one final Adoption record (though they may have many pending inquiries).

ER Diagram:


child_id
DOB
first_name
gender
last_name
Child
staff_id
salary
first_name
phone_no
last_name
Staff
Take Care
Orphanage Management System
donation_id
donation_type
amount_value
donation_date
donation_name
Donation
Medical Record
record_id
child_id
diagnosis
doctor_name
medication
child_id
adopter_name
Adoption
adoption_date
adoption_id
adopter_phone

2. SQL Queries and relational operations:

I) Select operation:

SELECT first_name, last_name, gender FROM Child;

first_name	last_name	gender
Alice	Brown	Female
Ben	Smith	Male
Clara	Johnson	Female
David	Lee	Male
Ella	Davis	Female

II) Project operation:

SELECT child_id, date_of_birth FROM Child;

child_id	date_of_birth
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

III) Union All:

```
SELECT child_id, date_of_birth FROM Child
UNION ALL
SELECT child_id, date_of_birth FROM Child
WHERE date_of_birth > '2015-01-01';
```

child_id	date_of_birth
1	2014-05-24
2	2016-11-12
3	2015-07-19
4	2016-01-05
5	2012-08-20
6	2015-07-19
7	2016-03-26

IV) Union:

```
SELECT child_id, date_of_birth FROM Child
UNION
SELECT child_id, date_of_birth FROM Child
WHERE date_of_birth > '2015-01-01';
```

child_id	date_of_birth
1	2014-05-24
2	2016-11-12
3	2015-07-19
4	2016-01-05
7	2016-03-26

V) Intersect:

```
SELECT child_id, date_of_birth FROM Child
INTERSECT
SELECT child_id, date_of_birth FROM Child
```

WHERE date_of_birth > '2015-01-01';

child_id	date_of_birth
[illegible]	[illegible]
[illegible]	[illegible]

VI) Sum:

SELECT SUM(child_id) AS sum_of_child_ids FROM Child;

sum_of_child_ids
[illegible]

VII) Count:

SELECT COUNT(*) AS total_children_after_2015
FROM Child
WHERE date_of_birth > '2015-01-01';

total_children_after_2015
[illegible]

VIII) AVG:

SELECT AVG(staff_caretaker_id) AS average_caretaker_id FROM Child;

average_caretaker_id
[illegible]

IX) Minimum and Maximum:

SELECT MIN(child_id) AS min_id, MAX(child_id) AS max_id FROM Child;

min_id	max_id
[illegible]	[illegible]

X) Nested Queries:

```
SELECT * FROM Child
WHERE child_id = (
  SELECT MAX(child_id) FROM Child
);
```

child_id	first_name	last_name	date_of_birth	date_admitted	gender	status	staff_care
[illegible]	Eva	Garcia	[illegible]	[illegible]	F	Active	[illegible]

XI) PL SQL Procedure:

```
DECLARE
    v_avg_child_id NUMBER;
BEGIN
    -- Calculate average child_id for children born after 2015
    SELECT AVG(child_id)
    INTO v_avg_child_id
    FROM Child
    WHERE date_of_birth > TO_DATE('2015-01-01', 'YYYY-MM-DD');

    -- Output the result
    DBMS_OUTPUT.PUT_LINE('Average child_id (born after 2015): ' || ROUND(v_avg_child_id, 2));
END;
/
```

Output:

```
Average child_id (born after 2015): 12.5
```

JOINS:

A) INNER JOINS:

```
SELECT
c.child_id,c.first_name,c.last_name,c.date_of_birth,s.staff_name,s.staff_role
FROM
  Child c
INNER JOIN
  Staff s
ON
  c.staff_caretaker_id = s.staff_caretaker_id;
```

child_id	first_name	last_name	date_of_birth	staff_name	staff_role
[illegible]	Alice	Brown	[illegible]	Mr. Adams	Supervisor
[illegible]	Ben	Smith	[illegible]	Ms. Taylor	Caregiver
[illegible]	Clara	Johnson	[illegible]	Mr. Clark	Nurse
[illegible]	David	Wilson	[illegible]	Mr. Adams	Supervisor

