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

NAAC A++ HIGHEST GRADE

Recognised as CATEGORY-1 DEEMED TO BE UNIVERSITY by the UGC University Grants Commission

School of Computing

Department of CSE (Data Science)

ACADEMIC YEAR 2025 – 2026 (Summer Semester)

BONAFIDE CERTIFICATE

NAME: B. Kanaka babu

VTU NO: 29144

REG.NO: 24UGDC0014

BRANCH: B.E (DS)

YEAR/SEM: II / III

SLOT: S1 & 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**

INDEX

NAME: B. Kanaka Lakshmi STD: BCS SEC : Roll No :

S. No.	Date	TITLE	Page No.	Teacher's Sign.
1	24/7/25	conceptual design EER	18	
2	31/7/25	hierarchy design other traditional database	18	
3	7/8/25	developing queries pm single row functions opon	18	
4	14/8/25	developing queries on multirow functions	18	
5	21/8/25	writing join queries, Equivalent ANSI	18	
6	28/8/25	TO implement pl/sql procedure Functionsons	18	
7	4/9/25	pl/sql procedure for loop	18	
8	11/9/25	normalizing database using function		
9	18/9/25	backing up and recovery database	17	
10	25/9/25	cord operations in documents	18	
11	9/10/25	cord operations in graph database	19	
12	30/10/25	JDBC BUS booking system	19	

Completed

Task-1:-

Conceptual Design after ETR

Using basic database design, methodology and ER modeler, design Entity relationship Diagram by satisfying following subtasks

1. a identifying entities
   b. attributes
   c. relationships, types of relationship
   d. Reframing relations with keys
   e. Develop ER diagram for stated case of task

Aim:- To draw bank management system of conceptual design through ETR using draw.io

procedure:-

a. identifying entities of b.m.s

Sample output:-

i) Branch
ii) Loan
iii) Account
iv) Customer
v) Avail'd by

b. identifying attributes for b.m.s

Sample output:

i) Branch ID
ii) Name
iii) ADDress
iv) CODE
v) Account_NO
vi) Account_type
vii) Balance
viii) phone

c) identification of Relationships, cardinality, type of relationship for bms

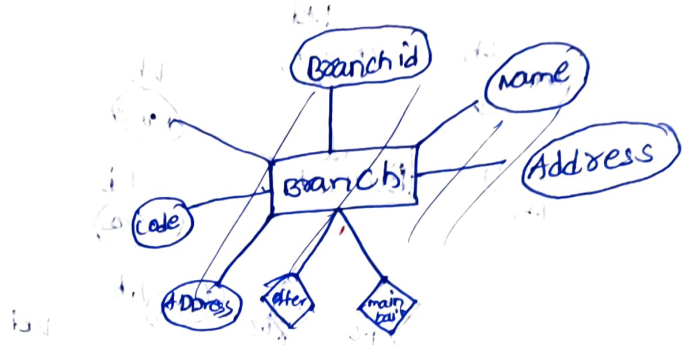
Sample output:

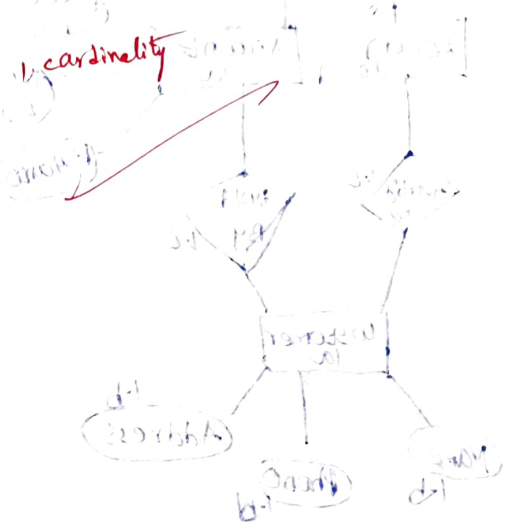
i) Avalid by
ii) Hold by
iii) offer
iv maintain

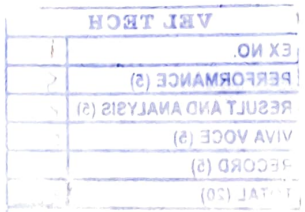
d) Reframing relations with keys and constraints for bms

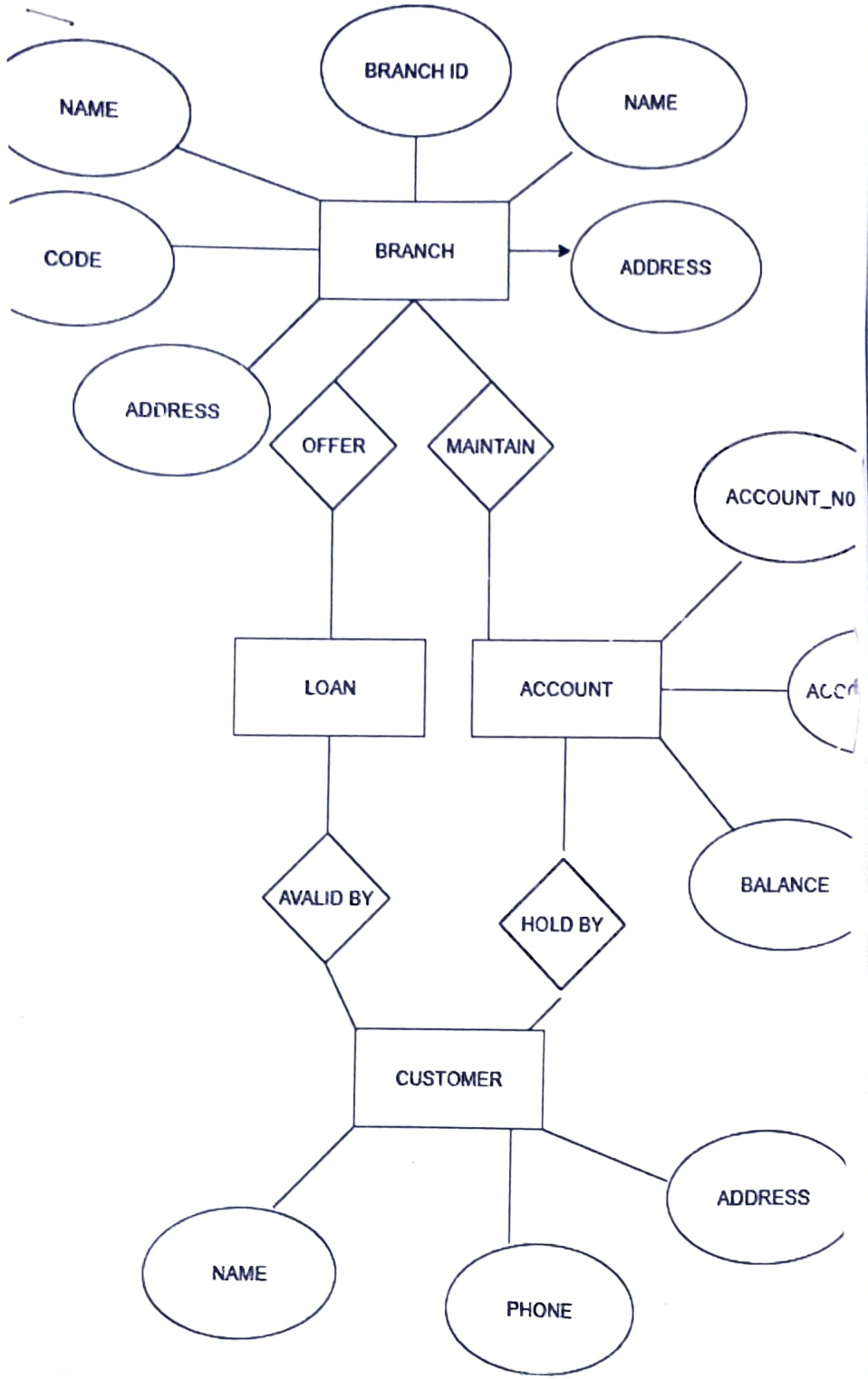
Sample output:

i) Branch id
ii) Account_NO
iii) phone









Task 2:-

Generating design of traditional database model

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

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

Objective:-

* to understood different issues involved in design and implementation of database system
* to understand and use data definition language to write query for database

Theory:-

oracle has many tools such as SQL* plus oracle forms, oracle Report writes, oracles graphics etc--

SQL* plees:-

The SQL* plus tool is made up of two distinct parts. these are

Interactive SQL:-

Interactive SQL is designed for create, access and manipulate data structures like tables and indexes

PL/SQL:- PL/SQL can be used to developed programs for different applications

Oracle Forms:- This tool allows you to create a data only screen along with suitable menu objects Thus it is oracle forms tool that handles data gathering and data validation.

Report writer:-

Report writer allows programmers to prepare informative reports using data from oracle structures like tables, views etc..

Oracle graphics:-

Some of data can be better represented in form of pictures oracle graphics tool allows programmers to prepare progs using data from oracle structure like tables, views etc..

Data Types:-

1) CHAR(size):

This data type is used to store character strings values of fixed len. The size in brackets determines the number of characters

2. VARCHAR(size)

This data types is used to store variable-length alphanumeric data. The maximum character can hold is 2000 characters

3) Number(p,s):

The number data type is used to store num. num of virtual any magnitude may be stored up to 38 digits of precision num as larger as $9.90*10^{24}$.

4) Date:

This data type is used stores date and time. the standard formed is DD-Name xx as in 17 sep-2009. to inter dates other than standard format, use

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

6) RAW: The Raw data type is used to store binary data such as digitized picture or image. Data, such as loaded into columns of data types are stored without any further conversation.

DATA DEFINITION LANGUAGES:-

A DATA definition Language statement are used to define database mangement structured

create:-

TO make a new data base, table, index or stored query. A create statement in SQL created an object inside of relational database mangement sys.

syntax:-

```
create table table_name
(column_name1 datatype(size)
column_name_n data_type (size))
```

Alter:

TO modify an existing database Object Alter structure of data base

syntax:-

```
ALTER TABLE table_name
column_name database (size))
```

* TO delete column in a table

Rename:-

syntax:

```
ALTER TABLE table_name RENAME
column add_column_name TO
new_column_name;
```

Describe:-

syntax:-

```
desc. table_name
```

TRUNCATE TABLE:-

Remove all records from a table, including all spaces allocated for records

syntax:-

```
TRUNCATE TABLE table_name
```

DATA MANIPULATION LANGUAGES

DATA manipulation Language allows users to query and manipulate data existing schema in object.

Insert:-

value can be inserted into table using insert commands. There are two types of insert commands.

Syntax:

INSERT INTO table_name
(value1, value2, value3.---);
(OR)
INSERT INTO table_name (column1,2,3--)

update:

This allows the user to update particular column value using where clause condition

Delete:-

These allows you to delete the particular column values using where clause condition

syntax:

DELETE from <table_name> where condition

Select:

The select statement is used to query a database.

