DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE
(AUTONOMOUS)
(Approved by AICTE & Affiliated to Anna University, Chennai)
Accredited with 'A' Grade by NAAC, Accredited by TCS
Accredited by NBA with BME, ECE & EEE
PERAMBALUR - 621 212. Tamil Nadu.
website : www.dsengg.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB MANUAL

U23ITP33 - DATABASE MANAGEMENT SYSTEMS LABORATORY

REGUALTION: 2023

Prepared By
Mr.S.Saravana Kumar, AP / IT

Approved By
Dr. V.Thiruppathy Kesavan, IT / HOD

U23ITP33	DATABASE MANAGEMENT SYSTEM LABORATORY	L	T	P	C
		0	0	4	2

COURSE OBJECTIVES

The main learning objective of this course is to:

1. To learn and implement important commands in SQL.
2. To learn the usage of nested and joint queries.
3. To understand functions, procedures and procedural extensions of databases.
4. To understand design and implementation of typical database applications.
5. To be familiar with the use of a front-end tool for GUI based application development.
6. To learn and implement important commands in SQL.

LIST OF EXPERIMENTS

1. Create a database table, add constraints (primary key, unique, check, Not null), insert rows, update and delete rows using SQL DDL and DML commands.
2. Create a set of tables, add foreign key constraints and incorporate referential integrity.
3. Query the database tables using different „where" clause conditions and also implement aggregate functions.
4. Query the database tables and explore sub queries and simple join operations.
5. Query the database tables and explore natural, equi and outer joins.
6. Write user defined functions and stored procedures in SQL.
7. Execute complex transactions and realize DCL and TCL commands.
8. Write SQL Triggers for insert, delete, and update operations in a database table.
9. Create View and index for database tables with a large number of records.
10. Create an XML database and validate it using XML schema.
11. Create Document, column and graph-based data using NOSQL database tools.
12. Develop a simple GUI based database application and incorporate all the above-mentioned features
13. Case Study using any of the real-life database applications from the following list

a) Inventory Management for a EMart Grocery Shop
b) Society Financial Management c) Cop
Friendly App – Eseva
d) Property Management – eMall
e) Star Small and Medium Banking and Finance

- Build Entity Model diagram. The diagram should align with the business and functional goals stated in the application.
- Apply Normalization rules in designing the tables in scope.
- Prepared applicable views, triggers (for auditing purposes), functions for enabling enterprise grade features.
- Build PL SQL / Stored Procedures for Complex Functionalities, ex EOD Batch Processing for calculating the EMI for Gold Loan for each eligible Customer.
- Ability to showcase ACID Properties with sample queries with appropriate settings

TOTAL: 60 PERIODS

EX.NO:1a **DATA DEFINITION COMMANDS**

AIM:

Commands

To implement database creation and do the operations in database using Data Definition

ALGORITHM:
Step 1: Create table by using create table command with column name, data type and size.
Step 2: Display the table by using desc command.
Step 3: Add any new column with the existing table by alter table command.
Step 4: Modify the existing table with modify command.
Step 5: Delete the records in files by truncate command.
Step 6: Delete the Entire table by drop command.

1) CREATE TABLESyntax:
create table table_name(column1 datatype, column2 datatype, column3 datatype,..... columnN datatype);

Example:

create table employee(id int,name varchar(100),age int,address varchar(100),salary float);
Table employee created

2) DESCRIBING THE TABLE

Syntax:
desc table_name;

Example:
desc employee;

Object Type **TABLE** Object **EMPLOYEE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE	ID	NUMBER	22	-	0	-	✓	-	-
	NAME	VARCHAR2	100	-	-	-	✓	-	-
	AGE	NUMBER	22	-	0	-	✓	-	-
	ADDRESS	VARCHAR2	100	-	-	-	✓	-	-
	SALARY	FLOAT	126	126	-	-	✓	-	-

1 - 5

3) ALTER TABLESyntax:
alter table table_name add column_name datatype;

Example:
alter table employee add dept varchar(10);

table employee altered

select * from employee;

ID	NAME	AGE	ADDRESS	SALARY	DEPT
1	ram	20	cbe	5000	-

Example:
alter table employee add (state varchar(100),weight float);
table employee altered.

select * from employee;

ID	NAME	AGE	ADDRESS	SALARY	DEPT	STATE	WEIGHT
1	ram	20	cbe	5000	-	-	-

table employee altered.

4) RENAME
To rename column nameSyntax:
alter table table_name rename column existing _name to new_name;

Example:
alter table employee rename column address to addr;
table employee altered.

select * from employee

ID	NAME	AGE	ADDR	SALARY	DEPT	STATE	WEIGH
1	ram	20	cbe	5000	-	-	-

To rename table nameSyntax:
alter table table_name rename to new_table_name;

Example:
alter table employee rename to empl;
table employee altered.

select * from employee;
ORA-00942: table or view does not exist 00942. 00000 - "table or view does not exist"
*Cause:
*Action:
Error at Line: 21 Column: 15

select * from empl;

ID	NAME	AGE	ADDR	SALARY	DEPT	STATE	WEIGH
1	ram	20	cbe	5000	-	-	-

5) TRUNCATESyntax:
truncate table table_name;

Example:
truncate table emp;select * from empl;

no data found

6) DROP TABLESyntax:
drop table table_name;Example:
drop table empl;
table customers dropped.

select * from empl;
ORA-00942: table or view does not exist

RESULT:

Thus the queries to implement data definition commands are executed successfully.

EX.NO:1b **DATA MANIPULATION COMMANDS**

AIM:

To implement and execute a query for manipulating & storing data items in a Oracledatabase using Structured Query Language commands

ALGORITHM:

Step 1: Create table by using create table command.Step 2: Insert values into the table
Step 3: Delete any records from the tableStep 4: Update any values in the table.
Step 5: Display the values from the table by using select command.

1) CREATE A TABLE

create table student(id int,Name varchar(30),Department varchar(10));desc student;

2) INSERTSyntax:

insert into table_name values(column1,column2,….);

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STUDENT	ID	NUMBER	22	-	0	-	✓	-	-
	NAME	VARCHAR2	30	-	-	-	✓	-	-
	DEPARTMENT	VARCHAR2	10	-	-	-	✓	-	-
									1 - 3

Example:

insert into student values(1,'ABC','CSE'); insert into student values(2,'DEF','CSE'); select * from student;

ID	NAME	DEPARTMENT
1	ABC	CSE
2	DEF	CSE

To insert NULL values to column:

The following syntax is used to insert NULL value to the column.

insert into student values(3,'null','CSE'); select * from student;

ID	NAME	DEPARTMENT
1	ABC	CSE
3	null	CSE
2	DEF	CSE

To insert default value to the column:

The following syntax is used to insert default value to the table.

Syntax:

insert into table_name values(column1,column2,…);

Example:

insert into student values(3,’alex’,default);1 rows inserted.

select * from student;

ID	NAME	DEPARTMENT
1	ABC	CSE
3	null	CSE
2	DEF	CSE
3	alex	-

To insert data using select statement:
The following syntax is used insert all the values from another table using select statement.
Syntax:
insert into table_name1 select * from table_name2;

Example:
create table stud(id int,Name varchar(30),Department varchar(10));insert into stud select * from student;

ID	NAME	DEPARTMENT
1	ABC	CSE
3	null	CSE
2	DEF	CSE
3	alex	-

3) UPDATESyntax:
update table_name set column1 = value1, column2 = value2. .., columnn = valuen
where [condition];
Example: (single column)

update stud set Department='ECE' where id=3;
1 rows updated.

