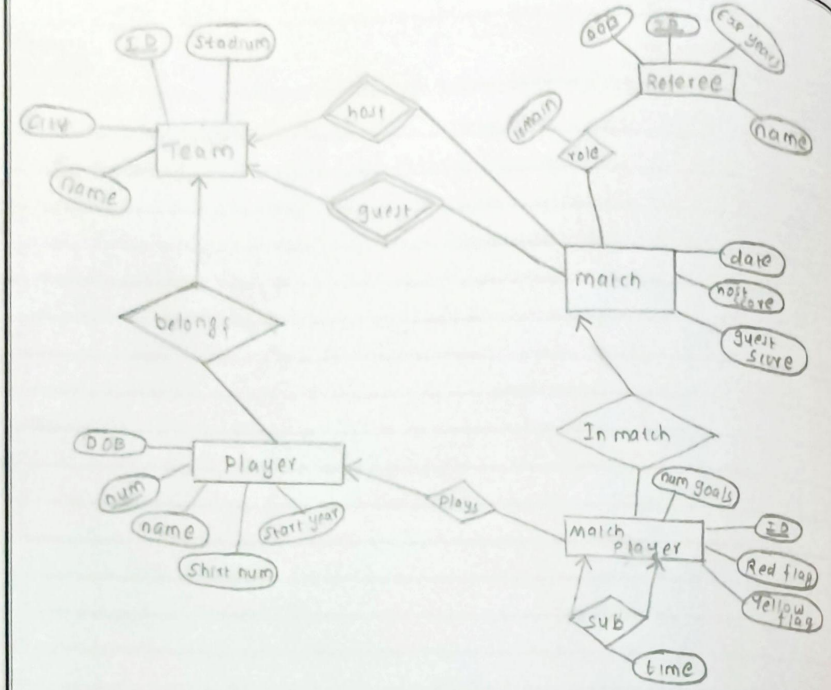


30/11/21

1

Experiment: 1

Q) Design an ER diagram for an application that models soccer teams by considering the games they play and the players in each ~~time~~ team. In the design, the following information are to be considered.

→ Set of teams, each team has an ID, name, main stadium and to which city the team belongs.

→ Each team has many players, each player belongs to one team. Each player has a number, name, DoB, start year and shirt number.

→ Teams play matches, each match there is a host and guest team. Match takes place in stadium of the host team.

→ For each match we need to keep track of the following.

→ Date on which game is played

→ Final result of the match

→ Players participated in the match, no. of goals scored, yellow card / red card taken or not

→ Substitutions and the time at which it took place.

→ Each match has exactly 3 referees. Each referee have an ID, name, DoB, year of experience (1 main & 2 assistant referees)

Write details about the assumptions made for designing ER diagram.

Result

The ER diagram has been created as per requirements.

Assumptions:

1) The match player, entity set, are added a unique identifier, for each record ID

2) The final result is match entity set is captured using two attributes, host score and guest score.

3) The attributes "is main" in relationship 'role' is true if the refree is the main refree in the match, otherwise it will be false.

Branch:

Branch_name	Branch_city	Assets

customer:

customer_name	customer_street

Loan:

loan_number	branch_name	loan_amount

Borrower:

customer_name	loan_number

Account:

Account_number	branch_name	balance

10/12/21 5

Experiment :-2

Q) Create the following database and insert values into tables using UI and SQL commands. Also find queries for the questions given below.

(a) Banking database

branch (branch_name, branch_city, assets)
loan (loan_number, branch_name, loan_amount)
customer (customer_name, customer_street)
borrower (customer_name, loan_number)
account (account_number, branch_name, balance)
depositor (customer_name, account_number)

i) Insert a new column customer_area to customer table.
ii) change the data type of loan_amount to integer.
iii) Rename the attribute, customer_area to customer_city.

(b) Employee database

Employee (employee_name, street, city)
Works (employee_name, company_name, salary)
Company (company_name, Manager name)
Manager (employee_name, Manager_name)

i) Add a new column Date_of_joining to the employee.
ii) Delete the column country from the company table.

Depositor:

customer_name	account_number

Customer:

customer_name	customer_street	customer_area

Employee:

Employee_name	Street	City

SQL Commands

a) Banking database

```
CREATE TABLE BRANCH (branch_name varchar(20) primary key,
branch_city varchar(20), assets varchar(20));

CREATE TABLE CUSTOMER (customer_name varchar(20) primary key,
customer_street varchar(20));

CREATE TABLE BORROWER (customer_name varchar(20), loan_num int,
primary key (customer_name, loan_num));

CREATE TABLE ACCOUNT (account_number int primary key, branch_name
varchar(20), balance int);

CREATE TABLE DEPOSITOR (customer_name varchar(20), account_number int);

i) ALTER TABLE CUSTOMER AND customer_area varchar(20);
ii) ALTER TABLE LOAN ALTER COLUMN loan_amount int;
iii) ALTER TABLE CUSTOMER RENAME COLUMN customer_area
to customer_city;
INSERT INTO DEPOSITOR ('Name1', '101');
```

b) Employee Database

```
CREATE TABLE EMPLOYEE (employee_name varchar(20) primary key,
street varchar(20), city varchar(20));
```

