

PROJECT WORK

on

**“PRTC BUS PASS ALLOTMENT SYSTEM. ”**

SUBMITTED BY-

PRIYANKA CHAUHAN-101603250

RAGAHAV MITTAL-101603255

SUBMITTED TO-

Dr. Santosh Singh Rathore

TABLE OF CONTENTS

* PROBLEM DEFINATION AND INTRODUCTION
* ER DIAGRAM
* ER TO TABLE CONVERSION
* NORMALISATION OF TABLE
* FUNCTIONAL DEPENDECIES
* PL /SQL CODE

PROBLEM DEFINATION AND INTRODUCTION

In this we have created a database for the PRTC bus pass allotment system. Bus pass is a facility provided by the PRTC for those student who travels regularly in buses. This facility helps them to travel at cheaper rate. Today, PRTC bus pass allotment system is a offline system.

So, this project is small effort to make the work of students to apply and issue the bus pass easy by making it online. With the help of this database student easily can apply for bus pass and give fill his information about roll no ,institute, address etc.

PROBLEM STATEMENT

For each student needs bus pass, should submit the details of it’s phone no, address pin-code, address, pincode, name. Student needs to tell institute id also. Institute gives their grant to the student that the he/she studies in their institute. After all the verification PRTC will issue the bus pass to the student by giving the information about the payment using payment database.

SCHEMAS

* PRTC\_BUSPASS
* STUDENT
* INSTITUTE
* PAYMENT

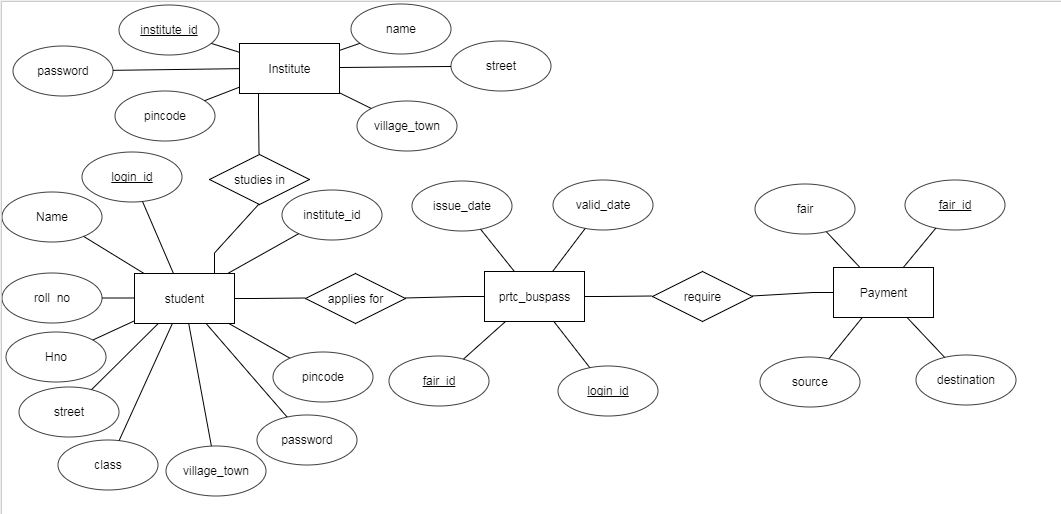
SYSTEM ANALYSIS

* Any Student can apply for bus pass and fill the information then he gets

a login id .

* Then institute verifies wither the student studies in their institute or not.
* Then prtc tells the payment to be paid for bus pass to the student.
* So, after the payment prtc issues the bus pass.