ID	NAME	DEPARTMENT
1	ABC	CSE
3	null	ECE
2	DEF	CSE
3	alex	ECE

Example: (multiple column)

update stud set Name=’XYZ’,Department='CIVIL' where id=2;

ID	NAME	DEPARTMENT
1	ABC	CSE
3	null	ECE
2	XYZ	CIVIL
3	alex	ECE

4) DELETE

Syntax: (delete particular row)
delete from table_name where condition;

Example:
delete from stud where id=2;

ID	NAME	DEPARTMENT
1	ABC	CSE
2	XYZ	CIVIL

Syntax:
delete from table_name;

Example:
delete from stud; select * from stud;

no data found

RESULT:

Thus, a query for manipulating & storing data items in an Oracle database using Structured Query Language commands were implemented and executed successfully.

EX.NO:2 **CREATE SET OF TABLES ADD FOREIGN KEY CONSTRAINT AND INCORPORATE REFERENTIAL INTEGRITY**

Aim:

To create set of tables add foreign key constraint and incorporate referential integrity using SQL.

Algorithm:

- Create two tables with a link that will not work.
- Create two tables with a link that will work.
- Add a foreign key to enforce referential integrity between the two tables.
- Delete data from the tables.
- Update and remove data to show how the foreign key protects the data.
- Use the cascade option of the foreign key.
- Companies table: List of Companies
- Employees table: List of Employees working at the Company
- Rule: Employees can only work at one Company, and a Company can employ multipleEmployees

Program:

Step 1:

```
--Create the HRDatabaseUSE master
GO
DROP DATABASE IF EXISTS HRDatabaseGO
CREATE DATABASE HRDatabaseGO
USE HRDatabaseGO
```

Step 2:

```
-1st Try: Create a Companies and an Employees table and link them togetherDROP TABLE IF EXISTS Companies;
GO
DROP TABLE IF EXISTS Employees;GO
-- SQL CREATE TABLE StatementCREATE TABLE Companies (
ID  INT CONSTRAINT PK_Companies PRIMARY KEY IDENTITY,
CompanyName  VARCHAR(80) NOT NULL, -- column name, data types and null valueCompAddress
VARCHAR(80) NOT NULL,
CompContactNo VARCHAR(20) NOT NULL,EmpID  INT      NOT NULL,
CreateDate                DATETIME NOT NULL constraint DF_Companies_CreateDate DEFAULTgetdate()
) CREATE TABLE Employees (
ID                        INT CONSTRAINT PK_Employees PRIMARY KEY IDENTITY,
EmployeeName  VARCHAR(80) NOT NULL,ContactNo                VARCHAR(20) NOT NULL, Email
                          VARCHAR(80) NOT NULL,CreateDate   DATETIME  NOT NULL constraint
DF_Employees_CreateDate DEFAULTgetdate())
INSERT INTO Companies (CompanyName, CompAddress, CompContactNo, EmpID)VALUES
('Alpha Company', '123 North Street, Garsfontein, Pretoria', '091 523 6987' , 1),
('Bravo Company', '456 South Street, Brooklyn, Pretoria', '091 523 4789' , 1),
('Charlie Company' , '987 West Street, Lynnwood, Pretoria', '091 523 1235' , 1),
('Delta Company', '258 East Street, The Meadows, Pretoria', '091 523 7414' , 1),
('Echo Company', '100 Amber Street, Hatfield, Pretoria', '091 523 9685' , 1) INSERT INTO Employees
(EmployeeName, ContactNo, Email) VALUES ('Joe Blogs' , '012 365 4789', 'joeblogs@gmail.com') ,
('Jane Doe' , '012 365 4789', 'janedoe@gmail.com')  ,
('John Smit' , '012 365 4789', 'johnsmit@gmail.com')  ,
('Eddy Jones' , '012 365 4789', 'eddyjones@gmail.com'),
('Steve Dobson', '012 365 4789', 'stevedobson@gmail.com')


SELECT * FROM CompaniesSELECT * FROM Employees
```

OUTPUT :

	ID	CompanyName	CompAddress	CompContactNo	EmpID	CreateDate
1	1	Alpha Company	123 North Street, Garsfontein, Pretoria	091 523 6987	1	2022-06-12 09:00:15.390
2	2	Bravo Company	456 South Street, Brooklyn, Pretoria	091 523 4789	1	2022-06-12 09:00:15.390
3	3	Charlie Company	987 West Street, Lynnwood, Pretoria	091 523 1235	1	2022-06-12 09:00:15.390
4	4	Delta Company	258 East Street, The Meadows, Pretoria	091 523 7414	1	2022-06-12 09:00:15.390
5	5	Echo Company	100 Amber Street, Hatfield, Pretoria	091 523 9685	1	2022-06-12 09:00:15.390

	ID	EmployeeName	ContactNo	Email	CreateDate
1	1	Joe Blogs	012 365 4789	joeblogs@gmail.com	2022-06-12 09:00:15.397
2	2	Jane Doe	012 365 4789	janedoe@gmail.com	2022-06-12 09:00:15.397
3	3	John Smit	012 365 4789	johnsmit@gmail.com	2022-06-12 09:00:15.397
4	4	Eddy Jones	012 365 4789	eddyjones@gmail.com	2022-06-12 09:00:15.397
5	5	Steve Dobson	012 365 4789	stevedobson@gmail.com	2022-06-12 09:00:15.397

RESULT:

Thus, to create a set of tables add foreign key constraint and incorporate referential integrity wereimplemented and executed successfully.

EX.NO:3 DATABASE TABLES WITH DIFFERENT 'WHERE CLAUSE' CONDITIONS

Aim:

To using query the database tables with different 'where clause' conditions also implement aggregate function with algorithm

Algorithm:

Step 1:

There are tables country and city. We'll use the following statements:

```
SELECT *
FROM country;
SELECT *
FROM city;
```

You can see the result in the picture below:

Results Messages

	id	country_name	country_name_eng	country_code
1	1	Deutschland	Germany	DEU
2	2	Srbija	Serbia	SRB
3	3	Hrvatska	Croatia	HRV
4	4	United States of America	United States of America	USA
5	5	Polska	Poland	POL
6	6	España	Spain	ESP
7	7	Rossiya	Russia	RUS

	id	city_name	lat	long	country_id
1	1	Berlin	52.520008	13.404954	1
2	2	Belgrade	44.787197	20.457273	2
3	3	Zagreb	45.815399	15.966568	3
4	4	New York	40.730610	-73.935242	4
5	5	Los Ang...	34.052235	-118.243...	4
6	6	Warsaw	52.237049	21.017532	5

Step 2:

```
SELECT *
FROM country
INNER JOIN city ON city.country_id = country.id;

SELECT COUNT(*) AS number_of_rowsFROM country
INNER JOIN city ON city.country_id = country.id;
```

