Name – Parth Medhekar

Roll No-44 Div – A(A3)

Experiment No 2 – Implement Inheritance in ORDBMS For Given Dataset

1. Create type name_ty with following attributes:

●Fname varchar

●Lname varchar

```
create or replace type name_ty as object
(
fname varchar(20),
lname varchar(20)
)not final;
```

2. Create type addr_ty with following attributes:

●City varchar

●Pincode number

```
create or replace type addr_ty as object
(
city varchar(20),
pincode int
)not final;
```

3. Create type employee with following attributes:

●emp_id number

●name name_ty

●address addr_ty

```
create or replace type emp_ty as object
(
emp_id number,
```

name name_ty,

address addr_ty

)not final

4. Create type fulltime_emp under type employee with following attribute: ●Salary number

create or replace type fulltime_emp under emp_ty

(

salary number

);

5. Create type parttime_emp under type employee with following attributes:

●Rate number

● Hours number

create or replace type parttime_emp under emp_ty

(

rate number,

hours number

);

6. Create table Fulltime of type fulltime_emp

create table fulltime of fulltime_emp;

7. Create table Parttime of type parttime_emp.

create table parttime of parttime_emp;

8. Insert the following data to Fulltime table:

insert into fulltime values(1,name_ty('Rahul','Kumar'),addr_ty('pune',411234),50000)

insert into fulltime values(2,name_ty('Aniket','Sharma'),addr_ty('Kop',410123),70000)

insert into fulltime values(3,name_ty('Abhi','Verma'),addr_ty('Kop',410124),40000)

insert into fulltime values(4,name_ty('Rohan','Kumar'),addr_ty('mumbai',416605),60000)

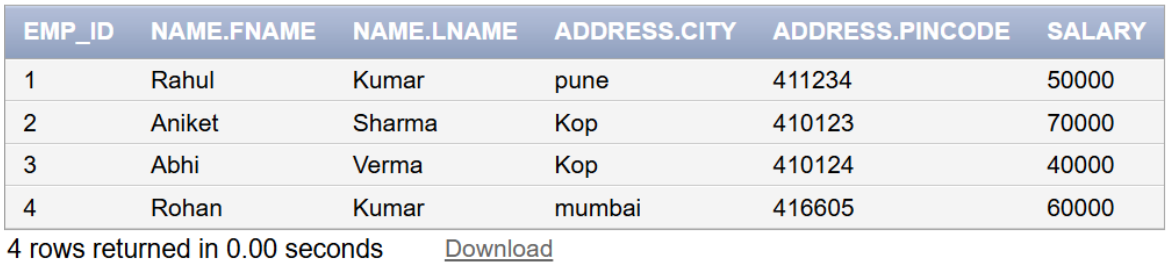
9. Insert the following data to Parttime table:

insert into parttime values(5,name_ty('vibhuti','Mitra'),addr_ty('Sangli',410298),1000,8)

```
insert into parttime values(6,name_ty('Kteaki','Bhave'),addr_ty('Kop',410222),500,7)

insert into parttime values(7,name_ty('mahesh','Kumbhar'),addr_ty('pune',416289),2000,5)

insert into parttime values(8,name_ty('raj','patil'),addr_ty('pune',409256),800,4)
```

10. Print the record of all Fulltime employees.

```
select f.emp_id,f.name.fname,f.name.lname,f.address.city,f.address.pincode,f.salary from
fulltime f
```

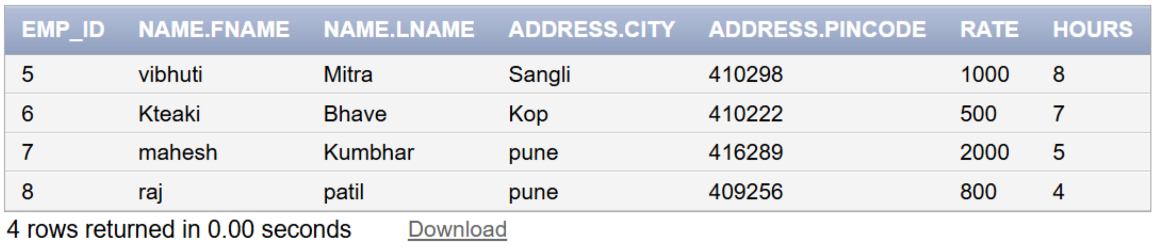


EMP_ID	NAME.FNAME	NAME.LNAME	ADDRESS.CITY	ADDRESS.PINCODE	SALARY
1	Rahul	Kumar	pune	411234	50000
2	Aniket	Sharma	Kop	410123	70000
3	Abhi	Verma	Kop	410124	40000
4	Rohan	Kumar	mumbai	416605	60000

4 rows returned in 0.00 seconds Download

11. Print the record of all Parttime employees.

```
select p.emp_id,p.name.fname,p.name.lname,p.address.city,p.address.pincode,p.rate,p.hours
from parttime p
```

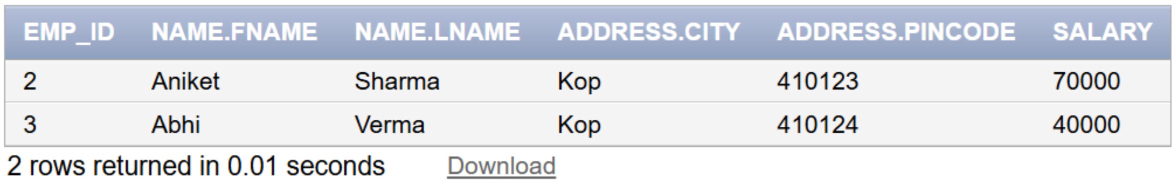


EMP_ID	NAME.FNAME	NAME.LNAME	ADDRESS.CITY	ADDRESS.PINCODE	RATE	HOURS
5	vibhuti	Mitra	Sangli	410298	1000	8
6	Kteaki	Bhave	Kop	410222	500	7
7	mahesh	Kumbhar	pune	416289	2000	5
8	raj	patil	pune	409256	800	4

4 rows returned in 0.00 seconds Download

12. Retrieve the record of all fulltime employees staying in Kolhapur.

```
select f.emp_id,f.name.fname,f.name.lname,f.address.city,f.address.pincode,f.salary from
fulltime f where f.address.city='Kop'
```