ER DIAGRAM



E-R TO TABLES

STUDENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ATTRIBUTES | DATATYPES | NULL | | CONSTRAINTS |
| LOGIN\_ID | NUMBER | NOT-NULL | | PRIMARY-KEY |
| ROLL\_NO | NUMBER | NOT-NULL | | - |
| NAME | VARCHAR2(10) | NOT-NULL | | - |
| CLASS | NUMBER | NOT-NULL | | - |
| PASSWORD | VARCHAR2(10) | NOT-NULL | | - |
| PHONE | NUMBER | NOT-NULL | | - |
| INSTITUTE\_ID | VARCHAR2(10) | NOT-NULL | | FOREIGN-KEY |
| STREET | VARCHAR2(10) | | - | - |
| VILLAGE\_TOWN | VARCHAR2(10) | | - | - |
| PINCODE | VARCHAR2(10) | | NOT-NULL | - |

PRTC\_BUSPASS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ATTRIBUTES | DATA-TYPES | | NOT-NULL | CONSTRAINTS |
| LOGIN\_ID | NUMBER | | NOT-NULL | PRIMARY-KEY,FOREIGN KEY |
| ISSUE\_DATE | DATE | | NOT-NULL | - |
| VALID\_DATE | DATE | | NOT-NULL | - |
| FAIR\_ID | NUMBER | NOT-NULL | | PRIMARY-KEY |

INSTITUTE

|  |  |  |  |
| --- | --- | --- | --- |
| ATTRIBUTES | DATA-TYPES | NOT-NULL | CONSTRAINTS |
| INSTITUTE\_ID | VARCHAR2 | NOT-NULL | PRIMARY-KEY |
| PASSWORD | VARCHAR2 | NOT-NULL | - |
| NAME | VARCHAR2 | NOT-NULL | - |
| VILLAGE-TOWN | VARCHAR2 | - | - |
| NAME | VARCHAR2 | - | - |
| PINCODE | NUMBER | NOT-NULL |  |

PAYMENT

|  |  |  |  |
| --- | --- | --- | --- |
| ATTRIBUTES | DATA-TYPE | NOT-NULL | CONSTRSAINT |
| FAIR-ID | NUMBER | NOT-NULL | PRIMARY-KEY |
| SOURCE | NUMBER | NOT-NULL | - |
| DESTINATION | NUMBER | NOT-NULL | - |
| FAIR | NUMBER | NOT-NULL | - |

NORMALISATION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ATTRIBUTES | DATATYPES | NULL | | CONSTRAINTS |
| LOGIN\_ID | NUMBER | NOT-NULL | | PRIMARY-KEY |
| ROLL\_NO | NUMBER | NOT-NULL | | - |
| NAME | VARCHAR2(10) | NOT-NULL | | - |
| CLASS | NUMBER | NOT-NULL | | - |
| PASSWORD | VARCHAR2(10) | NOT-NULL | | - |
| PHONE | NUMBER | NOT-NULL | | - |
| INSTITUTE\_ID | VARCHAR2(10) | NOT-NULL | | FOREIGN-KEY |
| STREET | VARCHAR2(10) | | - | - |
| VILLAGE\_TOWN | VARCHAR2(10) | | - | - |
| PINCODE | VARCHAR2(10) | | NOT-NULL | - |

1ST NORMAL FORM- Table Is in first normal form as all underlying

Or primary attributes are single valued.

2nd NORMAL FORM-Table is in 2 normal form as all non-key attribute is fully dependent on primary key.

3rd NORMAL –Table is in 3rd normal form as no non-prime attribute

transitively dependent on the primary key.

BC-NF –Table is in BCNF Normal form as every determinant is a candidate key.

PRTC\_BUSPASS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ATTRIBUTES | DATA-TYPES | | NOT-NULL | CONSTRAINTS |
| LOGIN\_ID | NUMBER | | NOT-NULL | PRIMARY-KEY,FOREIGN KEY |
| ISSUE\_DATE | DATE | | NOT-NULL | - |
| VALID\_DATE | DATE | | NOT-NULL | - |
| FAIR\_ID | NUMBER | NOT-NULL | | PRIMARY-KEY |

1ST NORMAL FORM- Table Is in first normal form as all underlying

Or primary attributes are single valued.

2nd NORMAL FORM-Table is in 2 normal form as all non-key attribute is fully dependent on primary key.

3rd NORMAL –Table is in 3rd normal form as no non-prime attribute

transitively dependent on the primary key.

BC-NF –Table is in BCNF Normal form as every determinant is a candidate key.