Results Messages

	id	country_name	country_name_eng	country_code	id	city_name	lat	long	country_id
1	1	Deutschland	Germany	DEU	1	Berlin	52.520008	13.404954	1
2	2	Srbija	Serbia	SRB	2	Belgrade	44.787197	20.457273	2
3	3	Hrvatska	Croatia	HRV	3	Zagreb	45.815399	15.966568	3
4	4	United States of America	United States of America	USA	4	New York	40.730610	-73.935242	4
5	4	United States of America	United States of America	USA	5	Los Angeles	34.052235	-118.243683	4
6	5	Polska	Poland	POL	6	Warsaw	52.237049	21.017532	5

	number_of_rows
1	6

Step 3:

```
SELECT *FROM country
LEFT JOIN city ON city.country_id =  country.id;

SELECT  COUNT(*)  AS  number_of_rowsFROM country
LEFT JOIN city ON city.country_id =  country.id;

SELECT COUNT(country.country_name) AS countries, COUNT(city.city_name) AS citiesFROM country
LEFT JOIN city ON city.country_id =  country.id;
```

Results Messages

	id	country_name	country_name_eng	country_code	id	city_name	lat	long	country_id
1	1	Deutschland	Germany	DEU	1	Berlin	52.520008	13.404954	1
2	2	Srbija	Serbia	SRB	2	Belgrade	44.787197	20.457273	2
3	3	Hrvatska	Croatia	HRV	3	Zagreb	45.815399	15.966568	3
4	4	United States of America	United States of America	USA	4	New York	40.730610	-73.935242	4
5	4	United States of America	United States of America	USA	5	Los Angeles	34.052235	-118.243683	4
6	5	Polska	Poland	POL	6	Warsaw	52.237049	21.017532	5
7	6	España	Spain	ESP	NULL	NULL	NULL	NULL	NULL
8	7	Rossiya	Russia	RUS	NULL	NULL	NULL	NULL	NULL

	number_of_rows
1	8

	number_of_countries	number_of_cities
1	8	6

RESULT:

Thus the SQL query the database tables with different 'where clause' conditions alsoimplement aggregate function are executed successfully.

EX.NO:4 **SIMPLE QUERIES, SUB QUERIES, JOINS**

AIM:

To implement Simple queries, Nested queries, Sub queries using structured query language

ALGORITHM:

STEP 1: Create a table with the columns that are needed
STEP 2: Write simple queries to insert, delete, update and implement some functions
STEP 3: Write nested queries using clauses to retrieve data from the table
STEP 4: The nested queries may have more than one sub queries.
STEP 5: Efficiently retrieve data from the table

QUERIES:

create table salary(id int, Name varchar(20),Age int,Salary float);insert into salary values(1,'Ajay',25,20000);
insert into salary values(2,'Archana',28,30000);insert into salary values(3,'Archana',23,25000);insert into salary values(4,'Sara',24,20000); insert into salary values(5,'Ajay',25,40000); insert into salary values(6,'Srinithi',26,35000);
select * from salary;

ID	NAME	AGE	SALARY
1	Ajay	25	20000
2	Archana	28	30000
3	Archana	23	25000
4	Sara	24	20000
5	Ajay	25	40000
6	Srinithi	26	35000

Sum function

Syntax:

select column_name1,sum(column_name2)from table_name group by column_name3;

Example:

select name, sum(salary) from salary group by name;

NAME	SUM(SALARY)
Ajay	60000
Sara	20000
Srinithi	35000
Archana	55000

Min function

Syntax:

select column_name1,min(column_name2)from table_name group by column_name3;

Example:

select name, min(salary) from salary group by name;

NAME	MIN(SALARY)
Ajay	20000
Sara	20000
Srinithi	35000
Archana	25000

Max function

Syntax:
select column_name1,max(column_name2)from table_name group by column_name3;
Example:
select name, max(salary) from salary group by name;

NAME	MAX(SALARY)
Ajay	40000
Sara	20000
Srinithi	35000
Archana	30000

Count function

Syntax:
select column_name1,count(column_name2)from table_name group by column_name3;
Example:
select name, count(salary) from salary group by name;

NAME	COUNT(SALARY)
Ajay	2
Sara	1
Srinithi	1
Archana	2

Mod function

Syntax:
select column_name1,mod(column_name2)from table_name;
Example:
select name,mod(salary,5) from salary;

NAME	MOD(SALARY,5)
Ajay	0
Archana	0
Archana	0
Sara	0
Ajay	0
Srinithi	0

DISTINCT and ORDER BY Clause

Syntax:
select distinct column1, column2,.....columnn from table_name;

Example:
select salary from salary order by salary;

SALARY
20000
20000
25000
30000
35000
40000

select distinct salary from salary order by salary**;**

SALARY
20000
25000
30000
35000
40000

HAVING clause

Example:

select min(salary)from salary group by age having max(salary)<30000;

MIN(SALARY)
20000
25000

Select using Conditions

AND

Example:
select * from salary where age >= 25 and salary >= 25000;

ID	NAME	AGE	SALARY
2	Archana	28	30000
5	Ajay	25	40000
6	Srinithi	26	35000

OR

Example:
select * from salary where age >= 25 or salary >= 25000;

ID	NAME	AGE	SALARY
1	Ajay	25	20000
2	Archana	28	30000
3	Archana	23	25000
5	Ajay	25	40000
6	Srinithi	26	35000

LIKE

select * from salary where name like 'Aj%';

ID	NAME	AGE	SALARY
1	Ajay	25	20000
5	Ajay	25	40000

IN

select * from salary where age in (25, 28);

ID	NAME	AGE	SALARY
1	Ajay	25	20000
2	Archana	28	30000
5	Ajay	25	40000

BETWEEN

select * from salary where salary between 20000 and 30000;

ID	NAME	AGE	SALARY
1	Ajay	25	20000
2	Archana	28	30000
3	Archana	23	25000
4	Sara	24	20000

IS NOT NULL

select * from salary where age is not null;

ID	NAME	AGE	SALARY
1	Ajay	25	20000
2	Archana	28	30000
3	Archana	23	25000
4	Sara	24	20000
5	Ajay	25	40000
6	Srinithi	26	35000

IS NULL

select * from salary where age is null;
no data found

(4,'Kumaran',25,'CBE',24000);

Sub queries with the INSERT Statement

INSERT INTO customers_backupSELECT * FROM CUSTOMERSWHERE ID IN
(SELECT ID
FROM CUSTOMERS) ;
4 rows inserted

select * from customers_backup;

ID	NAME	AGE	ADDRESS	SALARY
1	Sri	25	CBE	25000
2	Hari	28	CBE	35000
3	Sow	28	CHENNAI	45000
4	Kumaran	25	CBE	24000

Sub queries with the UPDATE Statement

UPDATE CUSTOMERS
SET SALARY = SALARY * 0.25WHERE AGE IN
(SELECT AGE FROM customers_backupWHERE AGE >= 27);
Select * from customers;

ID	NAME	AGE	ADDRESS	SALARY
1	Sri	25	CBE	25000
2	Hari	28	CBE	8750
3	Sow	28	CHENNAI	11250
4	Kumaran	25	CBE	24000

Subqueries with the DELETE Statement

DELETE FROM CUSTOMERSWHERE AGE IN
(SELECT AGE FROM CUSTOMERS_backupWHERE AGE >= 27);

select * from customers;

