SQL Practice - 1

1. To create a table Salesman

```
Create table Salesman (
        Salesman_id int primary key,
        Salesman_name Varchar (20),
        City Varchar,
        Commission float (10);
```

2. To create a table for customer

```
Create table customer (
        Customer_id int primary key,
        Customer_name Varchar (20),
        Customer_city Varchar (20),
        Customer_grade. int
        foreign key (Salesman id) references Salesman (Salesman_id);
```

. To create a table for orders

```
Create table order_num (
        order_no int primary key,
        pre_amt float (15),
        order_date date,
        Customer_id int,
        Salesman_id int,
```

Salesman_id	Salesman_name	City	Commission
5001	James Hoog	New York	0.15
5002	Nail Knite	Paris	0.13
5005	Pit Alex	London	0.11
5006	Mc Lyon	Paris	0.14
5007	Paul Adam	Rome	0.13

Customer_id	Name	City	Grade	Salesman_id
3002	Nick Rimando	New York	100	5001
3004	Fabian Johns	Paris	300	5006
3005	Zusi	California	200	5002
3007	Davis	New York	200	5001
3008	Green	London	300	5002

order_no	Purc_amt	order_date	Customer_id	Salesman_id
7001	150.5	2016-01-06	3002	5002
7002	270.65	2017-01-03	3009	5001
7005	240.6	2018-01-03	3001	5001
7007	938.5	2013-01-03	3002	5002
7008	1983.43	2019-01-03	3009	5002
70011	75.29	2013-07-03	3001	5002
70012	250.45	2015-09-03	3002	5002

4. Insert Values into Salesman

```
Insert into Salesman Values (5001, 'James Hoog', "New York", 0.5),
(5002, 'Nail Knite', 'Paris', 0.13),
(5005, 'Pit Alex', 'London', 0.11),
(5006, 'Mc Lyon', 'Paris', 0.14),
(5007, 'Paul Adam', 'Rome', 0.3),
```

5. Insert Values into customer value table

```
Insert into Customer Values (3002, 'Nick Rimando', 'New York',
                                                100, 3001)
(3004, 'Fabrian Johns', 'Paris', 300, 5005),
(3005, 'Graham Zusi', 'California', 200, 5002),
(3007, 'Brad Davis', 'New York', 200, 5001),
(3008, 'Julian Green', 'London', 300, 5002);
```

8. Insert Values into order table

```
Insert into order-num values (7002, 140.5, "2016-01-03", 3001, 5002),
(7003, 160.5, "2016-01-06", 3002, 5002),
(7004, 170.5, "2017-01-03", 3004, 5001),
(7005, 180.5, "2018-01-03", 3001, 5001),
(7006, 190.5, "2013-01-03", 3002, 5002),
(7007, 110.5, "2019-01-03", 3009, 5002),
(7008, 340.5, "2015-07-03", 3001, 5002),
(7009, 650.5, "2015-09-03", 3002, 5002);
```

Name	Commission
James Hoog	0.15
Nail Knite	0.13
Pit Alex	0.11
Mc Lyon	0.14
Paul Adam	0.13

Name	City
Nail Knite	Paris
Mc Lyon	Paris

Customer_id	Name	City	grade	Salesman_id
3007	Brad Davis	New York	200	5001
3005	Graham Zusi	California	200	5002
3003	Altidor	Moscow	200	5002

ord_no	ord_date	Purch_amt
7002	2012-10-05	65.26
7005	2012-07-27	2400.60
7008	2017-09-10	5760.00
70012	2012-04-25	3045.60

Winner:
Pabio Neruda.

Query 1 :- Display name and Commission for all the Salesman

=> Select Salesman_name, Commission from Salesman

Query 2 :- Retrive Salesman id of all Salesman from order table without any repeats

=> Select Distinct Salesman_id from orders

Query 3 :- Display names and City of Salesman, who belongs to the City of Paris

=) Select Salesman_name, city frome Salesman where City = 'Paris'

Query 4 :- Display all the information for those customer with a grade of 200.

=) Select * from customer where grade = 200;

Query 5 :-

=) Select ord_no, ord_date, Purch_amt From orders where Salesman_id = 5001;

Query 6 :-

=) Select winner from nobel_win where year = 1979 and Subject = 'Literature';

Output :-

emp - id	first - name	hourly pay	Job
1	x x x	51	manager
2	y y y	91	Cashier
3	z z z	81	Cook
4	e e e	290	teacher.