INSTITUTE

|  |  |  |  |
| --- | --- | --- | --- |
| ATTRIBUTES | DATA-TYPES | NOT-NULL | CONSTRAINTS |
| INSTITUTE\_ID | VARCHAR2 | NOT-NULL | PRIMARY-KEY |
| PASSWORD | VARCHAR2 | NOT-NULL | - |
| NAME | VARCHAR2 | NOT-NULL | - |
| VILLAGE-TOWN | VARCHAR2 | - | - |
| NAME | VARCHAR2 | - | - |
| PINCODE | NUMBER | NOT-NULL |  |

1ST NORMAL FORM- Table Is in first normal form as all underlying

Or primary attributes are single valued.

2nd NORMAL FORM-Table is in 2 normal form as all non-key attribute is fully dependent on primary key.

3rd NORMAL –Table is in 3rd normal form as no non-prime attribute

transitively dependent on the primary key.

BC-NF –Table is in BCNF Normal form as every determinant is a candidate key.

PAYMENT

|  |  |  |  |
| --- | --- | --- | --- |
| ATTRIBUTES | DATA-TYPE | NOT-NULL | CONSTRSAINT |
| FAIR-ID | NUMBER | NOT-NULL | PRIMARY-KEY |
| SOURCE | NUMBER | NOT-NULL | - |
| DESTINATION | NUMBER | NOT-NULL | - |
| FAIR | NUMBER | NOT-NULL | - |

1ST NORMAL FORM- Table Is in first normal form as all underlying

Or primary attributes are single valued.

2nd NORMAL FORM-Table is in 2 normal form as all non-key attribute is fully dependent on primary key.

3rd NORMAL –Table is in 3rd normal form as no non-prime attribute

transitively dependent on the primary key.

BC-NF –Table is in BCNF Normal form as every determinant is a candidate key.

FUNCTIONAL DEPENDENCIES

STUDENT FD

PHONE

ROLL\_N0

INSTITUTE\_ID

STREET\_

LOGIN\_ID

VILLAGE\_TOWN

PINCODE

NAME

PASSWORD

CLASS

INSTITUTE\_FD

STREET

VILLAGE\_TOWN

NAME

INSTITUTE\_ID

PINCODE

PASSWORD

PRTC BUSPASS FD

ISSUE\_DATE

LOGIN\_ID

FAIR ID

LOGIN\_ID

VALID\_DATE

PAYMENT FD

SOURCE

FAIR

FAIR\_ID

DESTINATION

CREATE TABLE

CREATE TABLE "STUDENT"

( "LOGIN\_ID" NUMBER NOT NULL ENABLE,

"PASSWORD" VARCHAR2(4000) NOT NULL ENABLE,

"INSTITUTE\_ID" VARCHAR2(4000) NOT NULL ENABLE,

"NAME" VARCHAR2(4000) NOT NULL ENABLE,

"ROLL\_NO" NUMBER(20,0) NOT NULL ENABLE,

"PHONE" NUMBER(20,0) NOT NULL ENABLE,

"CLASS" NUMBER(20,0) NOT NULL ENABLE,

"STREET\_" NUMBER(20,0),

"VILLAGE\_TOWN" VARCHAR2(4000),

"PINCODE" NUMBER(20,0) NOT NULL ENABLE,

CONSTRAINT "STUDENT\_PK" PRIMARY KEY ("LOGIN\_ID") ENABLE

)

CREATE TABLE "INSTITUTE"

( "INSTITUTE\_ID" VARCHAR2(4000) NOT NULL ENABLE,

"INSTITUTE\_PASSWORD" VARCHAR2(4000) NOT NULL ENABLE,

"INSTITUTE\_NAME" VARCHAR2(4000) NOT NULL ENABLE,

"INSTITUTE\_VILLAGE\_TOWN" VARCHAR2(4000) NOT NULL ENABLE,

"INSTITUTE\_STREET" NUMBER(20,0) NOT NULL ENABLE,

"INSTITUTE\_PINCODE" NUMBER(20,0) NOT NULL ENABLE,

CONSTRAINT "INSTITUTE\_PK" PRIMARY KEY ("INSTITUTE\_ID") ENABLE

)

CREATE TABLE "NEW"