DATA control language:-

1 create:

* create user kama identified by kunal;
user created

2) Grant:

* grant all privilges from kamal;
Revoke & Grant succeeded;

3) Revoke:

* Revoke all privileges from kamal
Revoke succeed

TRANSACTION control LANGUAGES

1. commit:

* commit;
commit complete

2. save point

* save pt k1;
save point created

3. ROll back:

* Rollback
Roll back complete

Result:- Implemt of DDL, DML, DCL and TCL commands are executed suitable examples

VEL TECH

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

Task 2:
Generating design of other bradibron ay database
21/7/25

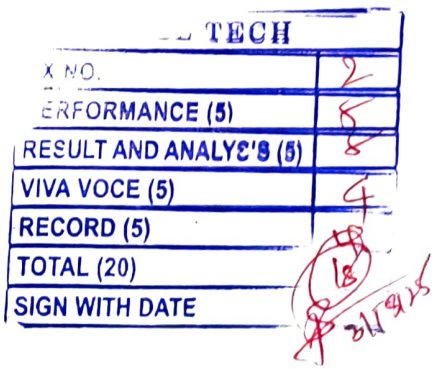
kanaka task 2 dbms.txt

```
SQL*Plus: Release 11.2.0.2.0 Production on Thu Aug 21 14:16:05 2025

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

SQL> connect
Enter user-name: system
Enter password:
ERROR:
ORA-28002: the password will expire within 7 days

Connected.
SQL> create table kanaka babu(name varchar(10),vtu_no number(5),address
varchar(15));
create table kanaka babu(name varchar(10),vtu_no number(5),address varchar(15))
                    *
ERROR at line 1:
ORA-00922: missing or invalid option

SQL> create table kanaka(name varchar(10),vtu_no number(5),address varchar(15));
create table kanaka(name varchar(10),vtu_no number(5),address varchar(15))
             *
ERROR at line 1:
ORA-00955: name is already used by an existing object

SQL> create table david(name varchar(10),vtu_no number(5),address varchar(15));

Table created.

SQL> create table dress(sl_no number(2),brand_name varchar(25),cost number(20));

Table created.

SQL> desc david;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                               VARCHAR2(10)
 VTU_NO                                             NUMBER(5)
 ADDRESS                                            VARCHAR2(15)

SQL> desc clothes;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SL_NO                                              NUMBER(2)
 BRAND_NAME                                         VARCHAR2(25)
 COST                                               NUMBER(20)

SQL> alter table david add phonenumber number (10);

Table altered.

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

```
---------------------------------------- -------- ----------------------------
                                                  VARCHAR2(10)
U_NO                                              NUMBER(5)
ADDRESS                                           VARCHAR2(15)
PHONENUMBER                                       NUMBER(10)

SQL> insrt into david values('lavanya',29144,'bhimavaram',9014018446);
1 row created.

SQL> select* from student;
no rows selected
SQL>  select* from david;

NAME           VTU_NO ADDRESS         PHONENUMBER
---------- ---------- --------------- -----------
lavanya         29144 bhimavaram       9014018446

SQL> insert into clothes values(1,'shirt',999);

1 row created.

SQL> select* from dress;

no rows selected

SQL> insert into dress values(1,'shirt',999);

1 row created.

SQL> select* from dress;

     SL_NO BRAND_NAME                    COST
---------- ------------------------- ----------
         1 shirt                            999

SQL> insert into david values('satya',29143,'tanuku',56341278);

1 row created.

SQL> select* from david;

NAME           VTU_NO ADDRESS         PHONENUMBER
---------- ---------- --------------- -----------
lavanya         29144 bhimavaram       9014018446
satya           29143 tanuku             56341278

SQL> update david set raju
  2
SQL> update david set name='blessy' where name='lavanya';

1 row updated.
```

```
select* from david;

NAME       VTU_NO ADDRESS         PHONENUMBER
---------- ---------- -------------- -----------
blessy      29144 bhimavaram       9014018446
satya       29143 tanuku             56341278

SQL> select distinct name,phonenumber number from david;
select distinct name,phonenumber number from david
                                    *

ERROR at line 1:
ORA-00923: FROM keyword not found where expected


SQL> select distinct name,phonenumber number from david;
select distinct name,phonenumber number from david
                                    *

ERROR at line 1:
ORA-00923: FROM keyword not found where expected


SQL> select distinct name,phonenumber from david;

NAME       PHONENUMBER
---------- -----------
blessy      9014018446
satya         56341278

SQL> insert into david values('ssp',30234,'chirala',1234567890);

1 row created.

SQL> select* from david;

NAME       VTU_NO ADDRESS         PHONENUMBER
---------- ---------- -------------- -----------
blessy      29144 bhimavaram       9014018446
satya       29143 tanuku             56341278
ssp         30234 chirala          1234567890

SQL> select* from david where vtu_no between 29000 and 30000;

NAME       VTU_NO ADDRESS         PHONENUMBER
---------- ---------- -------------- -----------
blessy      29144 bhimavaram       9014018446
satya       29143 tanuku             56341278

SQL> select name,phonenumber from david;

NAME       PHONENUMBER
---------- -----------
blessy      9014018446
```

```
          56341278
        1234567890

SQL> select address as addresses from david;

ADDRESSES
--------------
bhimavaram
tanuku
chirala

SQL> select vtu_no from david where name like'_f%';

no rows selected

SQL> select name from david where vtu_no=29144 and addresses='bhimavaram' and
name like'_f%';
select name from david where vtu_no=29144 and addresses='bhimavaram' and name
like'_f%'
                                                *

ERROR at line 1:
ORA-00904: "ADDRESSES": invalid identifier


SQL>  select name from david where vtu_no=29144 and phonenumber=9014018446 and
name like'_f%';

no rows selected

SQL> create user chinna identified by chinna;

User created.

SQL> commit;

Commit complete.

SQL> savepoint k1;

Savepoint created.

SQL> roll back to k1;
Rollback complete.
SQL>
```



om david

Vtu-no	Address	phonenumber
29144	bhimavaram	9014018446
29143	tanku	56341278
30234	chirala	1234567840

1) select * from david where Vtu-no b/w 29000 and 30500;

name	VTU-NO
blessy	29144
satya	29143

2) select address as addresses from david;

Addresses
bhimavaram
tanku
chirala

3) create user chinna identified by chinna;
User created.

» commit;
Commit complete.

» save point k1;
save point created

» roll back to k1;
roll back complete.

VEL TECH	
EX NO.	2
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
TOTAL (20)	18
WITH DATE	02/5/24

Task-3

Developing Queries with Oracle Single Row functions and operations

Aim:-

To perform query processing on database in different retrieval results of queries using DML, DRL single row operations using aggregate date, string, indent functions, set clauses and operations.

Procedure:-

Aggregative operations:-

In addition to simply retrieving data, we often want to perform some computation SQL allows use of arithmetic expressions we now consider a powerful class of constructs computing aggre values such as min and sum.

1. count:-

count following by column name returns count of tuple in column. If Distinct keyword it used then it will return only count of unique tuple in column. otherwise it will return count of all tuples (including duplicates) count(*) indicates all tuples of column.

Syntax:- count (column name)

Example: select count(*) from retail_store_employees

2. sum:- sum followed by a column name returns sum of all values in column.

Syntax:- sum (column name)

Example: select sum(salary from retail_store_employees

3. Avg:- Avg followed by column name returns avg value of column values.

Syntax:- AVG(n1,n2---)

Example: select AVG(salary) from retail_store_employees

4. min:- min followed by column name returns minimum value of column.

Syntax:- min (column name)

Example:- Selects MIN(salary) from emp;

SQL> select min(salary) retail_store_employee_name min(salary) from retail_store_employees grp by retail_store_employee_name having min (salary)>1000)

SQL string functions:-

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

1. UPPER()

Query: SELECT upper (retail_store_employee_name) from retail_store_employees where retail_store_employee_id=1;

2. LOWER()

Query: SELECT LENGTH (retail_store_employee_name) from retail_store_employee where retail_store_employee_id=1;

3. Length()

Query: SELECT Length (retail_store_employee_name) from retail_store_employees where retail_store_employee_id=1;

4. SUBSTR()

Query: SELECT Length (retail_store_employee_name, 1,5 retail_store_employees where retail_store_employee_id=1;

SQL Date and Time Functions:

The Date & time functions are built in functions in SQL these functions can be used in SQL queries to perform various date and time operations, such as filtering records based on dates, calculating date difference and formatting dates display purposes.

For storing a date and value in database my SQL offers following data types

Date	Format YYYY-mm-DD
Datetime	Format: yyyy-mm-DD-HH:MM:SS
Timestamp	Format: YYYY-MM-DD HH:MM:SS

CURDATE():

Query:- select CURRENDATE from dual;

CURTIME()

Query:- select CURRENT_TIMES() from dual;

ADDDATE(DATE, DAYS)

SQL> select ADDTIMES (2018-08-01 23:59 AM:HH

DAY of MONTH (date)

SQL> select Dayofmonth (2018-02-15);

DAY of WEEK(date)

SQL> select Day of week ('2018-02-15')

MONTH (date)

SQL> SELECT MONTH (2018-08-1001);

TIME(expr)

SQL > select Time ('2018-08-01 11:33:25');

Sysdate:

SQL> select sysdate from DUAL;

NEXT day:-

SQL> select NEXT_DAY(SYSDATE,'wed') from dual;

Last-day:-

SQL> select LAST_DAY(SYSDATE) from DUAL;

months_btw:-

SQL> select MONTHS_BETWEEN(SYSDATE, HIREDATE) from EMP;

add_months:-

SQL> SELECT ADD_MONTHS(SYSDATE,2) from dual;

Least:-

SQL> SELECT LEAST('10-JAN-07','12-OCT-07') from dual;

TRUNK:-

SQL> select TRUNCATE(DATE,'DAY') from dual;

Round:-

SQL> select Round(SYSDATE,'day') from dual;

TO_Char:

SQL> select TO_Char(sysdate,"dd/mm/yy") from dual;

TO_date:-

SQL> select TO_DATE(sysdate,"dd/mm/yy") from dual;

STRING FUNCTIONS:-

CONCAT:- concat returns char1 concatenated with char2. Both char1 and char2 can be only of datatypes

SQL> SELECT CONCAT('ORACLE','ORPORATION') from dual;

ORACL= 'orporation.

Lpad:- LPAD returns expr1 left padded to length n characters with sequence of characters in expr2.

SQL > select LPAD('ORACLE',15,'*') from DUAL;

Rpad:-

RPAD returns expr1, right padded to length characters with expr2, replicated as many times as necessary

SQL> SELECT RPAD('ORACLE',15,'*') from dual;

LTrim:-

Returns a character expression after removing leading

SQL> SELECT LTRIM('SSMS THSS','S') from DUAL;

Lower:-

Returns a character expression after converting uppercase character data to lower case.

SQL> select UPPER('dbms') from DUAL

Single - Row Operations:-

1. IS NULL:-

Query:- select * from retail_store_emp where salary is NULL;

2. IS not NULL

Query:- select * from retail_store_employees where salary is not NULL;

3. LIKE

Query:- select * from retail_store_employees where retail_store_employee_name LIKE '%John%';

4. NOT LIKE:-

Query:- select * from retail_store_employees where retail_store_employee_name NOT LIKE '%John%';

5. Between:-

Query:- select * from retail_store_employee where salary BW 50000 AND 100000

Result:- Thus Developed with DML single row functions and operators been successfully.

Task 3 :- Developing queries with Dml Row functions and operations

TASK 3 OP