Expense - id	expense - name	Expense - Total
1	Salaries	$9856.00
2	Supplies	0.00
3	taxes	0.00

Trigger, Delimiter, Cursor

=> Create table employee (

emp-id int primary key,

first-name Varchar (20),

hourly-pay int,

Job Varchar (20));

=> Insert into employees values (1, 'xxx', 100, 'manager');

(2, 'yyy', 90, 'cashier'),

(3, 'zzz', 80, 'cook'),

(4, 'eee', 750, 'teacher');

=> Alter table employees

add column Salary decimal (10, 2) after hourly-pay;

Select * from employees;

=> update employees

Set Salary = hourly-pay * 2080;

Select * from employees;

=> Create trigger before-hourly-pay-update

before update on employees

for each row

Set new.salary = (new.hourly-pay * 2080);

Select * from employees;

=) Show trigger;

update employees

Set hourly-pay = 5;

where emp-id = 1;

=) update employees

Set hourly-pay = hourly-pay + 1;

=) Delete from employees

where emp-id = 4;

=) create trigger before-hourly-pay-interest

before insert on employees

for each row

Set new.Salary = (new.$hourly-pay * 2080);

=) insert into employees

Values (4, 'aaa', 290, null, 'teacher');

=) Select * from employees;

=) Create table expenses(

Expense-id int primary key,

expense-name varchar(50),

Expense-total decimal(10,2));

=) insert into expenses values (1, "Salaries", 0);

(2, "Supplier", 0);

(3, "taxes", 0);

=) update expenses

Set expense_total = (Select Sum (Salary) from employees)

where expense_name = 'Salaries';

=) create trigger after Salary_delete

after delete on employees

for each row

update expenses

Set expense_total = expense_total - old. Salary

where expense_name = 'Salaries';

=) delete from employees where emp_id = 3;

Delimiter & cursor :-

Create table customer (

id int not null,

Name Varchar (20) not null,

Age int not null,

~~dddddd~~ Address char (25)

Salary decimal (18, 2),

Primary key (20));

=) Insert into customer values (1, 'Ramesh', 32, 'Ahmedabad', 2000)

(2, 'kitan', 25, 'Delhi', 1500.00),

(3, 'kaushik', 23, 'kolkata', 2000.00),

(4, 'chaitali", 25, 'Mumbai', 6500.00);

=) Select * from customer;

=) Create table customer-Backup (

ID int not null,

Name varchar (20) not null,

Primary key (ID));

=) Create procedure Fetch customer()

Begin

Declare done int default False;

Declare customer-id int;

Declare customer-name varchar (255);

Declare auto-id Int;

-- Declare cursor

Declare my-cursor Cursor for

Select id, name from customer;

-- Declare exit handler

Declare continue handler for not found set done = True;

```
-- open cursor

open my_cursor;

-- Fetch and insert rows

read_loop: loop

    Fetch_my_cursor into customer_id, customer_name;

    if done = 1 than

    leave read_loop;

end if;
```

Output :-

ID	name	dept - name	Salary
10101	Srinivasan	Comp Sci	65000
12121	[illegible]	Finance	90000
15151	mozarad	Music	60000
22222	Einstein	Physics	95000
32343	El Said	History	60000
45565	Crolad	Physics	87000
58583	katy	Comp Sci	75000
76543	Califeri	History	62000

ID	Course_id	Sec_id	Semester	Year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2018
76766	BIO-301	1	Summer	2017
83821	CS-190	1	Spring	2018
83821	CS-319	1	Spring	2018
98345	EE-181	1	Spring	2018

Accessing the database

1. Create the following relation with primary Key integrity Constraint instructor

```
Create table instructor (
        ID, integer primary key,
        name text not null,
        dept text not null,
        Salary int);
```

-- Insert

```
Insert into instructor Values (10101, 'Srinivasan', 'Comp', 65000)
(12121, 'wee', 'Finance', 90000),
(15151, 'Mozart', 'Music', 40000),
(22222, 'Einstein', 'physics', 95000),
(32343, 'El Said', 'History', 60000),
(33456, 'Gold', 'physics', 87000),
(58583, 'Califieard', 'History', 62000),
(76543, 'Singh', 'Finance', 80000),
(76766, 'brick', 'Biology', 72000),
(83821, 'Brandt', 'Comp.Sci', 92000);
```

ID	Name	dept-name	Salary
32343	El Said	History	60000
58583	Califori	History	62000