ID	NAME	AGE	ADDRESS	SALARY
1	Sri	25	CBE	25000
4	Kumaran	25	CBE	24000

select age from customers where exists (select age from salary where salary > 25000);

AGE
25
28
23
24
25
26

JOINS

CREATE and INSERT values in TABLE(cust)

create table cust(id int primary key,name varchar(20),age int); insert into cust values(1,'aaa',32);
insert into cust values(2,'aba',32); insert into cust values(3,'abc',32); insert into cust values(4,'ccc',25); select * from cust;

ID	NAME	AGE
1	aaa	32
2	aba	32
3	abc	32
4	ccc	25

CREATE and INSERT values in TABLE(order1)

create table order1(oid int primary key,custid int,amount int);insert into order1 values(101,1,1000);
insert into order1 values(102,2,2000);insert into order1 values(103,3,3000);

1. INNER JOINSyntax:

OID	CUSTID	AMOUNT
101	1	1000
102	2	2000
103	3	3000

SELECT table1.column1, table2.column2...FROM table1
INNER JOIN table2
ON table1.common_filed = table2.common_field;

Example:

select id,name,oid,amount

from cust
inner join order1
on cust.id=order1.custid;

ID	NAME	OID	AMOUNT
1	aaa	101	1000
2	aba	102	2000
3	abc	103	3000

2. LEFT JOIN:Syntax:

select table1.column1, table2.column2...from table1
left join table2
on table1.common_filed = table2.common_field;

Example:
select id,name,oid,amountfrom cust
left join order1
on cust.id=order1.custid;

ID	NAME	OID	AMOUNT
1	aaa	101	1000
2	aba	102	2000
3	abc	103	3000
4	ccc	-	-

3. RIGHT JOIN:
Syntax:
select table1.column1, table2.column2...from table1
right join table2
on table1.common_filed = table2.common_field;

Example:
select id,name,oid,amountfrom cust
right join order1
on cust.id=order1.custid;

ID	NAME	OID	AMOUNT
1	aaa	101	1000
2	aba	102	2000
3	abc	103	3000

4. FULL JOIN:Syntax:

select table1.column1, table2.column2...from table1
full join table2
on table1.common_filed = table2.common_field;

Example:
select id,name,oid,amountfrom cust
full join order1
on cust.id=order1.custid;

ID	NAME	OID	AMOUNT
1	aaa	101	1000
2	aba	102	2000
3	abc	103	3000
4	ccc	-	-

RESULT:
Thus the SQL commands to execute simple queries, sub queries and joins are executedsuccessfully.

EX.NO:5 **NATURAL, EQUI AND OUTER JOINDATE:**

Aim

To write DQL Commands to join, Restrict and Retrieve information from one or more tables execute it and verify the same.

Algorithm

Selecting Rows with Equijoin using table Aliases

1. Write a query to display empid,name,deptid,deptname andlocation for allemployees and verify it.

SQL> select employee.empid,employee.name,employee.depid,department.deptname,depa rtment.location from employee,department where employee.depid=department.depid;

EMPID	NAME	DEPID	DEPTNAME	LOCATION
a123	ragu	CS000	COMPUTER_SCIENCE	CHENNAI
a124	rama	CS000	COMPUTER_SCIENCE	CHENNAI
a125	ram	EE000	ELECT_ELECTRO	MUMBAI
a128	sag	EE000	ELECT_ELECTRO	MUMBAI
c128	dinesh	EC000	ELECT_COMM	DELHI
d124	abc	EC000	ELECT_COMM	DELHI

6 rows selected.

2. Write a query to display the “dinesh” depid and deptname and verify it.

SQL> select e.empid,e.name,e.depid,d.deptname from employeee,department d wheree.depid=d.depid and e.name='dinesh';

EMPID	NAME	DEPID	DEPTNAME
c128	dinesh	EC000	ELECT_COMM

Selecting Rows with Non-Equijoin using table Aliases; [Other Conditions such as >=, <= and BETWEEN, AND]

1. Write a query to display the name,salary and deptname of allemployees whosesalary is greater than 10000 and verify it.

SQL> select e.name,e.salary,d.deptname from employee e,department d wheree.salary>10000and e.depid=d.depid;

NAME	SALARY	DEPTNAME
ragu	200000	COMPUTER_SCIENCE
rama	200000	COMPUTER_SCIENCE
ram	30000	ELECT_ELECTRO

Selecting Rows using Outer Joins:[Left Outerjoin,Right Outerjoin using ”+”symbol]

1. Write a query to display the name, depid and deptname of all employees.Make surethat employees without department are included aswell and verify it.

SQL> select e.name,e.depid,d.deptname from employee e,department d where e.depid =d.depid(+);

NAME	DEPID	DEPTNAME
rama	CS000	COMPUTER_SCIENCE
ragu	CS000	COMPUTER_SCIENCE
sag	EE000	ELECT_ELECTRO
ram	EE000	ELECT_ELECTRO
abc	EC000	ELECT_COMM
dinesh	EC000	ELECT_COMM
www		

7 rows selected.

2. Write a query to display the name, salary,depid and deptname of all employees. Makesure that departments without employees are included as well and verify.

SQL> select e.name,e.salary,e.depid,d.deptname from employeee,department d where e.depid(+)=d.depid;

NAME	SALARY	DEPID	DEPTNAME
ragu	200000	CS000	COMPUTER_SCIENCE
rama	200000	CS000	COMPUTER_SCIENCE
ram	30000	EE000	ELECT_ELECTRO
sag	10000	EE000	ELECT_ELECTRO
dinesh	10000	EC000	ELECT_COMM
abc	5000	EC000	ELECT_COMM
			MECHANICAL
			CHEMICAL

8 rows selected

RESULT:
Thus the SQL commands to execute natural, equi and outer joins areexecuted successfully.

EX.NO:6 **PROCEDURES AND FUNCTIONS**

AIM:
To implement and execute Procedures in Oracle Database using Procedural Languageconcepts.

PROCEDURES:
Procedure is a sub program used to perform an action.
Replace-recreates the procedure if it already exists.

3 MODES:
IN – Means you must specify a value for the argument at the time execution of the procedure.

OUT-passes back a value to its calling program.

1) INOUT – When calling the procedure, yu must specify the value and that procedures passes value backto the calling procedure

CREATING A PROCEDURESYNTAX:
CREATE [OR REPLACE] PROCEDURE procedure_name[(column_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS} BEGIN
< procedure_body > END procedure_name;

CREATE OR REPLACE PROCEDURE greetingsAS
BEGIN
dbms_output.put_line('Hello World!');END;

To run this procedure

BEGIN
greetings;END;

DELETING A PROCEDURESYNTAX:
DROP PROCEDURE procedure-name;

EXAMPLE:

DROP PROCEDURE greetings;

Procedure dropped

EXAMPLE 2

Create a table user1

create table user1(id number(10) primary key,name varchar2(100));

Create the procedure

create or replace procedure
INSERTUSER(id IN NUMBER, name IN VARCHAR2)
is begin

insert into user1 values(id,name);end;

To execute the procedure

BEGIN INSERTUSER(101,'Rahul');
dbms_output.put_line('record inserted successfully');END;

PL/SQL FUNCTIONS

SYNTAX:

CREATE [OR REPLACE] FUNCTION function_name [parameters][(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}BEGIN
< function_body > END [function_name];

create or replace function adder(n1 in number, n2 in number)return number
is
n3 number(8);begin
n3 :=n1+n2;return n3; end;

EXECUTE THE FUNCTION

DECLARE
n3 number(2);BEGIN
n3 := adder(11,22); dbms_output.put_line('Addition is: ' || n3);
END;

Output:

Addition is: 33 Statement processed.