```
SQLselect * from retail_store_employees;

RETAIL_STORE_EMPLOYEE_ID  STORE_ID RETAIL_STORE_EMP    SALARY
DEPARTMENT
------------------------ --------- -------------- ---------- ---------------
STORE_PHONENO EMPLOYEE_PHONENO
------------ ----------

            30111      2001 kanna          50000 cashier
 9123456789     9876501111

            30222      2001 blessy     40000 supervisor
 9123456789     9876502222

            30333      2001 lavanya        60000 hr
 9123456789     9876503333

            30444      2001 ssp      35000 salesman
 9123456789     9876504444


SQLselect count(*) from retail_store_employees;

 COUNT(*)
----------
         4

SQLselect sum(salary) from retail_store_employees;

SUM(SALARY)
-----------
     185000

SQLselect avg(salary) from retail_store_employees;

AVG(SALARY)
-----------
     46250

SQLselect max(salary) from retail_store_employees;

MAX(SALARY)
-----------
     60000

SQLselect min(salary) from retail_store_employees;

MIN(SALARY)
-----------
     35000


SQLselect upper(retail_store_employee_name) from retail_store_employees where
retail_store_employee_id=30333;

UPPER(RETAIL_ST)
----------------
```

lavanya

SQLselect lower(retail_store_employee_name) from retail_store_employees where retail_store_employee_id=30333;

LOWER(RETAIL_ST)
---

lavanya
SQLselect length(retail_store_employee_name) from retail_store_employees where retail_store_employee_id=30333;

LENGTH(RETAIL_STORE_EMPLOYEE_NAME)

7

SQLselect substr(retail_store_employee_name,1,4) from retail_store_employees where retail_store_employee_id=30111;

SUBSTR(RETAIL_ST)
---
anir

SQLselect concat('retail_store_employee_name','salary') from retail_store_employees where retail_store_employee_id=30444;

CONCAT('RETAIL_STORE_EMPLOYEE_NA
---
retail_store_employee_namesalary

SQLselect sysdate from dual;

SYSDATE

27-AUG-25

SQLselect next_day(sysdate,'wed') from dual;

NEXT_DAY(

03-SEP-25

SQLselect add_months(sysdate,2) from dual;

ADD_MONTH

27-OCT-25

SQLselect last_day(sysdate) from dual;

LAST_DAY(

31-AUG-25

SQLselect least('10-jan-07','12-oct-07') from dual;

LEAST('10

10-jan-07

SQLselect greatest('10-jan-07','12-oct-07') from dual;

GREATEST(

12-oct-07

SQLselect trunc(sysdate,'day') from dual;

TRUNC(SYS

24-AUG-25

SQLselect round(sysdate,'day') from dual;

ROUND(SYS

31-AUG-25

SQLselect to_char(sysdate,'dd\mm\yy') from dual;

TO_CHAR(

27\08\25

SQLselect to_date(sysdate,'dd\mm\yy') from dual;

TO_DATE(S

27-AUG-25

SQLselect concat('oracle','corpuration') from dual;

CONCAT('ORACLE','CORPURATION')
----
oraclecorpuration

SQLselect lpad('oracle',15,'*') from dual;

LPAD('ORACLE',15,'*')

*********oracle

SQLselect rpad('oracle',15,'*') from dual;

RPAD('ORACLE',15,'*')
----
oracle*********

SQLselect ltrim('ssmithss','s') from dual;

LTRIM(
-----

mithss

```
SQLselect lower('dbms') from dual;

LOWE
----
dbms

SQLselect upper('dbms') from dual;

UPPE
----
DBMS

SQLselect rtrim('ssmithss','s') from dual;

RTRIM(
------
ssmith

SQLselect length('database') from dual;

LENGTH('DATABASE')
------------------
                 8

SQLselect substr('abcdefghij',3,4) from dual;

SUBS
----
cdef

SQLselect instr('corporate floor','or',3,2) from dual;

INSTR('CORPORATEFLOOR','OR',3,2)
--------------------------------
                              14


SQLselect * from retail_store_employees where salary is null;

no rows selected


SQLselect * from retail_store_employees where salary is not null;

RETAIL_STORE_EMPLOYEE_ID   STORE_ID RETAIL_STORE_EMP    SALARY
DEPARTMENT
------------------------ --------- --------------- ---------- ---------------
STORE_PHONENO EMPLOYEE_PHONENO
------------- ---------------
            30111      2001 pradeep        50000 cashier
 9123456789      9876501111

            30222      2001 nivas        40000 supervisor
 9123456789      9876502222
```

Result :- Done

30333 2001 kishore 60000 hr
9123456789 9876503333

30444 2001 Satish 35000 salesman
9123456789 9876504444

SQLselect * from retail_store_employees where retail_store_employee_name like '%anirudh%';

RETAIL_STORE_EMPLOYEE_ID STORE_ID RETAIL_STORE_EMP SALARY DEPARTMENT
---------- ---------- ---------- ---------- ----------
STORE_PHONENO EMPLOYEE_PHONENO
---------- ----------

30111 2001 pradeep 50000 cashier
9123456789 9876501111

SQLselect * from retail_store_employees where retail_store_employee_name not like '%anirudh%';

RETAIL_STORE_EMPLOYEE_ID STORE_ID RETAIL_STORE_EMP SALARY DEPARTMENT
---------- ---------- ---------- ---------- ----------
STORE_PHONENO EMPLOYEE_PHONENO
---------- ----------

30222 2001 nivas 40000 supervisor
9123456789 9876502222

30333 2001 Kishore 60000 hr
9123456789 9876503333

30444 2001 Satish 35000 salesman
9123456789 9876504444

SQLselect * from retail_store_employees where salary between 30000 and 70000;

RETAIL_STORE_EMPLOYEE_ID STORE_ID RETAIL_STORE_EMP SALARY DEPARTMENT
---------- ---------- ---------- ---------- ----------
STORE_PHONENO EMPLOYEE_PHONENO
---------- ----------

30111 2001 pradeep 50000 cashier
9123456789 9876501111

30222 2001 nivas 40000 supervisor
9123456789 9876502222

30333 2001 kishore 60000 hr
9123456789 9876503333

30444 2001 Satish 35000 salesman
9123456789 9876504444

VEL TECH

EX NO.	3
PERFORMANCE (5)	4
RESULT AND ANALYSIS (5)	4
VIVA VOCE (5)	4
RECORD (5)	4
TOTAL (20)	16

SIGN WITH DATE

Result

Developing queries with Dml single row functions and operations executed successfully

Task 4:

Developing queries on multirow function and operators perform advanced query processing and test its. strikes using designing optimal correlated and nested subqueries finding summary statistics consider schema for:-

Employees (emp-no, emp-name, dept, deptno, salary, age).

orders (empno, order id, price, qty-ord, qty-hand)

item file (item id, item name, qty ord, qty hand, it emrate)

Queries using union, intersect, MINUS

* union:-

The union operator returns all distinct rows selected by 1 or more queries.

SQL> select emp_no from employees.

output:-

SQL> select emp_no from orders:

output:-

SQL> select emp_no from employees union select emp_no orders

output:-

*union all:-

SQL> select emp.no from employee intersect

select emp.no from orders:

SQL:- select emp_no from employee intersect
select emp.no
from orders;

query using group by having clause and order by clause

Group By:- This query is used to grp to all the records into relation together for each and every value of specific key(s) and display them for selected set of fields relation

-> select deptno, count(*) from employees
grp by deptno;

oup by-'Having':-
The having clause was added to SQL because where keyword would not be used with aggregate functions. The having clause must follow the grp.

QL> select deptno, count(*) from employees
p by deptno having Dept no is not null;

der By:-
This query used to display selected of fields from relation an ordered manner base on same field.

Syntax:-

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

SQL> select empno, ename, salary from emp order by salary.

SQL> select empno, emp_name, salary from emp 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 + comm) annual_net_sal from emp_master;

Sub queries:-

SQL> select * from employees.

SQL> insert into employees. select * from employees where emp_id in (select emp_id from employees);

SQL> update employees set salary = salary * 10 where department in (select dept from employees where department = "sales";

delete from employees where department in (select department from employees where department = 'sales');

IN:

Query: select * from employees where department In ('sales', 'marketing');

NOT IN:

Query:- select * from employees where department Not IN ('sales', marketing');

Exists:-

Query: select * from employees where exists (select * from orders where orders emp-no: (link unavailable

not Exists:

Query: select * from employees where Not exists (select * from orders where orders emp_no: (link unavailable)

All:-

Query: select * from employee where salary > All (select salary from employees where department sale's);

Any:-

Query:- select * from employee where salary > any (select salary from employee where

Task 4:

SQL> select*from icecream;

NAME	IDNO	ADDRESS	PH_NO	DEPARTMENT
kanna i	345	plvd	3675890728	mechanical
chinnu	678	bhutan	2509760412	computers
ssp	296	chennai	6700855423	artist

SQL> select*from food;

NAME	IDNO	ADDRESS	PH_NO	DEPARTMENT
satya	276	chennai	3657892970	cse
bhuvi	292	pulivendula	8762596309	csd

296 nandyala 2676489907

NAME	IDNO	ADDRESS	PH_NO
DEPARTMENT			
vkanna		291 anatapuram	9392239773
aiml			
sai		299 vijayawada	9987603340
ece			

SQL> select name from icecream union select name from food;

NAME

---

ssp

sai

bhuvi

satya

4 rows selected.

SQL> select name from icecream intersect select name from food;

elected

> select name from icecream minus select name from food;

NAME

---

chinnu

mani

vardhan

SQL> select idno,count(*) from icecream group by idno;

IDNO	COUNT(*)
678	1
296	1
345	1

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

IDNO	COUNT(*)
678	1
296	1
345	1

SQL> select name,idno,address, from icecream order by department;

select name,idno,address, from icecream order by department

at line 1:

00936: missing expression

SQL> name,idno,address, from icecream order by department;

SP2-0734: unknown command beginning "name,idno,..." - rest of line ignored.

SQL> select name,idno,address from icecream order by department;

NAME	IDNO	ADDRESS
vardhan	296	chennai
chinnu	678	bhutan
mani	345	plvd

SQL> select*from food;

NAME	IDNO	ADDRESS	PH_NO	DEPARTMENT
chitti	276	chennai	3657892970	cse
pradeep	292	pulivendula	8762596309	csd
suri	296	nandyala	2676489907	

ME	IDNO	ADDRESS	PH_NO

DEPARTMENT

---

vishnu 291 anatapuram 9392239773

aiml

sai 299 vijayawada 9987603340

ece

SQL> insert into food select*from food where idno in(select idno from food);

5 rows created.

SQL> select*from food where department IN('cse','csd');

NAME	IDNO	ADDRESS	PH_NO

DEPARTMENT

---

chitti 276 chennai 3657892970

cse

pradeep 292 pulivendula 8762596309

cti 276 chennai 3657892970

csc

NAME IDNO ADDRESS PH_NO

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

DEPARTMENT

---

pradeep 292 pulivendula 8762596309

csd

SQL> select *from food where department NOT IN('cse',csd);

select *from food where department NOT IN('cse',csd)

*

ERROR at line 1:

ORA-00904: "CSD": invalid identifier

SQL> select *from food where department NOT IN('cse','csd');

NAME IDNO ADDRESS PH_NO

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

DEPARTMENT

---

suri 296 nandyala 2676489907

shnu 291 anatapuram 9392239773

aiml

sai 299 vijayawada 9987603340

ece

NAME IDNO ADDRESS PH_NO

DEPARTMENT

---

suri 296 nandyala 2676489907

eee

vishnu 291 anatapuram 9392239773

aiml

sai 299 vijayawada 9987603340

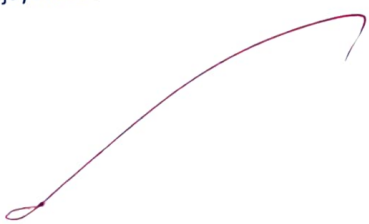
ece

6 rows selected.

SQL> SELECT*FROM food where ph_no>ALL(SELECT ph_no FROM food WHERE department='cse');

	IDNO	ADDRESS	PH_NO
EPARTMENT			
mohan		292 pulivendula	8762596309
csd			
moham		292 pulivendula	8762596309
csd			
vishnu		291 anatapuram	9392239773
aiml			

NAME	IDNO	ADDRESS	PH_NO
DEPARTMENT			
vishnu		291 anatapuram	9392239773
aiml			
sai		299 vijayawada	9987603340
ece			
sai		299 vijayawada	9987603340
ece			



292 pulivendula 8762596309

6 rows selected.

SQL> select

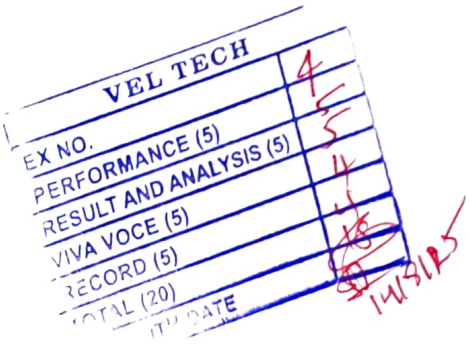
from order master where order

select order_no from orders where
r_no='0001');

t * from order_master where order_no
(select order_no from orders);

SQL> select * from order_master where order
no = any (select order_no from order
detail);

SQL> select * from order_detail where
qty = all (select qty from item
file where itemcode = 250);



Result: Thus Developing queries with executed successfully

Tasks :-

Writing join queries, Equivalent to and/or recursive queries

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

- A SQL join combine records from two tables
- A join Locate related column values in two table.
- A query contain zero, one, or multiple join operations.

Objective :-

To implement different types of joins and recursive queries.

Theory :-

The SQL joins clause is used to combine records from two or more table into database. A join is means for combining fields from two tables by using values common to each.

Syntax :-

select column1, column2, column3 --- from table name1, table-name2 where Table-name1, column name = table

types of joins

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

simple join:-

It is most common type of join. It retrieves rows from 2 tables having common column and feather classified into equi join.

A join, which is based on equalities, is called equi-join.

Example:-

select * from item, user where item-id : user-id;

In above statement, item-id = user id performance the join statement it retrieves rows from both tables provided they both have same id as specified by the where clause since where clause uses the comparsion operator(=) to perform join, it is said to be equision. It combines the matched rows of tables. It can be used as follows:

- To insert records in target table.
- To create tables and insert records in ta
- To create views.

Example:-

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

table Aliases

It was used to make multiple table queries shorted and more readable. we give alies name to table in 'from' clause and use instead of name thraghout query

self join:-

joining of table to itself is know as self join. It joins are row table to another

It can compare each row of table to itself and also with other rows of same table

outer join:-

It entered the result of sample join. an outer join returns all rows returned by simple join as well as those rows from table that do not match any row from table. symbol represents outer join.

Inner join :- Return records that have matching values in both tables

select column names(s) from table
Inner join table2 on table1 column-name
=table2. column_name;

left counter join:- Returns all records from left table, and matched records from right table

~~outer~~ join:-
~~select~~ column. name
Right (outer join : Return all records from right table, and matched records from left table.

select column name(s) from table1
Right join Table2 on table1 . column-name = table2. column_name;

Full Louter join:- Return all records when there is match in either left or right table select column-name(s)
from Table1.

Full ater join Table2 on table1.column
-name=tb2 column-name.

Tip: full OUTER join and full join

on the same select member.name,
borrowed membno from member.
full join borrowed on borrowed membno
no = member.membno.

Full (outer) join:-

Return all records when there
is a match in either left or right
table select column_name(s) from
table1.
full ater join table 2 on table1.column
-name = table2.column_name.

LEFT join

Right join

Inner join

full ater join

Tasks: Write Join Queries, Equivalent to/or Recursive queries.