WORKS

Employee_name	company_name	Salary

Company:

company_name	city	country

Manager:

Employee_name	Manager_name

Employee:

Employee_name	street	city	Date_of_joining

Company:

company_name	city

CREATE TABLE WORKS (employee_name varchar(20) primary key, company_name varchar(20), city varchar(20));

CREATE TABLE COMPANY (company_name varchar(20) primary key, city varchar(20), country varchar(20));

CREATE TABLE MANAGER (employee_name varchar(20) primary key, manager_name varchar(20));

INSERT INTO MANAGER ('Name1', 'Name2');

i) ALTER TABLE EMPLOYEE ADD Date_of_joining int;

ii) ALTER TABLE COMPANY DROP COLUMN COUNTRY;

Result

The queries are executed successfully and table have been created according to specification.

10/12/21

a)

customer_name	customer_street	customer_city
Joseph	Sreenagar	Bangalore
Sree	Marine Drive	Kochi
Kavya	Kowdiar	Tvm
John	Jawahar	Mumbai
Nived	SM	Delhi

b)

account_number
56418
32517

c)

customer_name
Joseph

d)

count (*)
4

e)

customer_street	customer_city
Marine Drive	Kochi

10/12/21

Experiment : 3

Banking Database

Q) Use the following banking database and find the queries for the questions below:

branch (branch_name, branch_city, asset)
customer (customer_name, customer_street, city)
loan (loan_number, branch_name, loan_amount)
borrower (customer_name, loan_number)
account (accno, branch_name, balance)
depositor (customer_name, account_number)
employee (emp_name, branch_name, dob, salary)

a) Display details of all customers

```
SELECT * from customer;
```

b) Find bank accounts with a balance under 1000 rs.

```
SELECT * from account
where balance < 1000;
```

c) Find names of customers whose city is Bangalore.

```
SELECT customer_name
from customer where customer_city = "Bangalore";
```

d) Find no:of tuples in the customer relation.

```
Select count (*)
from customer;
```

f)

customer_name
Sree

g)

count(*)
2

h)

Customer_name
Joseph

i)

Employee_name
Kavya

j)

Avg_salary
4000

k)

amount_no	balance
53789	2000

e) Find name and city where John lives
select customer_street, customer_city
from customer
where customer_name = 'John';

f) List the customers whose names end with letters 'E' and has exactly 4 characters
Select * from customer
where customer_name like '___E';

g) Find total no of customers who have taken a loan
Select count(*)
from (select distinct customer_name from borrower);

h) Find the names of all customers whose street includes substring 'Nagar'
Select customer_name
from customer
where customer_street like '%_Nagar_%';

i) Find all employees whose salary is greater than 1400 and working branch is not Mumbai
Select employee_name
from employee
where salary > 1000
and branch_name != 'Mumbai';

l)

Branch_name
Kochi
Bangalore

m)

Branch_name	Count
Kochi	1
Bangalore	2

n)

Customer_name
John
Sree

o)

Customer_name
Sathya

p)

Customer_name
Nived

j) calculate the average salary of all employees and show average salary as 'avg_salary';

```
Select AVG(salary) As avg_salary
From employee;
```

k) Find the account_no. and balance for all accounts from kolkata branch where the tolerance is greater than 100 rupees.

```
Select account_number, balance
from account
where branch_name = 'kolkata' and balance > 1000;
```

l) Find name of all branches with assets b/w 10 and 25 lakhs

```
Select branch_name From branch
where assets between 10,00000 and 2500000
```

m) Find the number of depositor for each branch

```
Select branch_name, count (*)
from account a, depositor d
where a.accno = d.cusno
group by branchname
```

n) Find the name of customers who have both loan and account

```
Select distinct b.customer_name
from borrower b, depositor d
where b.customer_name = d.customer_name;
```

6)

Branch_name
Delhi
Kochi

7)

Employee_name
Kavya
Joseph

8)

Employee_name
John

9)

Month
Jan
Oct
Apr

6) Find the name of all customers who have account but no loan

```
Select distinct b.customer_name
from depositor d, borrower b
where d.customer_name NOT IN customer_name;
```

7) List in alphabetical order the names of all customers having a loan in Delhi branch

```
Select customer_name
from loan l, borrower b
where branch_name = 'Delhi' AND l.loanno := b.loanno;
```

8) Find the name of all branches that have greater assets than some branch located in Bangalore.

```
Select branch_name
from branch
where (select Min (ASSETS)
from branch
where branch_city = 'Bangalore') < assets;
```

9) List the name of employee who are born in January

```
Select employee_name
from employee
where DOB Like '__-JAN-__';
```

5) List the name of employees where age is greater than 30

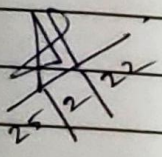
```
Select employee_name
from employee
where extract (year from dob) <= 1991;
```

6) Display month of birth of all employee

```
Select extract (month from DOB)
from employee;
```

Result

The queries are implemented successfully and the output is verified.