RESULT:
Thus the PL/SQL procedures and functions are executed successfully

EX.NO.7 TRANSACTION CONTROL AND DATA CONTROL STATEMENTSDATE:

AIM:

To implement Transaction Control statements using structured Query Language

ALGORITHM:

Step 1: Create table by using create table command.Step 2: Insert values into the table
Step 3: Commit the table
Step 4: Insert or Update any values in the table.Step 5: Rollback the table
Step 6: Insert or Update any values in the table.Step 7: Fix a save point
Step 8: Repeat step 6 and 7
Step 9: Rollback to any save point

1) CREATE A TABLE

create table employeepersonaldetails(id int primary key,name varchar(30) not null,age int not null,mobilenumber int not null unique);

insert into employeepersonaldetails values(1,'Ajay',20,9876543210); insert into employeepersonaldetails values(2,'Brindha',20,9874563210); insert into employeepersonaldetails values(3,'Kumaran',20,9673443210);select * from employeepersonaldetails;

ID	NAME	AGE	MOBILENUMBER
1	Ajay	20	9876543210
2	Brindha	20	9874563210
3	Kumaran	20	9673443210

3 rows returned in 0.03 seconds Download

commit;

Statement executed

insert into employeepersonaldetails values(4,'Archana',20,9876543410);

rollback;

insert into employeepersonaldetails values(5,'Braham',20,9878953210);

ID	NAME	AGE	MOBILENUMBER
5	Braham	20	9878953210
1	Ajay	20	9876543210
2	Brindha	20	9874563210
3	Kumaran	20	9673443210

4 rows returned in 0.00 seconds

savepoint a;

savepoint created

insert into employeepersonaldetails values(6,'Srimathi',20,9678953210);

ID	NAME	AGE	MOBILENUMBER
5	Braham	20	9878953210
6	Srimathi	20	9678953210
1	Ajay	20	9876543210
2	Brindha	20	9874563210
3	Kumaran	20	9673443210

savepoint b; rollback to a;

RESULT:
Thus Transaction Control and Data Control statements using structured Query Language isexecuted successfully

EX.NO:8 **TRIGGERS**

AIM:
To implement PL/SQL triggers and do the operations in database automatically.

ALGORITHM:

STEP1: Create a table emply101 STEP2: Insert values into the table STEP3: Create another table emply102STEP4: Create trigger
CREATE OR REPLACE TRIGGER <TRIGGER NAME>
{BEFORE/AFTER/INSTEAD OF}
{INSERT/UPDATE/DELETE}
ON <TABLENAME/VIEWNAME> REFRENCECING {OLD AS OLD /NEW AS NEW}
[FOR EACH STATEMENT /FOR EACH ROW [WHEN <CONDITION>]]
STEP5: Insert and delete record into emply101STEP6: View the updations in emply102

CREATE TABLE emply101
create table emply101(eid varchar(20),ename char(25), age number(10), salary number(10)); insert into emply101 values(1,'abi',23,25000);
insert into emply101 values(2,'abinaya',22,20000); insert into emply101 values(3,'abishek',23,25000); select * from emply101;

EID	ENAME	AGE	SALARY
1	abi	23	25000
2	abinaya	22	20000
3	abishek	23	25000

CREATE TABLE emply102
create table emply102(eid varchar(20),ename char(25),age number(10),salary number(10));
select * from emply102;
no data found
Creating trigger
create or replace trigger tfg1 before insert on emply101 for each rowbegin
insert into emply102 values(:new.eid,:new.ename,:new.age,:new.salary);end;

Insert values into emply101

insert into emply101 values(4,'bala',23,25000);select * from emply101;

EID	ENAME	AGE	SALARY
1	abi	23	25000
2	abinaya	22	20000
3	abishek	23	25000
4	bala	23	25000

select * from emply102;

EID	ENAME	AGE	SALARY
4	bala	23	25000

CREATING TRIGGER FOR UPDATING AND DELETING

```
create or replace trigger trgrr15 after delete or update on emply101 for each rowbegin
if deleting then
insert into emply102 values(:old.eid,:old.ename,:old.age,:old.salary);elsif updating then
insert into emply102 values(:old.eid,:old.ename,:old.age,:old.salary);end if;
end;
```

Update a value in emply101

```
update emply101 set salary=10000 where eid=1;select * from emply102;
```

EID	ENAME	AGE	SALARY
4	bala	23	25000
1	abi	23	25000

RESULT:
Thus the SQL triggers are executed successfully.

EX.NO:9 **VIEWS AND INDEX**

AIM:

To write a DDL command to create views to restrict data access from one or more tables, execute it and verify the same.

Creating views:

1. Create a view that contains employee id as "ID_NUMBER", employee nameas "NAME"and salary as "SALARY" for each employee in department 90.

SQL> create view vw_emp80(id,name,salary) as select empid,name,salary fromemployee wheredeptid=90;

OUTPUT:

View created.
SQL> select * from vw_emp80;

ID	NAME	SALARY
s203	vidhya	29000

2. To create a view to set the salary Rs.15000/- to the employee id 'a102' in the vw_dept_salary and verify the result in vw_emp12_sal and employees table.

SQL> update vw_emp12 set salary=15000 whereempid='a102';1 row updated.
To Verify: SQL> select * from vw_emp12;

SNO	NAME	SALARY	DESIG	LAST	EMPID	DEPTID	HIRE_DATE
1	monica	15000	rep	a	a102	102	13-MAR-08

Creating Index
Create table
CREATE TABLE Colleges (college_id INT PRIMARY KEY,college_codeVARCHAR(20),NOT NULL
college_name VARCHAR(50)
);
create index
CREATE INDEX college_index ON Colleges(college_code);Colleges

college_id	college_code	college_name
empty		

RESULT:
Thus the PL/SQL program for implementing views and index has been successfully executed.

EX.NO:10 **XML SCHEMA**

AIM:

To implement XML Database and validate it using XML Schema.

PROGRAM

```
CREATE TABLE t1 (id NUMBER,
xml XMLTYPE
);

INSERT INTO t1 VALUES (1, '<?xml version="1.0" encoding="UTF-8"?>
<shiporder orderid="889923">
<orderperson>John Smith</orderperson>
<shipto>
<name>Ola Nordmann</name>
<address>Langgt 23</address>
<city>4000 Stavanger</city>
<country>Norway</country>
</shipto>
<item>
<title>Empire Burlesque</title>
<note>Special Edition</note>
<quantity>1</quantity>
<price>10.90</price>
</item>
<item>
<title>Hide your heart</title>
<quantity>1</quantity>
<price>9.90</price>
</item>
</shiporder>');


INSERT INTO t1 VALUES (2, '<?xml version="1.0" encoding="UTF-8"?>
<shiporder orderid="889923">
<orderperson>John Smith</orderperson>
<shipto>
<name1>Ola Nordmann</name1>
<address>Langgt 23</address>
<city>4000 Stavanger</city>
<country>Norway</country>
</shipto>
<item>
<title>Empire Burlesque</title>
<note>Special Edition</note>
<quantity>1</quantity>
<price>10.90</price>
</item>
<item>
<title>Hide your heart</title>
<quantity>1</quantity>
<price>9.90</price>
</item>
```

</shiporder>');COMMIT;

OUTPUT

```
SELECT id,
XMLISVALID(xml, 'my_schema.xsd') AS is_validFROM t1;

ID  IS_VALID
        -------- ------------
1                       1
2                       0
```

RESULT:
Thus the XML Database and validate it using XML Schema is executed successfully.