B) LEFT OUTER JOIN:

```
SELECT
c.child_id,c.first_name,c.last_name,c.date_of_birth,s.staff_name,s.staff_rol
e
FROM
  Child c
LEFT OUTER JOIN
  Staff s
ON
  c.staff_caretaker_id = s.staff_caretaker_id;
```

child_id	first_name	last_name	date_of_birth	staff_name	staff_role
[illegible]	Alice	Brown	[illegible]	Mr. Adams	Supervisor
[illegible]	Ben	Smith	[illegible]	Ms. Taylor	Caregiver
[illegible]	Clara	Johnson	[illegible]	Mr. Clark	Nurse
[illegible]	David	Wilson	[illegible]	Mr. Adams	Supervisor
[illegible]	Eva	Garcia	[illegible]	NULL	NULL

C) RIGHT OUTER JOINS:

```
SELECT
c.child_id,c.first_name,c.last_name,c.date_of_birth,s.staff_name,s.staff_rol
e
FROM
  Child c
RIGHT OUTER JOIN
  Staff s
ON
  c.staff_caretaker_id = s.staff_caretaker_id;
```

child_id	first_name	last_name	date_of_birth	staff_name	staff_role
[illegible]	Alice	Brown	[illegible]	Mr. Adams	Supervisor
[illegible]	David	Wilson	[illegible]	Mr. Adams	Supervisor
[illegible]	Ben	Smith	[illegible]	Ms. Taylor	Caregiver
[illegible]	Clara	Johnson	[illegible]	Mr. Clark	Nurse
[illegible]	[illegible]	[illegible]	NULL	Ms. Lee	Volunteer

D) FULL OUTER JOIN:

```
SELECT
c.child_id,c.first_name,c.last_name,c.date_of_birth,s.staff_name,s.staff_rol
e
FROM
  Child c
FULL OUTER JOIN
  Staff s
ON
  c.staff_caretaker_id = s.staff_caretaker_id;
```

child_id	first_name	last_name	date_of_birth	staff_name	staff_role
[illegible]	Alice	Brown	[illegible]	Mr. Adams	Supervisor
[illegible]	David	Wilson	[illegible]	Mr. Adams	Supervisor
[illegible]	Ben	Smith	[illegible]	Ms. Taylor	Caregiver
[illegible]	Clara	Johnson	[illegible]	Mr. Clark	Nurse
[illegible]	Eva	Garcia	[illegible]	NULL	NULL
[illegible]	[illegible]	[illegible]	NULL	Ms. Lee	Volunteer

Normalization :

ormalization is the process of organizing data in a database to reduce

edundancy and improve data integrity.

/e normally go through stages called **Normal Forms (1NF → 2NF → 3NF → CNF)**.

or our **Orphanage Management System**, we can start from a sample normalize elation and use the

riffith Normalization Tool to automate the process.

. First Normal Form (1NF)

relation (table) is in **1NF** if all its attributes contain **atomic (indivisible) values**, nd each row

unique.

ey point: No repeating groups or arrays.

. Second Normal Form (2NF)

relation is in **2NF** if it is in 1NF and every non-prime attribute is fully functiona lependent on the whole primary key.

Key point: No partial dependency on a part of a composite key.

. Boyce-Codd Normal Form (BCNF)

A relation is in BCNF if it is in 2NF and every determinant is a superkey.

Key point: Even stricter than 2NF; eliminates anomalies caused by non-key attributes determining other attributes.

- In this databse we perform normalization using Griffith university normalizat tool.

First Normal Form:

Attributes in Table

Separate attributes using a comma (,)

Adopter_ID, Adopter_name, Adopter_date, Adopter_phone

Functional Dependencies

Enter the functional dependencies that are present in your table.

Adopter_ID ×	→	Adopter_name × Adopter_phone ×	Delete
Adopter_name ×	→	Adopter_ID × Adopter_phone ×	Delete
Adopter_phone ×	→	Adopter_name × Adopter_ID × Adopter_date ×	Delete

Add Another Dependency

Check Normal Form

2NF
The table is in 2NF

3NF
The table is in 3NF

BCNF
The table is in BCNF

Show Steps

2NF

```
find all candidate keys. The candidates keys are {Adopter_ID}, {Adopter_name}, {Adopter_phone}. The set of
key attributes are, {Adopter_ID,Adopter_name,Adopter_phone}
for each non-trivial F.D. check whether the LHS is a proper subset of some candidate key or the RHS are not
all key attributes
checking F.D. Adopter_ID --> Adopter_name,Adopter_phone
checking F.D. Adopter_name --> Adopter_ID,Adopter_phone
checking F.D. Adopter_phone --> Adopter_name,Adopter_ID,Adopter_date
```