Name	Course-ID
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
brick	CS-319
brick	BIO-101
Brandt	BIO-301
Kin	CS-319

Ib	name	dept-name	Salary
10101	Srinivasan	Comp-Sci	65000
22222	Einstein	Physics	95000
76543	Singh	Finance	80000

ID	name	dept-name	Salary
12121	Wu	Finance	90000
22222	Einstein	Physic	95000
83821	Brandt	Comp. Sci	92000

2. create the following Relation teacher

```
Create table Teacher (
    ID integer not null
    Course-ID varchar (20) not null,
    See-id Varchar not null,
    Semester char (50) not null,
    Year int not null,
    foreign key (ID) Reference instructor (ID));
-- Insert
Insert into teacher Values (10101, 'CS-101', 1, 'Fall', 2017),
(10101, 'CS-315', 1, 'Spring', 2018),
(10101, 'CS-347', 1, 'fall', 2017),
(12121, 'FIN-201', 1, 'Spring', 2018),
(15151, 'MU-199', 1, 'Spring', 2017),
(2222, 'PHY-101', 1, 'Fall', 2018),
(32343, 'HIS-351' 1, 'Spring', 2018),
(45565, 'CS-101', 1, 'Spring', 2018),
(45565, 'CS-315', 1, 'Spring' 2017),
(76766, 'BIO-101' 1, 'Summer', 2018),
(83821, 'BIO-301', 1, 'Summer', 2018);
```

Course - ID
CS - 101
CS - 347
PHY - 101

ID	name	dept_name	Salary
10212	Tom	Biology	NULL

Avg (Salary)
74153.8462

Count (*)
7

Count (*)
15

dept_name	Avg(Salary)
Comp. Sci	77333.3333
Biology	6819.6677
Finance	40000.0000
Physic	482500.0000
History	61000.0000
Elec. Eng	80000.0000

dept_name	Salary
Comp. Sci	71333.3333
Biology	6819.6667
Finance	40000.0000
Physics	482500.0000
History	61000.0000
Elec. Eng	80000.0000

Query 3:-

```
Insert into instructor values (10211, 'Smith', 'Biology', 66000);
```

Query 4:-

```
Delete into instructor where ID = '10211';
```

Query 5:-

```
Select * from instructor where dept = 'History';
```

Query 6:-

```
Select * from instructor cross join Teacher;
```

Query 7:-

```
Select name, Course-ID from instructor inner join Teacher
On Instructor-ID = Teacher-ID;
```

Query 8:-

```
Select * from instructor where name like "%. in%";
```

Query 9:-

```
Select * from instructor where Salary between 90000 and
100000;
```

Name
Srinivasan
Smith
Tom
wu
Elsaad
Gold
katz
Singh
brick
kim

Name
Wu
Einstein
Gold
Katz
Singh
brick
Brandt
Kim

dept_name
Physics

Experiment 2 :- Basic SQL

Query 1 :-

```
Select * from instructor order by Salary ASC;
```

Query 2 :-

```
Select Course-ID from teacher where (Semester = 'Fall' and
Year = "2017") or (Semester = 'Spring' and Year = '2018);
```

Query 3 :-

```
Select Course-ID from Teacher where (Semester = 'Fall'
and Year = '2017') and (Semester = 'Spring' and Year = '2018');
```

Query 4 :-

```
Select Course-ID from Teachers where (Semester = 'Fall' and
Year = '2017');
```

Query 5 :-

```
Insert into teacher Values ("10211", "Smith", "Biology", 60000),
(10212, "Tom", "Biology", Null);
```

Query 6 :-

```
Select * from instructor where Salary = 'Null';
```

Query 7 :-

```
Select AVG (salary) from instructor where Dept = 'Comp.sci';
```

Name	Course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	Fin-201
Mozart	Phy-101
Einstein	His-351
El Said	CS-101
Katz	CS-319
Brick	bio-101
Brick	bio-301
Brandit	CS-190
Brandit	CS-319
kim	ee-181

Id	Name	dept
10101	Srinivasan	CS
10211	Smith	Biology
10212	tom	Biology
10232	Adam	Music
10326	Wu	Finance
12121	Mozart	Music
15151	Einstein	physics
22222	El Said	History
32343	Gralel	physics
33456	Katz	Comp. Sci
45565	Singh	History
58583	Brick	

Exp 3: Intermediate SQL

1. Find the total no. of instructor who teach a course in the Spring 2018 Semester.

   Select Count (*) from instructor where (Semester = 'Spring' and year = '2018')