EX.NO:11 **NOSQL DATABASE TOOLSDATE**

Aim:
To design and develop MongoDB queries using CRUD operations.

Algorithm:

Step 1: Create table by using create table command.
Step 2: Insert values into the table
Step 3: Delete any records from the table
Step 4: Update any values in the table.
Step 5: Display the values from the table by using select command.
Step 6: Search any value using find.

Query:

```
>                db.createCollection('Student');
{ "ok" : 1 }
>                db.Student.insert({'Rno':'1','Name':'Piyush','Class':'TE COMP'});WriteResult({ "nInserted" : 1 })
>                db.Student.insert({'Rno':'2','Name':'Abhi','Class':'TE COMP'});WriteResult({ "nInserted" : 1 })
>                db.Student.insert({'Rno':'3','Name':'Ashley','Class':'TE COMP'});WriteResult({ "nInserted" : 1 })
>                db.Student.insert({'Rno':'4','Name':'Hitesh','Class':'TE COMP'});WriteResult({ "nInserted" : 1 })
>                db.Student.insert({'Rno':'5','Name':'Pratik','Class':'TE COMP'});WriteResult({ "nInserted" : 1 })
>                db.Student.insert({'Rno':'6','Name':'Pratik','Class':'TE COMP'});WriteResult({ "nInserted" : 1 })
>                db.Student.find();
{ "_id" : ObjectId("5b8fad4ef00832a0a50b5036"), "Rno" : "1","Name" : "Piyush", "Class" : "TE COMP" }
{ "_id" : ObjectId("5b8fad62f00832a0a50b5037"), "Rno" : "2","Name" : "Abhi", "Class" : "TE COMP" }
{ "_id" : ObjectId("5b8fad70f00832a0a50b5038"), "Rno" : "3","Name" : "Ashley", "Class" : "TE COMP" }
{ "_id" : ObjectId("5b8fad7ff00832a0a50b5039"), "Rno" : "4","Name" : "Hitesh", "Class" : "TE COMP" }
{ "_id" : ObjectId("5b8fad8df00832a0a50b503a"), "Rno" : "5","Name" : "Pratik", "Class" : "TE COMP" }
{ "_id" : ObjectId("5b8fada4f00832a0a50b503b"), "Rno" : "6","Name" : "Pratik", "Class" : "TE COMP" }
>                db.Student.find().pretty();
{
"_id" : ObjectId("5b8fad4ef00832a0a50b5036"),"Rno" : "1",
"Name" : "Piyush",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad62f00832a0a50b5037"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"}
{
"_id" : ObjectId("5b8fad70f00832a0a50b5038"),
"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad7ff00832a0a50b5039"),"Rno" : "4",
"Name" : "Hitesh",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad8df00832a0a50b503a"),"Rno" : "5",
"Name" : "Pratik",
```

```
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fada4f00832a0a50b503b"),"Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}
>                 show dbs; Abhi 0.078GB
admin (empty)local 0.078GB
>                 db.Student.update({'Name':'Hitesh'},{$set:
{'Name':'Henry'}});WriteResult({ "nMatched" : 1, "nUpserted" : 0,"nModified" : 1
})
>                 db.Student.find().pretty();
{
"_id" : ObjectId("5b8fad4ef00832a0a50b5036"),"Rno" : "1",
"Name" : "Piyush",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad62f00832a0a50b5037"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad70f00832a0a50b5038"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad7ff00832a0a50b5039"),"Rno" : "4",
"Name" : "Henry",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad8df00832a0a50b503a"),
"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"}
{
"_id" : ObjectId("5b8fada4f00832a0a50b503b"),"Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}
>                 db.Student.remove({'ADD':'MP'}); WriteResult({ "nRemoved" : 1 })
>                 db.Student.find().pretty();
{
"_id" : ObjectId("5b8fad62f00832a0a50b5037"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad70f00832a0a50b5038"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}
```

```
{
"_id" : ObjectId("5b8fad7ff00832a0a50b5039"),"Rno" : "4",
"Name" : "Henry",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad8df00832a0a50b503a"),"Rno" : "5",
"Name" : "Pratik","Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fada4f00832a0a50b503b"),"Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}
>               db.Student.remove({'Name':'Pratik'},1); WriteResult({ "nRemoved" : 1 })
>               db.Student.remove({'Name':'Pratik'},1); WriteResult({ "nRemoved" : 1 })
>               db.Student.find().pretty();
{
"_id" : ObjectId("5b8fad62f00832a0a50b5037"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad70f00832a0a50b5038"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}
{
"_id" : ObjectId("5b8fad7ff00832a0a50b5039"),"Rno" : "4",
"Name" : "Henry",
"Class" : "TE COMP"
}


>               db.Student.drop();true
>               db.Student.find().pretty()
{ "ok" : 1 }
>               db.Student.insert({'Rno':'1','Name':'Piyush','Class':'TE COMP'}); WriteResult({ "nInserted" : 1 })

>               db.Student.insert({'Rno':'2','Name':'Abhi','Class':'TE COMP'}); WriteResult({ "nInserted" : 1 })

>db.Student.insert({'Rno':'3','Name':'Ashley','Class':'TE COMP'}); WriteResult({ "nInserted" : 1 })

>               db.Student.insert({'Rno':'4','Name':'Hitesh','Class':'TE COMP'}); WriteResult({ "nInserted" : 1 })

>               db.Student.insert({'Rno':'5','Name':'Pratik','Class':'TE COMP'}); WriteResult({ "nInserted" : 1 })

>               db.Student.insert({'Rno':'6','Name':'Pratik','Class':'TE COMP'}); WriteResult({ "nInserted" : 1 })

>               db.Student.find();
{ "_id" : ObjectId("5ba1d618f5bbacd4ad81568d"), "Rno" : "1","Name" : "Piyush", "Class" : "TE COMP" }
{ "_id" : ObjectId("5ba1d625f5bbacd4ad81568e"), "Rno" : "2","Name" : "Abhi", "Class" : "TE COMP" }
{ "_id" : ObjectId("5ba1d63af5bbacd4ad81568f"), "Rno" : "3","Name" : "Ashley", "Class" : "TE COMP" }
{ "_id" : ObjectId("5ba1d647f5bbacd4ad815690"), "Rno" : "4","Name" : "Hitesh", "Class" : "TE COMP" }
{ "_id" : ObjectId("5ba1d65ef5bbacd4ad815691"), "Rno" : "5","Name" : "Pratik", "Class" : "TE COMP" }
{ "_id" : ObjectId("5ba1d66df5bbacd4ad815692"), "Rno" : "6","Name" : "Pratik", "Class" : "TE COMP" }
```