```
SQL*Plus: Release 11.2.0.2.0  kanna Vtu29144

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

SQL> connect
name: system-Enter user
:Enter password
.Connected

varchar(26),department  SQL>  create  table  employee5(employee_id  number(25),employee_name
;((varchar(5),department_no number(7),salary number(9

.Table created

<SQL
SQL> desc employee5
Name                          Null?   Type
----------------------------- ------- --------------------------------------
(EMPLOYEE_ID                          NUMBER(25
(EMPLOYEE_NAME                         VARCHAR2(26
(DEPARTMENT                           VARCHAR2(5
(DEPARTMENT_NO                         NUMBER(7
(SALARY                              NUMBER(9

SQL>  create  table  orders5(employee_no  number(5),order_id  number(6),price  number(5),qty_order
;((varchar(10),qty_hand char(9

.Table created

SQL> desc orders5
Null?   Type                          Name
----------------------------- ------- --------------------------------------
(EMPLOYEE_NO                           NUMBER(5
(ORDER_ID                            NUMBER(6
(PRICE                               NUMBER(5
(QTY_ORDER                            VARCHAR2(10
(QTY_HAND                            CHAR(9

qty_order  varchar(10),qty_hand ,(SQL>  create  table  itemfile9(item_id  number(6),item_name  varchar(15
;((char(9),itemrate number(6

.Table created

SQL> desc itemfile9
Name                          Null?   Type
----------------------------- ------- --------------------------------------
(NUMBER(6                            ITEM_ID
(ITEM_NAME                            VARCHAR2(15
(QTY_ORDER                            VARCHAR2(10
(QTY_HAND                            CHAR(9
(NUMBER(6                            ITEMRATE

(SQL> INSERT INTO employee9(employees_no,employee_name,department,department_no,salary,age
```

(VALUES(15,'ranjith','cse',18,5000,23 2
3
;(SQL> INSERT INTO employee5 values('21', 'Ranjith', 'CSE','23', '50003

.reatedrow c 1

;('SQL> INSERT INTO employee5 values('11', 'prudhvi', 'ECE','33', '10000

.row created 1

;('SQL> INSERT INTO employee5 values('19', 'bhanu', 'EEE','29', '7000

.row created 1

;('SQL> INSERT INTO employee5 values('17', 'jagan', 'MECH','39', '9000

.row created 1

;('SQL> INSERT INTO employee5 values('12', 'lenin', 'CIVIL','19', '3000

.row created 1

;SQL> select * from employee5

EMPLOYEE_ID	EMPLOYEE_NAME	DEPAR	DEPARTMENT_NO	SALARY
kanna	CSE	23	5000	21
lavanya	ECE	33	10000	11
satya	EEE	29	7000	19
jagan	MECH	39	9000	17
ssp	CIVIL	19	3000	12

;('SQL> INSERT INTO orders5 values('13','66','200','2','3000

.row created 1

;('values('14','77','500','2','5000 SQL> INSERT INTO orders5

.row created 1

;('SQL> INSERT INTO orders5 values('16','87','700','2','7000

.row created 1

;('SQL> INSERT INTO orders5 values('19','82','1000','5','9000

.row created 1

;('12000','9','00SQL> INSERT INTO orders5 values('53','242','60

.row created 1

```
;SQL> select * from employee5
 EMP_NO EMP_NAME DEPAR DEPT_NO SALARY
--------- --------- ---------- ---- --------------- ----------

lenin                19   3000 12
Ranjith              23    5000 21
bhanu                29    7000 19
prudhvi              33    10000 11
9000    39                jagan 17

.rows selected 5
SQL> update employee5 set salary=salary*10 where department in (select department from
;('employee5 where department='sales
.rows updated 0

romSQL> update employee5 set salary=salary*1 where department in (select department f
;('employee5 where department='CSE
.rows updated 2

;SQL> select * from employee5
 EMP_NO EMP_NAME DEPAR DEPT_NO SALARY
--------- --------- ---------- ---- --------------- ----------
lenin                19   3000 12
Ranjith              23    5000 21
bhanu                29    7000 19
prudhvi              33    10000 11
jagan                39    9000 17


.rows selected  .5
;('employee5 where department in ('CSE','ECE SQL> select * from
 EMP_NO EMP_NAME DEPAR DEPT_NO SALARY
--------- --------- ---------- ---- --------------- ----------
Ranjith              CSE          23          5000 21
10000       33         prudhvi              ECE 11

;('SQL> select * from employees where department not in ('CIVIL','MECH
 EMP_NO EMP_NAME DEPAR DEPT_NO SALARY
--------- --------- ---------- ---- --------------- ----------
000jagan               MECH        39          9 17
lenin                CIVIL          19          3000 12
.rows selected 2

;(SQL> select * from employee5 where exists (select * from orders5 where orders5.emp_no=22
 EMP_NO EMP_NAME DEPAR DEPT_NO SALARY
--------- --------- ---------- ---- --------------- ----------
Ranjith              CSE          23          5000 21
prudhvi              ECE          33          10000 11
bhanu                EEE          29          7000 19
jagan                MECH         39          9000 17
lenin                CIVIL          19          3000 12
.rows selected 5
```

```
SQL> select * from employee5 where not exists (select * from orders5 where
;(orders5.emp_no=23
rows selected no

SQL> select * from employee5 where not exists (select * from orders where
;(orders5.emp_no=19
no rows selected

SQL> select * from employee5 where not exists (select * from orders where
;(orders.emp_no=21
no rows selected

;(employee5 where exists (select * from orders5 where orders5.emp_no=39 SQL> select * from

 EMP_NO EMP_NAME DEPAR DEPT_NO SALARY
--------- --------- --------- ----- --------------- ----------
Ranjith              CSE         23          5000 21
rudhvi               ECE        33          10000p 11
bhanu                EEE         29          7000 19
jagan                MECH       39          9000 17
3000          19          lenin                CIVIL 12
.rows selected 5
SQL> select * from employee5 where salary > all (select salary from employee5 where
;('department='EEE
 EMP_NO EMP_NAME DEPAR DEPT_NO SALARY
bhanu                EEE         29          7000   19