2. Find the no. of tuples in the teacher relation.

   Select Count (*) from instructor;

3. Find the average Salary of instructor in each dept.

   Select dept-name, avg (Salary) from instructor group-by dept-name;

4. Find the names and average Salaries of all depart, whose average Salary is greater than 42000.

   Select dept-name, avg (Salary) from instructor group by dept-name having avg (Salary) > 42000;

5. Name all instructor whose name is neither "Mozart" nor "Einstein"

   Select name from instructor where name not in ('Mozart', 'Einstein');

6. Find the names of instructor with Salary greater than Salary of instructor in Biology dept.

   Select n.name from instructor where n. Salary > one (Select Salary from instructor where dept-name = `Biology');

dept_name	Total_Salary
Comp Sci	232000
Biology	138000
Finance	170000
Music	40000
Physics	182000
History	122000
Elec. Eng	80000

7. Find the names of all instructor whose Salary is greater than the Salary of all instructor in Biology.

Select name from instructor where Salary > all (Select Salary from instructor where dept-name = 'Biology');

Exp 4:- Intermediate & Advanced SQL

1. Find all department where the total Salary is greater than the average of the total Salary at all dept.

Select dept-name from instructor group by dept-name having Sum (Salary) > (Select avg (total-Salary) from (Select Sum (Salary as total-Salary from instructor group by dept-name) as Subquery);

2. List the names of instructor along with Course ID of the Course that they taught.

Select instructor.name, teacher.Course-ID from instructor join teacher on instructor_id = teacher_id;

3. List the names of instructor along with the Course ID of the Courses that they taught. In case, an instructor teaches no Courses Keep the Course ID as null.

Select instructor.name, teacher.Course ID from instructor left join teacher on instructor.id = teacher.id;

4. Create a view of instructor without their Salary called faculty.

Create view faculty as Select id, name, dept from instructor;

5. Give Select privileges on the view faculty to new user.

Grant Select on faculty to new user.

Exp 5 :- Advanced SQL

1. Create a View of instructor without their salary called faculty.

Create view faculty as Select ID, name, dept_name from Instructor where Salary is NULL;

2. Create a view of department Salary tools

Create a view department_salary_tools as Select dept_name, Sum (salary) as total_Salary from instructor group by Dept_name;

Select * from department_salary_tools;

3. Create a role of Student

Create role Student;

4. Give Select privileges on the view faculty to the role Student.

Grant Select on faculty to Student;

5. Create a new user and assign the role of Student.

Create User new_user identified by 'password';

4. Create a View of instructor without their Salary called faculty.

Create View faculty as Select id, name, dept from instructor;

5. Give Select privileges on the View faculty to new user.

Grant Select on faculty to new user.

Exp 5 :- Advanced SQL

1. Create a View of instructor without their Salary called faculty.

Create view faculty as Select ID, name, dept-name from Instructor where Salary is NULL;

2. Create a view of department Salary tools

Create a View department-Salary-tools as Select dept-name, Sum (salary) as total-Salary from instructor group by Dept-name;

Select * from department-Salary-tools;

3. Create a role of Student

Create role Student;

4. Give Select privileges on the View faculty to the role Student.

Grant Select on faculty to Student;

5, Create a new user and assign the role of Student.

Create User. new-user, identified by 'Password';

6. Login as the new user and find all instructor in the Biology department.

```
Select name from instructor where dept_name = 'Biology';
```

7. Revoke privileges to new user

```
Revoke Student from new user;
```

8. Remove the role of Student.

```
Drop Role Student;
```

9. Give Select privileges on the view faculty to the new user.

```
Grant Select on faculty to new_user;
```

10. Login as the new user and find all instructor in the finance dept.

```
Select * from instructor where dept_name = 'Finance';
```

11. Login again as root_user.

```
Select Host, user, Plugin from mysql.user;
```

12. Create table Teaches 2 with same column as teaches but with additional constraint that the Semester in one of fall, winter, Summer (or) Spring

```
Select * from Teaches 2;
```

13. Create index teaches_ID_index on Teaches (ID);

Exp 6:- Accessing Database through python.