> **db.**Student.find().pretty();
{
"_id" : ObjectId("5ba1d618f5bbacd4ad81568d"),"Rno" : "1",
"Name" : "Piyush",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d625f5bbacd4ad81568e"),"Rno" : "2","Name" : "Abhi",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d63af5bbacd4ad81568f"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),
"Rno" : "4",
"Name" : "Hitesh",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d65ef5bbacd4ad815691"),"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d66df5bbacd4ad815692"),"Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}

> **db.**Student.update({'Name':'Hitesh'},{$set: {'Name':'Henry'}}); WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1})

> **db.**Student.find().pretty();
{
"_id" : ObjectId("5b8fad4ef00832a0a50b5036"),"Rno" : "1",
"Name" : "Piyush",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fad62f00832a0a50b5037"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fad70f00832a0a50b5038"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fad7ff00832a0a50b5039"),"Rno" : "4",
"Name" : "Henry",
"Class" : "TE COMP"
}{ "_id" : ObjectId("5b8fad8df00832a0a50b503a"), "Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fada4f00832a0a50b503b"),"Rno" : "6","Name" : "Pratik",
"Class" : "TE COMP"
}

> **db.**Student.remove({'ADD':'MP'}); WriteResult({ "nRemoved" : 1 })

> **db.**Student.find().pretty();
{

"_id" : ObjectId("5b8fad62f00832a0a50b5037"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fad70f00832a0a50b5038"),"Rno" : "3",

"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fad7ff00832a0a50b5039"),"Rno" : "4",
"Name" : "Henry",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fad8df00832a0a50b503a"),"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"
} {"_id" : ObjectId("5b8fada4f00832a0a50b503b"), "Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}

>db.Student.save({_id:ObjectId("5b8fad4ef00832a0a50b5036"),"RNO ":"1","NAME":"PIYUSH","CLASS":"TE
COMP","ADD":"MP"});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1})

> **db.**Student.find().pretty();
{
"_id" : ObjectId("5b8fad4ef00832a0a50b5036"),"RNO" : "1",
"NAME" : "PIYUSH",
"CLASS" : "TE COMP",
"ADD" : "MP"
}{
"_id" : ObjectId("5b8fad62f00832a0a50b5037"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fad70f00832a0a50b5038"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fad7ff00832a0a50b5039"),"Rno" : "4","Name" : "Henry",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fad8df00832a0a50b503a"),"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5b8fada4f00832a0a50b503b"),"Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}

> **db.**Student.find({$and:[{"Name":"Piyush"},{"Rno":"2"}]});

> **db.**Student.find({$and:[{"Name":"Piyush"},
{"Rno":"1"}]}).pretty();
{
"_id" : ObjectId("5ba1d618f5bbacd4ad81568d"),

"Rno" : "1",

```
"Name" : "Piyush",
"Class" : "TE COMP"
}


> db.Student.find({$and:[{"Name":"Piyush"},{"Rno":"2"}]}).pretty();

> db.Student.find({$or:[{"Name":"Piyush"},{"Rno":"2"}]}).pretty();
{
"_id" : ObjectId("5ba1d618f5bbacd4ad81568d"),"Rno" : "1",
"Name" : "Piyush","Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d625f5bbacd4ad81568e"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}

> db.Student.find({$or:[{"Name":"Piyush"},{"Class":"TE COMP"}]}).pretty();
{
"_id" : ObjectId("5ba1d618f5bbacd4ad81568d"),"Rno" : "1",
"Name" : "Piyush",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d625f5bbacd4ad81568e"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d63af5bbacd4ad81568f"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),"Rno" : "4",
"Name" : "Hitesh","Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d65ef5bbacd4ad815691"),"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d66df5bbacd4ad815692"),"Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}

> db.Student.find({$nor:[{"Name":"Piyush"},{"Class":"TE COMP"}]}).pretty();

> db.Student.find({$nor:[{"Name":"Piyush"},
{"Rno":"2"}]}).pretty();
{
"_id" : ObjectId("5ba1d63af5bbacd4ad81568f"),"Rno" : "3",
"Name" : "Ashley",

"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),"Rno" : "4",
"Name" : "Hitesh",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d65ef5bbacd4ad815691"),"Rno" : "5","Name" : "Pratik",
"Class" : "TE COMP"
}{
```

"_id" : ObjectId("5ba1d66df5bbacd4ad815692"),"Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}
db.Student.find({"Rno": { $not:{$lt:"3"}}}).pretty();
{
"_id" : ObjectId("5ba1d63af5bbacd4ad81568f"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),"Rno" : "4",
"Name" : "Hitesh",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d65ef5bbacd4ad815691"),"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d66df5bbacd4ad815692"),"Rno" : "6","Name" : "Pratik",
"Class" : "TE COMP"
}
> **db.**Student.find({"Rno": { $eq:"5"}}).pretty();
{
"_id" : ObjectId("5ba1d65ef5bbacd4ad815691"),"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"
}

> **db.**Student.find({"Rno": { $ne:"5"}}).pretty();
{
"_id" : ObjectId("5ba1d618f5bbacd4ad81568d"),"Rno" : "1",
"Name" : "Piyush",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d625f5bbacd4ad81568e"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}{ "_id" : ObjectId("5ba1d63af5bbacd4ad81568f"), "Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),"Rno" : "4","Name" : "Hitesh",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d66df5bbacd4ad815692"),"Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}

> **db.**Student.find({"Rno": { $gt:"5"}}).pretty();
{
"_id" : ObjectId("5ba1d66df5bbacd4ad815692"),"Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}
> **db.**Student.find({"Rno": { $gte:"5"}}).pretty();
{
"_id" : ObjectId("5ba1d65ef5bbacd4ad815691"),"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"

```
}{
"_id" : ObjectId("5ba1d66df5bbacd4ad815692"),"Rno" : "6",
"Name" : "Pratik",
"Class" : "TE COMP"
}

> db.Student.find( {"Rno": { $lt:"5"}}).pretty();
{
"_id" : ObjectId("5ba1d618f5bbacd4ad81568d"),"Rno" : "1","Name" : "Piyush",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d625f5bbacd4ad81568e"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d63af5bbacd4ad81568f"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),"Rno" : "4",
"Name" : "Hitesh",
"Class" : "TE COMP"
}

> db.Student.find( {"Rno": { $lte:"5"}}).pretty();
{
"_id" : ObjectId("5ba1d618f5bbacd4ad81568d"),"Rno" : "1",
"Name" : "Piyush",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d625f5bbacd4ad81568e"),"Rno" : "2","Name" : "Abhi",
"Class" : "TE COMP"

}{
"_id" : ObjectId("5ba1d63af5bbacd4ad81568f"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),"Rno" : "4",
"Name" : "Hitesh",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d65ef5bbacd4ad815691"),"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"
}

> db.Student.find( {"Rno": { $lt:"5",$gt:"2"}}).pretty();
{
"_id" : ObjectId("5ba1d63af5bbacd4ad81568f"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),"Rno" : "4",
"Name" : "Hitesh","Class" : "TE COMP"
}

> db.Student.find( {"Rno": { $lte:"5",$gte:"2"}}).pretty();
{
"_id" : ObjectId("5ba1d625f5bbacd4ad81568e"),"Rno" : "2",
```