Employee-name
John
Manoj

14/12/21

Experiment :-4

Employees Relational Schema Queries

Q) Consider the following relational schema, Give an expression in SQL for each of the following queries:

Employee (employee-name, street, city)
works (employee-name, name, salary)
company (company-name, city)
manages (employee-name, manager-name)

a) Find the name, street address cities of residence for all employee who work for 'universal corporation' and earn more than 10000 ruppees.

```
Select E.employee-name, street, city
From employee E, works W
where E.employee name = W.employee-name AND
      company-name = 'Universal corporation' AND
      salary > 10000;
```

b) Find the names of all employees in database whose lives in same cities as the companies for which they work.

```
Select E.employee name
From employee E, works W, company C
Where E.employee-name = W.employee-name AND
      W.company-name = C.company-name AND
      C.city = E.city;
```

c)

employee_name
John

d)

employee-name
John
Manoj
William

e)

employee_name
John
Manoj
George

c) Find the names of all employees in the database who live in the same cities and on the same streets as do their managers.

```
Select E1.employee_name
from employee E1, employee E2, manages m
where E1.employee_name = m.employee-name AND
      m.manager-name = E2.employee_name AND
      E1.city = E2.city AND E1.street = E2.street
```

d) Find the names of all employees in the database who do not work for 'ABC corporation'. Assume that all people work for exactly one company.

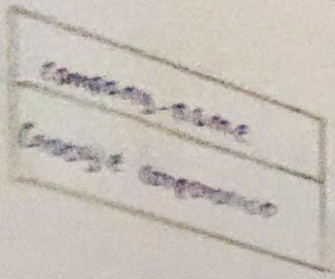
```
Select employee_name
from works
where company_name != 'ABC Corporation';
```

e) Find the names of all employees in the database who earn more than every employee of 'Groogle corporation'. Assume that all people work for atmost one company

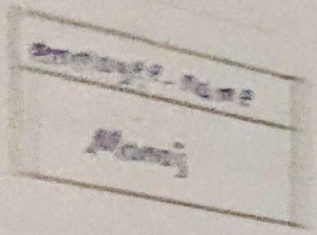
~~Select E.employee_name~~
~~from employee as E, works w~~
~~where E.employee_name = W.employee_name AND~~
~~W.salary > (Select Avg(salary) From works);~~

```
select distinct employee_name
from works
where salary > (select max(salary)
                From works
                where company_name = 'Groogle corporation');
```

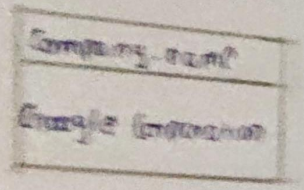
5)



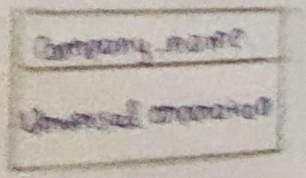
6)



7)



8)



f) Assume that the company may be located in several cities. Find all companies located in every city in which 'ABC corporation' is located.

```
Select C1.company_name from company C1, company C2
where C1.city = C2.city AND
      C2.company = 'ABC corporation';
```

g) Find the names of all employees who earn more than average salary of all employees of their company. Assume that all people work for atmost one company.

```
Select E.employee_name
from employee E, works W
where E.employee_name = W.employee_name and
      W.salary > (select Avg (salary) from works);
```

h) Find the name of company that has smallest payroll.

```
Select company_name group by company_name
Having Avg(salary) <= (Street Avg (salary From works
Having (salary) <= (Select Avg(salary) from works);
```

i) Find the company that has most employees.

```
Select company_name from (select company_name,
                          count (employee_name))
a2 WT from works
group by company_name
order by int desc
```

17) Find those companies whose employees earn a high salary on average, than the avg salary at 'Google corporation'.

```
Select company_name
from works
group by company_name
having Avg(salary) >[select Avg(salary) from works]
where company_name = 'Google Corporation'
```

Result
The queries are executed successfully.

15/2/27

Classroom

Building	room-no	capacity
Block-A	212	60
Block-B	305	100
Block-C	124	80

Department

Dept-name	building	budget
Chemistry	Block-A	10,000
Physics	Block-B	15,000
Maths	Block-C	8,000

14/10/21

Experiment :- 5

View - University Database

Q) Create the view with all course sections offered by Physics dept in the Fall 2009 semester, with the building and room number of each section.