;('QL> select * from orders5 where order_id=(select order_id from orders5 where order_id='53S
EMP_NO ORDER_ID PRICE QTY_ORD QTY_HAND
--------- --------- --------- --------- ----------
12000        9 6000      242       53

;(where order_id=any(select order_id from orders5 SQL> select * from orders5
EMP_NO ORDER_ID PRICE QTY_ORD QTY_HAND
--------- --------- --------- --------- ----------
3000        2 200       66        13
5000        2 500       77        14
7000        2 700       87        16
9000        5 1000      82        19
12000        9 6000      242       53

;(SQL> select * from orders5 where order_id in(select order_id from orders5

 EMP_NO ORDER_ID PRICE QTY_ORD QTY_HAND
 --------- --------- --------- --------- ----------
 4 5 500 789 21
 5 6 400 456 22
 3 4 600 123 23
 7 8 200 159 24
 2 3 900 357 25

 SQL> select salary+price from employee5,orders5
 ; 2
```

EMPLOYEE_NO

13
14
16
19
53

;SQL> select EMPLOYEE_ID from employee5 union select EMPLOYEE_NO from orders5

EMPLOYEE_ID

11
12
13
14
16
17
19
21
53

.rows selected 9

;SQL> select EMPLOYEE_ID from employee5 union all select EMPLOYEE_NO from orders5

EMPLOYEE_ID

21
11
19
17
12
13
14
16
19
53

.rows selected 10

;SQL> select EMPLOYEE_ID from employee5 intersect select EMPLOYEE_NO from orders5

EMPLOYEE_ID

19
MP_NO. EMP_NAME DEPAR DEPT_NO SALARY

prudhvi raj. ECE. 33. 10000 11

;(into employee5 select * from employee5 where emp_no in (select emp_no from employee5 SQL> insert
.rows created 5

;SQL> select * from orders5

EMPLOYEE_NO	ORDER_ID	PRICE	QTY_ORDER	QTY_HAND
3000	2	200	66	13
5000	2	500	77	14
7000	2	700	87	16
9000	5	1000	82	19
12000	9	6000	242	53

;('4000','3','600','SQL> INSERT INTO itemfile9 values('66','cement

.row created 1

;('SQL> INSERT INTO itemfile9 values('36','milk','800','4','6000

.row created 1

;('SQL> INSERT INTO itemfile9 values('17','pen','100','2','2000

.row created 1

;('values('47','chocalate','700','9','5000 SQL> INSERT INTO itemfile9

.row created 1

;('SQL> INSERT INTO itemfile9 values('44','drink','400','8','3000

.row created 1

;SQL> select * from itemfile9

ITEM_ID	ITEM_NAME	QTY_ORDER	QTY_HAND	ITEMRATE
cement	600	3		4000 66
milk	800	4		6000 36
pen	100	2		2000 17
5000	chocalate	700	9	47
drink	400	8		3000 44

;SQL> select EMPLOYEE_ID from employee5

EMPLOYEE_ID
21
11
19
17
12

;SQL> select EMPLOYEE_NO from orders5

```
SALARY+PRICE
----------
5200
10200
7200
9200
3200
SALARY+PRICE
----------
5500
10500
7500
9500
3500
SALARY+PRICE
----------
5700
10700
7700
9700
3700
SALARY+PRICE
----------
6000
1100
8000
10000
4000

SALARY+PRICE
11000
16000
13000
15000
90000
```

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	4
VIVA VOCE (5)	4
RECORD (5)	4
TOTAL (20)	19
WITH DATE	21/8/25

Result: Join Queries, Equivalent AND / OR Recursive Queries executed successfully

Task 6: procedure, functions, Loops

Aim: To implement pl/sql procedure functions and loops on number theory and business scenarios

procedure:-

PL/SQL is a combination of SQL along procedural features programming Language. It was developed oracel. corporation in early 90's to enhance capabilities of SQL. pl/sql is one of 3 key programming languages embedded in oracle database.

S.no	Sections Description:
1.	Declarations This sections with keyword declare. It is optinal section defines all variable, cures subprograms.
2.	Executable commands: This section is enclosed b/w keywords BEGIN and END and it is mandatory section. It consists of executable pl/sql statements of program.
3.	Exception handling: This section ets with keyword EXCEPTION. This optimal section contains exceptions handle errors in program.

simple program to print sentence!

Syntax:-

```
DECLARE
    <declarations section>
BEGIN
    <executable command(s)>
EXCEPTION
    <exception handling>
END;
```

program:-

```
DECLARE
    message varchar(20):booking closed
BEGIN
    dbms.output.put_line(message);
END;
```

Static input:-

```
SQL> set serve output on
SQL> declare
  2   x num(5);
  3   y
  4   z
  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

Dynamic Inputs:-

```
SQL> declare
  2  var1 degree;
  3  var2 degree;
  4  var3 degree;
  5  begin
  6  var1 := &var1;
  7  -- 2 := 1 + 2;
  8  -- 3 := var1 + var 2;
  9  dbms_output.put_line (var3);
 10  end;
 11  /
Enter value for 1: 20
old   6: var1 := &var1;
new   6: var1 := 20;
Enter value for var2: 30
```

SO

PL/SQL procedure successfully completed

```
DECLARE
    hid num(3) := 100;
BEGIN
    IF (hid = 10) THEN
        dbms_output.put_line ('value of hid is 10');
    ELS IF (hid = 20) THEN
        dbms.output.put_line ('value of hid 20');
    ELSE
        dbms_output.put_Line('none of value
is matching');
```

Sample program for only procedure:-

```
SQL> create or replace procedure
  2  cc_id in num, c_name in varchar)
  3  is
  4  begin
  5  dbms_output.put_line('ID :'||c_id);
  6  dbms .. .. .. .. 'name :'||c_name);
  7  end;
  8  /
Procedure created
SQL> exec cinformation(101,'ram');
PL/SQL procedure successfully completed
SQL> set serve output on;
SQL> exec cinformation(101,'ram');
ID:101
Name: ram
PL/SQL procedure successfully
completed.
```

Task 6: To Implement PL/SQL procedure functions and loops on num theory and bussiness scenarios

```
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  /
```

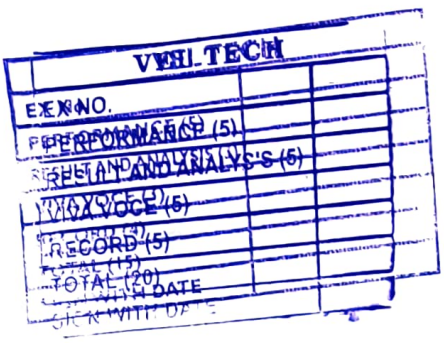
```
Enter value for var1: 20
old   6: var1:=&var1;
new   6: var1:=20;
Enter value for var2: 30
old   7: var2:=&var2;
new   7: var2:=30;
50

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.
```

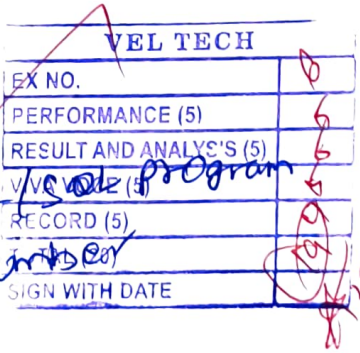
VEL TECH	
EX NO.	6
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	5
VIVA VOCE (5)	4
RECORD (5)	4
TOTAL (30)	18
WITH	

Result: Thus implementation PL/SQL procedure Function and Loops on numbers Theory and bussiness Scenario has executed successfully

program for only function:

```
create or replace function
(in num, c_name in varchar2)
return varchar2
is
begin
If c_id >200 then
Return ('no booking available');
else
Return ('booking open');
End if;
End;
```





Result:
TO implement PL/SQL program procedure function and loop on number theory and business

Task 7:- PL/SQL procedure for Loops:-

Aim:- To write PL/SQL programs using Loops for printing prime number customer IDs and for demonstrating loop control in different scenarios.

procedure:-

* Set a PL/SQL block or procedure
* use a cursor to fetch customer ID's from table.
* for each ID, check whether it is a prime num using a loop
* use for loop/while loop to demonstrate prime number checking.
* Print result using DBMS output PUT_LINE
* End the block

ex1: using while loop with cursor prime check using while loop.

```
create or replace procedure print_prime
.customer IS cursor cust_cur IS
    select customer_id from customers

v_id numbers;
v_is_prime Boolean;
v_i numbers;
BEGIN
    OPEN cust_cur;
    LOOP
    FETCH cust_cur INTO v_id;
    EXIT WHEN cust_cur%NOT
                    Found;

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;
```

```
END IF;
v_i := v_i+1;
END LOOP;
END IF;

IF v_is_prime THEN
DBMS_OUTPUT.PUT_LINE('prime
customer ID: ' || v_id);
END IF;
close cust_cur;
END;
```

This procedure checks all customer IDs in table and print the prime ones using while loop

Ex:- Using for loop for 1st n prime numbers

```
create or replace procedure
print_first
_n_primes (n number) IS
    v_num number:=2;
    v_count number:=0;
    v_is_prime, BOOLEAN;
BEGIN
```

4/9/25

Task 7:

— PL/SQL pulldown/ function loops simple program to print sentence

```
SQL*Plus: Release 11.2.0.2.0 Production on Wed Oct 15 21:34:44 2025

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

SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> CREATE OR REPLACE PROCEDURE print_prime_customers IS
  2  CURSOR cust_cur IS
  3   SELECT customer_id FROM customers;
  4
  5  v_id       NUMBER;
  6  v_is_prime  BOOLEAN;
  7  v_i        NUMBER;
  8  BEGIN
  9  OPEN cust_cur;
 10  LOOP
 11   FETCH cust_cur INTO v_id;
 12   EXIT WHEN cust_cur%NOTFOUND;
 13
 14   -- Prime check
 15   IF v_id< 2 THEN
 16    v_is_prime := FALSE;
 17   ELSE
```

..... prime num and demon by loop concept

```
    v_is_prime := TRUE;
    v_i := 2;

    WHILE v_i<= FLOOR(SQRT(v_id)) LOOP
     IF MOD(v_id, v_i) = 0 THEN
      v_is_prime := FALSE;
      EXIT;
     END IF;
     v_i := v_i + 1;
    END LOOP;
   END IF;

   IF v_is_prime THEN
    DBMS_OUTPUT.PUT_LINE('Prime customer ID: ' || v_id);
   END IF;

  END LOOP;
  CLOSE cust_cur;
END;
/
```

Warning: Procedure created with compilation errors.

```
SQL> CREATE OR REPLACE PROCEDURE print_first_n_primes(n NUMBER) IS
  2  v_numNUMBER := 2;
  3  v_countNUMBER := 0;
```

... prime num and demon by loop errors

```
    _prime  BOOLEAN;
  N
WHILE v_count< n LOOP
  v_is_prime := TRUE;
8
9   -- Prime check using FOR loop
10   FOR i IN 2 .. TRUNC(SQRT(v_num)) LOOP
11    IF MOD(v_num, i) = 0 THEN
12     v_is_prime := FALSE;
13     EXIT;
14    END IF;
15   END LOOP;
16
17   IF v_is_prime THEN
18    DBMS_OUTPUT.PUT_LINE('Prime: ' || v_num);
19    v_count := v_count + 1;
20   END IF;
21
22   v_num := v_num + 1;
23  END LOOP;
24 END;
25 /
```

Procedure created.