"Name" : "Abhi",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d63af5bbacd4ad81568f"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),"Rno" : "4",
"Name" : "Hitesh",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d65ef5bbacd4ad815691"),"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"
}
> **db.**Student.find({"Rno": { $lte:"5",$gt:"2"}}).pretty();
{
"_id" : ObjectId("5ba1d63af5bbacd4ad81568f"),"Rno" : "3",
"Name" : "Ashley","Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),"Rno" : "4",
"Name" : "Hitesh",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d65ef5bbacd4ad815691"),"Rno" : "5",
"Name" : "Pratik",
"Class" : "TE COMP"
}

> **db.**Student.find({"Rno": { $lt:"5",$gte:"2"}}).pretty();
{
"_id" : ObjectId("5ba1d625f5bbacd4ad81568e"),"Rno" : "2",
"Name" : "Abhi",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d63af5bbacd4ad81568f"),"Rno" : "3",
"Name" : "Ashley",
"Class" : "TE COMP"
}{
"_id" : ObjectId("5ba1d647f5bbacd4ad815690"),"Rno" : "4",
"Name" : "Hitesh",
"Class" : "TE COMP"}

RESULT

Thus the CRUD operation in NoSQL database MongoDB has been successfully executed.

EX.NO:12 **GUI BASED DATABASE APPLICATIONS**

AIM:

To implement an online exam registration page with database connectivity using php

ALGORITHM:

STEP 1: Develop a webpage which includes all the fields for registering for an online exam
STEP 2: Develop a PHP page which receives the values that are given as an input in the registration page
STEP 3: Connect the registration page with the PHP page
STEP 4: Receive the values from registration page
STEP 5: Establish a connection with the database
STEP 6: Store the received values in the database

PROGRAM:

Registration.html

```
<html>
<body bgcolor="lightblue">
<p><center>REGISTRATION FORM</center></p>
<form action="Registration.php" method="post">
<center><pre><b>  Student name: <input type="text" name="n" value=" "> Register Number: <input type="text" name="reg" value=" ">
CGPA: <input type="text" name="cgpa" value=" ">YEAR: <input type="text" name="y" value=" ">
Branch: <input type="text" name="branch" value=" ">
<input type="submit" name="button" value="SUBMIT">
</b></center></pre>
</body>
</html>
```

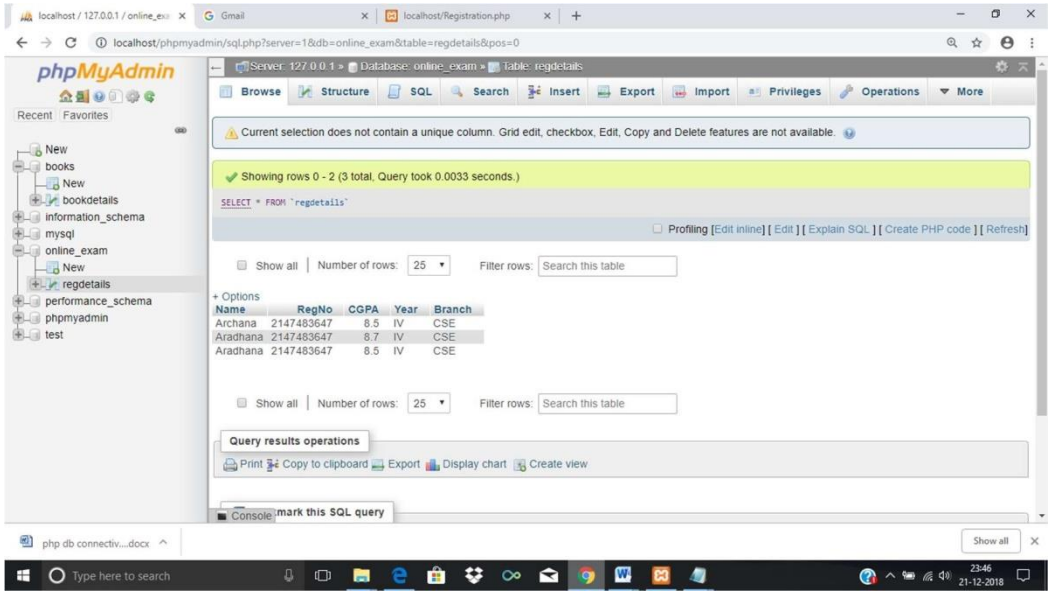
Registration.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Online_exam";
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error)
{
die("Connection failed: " . $conn->connect_error);
}
/*else
echo "Connection established";*/

$Name=$_POST['n'];
$RegNo=$_POST['reg'];
$CGPA=$_POST['cgpa'];
$Year=$_POST['y'];
$Branch=$_POST['branch'];
/*echo $Name; echo $RegNo; echo $CGPA; echo $Year; echo $Branch;*/
```

```
$sql = "CREATE TABLE regDetails(Name varchar(30) not null, RegNo int(15) not null,CGPA floatnot null, Year varchar(5) not null, Branch varchar(5) not null)";

if ($conn->query($sql) === TRUE)
{
echo "Table regDetails created successfully";
}
else
{
echo "Error creating table: " . $conn->error;
}
$sql = "INSERT INTO regDetails (Name,RegNo,CGPA,Year,Branch) VALUES
('$Name',$RegNo,$CGPA,'$Year','$Branch')";
if ($conn->query($sql) === TRUE)
{
echo "New record created successfully";
}
else
{
echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>
```



RESULT:

Thus the real life database application for exam registration has been successfully executed.

EX.NO:13 DATABASE DESIGN USING ER MODELING, NORMALIZATION AND IMPLEMENTATION FOR ANY APPLICATION

AIM:

To implement Database Design using ER modeling, normalization for an application

ALGORITHM:

STEP 1: Create a database schema for an application
STEP 2: Draw an E-R diagram for the schema
STEP 3: Check for atomic values and change it to first normal form
STEP 4: Check for functional dependency and modify the schema to second normal form
STEP 5: Check for transitive dependency and convert to third normal form
STEP 6: Check that every determinant is a candidate key and convert to Boyce Codd normal form
STEP 7: Check for multi-valued dependency and change it to fourth normal form
STEP 8: Check whether the table is fully normalized

First Normal Form:

- It should only have single(atomic) valued attributes/columns.
- Values stored in a column should be of the same domain
- All the columns in a table should have unique names

Changing it to 1NF:

roll_no	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C++

Second Normal Form (2NF)

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency.

Changing it to 2NF:

Student:

student_id	name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat

Subject:

subject_id	subject_name
1	Java
2	C++
3	Php

Score:

score_id	student_id	subject_id	marks	teacher
1	10	1	70	Java Teacher
2	10	2	75	C++ Teacher
3	11	1	80	Java Teacher

subject_id	subject_name	teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

And our Score table is now in the second normal form, with no partial dependency.

score_id	student_id	subject_id	marks
1	10	1	70
2	10	2	75
3	11	1	80

Third Normal Form (3NF)

For a table to be in the third normal form,

1. It should be in the Second Normal form.
2. And it should not have Transitive Dependency.

Student Table

student_id	name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat
12	Bkon	09-WY	IT	Rajasthan

Subject Table

subject_id	subject_name	teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

RESULT:

Thus normalization has been implemented for student database successfully