University database

classroom (building, room-no, capacity)
department (dept-name, building, budget)
course (course-id, title, dept-name, credits)
instructor (ID, name, dept-name, salary)
section (course-id, sec-id, semester, year, building, room no, time-slot-id)
teaches (ID, course-id, sec-id, semester, year)

```
Create table classroom (
    building varchar(20)
    room_no int,
    capacity int,
    primary key (building, room_no));

Create table department (
    dept-name varchar(20) primary key;
    building varchar (20),
    budget int);
```

Course

course_id	Title	dept_name	Credits
101	Fretron	Physrcs	4
102	Gravitation	physrcs	5
121	Bio-chemicals	chemistry	3

Instructor

ID	Name	dept_name	Salary
20	Manoj	Physrcs	30000
21	Deepa	Chemistry	40000
22	Annie	Maths	40 000
19	Gayathri	Physrcs	30 000

Section

course_id	sec_id	semester	year	building	room_no	time_slot_id
101	1	Fall	2009	Block-A	212	1
102	2	Fall	2009	Block-A	212	2
121	3	Fall	2009	Block-B	305	3


```
Create table course(
course_id int primary key,
title varchar(20),
dept_name varchar(20),
credits int,
foreign key(dept_name) references department(dept_name));

Create table instruction(
ID int primary key, name varchar(20),
dept_name varchar(20),
salary int,
foreign key(dept_name) references department(dept_name));

Create table section(
course_id int,
sec_id int,
semester varchar(20)
year int,
building varchar(20)
room_no int,
time_slot_id int,
primary key(course_id, sec_id, semester, year)
foreign key(course_id) references courses(course_id),
foreign key(building, room_no) references classroom
                    (building, room no));
```

course

course-id	Title	dept-name	Credits
101	Friction	Physics	4
102	Gravitation	physics	5
121	Bio-chemicals	chemistry	3

Instructor

ID	Name	dept-name	salary
20	Manoj	Physics	30000
21	Deepa	chemistry	40000
22	Annie	Maths	40 000
19	Gayathri	Physics	30 000

Section

course-id	sec-id	semester	year	building	room-no	time-slot-id
101	1	Fall	2009	Block-A	212	1
102	2	Fall	2009	Block-A	212	2
121	3	Fall	2009	Block-B	305	3


```
Create table course(
    course_id int primary key,
    title varchar(20),
    dept_name varchar(20),
    credits int,
    foreign key(dept_name) references department(dept_name));

Create table instruction(
    ID int primary key, name varchar(20),
    dept_name varchar(20),
    salary int,
    foreign key (dept_name) references department (dept_name)),

Create table section (
    course_id int,
    sec_id int,
    semester varchar(20)
    year2 int,
    building varchar(20)
    room_no int,
    time_slot_id int,
    primary key(course_id, sec_id, semester, year)
    foreign key(course_id) references courses(course_id),
    foreign key(building, room_no) references classroom
                (building, room no));
```

Teacher

ID	course_id	sec_id	semester	year
19	101	1	Fall	2009
20	102	2	Fall	2009
21	121	3	Fall	2009

View

Physics_courses

course_id	sec_id	Building	Room_no
101	1	Block-A	212
102	2	Block-A	212


```
Create table teacher(
    ID int,
    course_id int,
    sec_id int,
    semester varchar(20),
    year int,
    primary key (ID, course_id, sec_id, semester, year),
    foreign key (ID) references instructor(ID),
    foreign key (course_id, sec_id, semester, year) references
            section(course_id, sec_id, semester, year));

Create view physics_courses as
    select s.courseid, s.sec_id, s.building, s.room_no,
    from section s, course c
    where s.course_id = c.course_id AND
            c.dept_name = 'physics' AND
            s.semester = 'Fall' AND
            s.year = '2009';

SELECT * from physics_courses;
```

Result

Tables are created according to given specifications and view is also created satisfying given conditions.

12/22

17/12/21

Experiment :- 6

PL/SQL - Area of the circle

Q) Write PL/SQL code to enter radius from user and calculate the area of a circle. Write the radius as well as area into a table called circle area. Also write PL/SQL code to calculate the area of a circle for radius ranging from 3 to 7.

```
Create table circle_area (
        radius number,
        area number(7,2));
```

Program

```
Declare
    r number;
    area number(7,2);
    pi constant number := 3.14;

Begin
    r := &r;
    area := pi*r*r;
    Insert into circle_area values(r,area);
     r:=3;
    While r<=7
    Loop
        area := pi*r*r
        insert into circle_area values(r, area);
        r := r+1;
```

Output : circle_area

Radius	Area
2	12.56
3	28.26
4	50.24
5	78.50
6	113.04
7	153.86


```
End loop;
End;

Select *
from circle_area;
```

Result

Table is created and PL/SQL code is written to calculate the area of circle and to insert into the table created.