3NF

```
find all candidate keys. The candidates keys are {Adopter_ID}, {Adopter_name}, {Adopter_phone}. The set of
key attributes are {Adopter_ID,Adopter_name,Adopter_phone}
for each F.D. check whether the LHS is superkey or the LHS are all key attributes
checking functional dependency Adopter_ID --> Adopter_name,Adopter_phone
checking functional dependency Adopter_name --> Adopter_ID,Adopter_phone
checking functional dependency Adopter_phone --> Adopter_name,Adopter_ID,Adopter_date
```

BCNF

```
A table is in BCNF if and only if for every non-trivial F.D. the LHS is a superkey.
```

Check Normal Form

2NF

The table is in 2NF

3NF

The table is in 3NF

BCNF

The table is in BCNF

Show Steps

2NF

```
find all candidate keys. The candidates keys are { Adopter_ID}, { Adopter_name}, { Adopter_phone}. The set of
key attributes are { Adopter_ID,Adopter_name,Adopter_phone }
for each non trivial FD, check whether the LHS is a proper subset of some candidate key or the RHS are not
all key attributes
checking FD Adopter_ID -> Adopter_name,Adopter_phone
checking FD Adopter_name -> Adopter_ID,Adopter_phone
checking FD Adopter_phone -> Adopter_name,Adopter_ID,Adopter_date
```

3NF

```
find all candidate keys. The candidates keys are { Adopter_ID}, { Adopter_name}, { Adopter_phone}. The set of
key attributes are { Adopter_ID,Adopter_name,Adopter_phone }
for each FD, check whether the LHS is a superkey or the RHS are all key attributes
checking functional dependency Adopter_ID -> Adopter_name,Adopter_phone
checking functional dependency Adopter_name -> Adopter_ID,Adopter_phone
checking functional dependency Adopter_phone -> Adopter_name,Adopter_ID,Adopter_date
```

BCNF

```
A table is in BCNF if and only if for every non trivial FD, the LHS is a superkey.
```

Normalize to 2NF

Attributes in Table

Separate attributes using a comma (,)

Adopter_ID, Adopter_name, Adopter_date, Adopter_phone

Functional Dependencies

Enter the functional dependencies that are present in your table

Adopter_ID ×	→	Adopter_name × Adopter_phone ×
Adopter_name ×	→	Adopter_ID × Adopter_phone ×
Adopter_phone ×	→	Adopter_name × Adopter_ID × Adopter_date ×

Add Another Dependency

Normalize to 2NF

Attributes

Adopter_ID | Adopter_name | Adopter_date | Adopter_phone

Functional Dependencies

Adopter_ID → Adopter_phone

Adopter_name → Adopter_phone

Adopter_phone → Adopter_name, Adopter_ID, Adopter_date

Show Steps

Step: find the minimal cover of the FDs, which includes the FDs

Adopter_ID → Adopter_phone

Adopter_name → Adopter_phone

Adopter_phone → Adopter_name

Adopter_phone → Adopter_ID

Adopter_phone → Adopter_date

Initially split for the original table:

R (each) detailing table will 1)

Step: This table is 2NF already, result is in output

Attributes

Adopter_ID | Adopter_name | Adopter_date | Adopter_phone

Functional Dependencies

Adopter_ID → Adopter_phone

Adopter_name → Adopter_phone

Adopter_phone → Adopter_name

Adopter_phone → Adopter_ID

Adopter_phone → Adopter_date

Show Steps

Table already in 3NF

Normalize to BCNF:

Normalize to BCNF

Attributes

Adopter_ID | Adopter_name | Adopter_date | Adopter_phone

Functional Dependencies

Adopter_ID → Adopter_phone

Adopter_name → Adopter_phone

Adopter_phone → Adopter_name, Adopter_ID, Adopter_date

how Steps

Table already in BCNF, return itself

5.Document database using MONGODB :

CRUD,Which stands for Create,Read,Update,and Delete,represents a set of fundamental operations used to insert with and manipulate data stored in a database.These operations serve as the building blocks upon which countless applications,from simple to highly complex,rely.