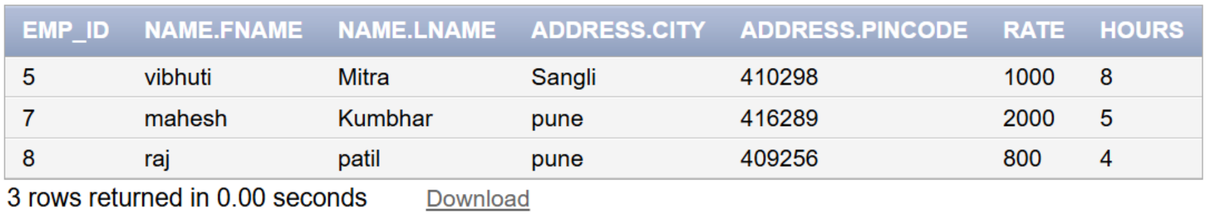


EMP_ID	NAME.FNAME	NAME.LNAME	ADDRESS.CITY	ADDRESS.PINCODE	SALARY
2	Aniket	Sharma	Kop	410123	70000
3	Abhi	Verma	Kop	410124	40000

2 rows returned in 0.01 seconds Download

13. Find all the employees whose rate is more than 500.

```
select p.emp_id,p.name.fname,p.name.lname,p.address.city,p.address.pincode,p.rate,p.hours
from parttime p where rate > 500
```

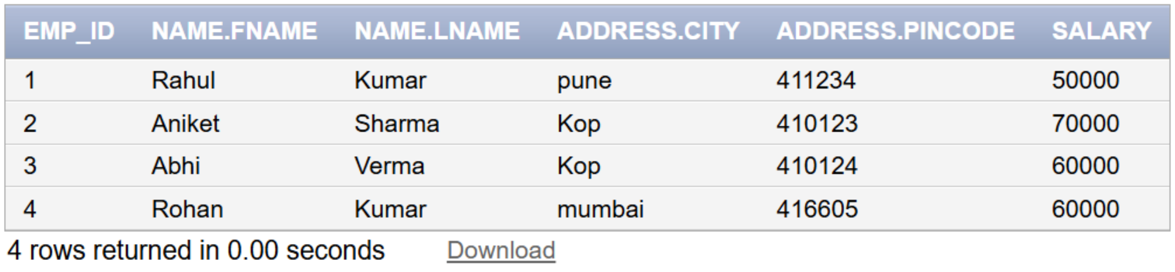


EMP_ID	NAME.FNAME	NAME.LNAME	ADDRESS.CITY	ADDRESS.PINCODE	RATE	HOURS
5	vibhuti	Mitra	Sangli	410298	1000	8
7	mahesh	Kumbhar	pune	416289	2000	5
8	raj	patil	pune	409256	800	4

3 rows returned in 0.00 seconds Download

14. Change the salary of employee to 60000 whose current salary is 40000.

update fulltime f set salary=60000 where f.salary=40000

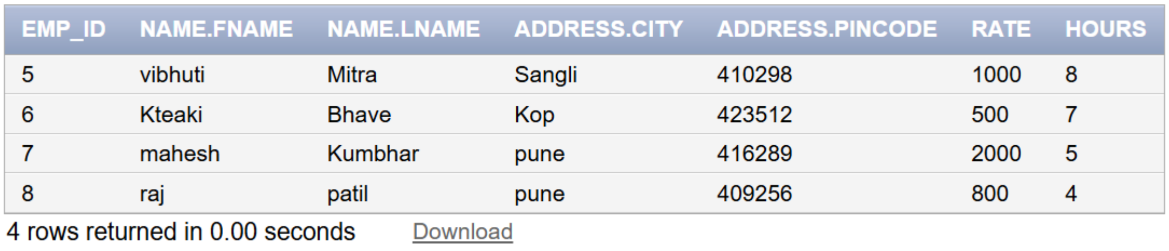


EMP_ID	NAME.FNAME	NAME.LNAME	ADDRESS.CITY	ADDRESS.PINCODE	SALARY
1	Rahul	Kumar	pune	411234	50000
2	Aniket	Sharma	Kop	410123	70000
3	Abhi	Verma	Kop	410124	60000
4	Rohan	Kumar	mumbai	416605	60000

4 rows returned in 0.00 seconds Download

15. Change the pincode of employee to 423512 whose hours are 7.

update parttime p set p.address.pincode=423512 where p.hours=7

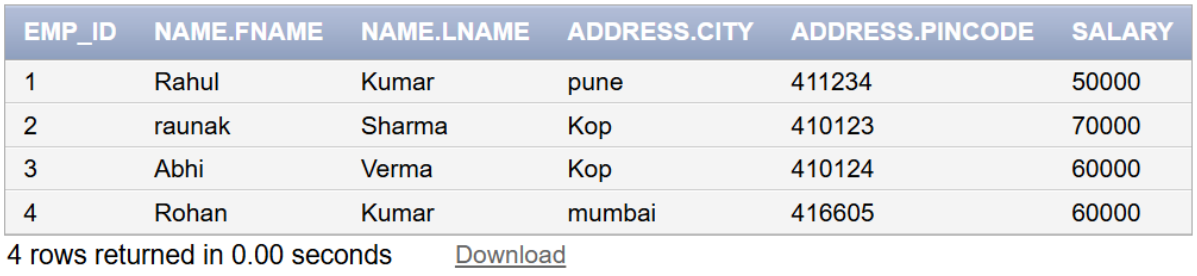


EMP_ID	NAME.FNAME	NAME.LNAME	ADDRESS.CITY	ADDRESS.PINCODE	RATE	HOURS
5	vibhuti	Mitra	Sangli	410298	1000	8
6	Kteaki	Bhave	Kop	423512	500	7
7	mahesh	Kumbhar	pune	416289	2000	5
8	raj	patil	pune	409256	800	4

4 rows returned in 0.00 seconds Download

16. Change the first name of fulltime employee to raunak whose last name is Sharma.

update fulltime f set f.name.fname='raunak' where f.name.lname='Sharma'

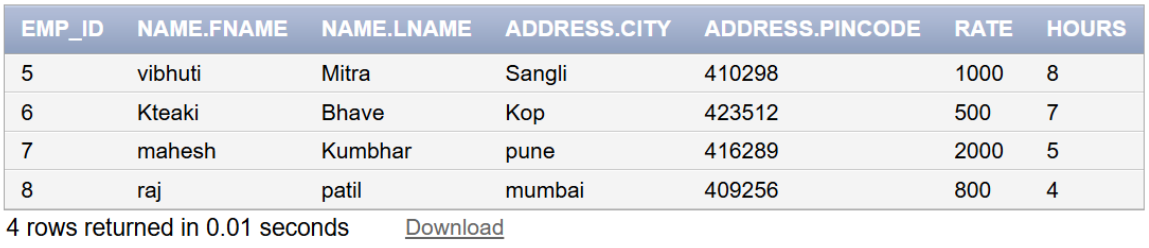


EMP_ID	NAME.FNAME	NAME.LNAME	ADDRESS.CITY	ADDRESS.PINCODE	SALARY
1	Rahul	Kumar	pune	411234	50000
2	raunak	Sharma	Kop	410123	70000
3	Abhi	Verma	Kop	410124	60000
4	Rohan	Kumar	mumbai	416605	60000