... prime num and demon ...

```
declare
onumber(3);
hinumber(3);
nnumber(2);
mnumber(2);
cnumber(20);
begin
dbms_output.put_line('enter the customer id from to limit:');
lo:=&lo;
hi:=&hi;
for n in lo.. hi
loop
c:=0;
for m in 1..n
loop
if mod(n,m)=0 then
c:=c+1;
end if;
endloop;
if c<=2 then
dbms_output.put_line(n || '\n');
end if;
endloop;
end;
```

... prime (NUM) and demon ...

```
ue for lo: 101
lo:=&lo;
9: lo:=101;
er value for hi: 120
ld 10: hi:=&hi;
new 10: hi:=120;

PL/SQL procedure successfully completed.
```

```
SQL> declare
bknumber(5);
snumber:=0;
r number;
len number;
m number;
begin
bk:=&bk;
m:=bk;
len:=length(to_char(bk));
while bk>0
loop
r:=mod (bk,10);
s:=s+power(r,len);
```

prime num and demon

```
unc(bk/10);
loop;
f
m=s
then
dbms_output.put_line('given number is armstrong');
else
dbms_output.put_line('given number is not an armstrong');
end if;
end;
/
Enter value for bk: 234
old   8: bk:=&bk;
new   8: bk:=234;

PL/SQL procedure successfully completed.

SQL>
```

EX NO.	[illegible]
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	4/9/25

v_count <n LOOP

- prime:= true;

IN 2 TRUNC (SQRT (v_num)) LOOP

MOD (v_num, i) = 0 THEN

- 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;

This procedure prints first N prime num using for loop

for Example

BEGIN

print_fr... A ... (10);

END;

VEL TECH	
EX NO.	8
PERFORMANCE (5)	
RESULT AND ANALYSIS (5)	
VIVA VOCE (5)	
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	

Result:

Thus write a PL/SQL program using LOOP for printing prime num and demon by LOOP control

Task8:- normalizing database using function dependencies upon BCNF

upon relational tables created task-2 perform normalization to BCNF based on dependencies as following assumed relations specified below employee database.

1. identify employee attributes: Employee ID, name. department JOB_title.
2. Define relational schema employee (Employee_ID, Name, department, JOB_title, manager_ID)
3. Determine functional dependecies b/w attributes Emplote_ID -> name, department, JOB_title, manger_ID

Step2: convert to 1NF

1. Eliminate repeating grps or arrays
2. create separate tables for each repeating grp

Step3: convert to 2NF

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

Step4: convert to 3NF

1. Ensure there no transitive dependecies
2. move non-key attributes to separate tables if they depend on another non key attributes.

14/9/25

TASK 8:

FUNCTIONAL DEPENDENCY :

Attributes in Table

Separate attributes using a comma (,)

employee_id, name, department, job_title, manager_id, hire_date, salary

Functional Dependencies

employee_id × → name × department × job_title × hire_date × manager_id × salary × **Delete**

Add Another 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 candiates keys are { employee_id}, The set of key attributes are: { employee_id }
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: employee_id –> name,department,job_title,hire_date,manager_id,salary

3NF

find all cadnidate keys. The candiates keys are { employee_id}, The set of key attributes are: { employee_id }
for each FD, check whether the LHS is superkey or the RHS are all key attributes
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.

and executed successfully

NVERT 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

Initially 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

NVERT BCNF :

Normalize to BCNF

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

Table already in BCNF, return itself.

VEL TECH	
EX NO.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	

4/11/25

and executed successfully

; to BCNF

ery determination candidate key
verlapping candidate keys
pose relations to eliminate
dancy.

g Griffith TOOL:-

nput relational schema & functional
2. Griffith tool generates dependency graph
3. Analyze graph to identify normalization
4. Apply normalization rules to transform schema

Griffith TOOL:-

1. Create a new project in Griffith
2. Define relational schema and FDs
3. Run 'Dependency Graph' tool.
4. Analyze graph for normalization issues.
5. Apply transformations using "normalize" tool

VEL TECH	
EX NO.	9
PERFORMANCE (5)	5
RESULT AND ANALYS'S (5)	5
VIVA VOCE (5)	5
RECORD (5)	20
TOTAL (20)	

Result:-

Thus Normaliz database using functional dependence upto BCNF was run and executed sucessfully

Task 9: Backing up and recovery database

perform following backup and recovery

a) Recovering a NOARCHIVELOG Database with incremental Backups

b) Restoring server parameter file

c) performing Recovery with Backup control file

Scenario 1:-

Step 1:-

Backup Database [database_name] to DISK

="backup_file.bak with nofORMAT, Name

="Full Database Backup', SKIP, Rewind

Step 2:- create incremental Backup

Backup Database [database_name] to DISK=

'incremental_backup.bak' with differential,

noformat, noinit.

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] from

DISK="incremental_backup.bak' with replace.

Step 6:- Recover Database

Recover Database [database_name]

Scenario2: Restoring server parameter file

Step1: Backup .spfile

Backup sever 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

Shutdown

Startup

Scenario3:- performing Recovery with Backup control file.

Step1:- Backup control file

Backup controlfile to file='control file bak';

Step2:- Simulate control file loss

. delete or modify control file

Step3:- Restore control file

Startup mount

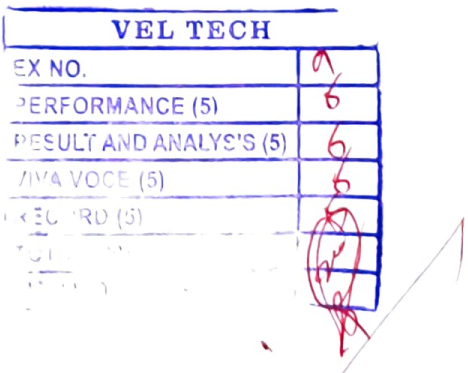
Restore controlfile from file='control bak';

Steps:- Recover Database

Recover Database using Backup controlfile.

SQL Server commands:

- Backup Database
- Restore Database
- Recover Database
- Alter Database
- Backup controlfile
- Restore controlfile



Result:-

Thus implement of Backing up and recovery data base was run executed successfully.

Task 10:- Crud operations in documents database

Aim:-
To perform mongoose using npm design mongo DB designing & document database and performing

Steps:-
Step1:- install mongo db using following link
https://www.mongodb.com/try/download

Step2:- install mongosh using below link

Step3) To add mongoDB shell binary; Locatio to your path environment variable:
Open control panel
In system and security category, click Sys
Click advanced System setting - The system properties modal display Click environmen

Step4) Open mongo shell to from

Step5 Type Crud (Create Read Update delete) commands file

CRUD operations:-

```
db.create collection ("mylab")
{"ok":1}
>db.mylab.insertone({item:"canvas",qty:100,tags:
["cotton"],size{h:28,w:35.5,uom:"cm"}}
}
{
"acknowledged": true
"insertedid":objectid("627d13acc7399007e6397e")
}
>db.mylab.find({item:"canvas"})
{"_id":objectid("62713ac c7399007 4e63),"item
["cotton"],"size":{"h":28,"w":35-5"uom":"cm"}}
>
db.mylab.insertmany([{item"journal",qty25.tags
["black","red"],size:{h:14.w:21.uom:c
{
"acknowledge":true
insertedids
        objectid("627d1598c7399c074 e6397d")
        objectid("627d1598c7399c0774e6397d")
        objectid("627d1598c7399c074e6397d")

}
}
_id: objectid("627d13acc7399c074e6397c"),
"item:"canvas"
```

25-9-25

Task 10: crud Operation in document database

```
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()
```

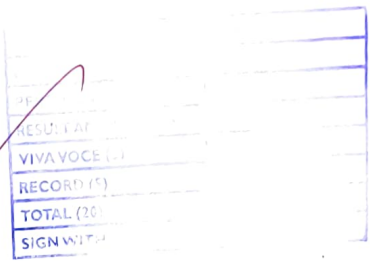
output:

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" : {
                "0" : ObjectId("68efcd6e4b9c34878362b38f"),
                "1" : ObjectId("68efcd6e4b9c34878362b390"),
                "2" : 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}
```



ect ("627d1598c73940c07e6397d");

"journal";

mylab delete one({item: "journal"})

mylab find ({}, {item: 1, qty: 1, _id: 0})

id": objectId("627d13acc73990c73990c074e6347c");

"item": "journal",

"qty": 25

}

{"_id": ObjId("627d1598c73990c074e6347e"), item

{

"_id": ObjId("627d1598c73990c074e6347f"), "item

VEL TECH	
EX NO.	10
ERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
ECORD (5)	5
OTAL (20)	20

Result:-

The implementation CRUD operations like Creating, inserting, removing operation using mongoDB is successfully executed.

Task 1:- CRUD operations in graph database.

Aim:-
To perform CRUD operations like creating, inserting, querying, finding, deleting, graph spaces

- Create Node with properties:-

properties are key value pairs using which a node stores data. you can create node with properties.

Syntax:-

Following is syntax to create node with properties.

Create (node: label {key1: value; key 2: value ---})

- Returning created node

To verify creation of node, type and execute following query in dollar prompt

match (n) Return n

- Creating Relationships

we can create relationship using create clause. we specify relationship within square

Syntax:-

Following syntax to create relationship using create clause create (node1)-[Relationship type]->(node2)

creating a relationship b/w existing nodes:-

you can also create relationship b/w existing nodes using match clause

syntax:-

following is syntax to create relationship using match clause.

- deleting a particular node:-

To delete particular node. you need to specify details of node in place "n" in above query

syntax:-

following in syntax to delete particular node from neo4j using delete clause.