15/2/22

Before:

Account

Acc.no	Branch-name	Balance
2345	TVM	20000
5627	TVM	50000
7214	KLM	4000

Output:

account_no := 7214

After:

Account

Acc-no	Branch-name	Balance
2345	TVM	20,000
5627	TVM	50000
7214	KLM	3900

Experiment:-7

PL/SQL - Bank Account Details

Write a PL/SQL code that accepts an account number from user. If the user balance is less than minimum balance then deduct 100 rupees from balance. This process is to be done on account table.

Program:

```
DECLARE
    account_no int;
    bal int;
    min constant int := 5000;

Begin
    account_no := &account_no;
    Select balance into bal
    from account
     where acc_no = account_no;
    if bal < min
    then
       bal := bal-100;
       update account
        set balance = bal
        where acc_no = account_no;
    end if;
End;
```

Result :- PL/SQL code is written for updating the balance on account table

15/12/22

17/12/21

Experiment:-8

PL/SQL Employee

Q) Write a PL/SQL code to obtain employee number if employee number is present in the table, then display corresponding salary of employee along with the message 'employee record found'. Update the salary of employee having total salary less than ₹1000 by increasing with ₹1000. Display no. of rows affected by this query

Program

```
DECLARE
    name varchar(20);
    empno int := 0;
    sal int;
    c int;
BEGIN
    name := &name;
    Select emp_no into empno
    from employee
    where employee name = name;
    if empno != 0
    then
        DBMS_OUTPUT.PUT_LINE('Employee record found');
        DBMS_OUTPUT.PUT_LINE('Employee no:' || empno);
        Select salary into sal
        from employee
        where emp_no = empno;
        DBMS_OUTPUT.PUT_LINE('Salary:' || sal);
end if
```

BEFORE:

employee_name	salary	emp_no
MANU	10 000	1234
ANU	10000	4562
DENNY	8000	5001

OUTPUT:

Enter value for name : 'ANU'
Employee record found
Employee no : 4562
Salary- 10000
No:of rows affected -1

After:

employee_name	Salary	emp_no
MANU	15000	1234
ANU	10000	4562
DENNY	9000	5001


```
select count(*) in c
from employee
where salary < 1000;

Update employee
set salary = salary + 1000
where salary < 1000;

DBMS_OUTPUT.PUT_LINE('No:of rows affected - '||c);
END;
/
```

Result:- PL/SQL code implemented successfully

15/2/22

Experiment:-9

Procedure- Min, Avg & Sum

Write a procedure to find the minimum, sum and average of two numbers.

Program

```
DECLARE
    a number:=&a;
    b number:=&b;
    s number;
    minimum number;
    average number;

PROCEDURE MIN(a in number, b in number, minimum out number) AS
BEGIN
    if a<b
    then
        minimum:=a;
    else
        minimum:=b;
    end if;
END MIN;

PROCEDURE sum(a in number, b in number, s out number) AS
BEGIN
    s=a+b;
END sum;
```

OUTPUT:

Enter value for a : 6
Enter value for b : 4
min -4
sum -10
avg -5


```
PROCEDURE Avg(s in number, average out number)As
BEGIN
    average:=(s/2);
END AVERAGE
BEGIN
    MIN(a,b,minimum);
    DBMS_OUTPUT.PUT_LINE('min'||minimum);
    sum(a,b,s);
    DBMS_OUTPUT.PUT_LINE('sum'||s);
    Avg(s,average)
    DBMS_OUTPUT.PUT_LINE('avg'||average);
END;
/
```

Result:- Procedure for finding minimum, sum and average implemented successfully

15/2/22

4/1/22

Experiment :- 10

Procedure - Calculation of Salary

Write a procedure that checks for employee having salary greater than 10000 and give an increment of 6 %. If more than one employee exists, display the message 'too many employees'. If no such employee exist display 'no such employee'.

Program

```
DECLARE
   c number;
PROCEDURE EMP AS
BEGIN
    Select count (*) into c from employee
    where salary > 10000;
    if (c>0)
    then
        update employee
        set salary = (salary + (0.06 x salary));
    if (c>1)
    then
        DBMS_OUTPUT.PUT_LINE ('Too many employees');
    elseif (c=0)
    then
        DBMS_OUTPUT.PUT_LINE ('No such employee');
    endif;
End Emp;
```

Before

employee name	Salary	emp-no
MANU	12000	1234
ANU	10000	4562
DENNY	8000	5001

AFTER:

employee name	Salary	emp-no
MANU	12720	1234
ANU	10000	4562
DENNY	8000	5001


```
BEGIN
    EMP;
END;
```

Result: Procedure for updating employee salary implemented successfully.