Create:

```
db.children.deleteOne({ first_name: "Anya", last_name:
"Johnson"
});
db.children.deleteMany({ status: "Left"
});
```

Output:

```
{
  acknowledged
  deletedCount
}
```

Read:

```
db.ch
first
last_name
});
db.ch  Left
});
```

[redacted]

[redacted]

database using MONGODB(using neo4j online compiler)

a graph database:

1:Child {child_id: 'C001', first_name: 'Anjali', last_name: 'Devi', DOB:
15', gender: 'F'})

1 node, set 2 properties, added 1 label

s1:Staff {staff_id: 'S01', first_name: 'Meena', last_name: 'Reddy', salary:
phone_no: '9876543210'})

1 node, set 4 properties, added 1 label

(d1:Donation {donation_id: 'D101', donation_type: 'Cash', amount_value:
donation_date: '2025-09-01', donation_name: 'Gita Trust'})

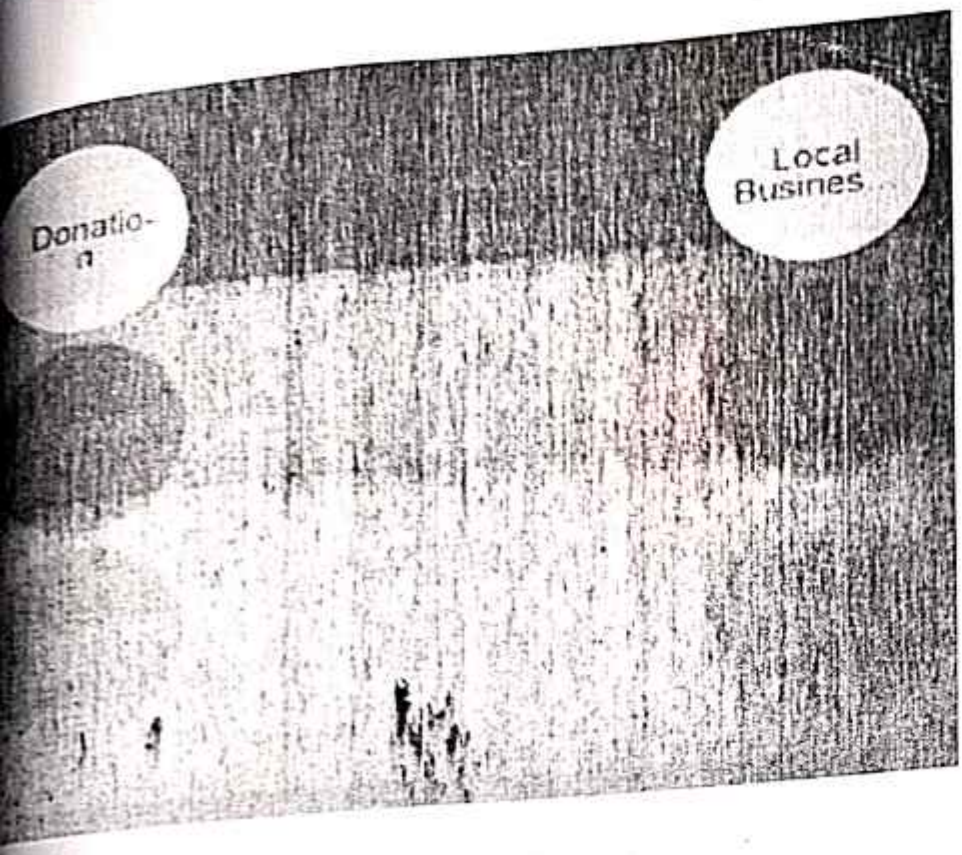
d 1 node, set 3 properties, added 1 label

ATCH (c1:Child {child_id: 'C001'}), (c2:Child {child_id: 'C002'}),

:Staff {staff_id: 'S01'}), (s2:Staff {staff_id: 'S02'})
ATCH (s:Staff)-[r:TAKES_CARE_OF]->(c:Child)

TURN s, r, c

utput:



s,the document database and graph database by using mongodb is
lemented.

ult:

s,Micro Project for Orphanage Management System was developed and
lemented successfully.

VEL TECH	
EX NO.	7
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	23/10/25