4 rows returned in 0.00 seconds Download

17. Change the city of employee to Mumbai whose rate is 800.

update parttime p set p.address.city='mumbai' where p.rate=800

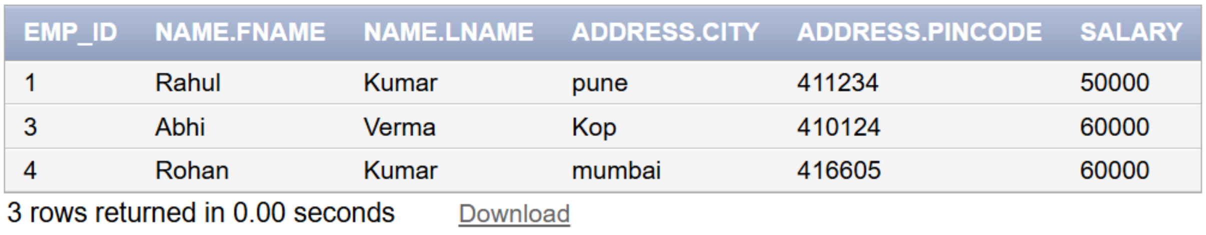


EMP_ID	NAME.FNAME	NAME.LNAME	ADDRESS.CITY	ADDRESS.PINCODE	RATE	HOURS
5	vibhuti	Mitra	Sangli	410298	1000	8
6	Kteaki	Bhave	Kop	423512	500	7
7	mahesh	Kumbhar	pune	416289	2000	5
8	raj	patil	mumbai	409256	800	4

4 rows returned in 0.01 seconds Download

18. Delete the record of fulltime employee with pincode 410123.

delete from fulltime f where f.address.pincode=410123

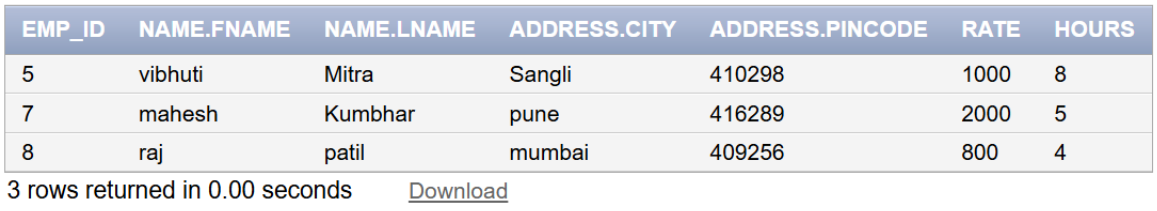


EMP_ID	NAME.FNAME	NAME.LNAME	ADDRESS.CITY	ADDRESS.PINCODE	SALARY
1	Rahul	Kumar	pune	411234	50000
3	Abhi	Verma	Kop	410124	60000
4	Rohan	Kumar	mumbai	416605	60000

3 rows returned in 0.00 seconds Download

19. Delete the record of parttime employee whose last name is bhave.

delete from parttime p where p.name.lname='Bhave'



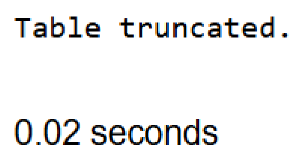
EMP_ID	NAME.FNAME	NAME.LNAME	ADDRESS.CITY	ADDRESS.PINCODE	RATE	HOURS
5	vibhuti	Mitra	Sangli	410298	1000	8
7	mahesh	Kumbhar	pune	416289	2000	5
8	raj	patil	mumbai	409256	800	4

3 rows returned in 0.00 seconds Download

20. Delete all the records of fulltime and parttime employees.

truncate table fulltime;

truncate table parttime;



```
Table truncated.

0.02 seconds
```

Experiment 3: Implement procedures, functions and cursors in PL/SQL

A. Implement Procedures in PL/SQL.

1. Create a schema level procedure to display a simple message “Hello”. Call the procedure by passing appropriate arguments.

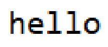
Query -
```
create or replace procedure display as
        begin
                dbms_output.put_line('hello');
        end;
```

Output - **Procedure Created**

2. Create a block level procedure to display a simple message “Hello”.

Query -
```
declare procedure display is
begin
        dbms_output.put_line('hello');
end;
begin
        display;
end;
```

Output –



```
hello
```

3. Create a procedure to find square of a number using two different modes of parameter passing.

a. IN , OUT mode

Query -
```
declare c number;
   procedure square(x in int, y out int) is
   begin
      y := x * x;
      dbms_output.put_line(y);
   end;
begin square(10, c); end;
```

Output -

```
100
```

b. IN OUT mode.

Query - declare
```
    num number := 10;
    procedure square(x in out number) is
    begin
        x := x * x;
        dbms_output.put_line(x);
    end;
begin square(num);  end;
```

Output –

```
100
```

4. Create table Student with attributes roll_no, name, address, contact_no.

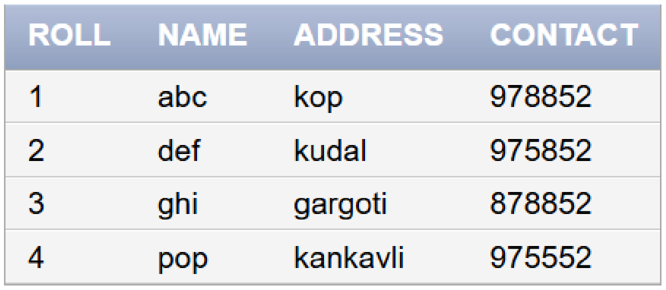
Query - create table student(roll int, name varchar(20), address varchar(20), contact int)

Output – **Table Created**

5. Create a schema level procedure to insert values in Customer table. Call the procedure and insert 4 rows in the table. Print the table using SQL statement.

Query - create or replace procedure insertdata(sroll student.roll%type, sname student.name%type, sadd student.address%type, sphone student.contact%type) as
```
begin
            insert into student values(sroll, sname, sadd, sphone);
end;
begin insertdata(1, 'abc', 'kop', 978852);
          insertdata(2, 'def', 'kudal', 975852);
            insertdata(3, 'ghi', 'gargoti', 878852);
          insertdata(4, 'pop', 'kankavli', 975552);
end;
select * from student;
```

Output - **Procedure Created**



ROLL	NAME	ADDRESS	CONTACT
1	abc	kop	978852
2	def	kudal	975852
3	ghi	gargoti	878852
4	pop	kankavli	975552

6. Create a block level procedure to find name of the student if roll_no and address is given.

Call the procedure by passing appropriate arguments.