```
match (node.label {properties.....})
detach delete node
```

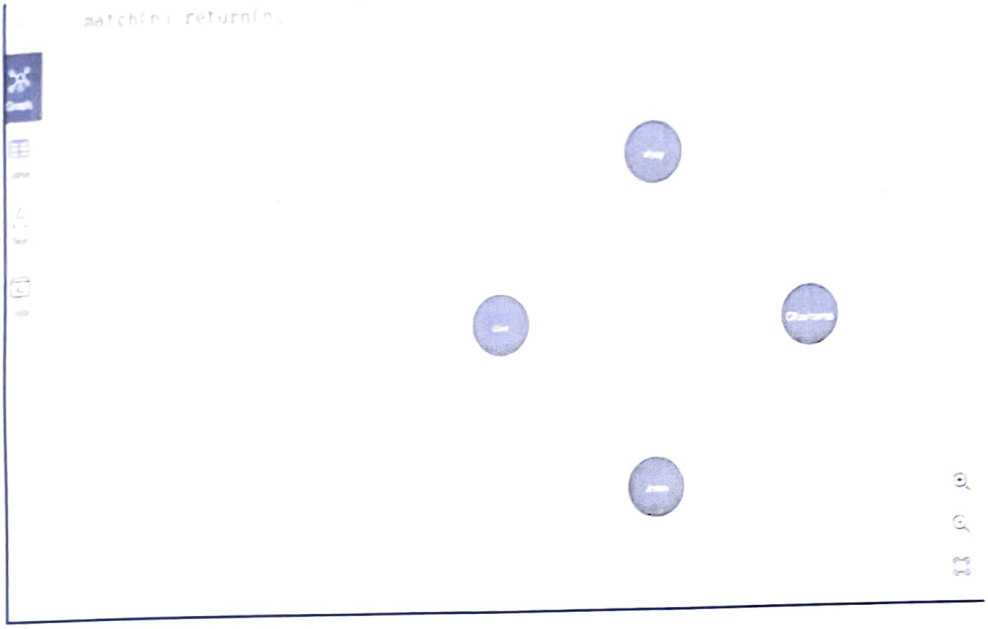
- create (n: dept {dept name "cse", dept id: "dept"})
- select all nodes in your db using command

```
match (n) return (n)
match (n:student) return (n)
```

9/10/25
Task 4:

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
```

Executed successfully

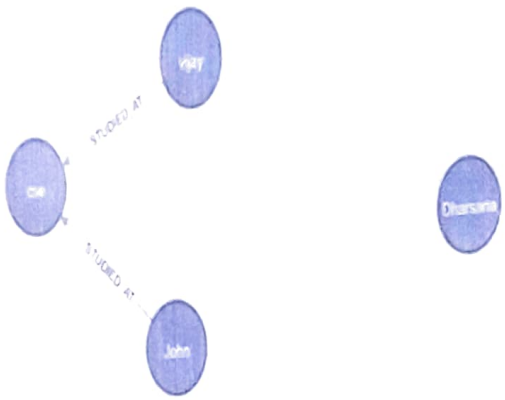
MATCH(s:student),(d:dept) WHERE s.Sname ='John' AND d.deptname='cse'

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

return s,d

n) return(n)

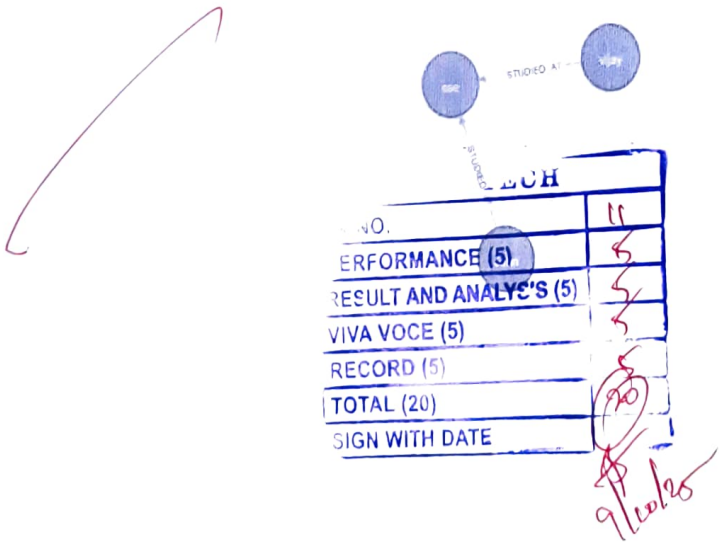
match(n) return(n)



b) Delete a node from student

match(n:student{Sname:'Dharsana'}) Delete(n)

ma ch(n)return(n)



: std), (d: dept) where
e = 'John' and d.depet name = 'c

n, d

arch(n) return(n)

delete a node from std

arch (n: std {sname: char sem: y)
delete (n)

VEL TECH	
EX NO.	11
PERFORMANCE (5)	4
RESULT AND ANALYSIS (5)	4
VIVA VOCE (5)	4
RECORD (5)	4
(20)	16

30/10/25

Result :-

The implementation CRUD operation creating inserting, finding and removing operations using graph DB successfully

DATABASE MANAGEMENT SYSTEMS
(10211DS207)

TASK:12

IRTC Bus booking system

Team details :

Team leader: G. Jagan Mohan Reddy

Reg No:24UEDC0024

Team members:

Names:	VTU NO:	REG NO:
p.Abhinay	vtu28956	24UEDC0050
B. kanaka babu	vtu29144	24UEDC0011
k. jagadeeswara Reddy	vtu29544	24UEDC0030
T.Jahnavi	vtu29550	24UEDC0063

AIM:

To develop a microproject on IRTC Bus booking system.

1.ER Diagram

An Entity -Relationship (ER) model for a IRTC Bus booking system would involve identifying and defining entities, their attributes, and the relationships between them. Here is a simplified example:

Entities:

1.Passengers

-attributes: passenger_id(pk), nameI, Email, phone_no, Gender.

2.booking:

-attributes: booking date, seatnumber, fare, booking_id, payment_status, seat_no.

3.bus

-attributes: bus_id(pk), bus number, source, destination, departuretime

4.route:

-attributes: routeid(pk), licence_no, contact_no, experience.

5. drive

-attributes: name, driver_id(pk).

6.driver:

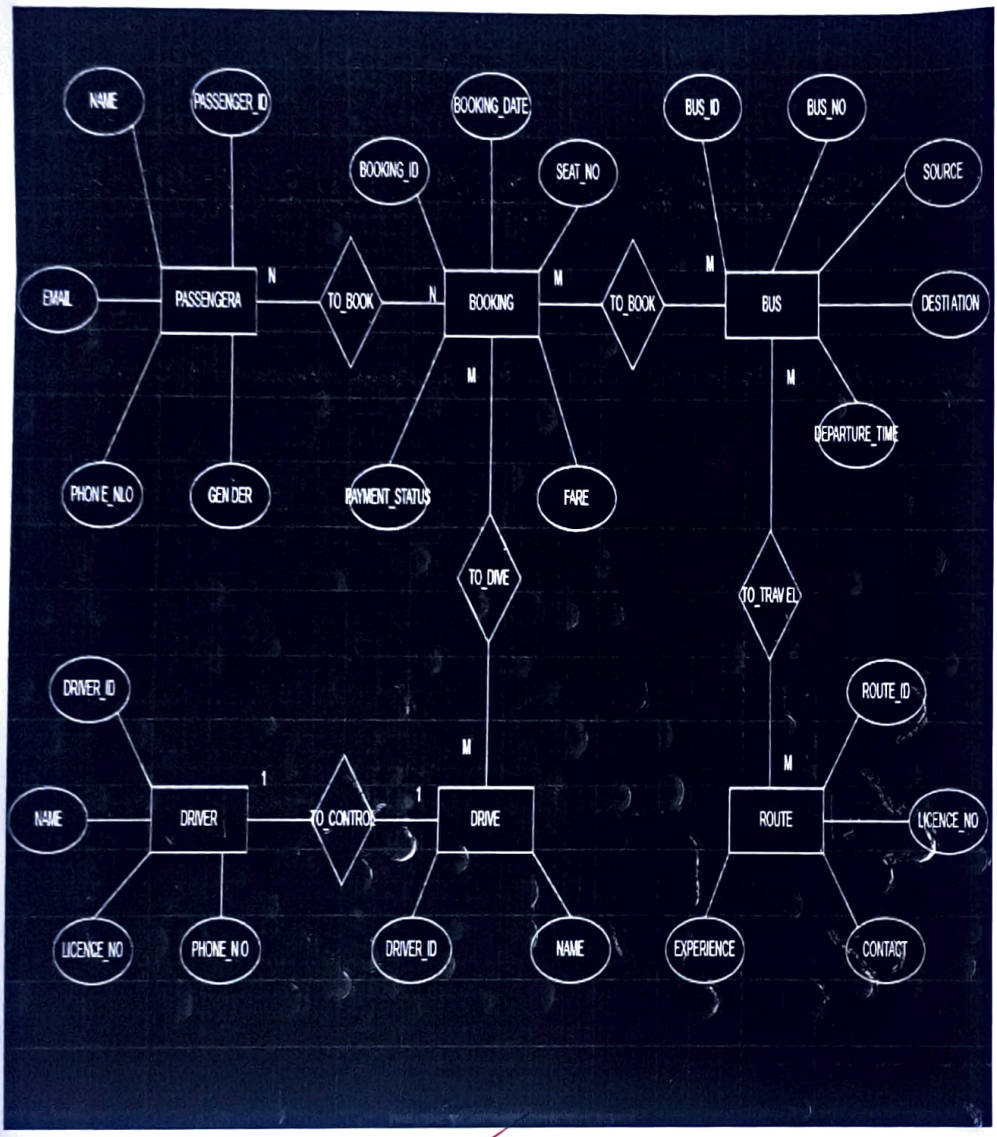
-attributes: contact_no, licence_no(pk), name, driver_id

Relationships:

2. Relationships:

PASSENGER-BOOKING	N-TO-N
BOOKING-BUS	N-TO-M
BOOKING-DTOVE	M-TO-M
BUS-ROUTE	M-TO-M
DRIVE-DRIVER	1-TO-1

- Each box represents an entity with its attributes inside.
- PK stands for Primary Key,which uniquely identifies each record in the table.
- FK stands for Foreign Key,which establishes a relationship between tables.
- The line connecting the entities represent the relationships between them.
- The notation "1" and "N" indicates the cardinality of the relationships.


NAME
PASSENGER_ID
BOOKING_DATE
BUS_ID
BUS_NO
BOOKING_ID
SEAT_NO
SOURCE
EMAIL
PASSENGERA
N
TO_BOOK
N
BOOKING
M
TO_BOOK
M
BUS
DESTIATION
M
M
DEPARTURE_TIME
PHONE_NO
GENDER
PAYMENT_STATUS
FARE
TO_DIVE
TO_TRAVEL
DRIVER_ID
ROUTE_ID
M
M
1
1
NAME
DRIVER
TO_CONTROL
DRIVE
ROUTE
LICENCE_NO
LICENCE_NO
PHONE_NO
DRIVER_ID
NAME
EXPERIENCE
CONTACT

2.SQL Queries &relational operations :

1.select operation

```
SQL> select* from booking;
```

Output:

```
PASSENGER_ID NAME           AGE   PHONE_NO
------------ ------- ---------- ----------
           1 babu            19         99
           2 abhi            20         98
           3 jagades         18         97
           3 jagan           21         97
           3 jahnavi         21         96
```

2.Project operation:

```
SQL> select name from booking;
```

Output:

```
PASSENGER_ID NAME           AGE   PHONE_NO
------------ ------- ---------- ----------
           1 babu            19         99
           2 abhi            20         98
           3 jagades         18         97
           3 jagan           21         97
           3 jahnavi         21         96
```

3.UNION ALL:

```
SQL> select name from booking union all select startpoint from bus;
```

Output:

```
NAME
--------
babu
abhi
jagades
jagan
jahnavi
atp
bangolore
vijaywada

8 rows selected.
```