Employee

employee-name	salary	emp-no
MANU	12720	1234
ANU	10000	4562
DENNY	8000	5001

OUTPUT

No :of tuples in employee table :3

Experiment : 11

Function

Write a function to count the number of tuples in employee relation and return it

Program

```
DECLARE
    c number;
FUNCTION COUNT_TUPLES
RETURN NUMBER IS
    temp number;
BEGIN
    select count(*) into temp from employee;
    return temp;
END;

BEGIN
    c := COUNT_TUPLES;
    DBMS_OUTPUT.PUT_LINE('No: of tuples in employee table' || c);
END;
```

Result

Function to count no:of tuples successfully implemented.

Employee

employee-name	salary	emp-no
MANU	12720	1234
ANU	10000	4562
DENNY	8000	5001

OUTPUT

Enter value for empno : 1234
NAME : MANU
SALARY : 12720

Enter value for empno : 200
No such employee

Experiment:-12
PL/SQL Exception Handling

Draw a PL/SQL code that accepts an employee number from the user and if the employee number is present in the table then corresponding name and salary are displayed. If the employee is not in the table an exception should be raised which displays the message 'No such employee'.

Program

```
DECLARE
    empno number;
    e_name varchar(20);
    e_sal number;
BEGIN
    empno := &empno;
    Select employee_name, salary into e_name, e_sal
    from employee
    where emp_no = empno;
    DBMS_OUTPUT.PUT_LINE('NAME -'||e_name||CHR(13)||CHR(10)||
                         'SALARY': ||e_sal);
EXCEPTION
    when no_data_found then
    DBMS_OUTPUT.PUT_LINE('No such employee');
END;
```

Result:- PL/SQL code with exception successfully implemented.

E_id	E_name	E_city	E_age	E_company
E01	JENNY	TVM	25	ABC
E02	RAVI	KOCHI	27	DEF
E03	RAM	MUMBAI	30	MNO
E05	RAJ	DELHI	28	XYZ

Experiment :-13

Exception Handling and Checking Conditions

Consider the schema (eid, ename, ecity, eage, company). Write a PL/SQL program that satisfies following conditions after accepting employee ID as input.

a) If eid = E01, then raise an exception to display corresponding city.
b) If eid = E05, then raise an exception to display name
c) Otherwise raise exception to display corresponding age

```
Declare
    eid varchar(20);
    city varchar(20);
    company varchar(20);
    age number;
    city_exception EXCEPTION;
    age_exception EXCEPTION;
    company_exception EXCEPTION;

Begin
    eid:= &eid
    select e_city, e_age, e_company INTO city,age,company
    From employee
    where e_id = eid;
    if eid = 'E01' then
        raise city_exception;
```

OUTPUT

employee_id : E01
city : TVM
employee-id : E02
age : 27
employee-id : E05
company : XYZ


```
elsif eid = 'E05' then
    raise company_exception;
else
    raise age_exception
end if;

EXCEPTION
    when city_exception then
        dbms_output.put_line('city: ' || city);
    when company_exception then
        dbms_output.put_line('company: ' || company);
    when age_exception then
        dbms_output.put_line('age: ' || age);

END;
```

Result

PL/SQL code to execute exception handling implemented successfully.

4/3/22

Book

Book-no	Title	Cost
302	MNO	50
101	ABC	100

→insert into book
values (200, 'DEF', 40);

ORA - 20000 : cost less than average
ORA - 06512 : at "RSA. INSERTION", line 9
ORA - 04088 : Error during execution of trigger 'RSA. INSERTION

→insert into book
values (205, 'XYZ', 200);

Book-no	Title	Cost
302	MNO	50
101	ABC	100
205	XYZ	200

11/2/22

61

Experiment :-14

Trigger

Consider the following relation
Book (book-no, title, cost)
Create a trigger to do the following
Insertion is possible in the book table if cost is greater than average cost of books available.

```
CREATE OR REPLACE TRIGGER insertion
BEFORE INSERT ON book
FOR EACH ROW
DECLARE
  average number;
BEGIN
  Select avg (cost) into average
  from book;
  if :new.cost < average
  then
    raise_application_error(-2000, 'cost less than average');
  end if;
END;
```

Result
Trigger created successfully and executed.