Query -
```
declare sname student.name%type;
procedure find(sroll student.roll%type, sadd student.address%type) is
begin
        select name into sname from student where roll = sroll and address = sadd;
        dbms_output.put_line(sname);
end;
begin find(1, 'kop'); end;
```
Output -

```
abc
```

7. Create a schema level procedure to update contact_no of student if roll_no is given. Call

the procedure by passing appropriate arguments.

Query -
```
create or replace procedure updatedata(sroll student.roll%type, sphone
student.contact%type) as
begin
        update student set contact = sphone where roll = sroll;
end;

begin updatedata(1, 888888); end;
```

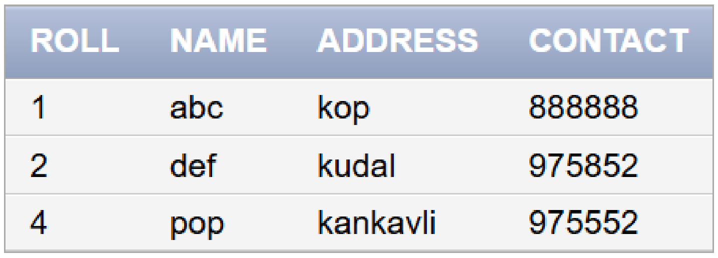
Output - **Procedure Created**
Statement Processed

8. Create a block level procedure to delete a student record if roll_no and name is given.

Call the procedure by passing appropriate arguments.

Query -
```
declare procedure removedata(sroll student.roll%type, sname student.name%type) is
begin
        delete from student where roll = sroll and name = sname;
end;
begin removedata(3, 'ghi'); end;
```

Output - 1 rows(s) deleted



ROLL	NAME	ADDRESS	CONTACT
1	abc	kop	888888
2	def	kudal	975852
4	pop	kankavli	975552

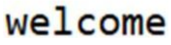
B. Implement Functions in PL/SQL.

1. Create a schema level function to display a message and call the function.

Query -

```
create or replace function msg return varchar as
begin
return 'welcome'; end;
declare y varchar(20);

begin
y := msg; dbms_output.put_line(y); end;
```

Output –



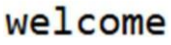
```
welcome
```

2. Create a block level function to display a message.

Query -

```
declare y varchar(20);
    function mssg return varchar is begin
    return 'welcome'; end;
    begin
y:= mssg; dbms_output.put_line(y); end;
```

Output –



```
welcome
```

3. Create table customer with attributes cust_id, first_name, last_name, city.

Query - create table cust(cust_id int, fname varchar(20), lname varchar(20), city varchar(20))

Output – **Table Created**

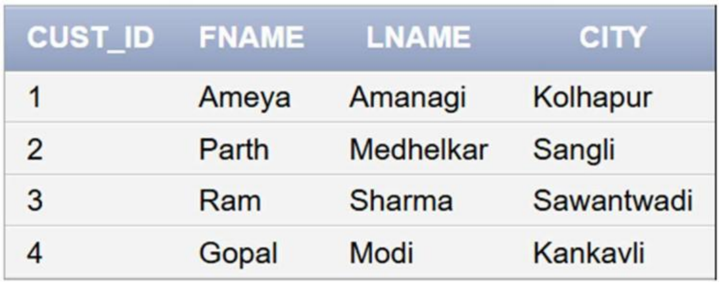
4. Create a block level function to insert values in customer table. Insert 4 rows in the table

and print the table using SQL statement.

Query -

```
declare a int; b int; c int; d int;
    function insertdata(cid cust.cust_id%type, cfname cust.fname%type,clname
    cust.lname%type, ccity cust.city%type) return int is
    begin insert into cust values(cid, cfname, clname, ccity); return 1; end;
    begin a:= insertdata(1, 'Ameya', 'Amanagi', 'Kolhapur');
    b:= insertdata(2, 'Parth', 'Medhelkar', 'Sangli'); c:= insertdata(3, 'Ram', 'Sharma',
'Sawantwadi');
d:= insertdata(4, 'Gopal', 'Modi', 'Kankavli'); end;

select * from cust;
```

Output – **1 row inserted**

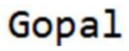


CUST_ID	FNAME	LNAME	CITY
1	Ameya	Amanagi	Kolhapur
2	Parth	Medhelkar	Sangli
3	Ram	Sharma	Sawantwadi
4	Gopal	Modi	Kankavli

5. Create a schema level function to find all the customers whose first name contains a specific letter. Call the function by passing appropriate arguments.

Query -
```
declare a varchar(20); function find return varchar is begin
   select fname  into a from cust where fname like '%G%'; return a;
   end; begin
a := find(); dbms_output.put_line(a); end;
```

Output –

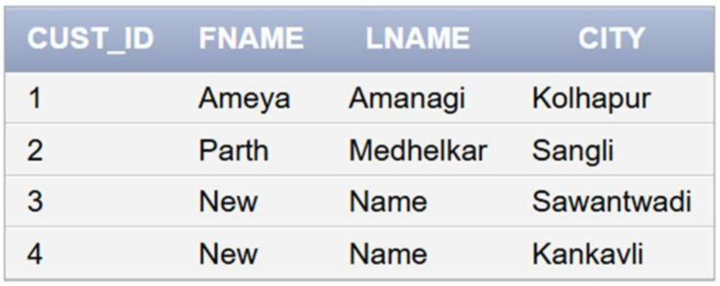


```
Gopal
```

6. Create a block level function to update first name and last name of the customer where a group of 3 cities are mentioned.

Query -
```
declare y varchar(20);
          function updated(c1 cust.city%type, c2 cust.city%type, c3 cust.city%type, newfname
          cust.fname%type, newlname cust.lname%type)
          return varchar is begin
          update cust set fname = newfname where city
          = c1 or city = c2 or city = c3;
          update cust set lname = newlname where city
          = c1 or city = c2 or city = c3; return 'done';
          end; begin
          y := updated('Kudal', 'Kankavli', 'Sawantwadi', 'New', 'Name');
          end;
```

Output –

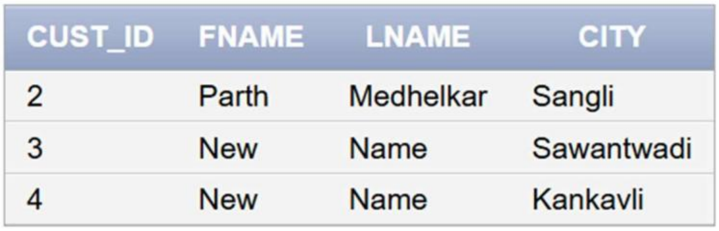


CUST_ID	FNAME	LNAME	CITY
1	Ameya	Amanagi	Kolhapur
2	Parth	Medhelkar	Sangli
3	New	Name	Sawantwadi
4	New	Name	Kankavli

7. Create a schema level function to delete a customer based on cust_id.

Query - create or replace function del(cid cust.cust_id%type) return number as begin
```
 delete from cust where cust_id = cid; return 1;
 end;
declare
x number; begin x:=del(1);
end;
```

Output –



CUST_ID	FNAME	LNAME	CITY
2	Parth	Medhelkar	Sangli
3	New	Name	Sawantwadi
4	New	Name	Kankavli

C. Implement Cursors in PL/SQL.

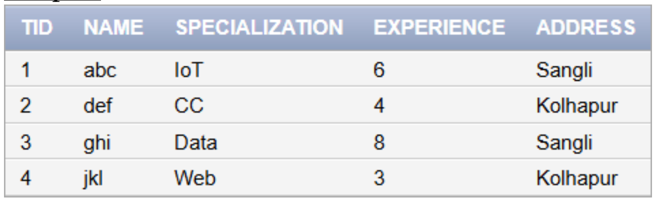
1. Create table teacher with attributes tid, name, specialization, experience and address.