4.intersect:

```
SQL> select name from booking intersect select startpoint from bus;
```

output:

```
no rows selected
```

5.Sum :

```
SQL> select sum(age) as totalage from booking;
```

OUTPUT:

```
  TOTALAGE
----------
        99
```

6.Count :

```
SQL> select count(*) as passengercount from booking;
```

OUTPUT:

```
PASSENGERCOUNT
--------------
             5
```

7.AVG :

```
SQL> select avg(age) as averageage from booking;
```

OUTPUT:

```
AVERAGEAGE
----------
      19.8
```

8.Minimum :

```
SQL> select min(age) as minimumage from booking;
```

Output:

```
MINIMUMAGE
----------
        18
```

9.Maximum :

```
SQL> select max(age) as maximumage from booking;
```

Output:

```
MAXIMUMAGE
----------
        21
```

10.Nested Queries :

a.innerjoin

```
SQL> select booking.name, bus.startpoint from booking inner join bus on booking.passenger_id=bus.bus_id;
```

Output:

```
NAME   STARTPOIN
------ ---------
babu   atp
abhi   vijaywada
abhi   bangolore
```

11. left outer join:

```
SQL> select booking.name, bus.startpoint from booking left outer join bus on booking.passenger_id=bus.bus_id;
```

Output:'

```
NAME   STARTPOIN
------ ---------
babu   atp
abhi   bangolore
abhi   vijaywada
jahnavi
jagan
jagades
```

12.right outer join:

```
SQL> select booking.name, bus.startpoint from booking right outer join bus on booking.passenger_id=bus.bus_id;

NAME   STARTPOIN
------ ---------
babu   atp
abhi   vijaywada
abhi   bangolore
```

13.full outerjoin:

```
SQL> select booking.name, bus.startpoint from booking full outer join bus on booking.passenger_id=bus.bus_id;
```

Output:

```
NAME       STARTPOIN
---------- ----------
babu       atp
abhi       vijaywada
abhi       bangolore
jagades
jagan
jahnavi

6 rows selected
```

4.Normalization :

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity.

We normally go through stages called Normal Forms (1NF → 2NF → 3NF → BCNF).

For our Dairy Farming System, we can start from a sample normalized relation and use the Griffith Normalization Tool to automate the process.

1. First Normal Form (1NF) A relation (table) is in 1NF if all its attributes contain atomic (indivisible) values, and each row is unique. Key point: No repeating groups or arrays.

2. Second Normal Form (2NF) A relation is in 2NF if it is in 1NF and every non-prime attribute is fully functionally dependent on the whole primary key.

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

3. 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 normalization tool.

First Normal Form :

Attributes in Table

Separate attributes using a comma (,)

passenger_id,name,email,phone_no,gender

Functional Dependencies

passenger_id × → name × email × gender × phone_no × 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 candiates keys are { passenger_id}, The set of key attributes are: { passenger_id }
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: passenger_id –> name,email,gender,phone_no

3NF

find all cadnidate keys. The candiates keys are { passenger_id}, The set of key attributes are: { passenger_id }
for each FD, check whether the LHS is superkey or the RHS are all key attributes
checking functional dependency passenger_id –> name,email,gender,phone_no

BCNF

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

Normalize to 2NF :

Normalize to 2NF

Attributes

passenger_id | name | email | phone_no | gender

Functional Dependencies

passenger_id → name email gender phone_no

Show Steps

First, find the minimal cover of the FDs, which includes the FDs
passenger_id –> name
passenger_id –> email
passenger_id –> gender
passenger_id –> phone_no

Initially rel[1] is the original table:

Round1: checking table rel[1]

***** The table is in 2NF already, send it to output *****

Normalize to 3NF :

1NF to 3NF

Attributes

passenger_id | name | email | phone_no | gender

Functional Dependencies

passenger_id → name
passenger_id → email
passenger_id → gender
passenger_id → phone_no

Show Steps

Table already in 3NF

Normalize to BCNF :

Normalize to BCNF

Attributes

passenger_id | name | email | phone_no | gender

Functional Dependencies

passenger_id → name email gender phone_no

Show 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.createCollection("irtc")

Output:

{ "ok" : 1 }

INSERT ONE:

db.irtc.insertOne({item:"name",qty:24,tags:["character"],size:{h:28,w:35.5,uom:"cm"}})

Output:

{

"acknowledged" : true,

"insertedId" : ObjectId("68f7a50c328226b6c747cc88")

}

FIND():

db.irtc.find({item:"name"})

Output:

{ "_id" : ObjectId("68f7a5d0e9c83b3f4509eced"), "item" : "name", "qty" : 24, "tags" : ["character"], "size" : { "h" : 28, "w" : 35.5, "uom" : "cm" } }

INSERT MANY:

db.irtc.insertMany([{item:"gender",qty:6,tags:["black","red"],size:{h:14,w:21,uom:"cm"}},{item:"phone_no",qty:2,tags:["stable"],size:{h:19,w:22,uom:"cm"}},{item:"passenger_id",qty:25,tags:["solid","siver"],size:{h:20,w:23,uom:"cm"}}])

Output:

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("68f7a6ea2e66dff570e9e109"),

ObjectId("68f7a6ea2e66dff570e9e10a"),

ObjectId("68f7a6ea2e66dff570e9e10b")

]

}

FIND() with Projection:

db.irtc.find({},{item:1,qty:1}).pretty()

Output:

{ "_id" : ObjectId("68f7a79b3aa6d2e1fb712d70"), "item" : "name", "qty" : 24 }

{ "_id" : ObjectId("68f7a79b3aa6d2e1fb712d71"), "item" : "gender", "qty" : 6 }

{ "_id" : ObjectId("68f7a79b3aa6d2e1fb712d72"), "item" : "phone_no", "qty" : 2 }

{ "id" : ObjectId("68f7a79b3aa6d2e1fb712d73"), "item" : "passenger_id", "qty" : 25 }

db.irtc.find({},{item:1,qty:1}).pretty()

Output:

{

"_id" : ObjectId("68f7a8a8f7331254e4f3157d"),

"item" : "gender",

"qty" : 6

}

{

"_id" : ObjectId("68f7a8a8f7331254e4f3157e"),

"item" : "phone_no",

"qty" : 2

}

{

"_id" : ObjectId("68f7a8a8f7331254e4f3157f"),

"item" : "passenger_id",

"qty" : 25

}

FIND() with Sort:

db.irtc.find({item:"name"}).pretty().sort({item:-1})

Output:

{

```
    "_id" : ObjectId("68f7a93e2f547bb56c7fa290"),
    "item" : "name",
    "qty" : 24,
    "tags" : [
        "character"
    ],
    "size" : {
        "h" : 28,
        "w" : 35.5,
        "uom" : "cm"
    }
}
```


DELETE:

```
db.irtc.deleteOne({item:"name"})
```

Output:


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


FIND() after DELETE:

```
db.irtc.find({},{item:1,qty:1}).pretty()
```

output:


```
{
    "_id" : ObjectId("68f7aa317353abb7f17ea765"),
    "item" : "gender",
    "qty" : 6
}
{
    "_id" : ObjectId("68f7aa317353abb7f17ea766"),
    "item" : "phone_no",
    "qty" : 2
}
{
```

"_id" : ObjectId("68f7aa317353abb7f17ea767"),

"item" : "passenger_id",

"qty" : 25

6.Graph Database using MONGODB(using neo4j online compiler)

a)Create a graph database

create(p1:Passenger{id:"p001",name:"Jagan"})

Output:

Created 1 node, set 2 properties, added 1 label

CREATE (n1:passengername {id: "n001", paymentstatus: "cash", gender: "Female", age:22})

Output:

Created 1 node, set 4 properties, added 1 label

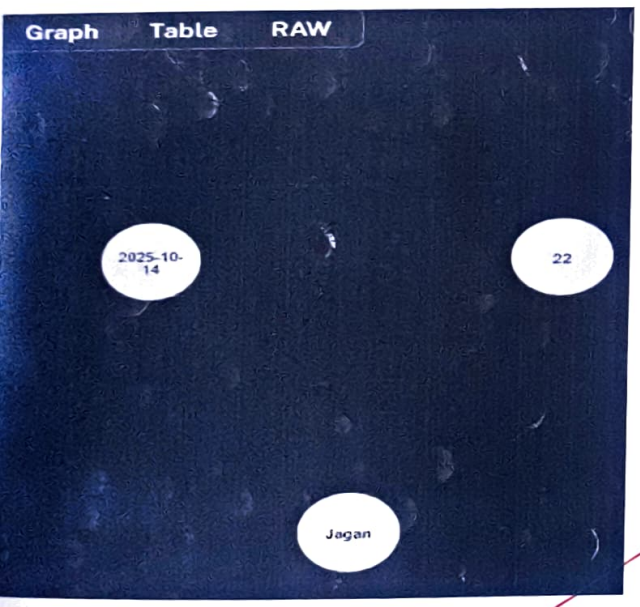
CREATE (b1:bookingstatus {id: "b001", seat_no: 15, date: "2025-10-14"})

Output:

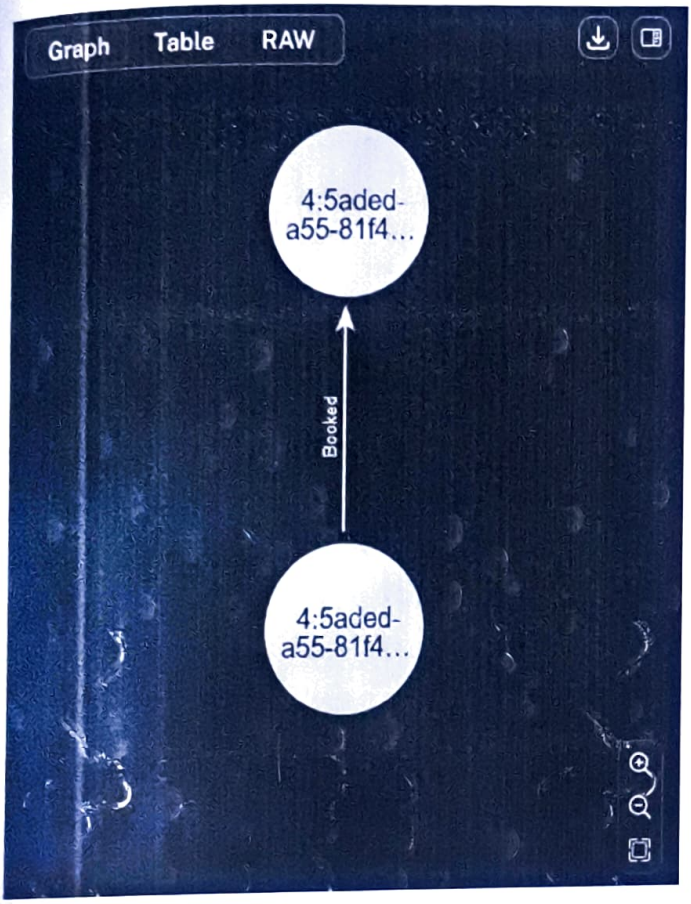
Created 1 node, set 3 properties, added 1 label

match(n) return(n)

Output:



create (p)-[:Booked]->(b)



match (p)-[r:Booked]->(b) DELETE r

Deleted 1 relationship

EX NO.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	4
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	19
SIGN WITH DATE	30/10/24

Result:

Thus, the micro project for IRTC Bus booking system was developed and implemented successfully completed.