AP
4/3/22

employee_name	employee_id	salary
CHRISTY	1234	13000
ANU	4562	10000
DENNY	5001	8000

Output:

enter value for id : = 1234

valid id

Salary incremented

employee-name	employee-id	salary
CHRISTY	1234	13000
ANU	4562	10000
DENNY	5001	8400

11/2/22

Experiment :-15

Package

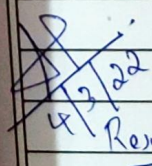
Write a package with function_id to get an employee ID and check whether it is valid. Also include a procedure inc-sal to increment the salary of employees by 5% if it is less than 1000.

```
CREATE PACKAGE emp_pack AS
    FUNCTION check_id;
    RETURN number;
    PROCEDURE inc_sal;
END emp_pack;
/
CREATE OR REPLACE PACKAGE BODY emp_pack AS
FUNCTION check_id (id in number)
RETURN number
  IS
    cnt number;
BEGIN
    Select count(*) into cnt
    from employee
    where employee_id = id;
    if cnt=1 then
      RETURN 1;
    end if;
      RETURN 0;
END check_id;
```

```
PROCEDURE inc-sal AS
BEGIN
    update employee
    set salary = salary + 0.05x salary;
    where salary > 10000;
    dbms-output-put-line ('salary incremented');
    END inc-sal;
END;
```

Program

```
DECLARE
    id number;
    val number;
    BEGIN
      id := &id;
    val := emp-pack.check-id (id);
    if val = 1 then
       dbms-output-put-line('valid id');
    else
       dbms-output-put-line ('Invalid id');
    end if;
    emp-pack.inc-sal;
    END;
/
```



Result

Package created and program implemented successfully.

Experiment:-16

Aim

Create a NOSQL restaurant database and write a mongoDB query to display the fields restaurant-id, name, address and zipcode, but exclude the field emp-id for all documents in the collection restaurant. Also write a mongo DB query to find the restaurant name, longitude and lattitude for those restaurants which contain 'gov' as first three letters of its name.

CODE

```
db.createCollection('Restaurant')

db.Restaurant.insertOne({restaurant_id:1, name:"govind",
address:"ABC street", zipcode:6161, emp-id:420, long:"42N",
lat:"62E"});

db.Restaurant.insertOne({restaurant_id:2, name:"aaryabhavan",
address:"XYZ street", zipcode:6262, emp-id:421, long:"52N",
lat:"72E"});

db.Restaurant.find({},{"emp-id":0});

db.Restaurant.find({name:{$regex:"^gov"}},{"name":1,
"long":1, "lat":1}).pretty();
```

3/3/22

Result

MongoDB queries successfully executed.

Experiment: 17

JDBC

Aim

Write GUI Java program to create student database and insert name and to retrieve marks from student (roll no, marks) for a particular roll. Also give provision to modify the marks of a particular rollno. Also give provision to modify the marks of a particular student and to display rank list.

Program

```
import java.sql.*;
import java.util.*;

Public class jdbc {
    public static void main (String arg s[]) {
        Scanner J = new Scanner (system.in);
        int nroll, nmarks, ch;
        String nname;
        try {
            Class.forName("oracle.jdbc.driver.Oracle Driver");
            Connection con = DriverManager.getConnection.
            ("jdbc:oracle:thin:@localhost:1521:ORCL", "system",
              "Oracle");
            Statement stmt = con.createStatement();
            String table = "CREATE TABLE student (NAME varchar(50),
                                ROLLNO int, MARKS int)";
        System.Out.println("Table Created");
```

```
Stmt.executeUpdate(table);
PreparedStatement ps1 = con.prepareStatement
        ("insert into student values(?,?,?)");
PreparedStatement ps2 = con.prepareStatement
        ("update student set marks = ? where rollno
        = ?");
PreparedStatement ps3 = con.prepareStatement
        ("select marks from student where rollno = ?");
do{
    System.out.println("MENU");
    System.out.println("-----");
    System.out.println("1. Add student");
    System.out.println("2. Modify marks");
    System.out.println("3. Find marks");
    System.out.println("4. Show rank list");
    System.out.println("5. Exit");
    System.out.println("Enter choice-");
    ch = s.nextInt();
    if (ch==1)
    {   System.out.println("Enter name-");
        nname = s.nextLine();
        System.out.println("Enter rollno-");
        nroll = s.nextInt();
        System.out.println("Enter marks-");
        nmarks = s.nextInt();
        ps1.setString(1,nname);
        ps1.setInt(2,nroll);
        ps1.setInt(3,nmarks);
```

```
        ps1.executeUpdate();
    }
    else if (ch==2)
    {   System.out.println("Enter roll no- ");
        nroll = s.nextInt();
        System.out.println("Enter new marks - ");
        nmarks = s.nextInt();
        ps2.setInt(1, nmarks);
        ps2.setInt(2, nroll);
        ps2.executeUpdate();
    }
    else if (ch==3)
    {   System.out.println("Enter roll no- ");
        nroll = s.nextInt();
        ps3.setInt(1, nroll);
        ps3.executeUpdate();
    }
    else if (ch==4)
    {   ResultSet rs = stmt.executeQuery("select
        * from student order by marks desc");
        while (rs.next())
            System.out.println(rs.getString("Name")
        + " " + rs.getInt("Roll no") + " " + rs.getInt
                                    ("marks"));
    }
    else if (ch==5)
        break;
    else
        System.out.println("Invalid choice");
```

OUTPUT:

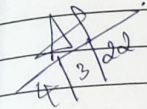
MENU
1. Add student
2. Modify marks
3. find marks
4. Show rank list
5. Exit

Enter choice 1
Enter name : Ravi
Enter rollno : 30
Enter marks : 48

Enter choice 4
Name - Ravi
Rollno - 30
Marks - 48


```
} while (true);
con.close();
}
catch (Exception e) {System.out.println(e);}
}
}
```

Result
JDBC program implemented successfully.