Query - create table teacher(tid number, name varchar(20), specialization varchar(20), experience number, address varchar(20))

Output – **Table Created**

2. Insert 4 records in the table teacher.

Query -
```
insert into teacher values(1,'abc','IoT',6,'Sangli')
insert into teacher values(2,'def','CC',4,'Kolhapur')
insert into teacher values(3,'ghi','Data',8,'Sangli')
insert into teacher values(4,'jkl','Web',3,'Kolhapur')
```
Output –



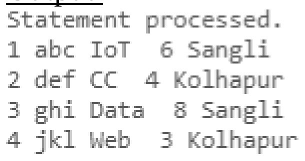
TID	NAME	SPECIALIZATION	EXPERIENCE	ADDRESS
1	abc	IoT	6	Sangli
2	def	CC	4	Kolhapur
3	ghi	Data	8	Sangli
4	jkl	Web	3	Kolhapur

3. Create a cursor to print all the values from teacher table.

Query -

```
declare
id teacher.tid%type;
tname teacher.name%type;
tspec teacher.specialization%type; texp teacher.experience%type; taddr
teacher.address%type; cursor teach is select tid, name,
specialization, experience, address from teacher;

begin
   open teach; loop
   fetch teach into id, tname, tspec, texp, taddr;
   exit when teach%notfound; dbms_output.put_line(id || tname || tspec || texp|| taddr);
end loop; close teach; end;
```

Output –



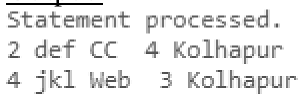
```
Statement processed.
1 abc IoT  6 Sangli
2 def CC  4 Kolhapur
3 ghi Data  8 Sangli
4 jkl Web  3 Kolhapur
```

4. Display information of all the teachers who are staying in Kolhapur.

Query -

```
   declare
   id teacher.tid%type;
   tname teacher.name%type;
   tspec teacher.specialization%type; texp teacher.experience%type; taddr
   teacher.address%type; cursor kol is select tid, name,
   specialization, experience, address from teacher where address= 'kolhapur' ;
   begin open kol; loop
   fetch kol into id, tname, tspec, texp, taddr; exit when kol%notfound; dbms_output.put_line(id ||
   ' ' ||tname || ' ' || tspec || ' '  || ' ' || texp || ' ' || taddr);
   end loop; close kol; end;
```

Output –



```
Statement processed.
2 def CC  4 Kolhapur
4 jkl Web  3 Kolhapur
```

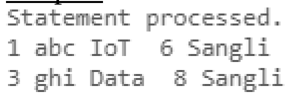
5. Display information of all the teachers whose experience is more than 5 years.

Query -

```
   declare
   id teacher.tid%type;
   tname teacher.name%type;
   tspec teacher.specialization%type; texp teacher.experience%type; taddr
   teacher.address%type; cursor exp is select tid, name,
   specialization, experience, address from teacher where experience>5 ;
```

```
begin open exp; loop
fetch exp into id, tname, tspec, texp, taddr; exit when exp%notfound; dbms_output.put_line(id
|| ' ' ||tname || ' ' || tspec || ' '  || ' ' || texp || ' ' || taddr);
end loop; close exp; end;
```

Output –



```
Statement processed.
1 abc IoT  6 Sangli
3 ghi Data  8 Sangli
```

Name: Parth Medhekar TY CSE

Div: A

Roll no: A44

Experiment No. 4: Implement Synonyms, Sequences, Triggers and Packages in PL/SQL

1. Create a table student with attributes id, roll_no, name, address,

Query **-** create table student(id int,roll int,name varchar(20), address varchar(20), contact int

Output – **Table Created**

2. Create a sequence to generate 'id' of a student automatically.

Query **-**
```
create sequence idseq
start with 1
increment by 1
```
Output – **Sequence Created**

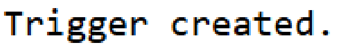
3. Insert following values in 'Student' table.

Query **-**
```
insert into Student values(idseq.nextval, 1, 'Ravi', 'Mumbai',9456723450);
insert into Student values(idseq.nextval, 2, 'Tina', 'Pune',8736492301);
insert into Student values(idseq.nextval, 3, 'Raj', 'Kolhapur',7829034658);
insert into Student values(idseq.nextval, 4, 'Madhuri', 'Sangli',9959310832);
```

Output – **1 row(s) inserted**

4. Create a trigger to prompt an error message when value entered for roll number is 0.

Query **-**
```
create or replace trigger sample
before insert on studentfor each row
when(new.roll <=0)
begin raise_application_error(-2,'Invalid input');
end;
```
Output –

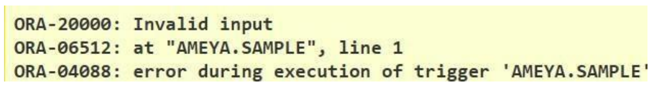


```
Trigger created.
```

5. Instantiate the created trigger by passing roll number of a student as 0.

Query **-** insert into Student values(idseq.nextval, 0, 'Madhur', 'Goa',9959310832);

Output –



```
ORA-20000: Invalid input
ORA-06512: at "AMEYA.SAMPLE", line 1
ORA-04088: error during execution of trigger 'AMEYA.SAMPLE'
```

6. Create a synonym ‘Stud’ for ‘Student’ table.

Query - create or replace synonym stud for student

Output –

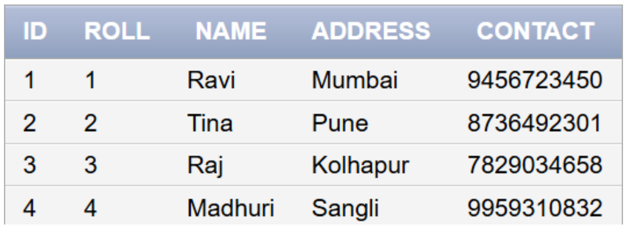
```
Synonym created.
```

7. Print the table ‘Student’ and ‘Stud’.

Query - select * from student

Select * from stud

Output –



ID	ROLL	NAME	ADDRESS	CONTACT
1	1	Ravi	Mumbai	9456723450
2	2	Tina	Pune	8736492301
3	3	Raj	Kolhapur	7829034658
4	4	Madhuri	Sangli	9959310832

8. . Create a package with following procedures:
a. Create a procedure to find name of the student if roll number is given

Query - create or replace package pack as sname varchar(20);
procedure find(sroll student.roll%type);
end pack;

create or replace package body pack as procedure find(sroll student.roll%type) is begin
select name into sname from student where roll = sroll;
end find;
end pack;

Output –

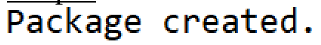
```
Package created.

Package Body created.
```

b. Create a function to delete a student record if roll number is given.