1. Insert following additional tuple in instructor:

(10211, 'Smith', 'Biology', 66000)

```
mycursor : mydb. cursor()
mycursor . execute ("Insert into instructor (ID, name, dept_name,
Salary). values (%s, %s, %s, %s)" val = (10211, 'Smith',
'Biology', 66000);
For i in my cursor:
    print (i);
```

2. Delete this tuple from instructor

```
mycursor. execute ("Delete from instructor whose ID = "10214");
```

3. Select tuple from instructor where dept_name = `History':

```
mycursor. execute ("Select * from instructor where dept_name
= `History');
```

4. Find the cartesian product instructor x teaches.

```
mycursor. execute ("Select * from instructor Cross join
Teaches);
```

5. Find the name of all instructor who have taught Some cursors and the Course_id

```
mycursor. execute ("Select name, Course_id from instructor inner
Join Teaches on instructor. id = Teaches_id);
```

6. Find the name of all instructor whose name includes the Sub String.

```
Select * from instructor where name like "Sama";
```

Exp 7:- Advanced techniques through python.

1. order the tuples in the instructor relation as per their Salary.

```
mycursor.execute ("Select * from teaches order by Salary');
```

2. Find the courses that ran in Fall 2017 or in Spring 2018.

```
mycursor.execute ("Select course_id from teaches where (Semester = 'fall' and year = '2017') or (Semester = 'Spring' and year = '2018');
```

3. Find Course that ran in Fall 2017 and in Spring 2018.

```
mycursor.execute ("Select Course_id from teaches where (Semester = 'fall' and year = '2017') and (Semester = 'Spring' and year = '2018')");
```

4. Find Courses that ran in fall 2017 but not in Spring 2018.

```
mycursor.execute ("Select Course_id from teaches where (Semester = 'Spring' and year = '2018')");
```

5. Insert following additional tuples in instructor

```
mycursor.execute ("Insert into instructor Values (10211, 'Smith', 'Biology', 66000), (10212, 'Tom', 'Biology', 'null')");
```

6. Find all instructor whose Salary in null.

```
mycursor.execute ("Select * from instructor where Salary is null");
```

8. Find the total no. of instructor who teach a course in the Spring 2018 Semester.

mycursor. execute ("Select Count (*) from instructor where (Semester = 'Spring' and year = '2018')");

9. Find the no. of tuples in the teaches relation.

mycursor. execute ("Select Count (*) from instructor");

10. Find the average Salary of instructor in each dept.

mycursor. execute ("Select dept-name, avg (Salary) from instructor group by dept-name");

11. Find the names and average Salaries of all dept whose average Salary is greater than 42000.

mycursor. execute ("Select dept-name, avg (Salary) from instructor group by dept-name having avg (Salary) >42000);

12. Name all instructor whose name is either Mozart nor Einstein.

mycursor. execute ("Select name from instructor where name not in ("Mozart", 'Einstein");

5. mycursor. execute ("Select n.name from instructor n where n.Salary > avg (Select Salary from details where dept-name = 'biology')");

15. mycursor.execute ("Select avg (Salary) from instructor group by dept_name having avg (Salary) > 42000");

16. mycursor.execute ("Select instructor.name, Teaches.Course_id from instructor join teaches on instructor.id = Teaches.id");

17. mycursor.execute ("Select instructor.name, Teaches.Course_id from instructor left join Teaches on instructor.id = Teaches.id);

Object oriented programming - oracle

1. Create a type addr_ty

```
Create type addr_ty as object
    ( Street  Varchar (60),
      City  Varchar (30),
      State  char (3),
      Zip  Varchar (9));
```

2. Create a type person_ty

```
Create type person_ty as object
    (name  Varchar (25),
     address  addr_ty);
```

3. Create a type emp_ty

```
Create type emp_ty as object
    (empt_id  Varchar (9),
     person  person_ty );
```

4. Create table emp_oo;

```
Create table emp_oo
(full_emp  emp_ty);
```

5. Enter values:

```
Insert into emp_oo values (emp_ty ('100', person_ty
    ('Ram', addr_ty ('10001o', 'patiala', 'PB', '147001'))));
```

Output :-

```
Full - emp (emp_id , person (Name , Address (Street , City ,

Emp_ty ('100' , person_ty ('Ram' , Addr_ty ('1000TU' , 'Patiala' , 'PB' , '147001' ,

Emp_ty ('101' , Person_ty ('Sham' , Addr_ty ('10001TU' , 'Patiala' , 'PB' , '147001' ;
```

Name	Null ?	Type
Full - emp		Emp - ty

ID	Name	City
101	Ram	Patiala
101	Sham	Patiala

ID	Name	City
100	Ray	Patiala
101	Sham	Patiala

Insert into emp-oo values

(emp-ty ('101',

person-ty ('Sham',

addr-ty ('1001 TU', 'patiala', 'PB', '147001')));