3/3/22

Experiment :-18

Aim

Practice SQL TCL commands like Rollback, commit and savepoint.

Code

```
create table class (id int, name varchar(20));
insert into class values (1, 'Abhi');
insert into class values (2, 'Vishnu');
insert into class values (3, 'Hari');
commit;

insert into class values (4, 'Divin');
savepoint A;
update class set id = 5 where name = 'Divin';
savepoint B;

insert into class values (6, 'Sree');
Savepoint C;

ROLLBACK TO B;
```

Result

TCL commands successfully executed.

4/3/22

Connect
Username : temp
password : temppassword

```
create table table1 (name varchar(20));
insert into table1 values ('name1');
```

3/3/22

Experiment :-19

DCL

Aim

Practice SQL DCL commands for ~~getting~~ granting and revoking user privileges

GRANT : Allow user access privileges to the database
REVOKE : withdraw user access privileges given by using GRANT command

CODE

To create new user,

```
CREATE user temp IDENTIFIED BY temppassword;
GRANT create session to temp;
```

To grant all priveleges to temp:

```
GRANT ALL PRIVILEGES ON *.* TO temp;
```

To grant selective permissions,

```
GRANT CREATE TABLE to temp;
GRANT Select, insert, update on temp.table1 to temp;
```

To revoke privileges,

```
revoke select, insert, update on temp.table1 from temp;
```

Result

DCL commands executed successfully

AP
4/3/22

Experiment:-20

XML

Aim

Create an XML document that contains the markup tags such as <name>, <dept>, and <rno> and create an XSL document that defines the style to display an XML document and load the XML document in the browser.

Code

rule.xsl file

```
<?xml version: "1.0" encoding="UTF-8"?>
<xsl:stylesheet version: "1.0" xmlns:xsl="http://www.w3.org/1999
  /xsl/Transform">
<xsl:template match="/">
 <html>
    <body>
    <h1 align="center">Student Detail</h1>
    <table border="3" align="center">
     <tr>
          <th>Name</th>
          <th>Dept</th>
          <th>Rno</th>
   </tr>
        <xsl:for-each select="student/s">
```

```
<tr>
    <td> <xsl:value of select="name"/></td>
    <td> <xsl:value-of select="dept"/></td>
    <td> <xsl:value-of select="rno"/></td>
</tr>
    </xsl:for-each>
    </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

student.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="C:\Users\User\Desktop\
    XML\rule.xsl"?>
      <student>
       <s>
          <name> HARI </name>
          <dept> CSE </dept>
          <rno> 230 </rno>
       </s>
       <s>
          <name> DIVIN </name>
          <dept> IT </dept>
          <rno> 56 </rno>
       </s>
       </s>
```

OUTPUT:

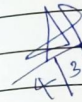
Student details

Name	Dept	Rno
HARI	CSE	230
DIVIN	IT	56
KAMAL	CSE	250

<name> KAMAL </name>
<dept> CSE </dept>
<rno> 250 </rno>
</s>
</student>

Result

xml and style sheet implemented successfully.



Certified

INDEX

Sl. No.	NAME OF EXPERIMENT	DATE	PAGE No.	INITIAL OF TEACHER
1	ER diagram	30-11-21	1-3	
2	Banking and Employee database	10-12-21	5-9	
3	Banking Database	10-12-21	11-19	
4	Employees Relational Schema Queries	14-12-21	21-27	
5	View University Database	14-12-21	29-33	
6	PL/SQL (Area of the Circle)	17-12-21	35-37	
7	PL/SQL (Bank Account Details)	17-12-21	39	
8	PL/SQL (Employee)	17-12-21	41-43	
9	PL/SQL (Procedure - Min, Avg and sum)	4-1-22	45-47	
10	PL/SQL (Procedure - Calculation of Salary)	4-1-22	49-51	
11	PL/SQL (Function - no: of tuples)	4-1-22	53	
12	PL/SQL (Exception Handling)	11-2-22	55	
13	PL/SQL (Exception Handling & Checking Conditions)	11-2-22	57-59	
14	PL/SQL (Trigger)	11-2-22	61	
15	PL/SQL (Package)	11-2-22	63-65	
16	NO SQL Restaurant Database	3-3-22	67	
17	JDBC	3-3-22	69-75	
18	SQL TCL Commands	3-3-22	77	
19	SQL DCL Commands	3-3-22	79	
20	XML	3-3-22	81-85	