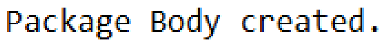
Query -create or replace package discard as function delete(sroll Student.roll%type)
return int;
end discard;

```
create or replace package body discard as function delete(sroll Student.roll%type)
return int
is begin
delete from Student where sroll = roll;
return 1;
end delete;
end discard
```

Output –



```
Package created.
```

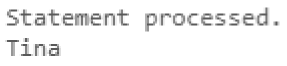


```
Package Body created.
```

9. Find name of the student whose roll number is 2 using a function created in a package

Query –
```
declare  x varchar(20);
begin
        pack.find(2);
        x := pack.sname; dbms_output.put_line(x);
end;
```

Output –

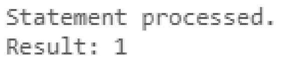


```
Statement processed.
Tina
```

10. Delete a student record whose roll number is 4using a procedure created in a package.

Query –
```
declare  x varchar(20);
begin
        x := discard.deleted(4);
        dbms_output.put_line('Result: ' || x);
end;
```

Output –



```
Statement processed.
Result: 1
```

Name – Parth Medhekar

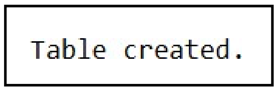
Div – A(A3) **Roll No** – 44

Experiment No 5 – Implementation Of Embedded And Dynamic SQL

Part 1 – Embedded SQL

1. **Create a table Teacher with attributes id,emp_id,name,department,address,contact.**

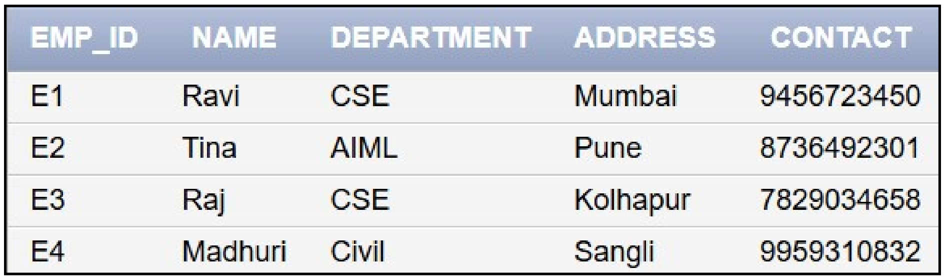
```
CREATE TABLE Teacher (
 id INT,
 emp_id VARCHAR(50),
name VARCHAR(100),
department VARCHAR(100),
 address VARCHAR(200),
 contact VARCHAR(15)
);
```



```
Table created.
```

2. **Insert Follwing in ‘Teacher’ table.**

```
INSERTINTOTeachervalues('E1','Ravi','CSE','Mumbai','9456723450')
INSERT INTO Teacher values('E2','Tina','AIML','Pune','8736492301')
INSERT INTO Teacher values('E3','Raj','CSE','Kolhapur','7829034658')
```

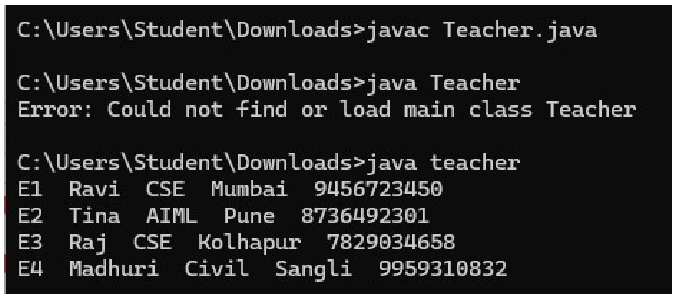


EMP_ID	NAME	DEPARTMENT	ADDRESS	CONTACT
E1	Ravi	CSE	Mumbai	9456723450
E2	Tina	AIML	Pune	8736492301
E3	Raj	CSE	Kolhapur	7829034658
E4	Madhuri	Civil	Sangli	9959310832

3. **Java Program to Embed SQL Code For Databse Connectivity**

```
publicclassSample{
publicstaticvoidmain([]arg
s){ try {
// Step 1: Load the driver class
Class.forName("oracle.jdbc.driver.OracleDriver");
// Step 2: Create the connection object
```

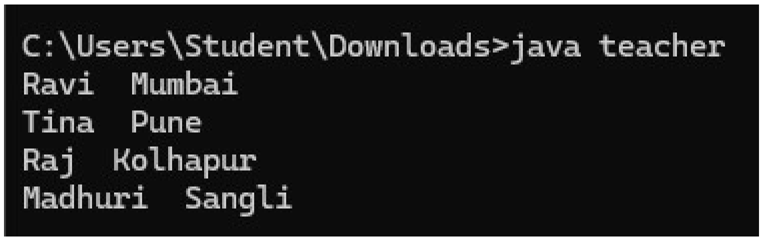
```
    Connectioncon=DriverManager.getConn
    ection(
    "jdbc:oracle:thin:@localhost:1521:XE","anushka","amoghnevgi");
    // Step 3: Create the statement
    object
    Statementstmt=con.createState
    ment();
    //Step4:Executeque
    ry ResultSet rs;
    rs=stmt.executeQuery("SELECT*FROMteacher"
    ); while (rs.next()) {
    System.out.println( rs.getString(1)+""+ rs.getString(2)+""+

    rs.getString()+""+
    rs.getString(4)+""
    + rs.getString(5)
       );}
    con.close();
    } catch (Exception
    e) {
    System.out.println(e
    );
  }}}
```



```
C:\Users\Student\Downloads>javac Teacher.java

C:\Users\Student\Downloads>java Teacher
Error: Could not find or load main class Teacher

C:\Users\Student\Downloads>java teacher
E1  Ravi  CSE  Mumbai  9456723450
E2  Tina  AIML  Pune  8736492301
E3  Raj  CSE  Kolhapur  7829034658
E4  Madhuri  Civil  Sangli  9959310832
```