6. Select * from emp-oo;

7. desc emp-oo;

8. Select e.full-emp. empt-id ID,

e.full-emp. person.name NAME,

e.full-emp. person.address.city CITY

from emp-oo e;

9. update

update emp-oo e Set

e.full-emp. person-name = 'Ajay'

where

e.full-emp. empt-id = '100';

10. Select e.full-emp. empt-id ID,

e.full-emp. person-name. NAME,

e.full-emp. person. address. city CITY

From emp-oo e;

11. Create (or) replace type newemp-ty as object (

firstname varchar (25),

lastname varchar (25),

birthdate date,

Employee.Firstname	E.Employee.Age (employee.Rd)
Ram	12152
Shan	10721

Name	Null	Type
Empt-id		Varchar(9)
Person		Person-ty

Emp-id	Person (Name, Address (Street, city)
100	Person-ty ('Ray', addr-ty ('100 TU', 'patala');

Ref(p)
0000 2802 0994 7142 B7FF 3477 6AAB A755 D64F 64FD A512

member function age [Birthdate in DATE]
return number)

12. Create or replace type body newemp-ty as
member function age (Birthdate in Date) retar number
begin

Return Round (SysDate - Birthdate);
end;
end;

13. Create table new-emp-oo
(employee newemp-ty);

4. insert into new-emp-oo values
(newemp-ty ('Ram', 'Lal', '12-dec-1976'));

5. How to call the function age ()
Select e.employee.firstname, e.employee.age
(e.employee.birthdate) from
new_emp-oo e;

- Creation of object table
Create table new-empl of emp-ty;

∴ Create type emp-ty as object
(empt-id varchar (9),
person person_ty);

Name	Null ?	Type
Emp no		Number (3)
Name		Varchar 2 (10)
Dept		Ref of new_dept_one

Emp no	Name	Deref (E.dept) (Dep no, Dname)
100	Raj	New-dept-00 (10, 'Comp')
101	Sham	New-dept-00 (20, 'Chem')

Emp no	Name	Deref (E.dept) (Dep no, Dname)	Dep no	Dname
100	Raj	New-dept-00 (10, 'Comp')	10	Comp
101	Sham	New-dept-00 (20, 'Chem')	20	Chem

18. Create table emp-oo
(full-emp. emp-ty);

19. Insert into emp-oo values
(emp-ty ('100',
person-ty ('Ram',
addr-ty('1000 TU', 'patiala', 'PB', '147001')))));

20. Insert into new-emp1 values ('100', person-ty ('Ray',
addr-ty ('100 TU', 'PTA', 'PB', '147001'))));

21. Select * from new-emp1;
Select ref (P) from new-emp1 P;

22. Implementing the concept of fk
Create type new-dept-oo as object
(deptno number (3), dname varchar (20);
Create table dept-table of new-dept-oo;

23. insert into dept-table values (10, 'comp');
insert into dept-table values (20, 'chem');
insert into dept-table values (30, 'math');

24. Select ref (P) from dept-table P;

25. Create table emp-test-fk (
empno number (3),
name varchar (10),
dept. ref new-dept-oo);

26. desc emp - test - fk

Set desc depth

desc emp - test - fk

27. insert into emp - test - fk

Select 100, 'ram', ref (p) from dept - table, where

depno = 10;

28. insert into emp - test - fk

Select 101, 'Sham', ref (p) from dept - table p where

depno = 20;

29. Accessing the values :-

Select * from emp - test - fk;

Select empno, name, deref (e.dept) from emp - test . fk;

Select empno, name, deref (e.dept, deref (e.dept),

depno depno (e.dept). dname

dname from emp - test - fk e;

30. Create table emp - table - fk

(employee emp - ty,

dept ref new - dept - 00);

31. Set descrete depth : 1

desc emp - table - fk

Set descrete depth 2

desc - emp - table - fk

Set descrete depth 3
desc emp-table-fk

32. Set describe depth 3
desc emp-table-fk
Set describe depth 4
desc emp-table-fk

33. insert into emp-table-fk values (
emp-ty (100, person-ty ('ram', addr-ty ('10tu', 'pat',
(Select ref(p)
from dept-table p
where deptno = 10)));

Select * from emp-table-fk;

Select e.employee.empt-id id, e.employee.person.name Name
deref(e.dept).deref(e.dept).deptno deptno,
deref(e.dept).dname from emp-table-fk e;

04/06/2024