4. Find name and address of employees using java code created in Q3

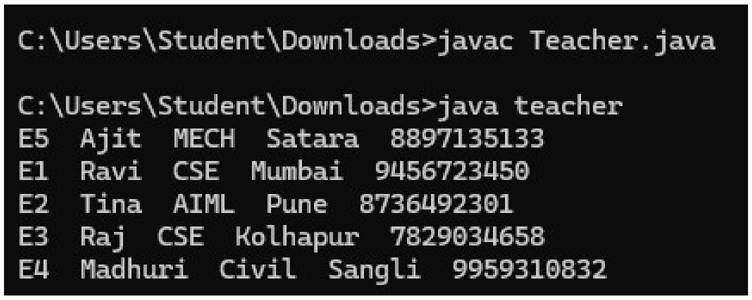
```
    //Step4:Executequery
    ResultSetrs;
    rs=stmt.executeQuery("SELECTname,addressFROMteacher");
    while (rs.next()) {
    System.out.println(
rs.getString(1) + "" +
rs.getString(2) + ""
);}
```



```
C:\Users\Student\Downloads>java teacher
Ravi  Mumbai
Tina  Pune
Raj  Kolhapur
Madhuri  Sangli
```

5. **Add new employee with employee id E5,name AJIT,dept MECH,address satara and contact 8897135133**

```
//Step4:Executequery
ResultSetrs;
rs=stmt.executeQuery("INSERTINTOteachervalues('E5','Ajit','MECH','Satara',
'8897135133')");
rs=stmt.executeQuery("SELECT*FROMteacher"); while
(rs.next()) {
System.out.println(
rs.getString(1)+""+
rs.getString(2)+""+
rs.getString(3)+""+
rs.getString(4)+""+
rs.getString(5)
);}
```

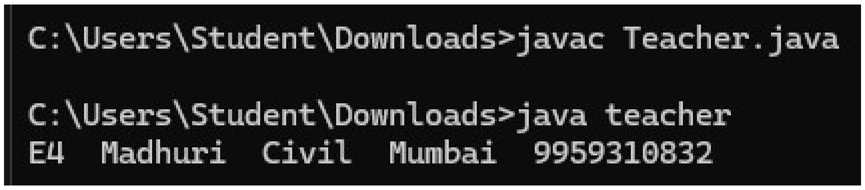


```
C:\Users\Student\Downloads>javac Teacher.java

C:\Users\Student\Downloads>java teacher
E5  Ajit  MECH  Satara  8897135133
E1  Ravi  CSE  Mumbai  9456723450
E2  Tina  AIML  Pune  8736492301
E3  Raj  CSE  Kolhapur  7829034658
E4  Madhuri  Civil  Sangli  9959310832
```

6. **Modify address of Madhurito Mumbai using Java code created in Q.3.**

```
//Step4:Executequery
ResultSetrs;
rs=stmt.executeQuery("UPDATEteacherSETaddress='Mumbai'WHEREname=
'Madhuri'");
rs=stmt.executeQuery("SELECT*FROMteacherwherename='Madhuri'"); while
(rs.next()) {
```

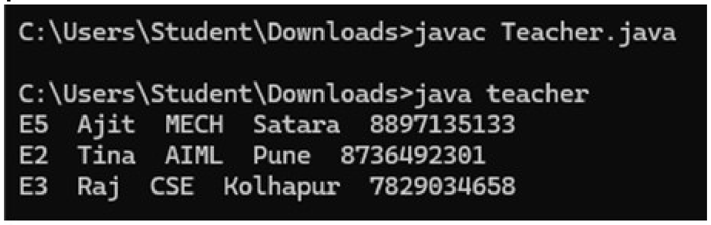
```
System.out.println(
rs.getString(1)+""+
rs.getString(2)+""+
rs.getString(3)+""+
rs.getString(4)+""+
rs.getString(5)
```



```
C:\Users\Student\Downloads>javac Teacher.java

C:\Users\Student\Downloads>java teacher
E4  Madhuri  Civil  Mumbai  9959310832
```

7. **DeleteemployeesstayinginMumbaiusingJavacodecreatedinQ.3.**

```
//Step4:Executequery
          ResultSetrs;
             rs=stmt.executeQuery("DELETEFROMteacherWHERE address='Mumbai'"); rs =
          stmt.executeQuery("SELECT * FROM teacher");
          while   (rs.next())   {
             System.out.println(
                rs.getString(1)+""+
                rs.getString(2)+""+
                rs.getString(3)+""+
                rs.getString(4)+""+
                rs.getString(5));}
```

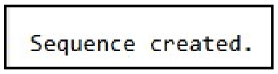


```
C:\Users\Student\Downloads>javac Teacher.java

C:\Users\Student\Downloads>java teacher
E5  Ajit  MECH  Satara  8897135133
E2  Tina  AIML  Pune  8736492301
E3  Raj  CSE  Kolhapur  7829034658
```

Part 2 – Dyanmic SQL

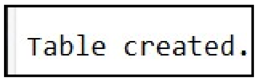
1. **Create table location with attributes location idand location name. Location id should be set as primary key which will increment automatically.**

```
create sequence
sample start with 1
increment by1;
```



```
Sequence created.
```

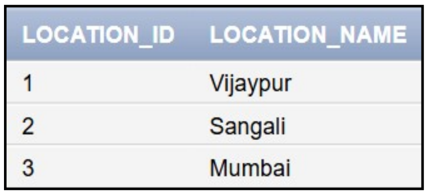
```
create table Location(Location_id int PRIMARYKEY,Location_name varchar(100));
```



```
Table created.
```

2. **Add 3 records in location table.**

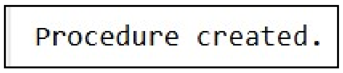
```
Insert t intoLocationvalues(sample.nextval,'Vijaypur');
 insert into Location values(sample.nextval,'Sangali');
insert into Location values(sample.nextval,'Mumbai');
select * from Location;
```



LOCATION_ID	LOCATION_NAME
1	Vijaypur
2	Sangali
3	Mumbai

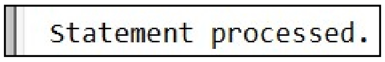
3. **Create a schema level procedure which will accept locationasaparameter.Itshouldinsertthat location into location table. It should also create a new table with the name emp_location (here location should be passed dynamically as a parameter to the procedure) (Hint: Use EXECUTE IMMEDIATE)**

```
createorreplaceprocedureHarsh(Locvac
har) as
begin
insertintoLocationvalues(sample.nextv
al,Loc); EXECUTE IMMEDIATE
'CreateTable'||'Emp_'||Loc||'
(Emp_noint,Emp_namevarchar(15),Emp_jobvarchar(10)
)'; end;
```



```
Procedure created.
```

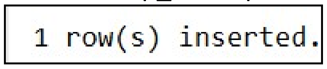
4. **Call the created procedure by passing values as‘Rajarampuri’and‘Shahupuri’.(Hint:Inthe background,tableswithnameemp_Rajarampuriandemp_shahupurishouldgetcreated)**
begin Parth('Rajarampuri'); end;
beginParth('Shahupuri');
end;



```
Statement processed.
```

5. **Insert wore cords in the tables created after execution of Q4.**

insert into Emp_Rajarampuri values (1,'ABC','XYZ');
insertintoEmp_Rajarampurivalues(1,'LMN','PQR'); insert into Emp_Shahupuri values (11,'CBA','ZYX');
insert into Emp_Shahupuri values (12,'NML','RQP');

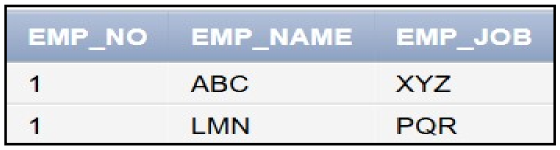


```
1 row(s) inserted.
```

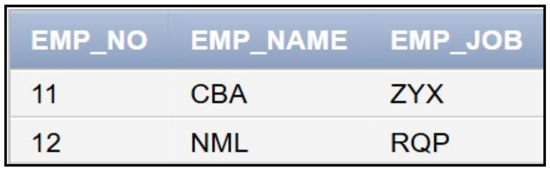
6. **Display those two tables**

select*fromEmp_Rajarampuri;

select * from Emp_Shahupuri;



EMP_NO	EMP_NAME	EMP_JOB
1	ABC	XYZ
1	LMN	PQR



EMP_NO	EMP_NAME	EMP_JOB
11	CBA	ZYX
12	NML	RQP

Name – Parth Medhekar

Div – A(A3) **Roll No** - A44

Experiment No 7 – Design an XML Document, XML DTD and XML Schema For Given Database

XML Document

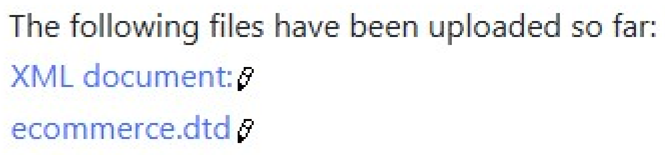
```
<?xml version="1.0" standalone="no"?>

<!DOCTYPE ecommerce SYSTEM "ecommerce.dtd">

<ecommerce>

 <Store>

  <Product>

   <pid>P01</pid>

   <pName>Oneplus Z2 earbuds</pName>

   <price>1699</price>

  </Product>

  <Customer>

   <cid>C01</cid>

   <cName>

    <Fname>Parth</Fname>

    <Lname>Medhekar</Lname>

   </cName>

   <email>p55112846@gmail.com</email>

   <age>20</age>

  </Customer>

  <Order>

   <oid>O9001</oid>

   <oDate>2025-04-18</oDate>

   <total>1699</total>
```

```
    </Order>
    <Seller>
      <sid>SS01</sid>
      <sName>SS Electronics</sName>
      <phone>1234567890</phone>
    </Seller>
  </Store>
</ecommerce>
```

XML DTD

```
<!ELEMENT ecommerce (Store+)>
<!ELEMENT Store (Product, Customer, Order, Seller)>
<!ELEMENT Product (pid, pName, price)>
<!ELEMENT pid (#PCDATA)>
<!ELEMENT pName (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT Customer (cid, cName, email, age)>
<!ELEMENT cid (#PCDATA)>
<!ELEMENT cName (Fname, Lname)>
<!ELEMENT Fname (#PCDATA)>
<!ELEMENT Lname (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT Order (oid, oDate, total)>
<!ELEMENT oid (#PCDATA)>
<!ELEMENT oDate (#PCDATA)>
```

```
<!ELEMENT total (#PCDATA)>

<!ELEMENT Seller (sid, sName, phone)>

<!ELEMENT sid (#PCDATA)>

<!ELEMENT sName (#PCDATA)>

<!ELEMENT phone (#PCDATA)>
```

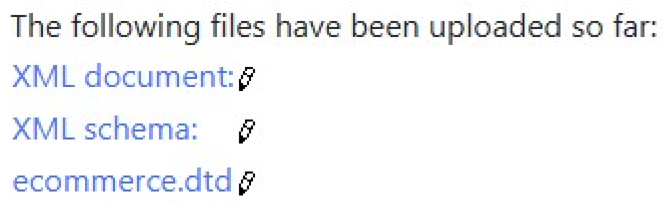

The following files have been uploaded so far:
XML document:
ecommerce.dtd

XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="ecommerce">

    <xs:complexType>

      <xs:sequence>

        <xs:element name="Store" type="StoreType"/>

      </xs:sequence>

    </xs:complexType>

  </xs:element>

  <xs:complexType name="StoreType">

    <xs:sequence>

      <xs:element name="Product" type="ProductType"/>

      <xs:element name="Customer" type="CustomerType"/>

      <xs:element name="Order" type="OrderType"/>

      <xs:element name="Seller" type="SellerType"/>

    </xs:sequence>
```

```
 </xs:complexType>
 <xs:complexType name="ProductType">
   <xs:sequence>
     <xs:element name="pid" type="xs:string"/>
     <xs:element name="pName" type="xs:string"/>
     <xs:element name="price" type="xs:decimal"/>
   </xs:sequence>
 </xs:complexType>
 <xs:complexType name="CustomerType">
   <xs:sequence>
     <xs:element name="cid" type="xs:string"/>
     <xs:element name="cName" type="NameType"/>
     <xs:element name="email" type="xs:string"/>
     <xs:element name="age" type="xs:integer"/>
   </xs:sequence>
 </xs:complexType>
<xs:complexType name="NameType">
   <xs:sequence>
     <xs:element name="Fname" type="xs:string"/>
     <xs:element name="Lname" type="xs:string"/>
   </xs:sequence>
 </xs:complexType>
 <xs:complexType name="OrderType">
   <xs:sequence>
     <xs:element name="oid" type="xs:string"/>
```

```
      <xs:element name="oDate" type="xs:date"/>

      <xs:element name="total" type="xs:decimal"/>

    </xs:sequence>

  </xs:complexType>

  <xs:complexType name="SellerType">

    <xs:sequence>

      <xs:element name="sid" type="xs:string"/>

      <xs:element name="sName" type="xs:string"/>

      <xs:element name="phone" type="xs:string"/>

    </xs:sequence>

  </xs:complexType>

</xs:schema>
```



The following files have been uploaded so far:

XML document:

XML schema:

ecommerce.dtd