( "ROLL\_NO" NUMBER NOT NULL ENABLE,

"RINST" NUMBER NOT NULL ENABLE,

"INSTITUTE\_ID" VARCHAR2(4000) NOT NULL ENABLE,

CONSTRAINT "NEW\_PK" PRIMARY KEY ("ROLL\_NO", "INSTITUTE\_ID") ENABLE

)

CREATE TABLE "PRTCBUSPASS"

( "LOGIN\_ID" NUMBER(20,0) NOT NULL ENABLE,

"FAIR\_ID" VARCHAR2(4000) NOT NULL ENABLE,

"ISSUE\_DATE" DATE NOT NULL ENABLE,

"VALID\_DATE" DATE NOT NULL ENABLE,

CONSTRAINT "PRTCBUSPASS\_PK" PRIMARY KEY ("LOGIN\_ID") ENABLE

)

CREATE TABLE "PAYMENT"

( "FAIR\_ID" NUMBER(20,0) NOT NULL ENABLE,

"FAIR" NUMBER(20,0) NOT NULL ENABLE,

"SOURCE" VARCHAR2(4000) NOT NULL ENABLE,

"DESTINATION" VARCHAR2(4000) NOT NULL ENABLE,

CONSTRAINT "PAYMENT\_PK" PRIMARY KEY ("FAIR\_ID") ENABLE

)

PL/SQL CODES

<----declare insert------>

declare

login student.login\_id%type;

pwd student.password%type;

iid student.institute\_id%type;

name student.name%type ;

roll student.roll\_no%type;

ph student.phone%type;

class student.class%type;

street student.street\_%type;

villto student.village\_town%type;

pin student.pincode%type;

begin

login := :login;

pwd := :pwd;

iid := :iid;

name := :name;

roll := :roll;

ph := :ph;

class := :class;

street := :street;

villto := :villto;

pin := :pin;

--procedure call

stud\_insert(login,pwd,iid,name,roll,ph,class,street,villto,pin);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

iid institute.institute\_id%type;

pwd institute.institute\_password%TYPE;

name institute.institute\_name%type ;

villto institute.institute\_village\_town%type;

street institute.institute\_street%type;

pin institute.institute\_pincode%type;

INTEGRITY\_VIOLATED EXCEPTION;

PRAGMA EXCEPTION\_INIT(INTEGRITY\_VIOLATED, -2291);

begin

iid := :iid;

pwd := :pwd;

name := :name;

villto := :villto;

street := :street;

pin := :pin;

--procedure call

inst\_insert(iid,pwd,name,villto,street,pin);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN INTEGRITY\_VIOLATED THEN

DBMS\_OUTPUT.PUT\_LINE('CAN’T FIND REFERENCED PARENT RECORD');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

fair\_id payment.fair\_id%type;

fair payment.fair%TYPE;

source payment.source%type ;

desti payment.destination%type;

INTEGRITY\_VIOLATED EXCEPTION;

PRAGMA EXCEPTION\_INIT(INTEGRITY\_VIOLATED,-2291);

begin

fair\_id := :fair\_id;

fair := :fair;

source := :source;

desti := :desti;

--procedure call

payment\_insert(fair\_id,fair,source,desti);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN INTEGRITY\_VIOLATED THEN

DBMS\_OUTPUT.PUT\_LINE('CAN’T FIND REFERENCED PARENT RECORD');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

login prtcbuspass.login\_id%type;

fair prtcbuspass.fair\_id%TYPE;

issue\_date prtcbuspass.issue\_date%type ;

valid\_date prtcbuspass.valid\_date%type;

INTEGRITY\_VIOLATED EXCEPTION;

PRAGMA EXCEPTION\_INIT(INTEGRITY\_VIOLATED, -2291);

begin

login := :login;

fair := :fair;

issue\_date := :issue\_date;

valid\_date := :valid\_date;

if valid\_date>issue\_date then

raise invalid\_date;

--procedure call

prtc\_insert(login,fair,issue\_date,valid\_date);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN INVALID\_DATE THEN

DBMS\_OUTPUT.PUT\_LINE('VALID DATE SHOULD BE HIGHER THAN ISSUE DATE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN INTEGRITY\_VIOLATED THEN

DBMS\_OUTPUT.PUT\_LINE('CAN’T FIND REFERENCED PARENT RECORD');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

<------declare update------->

declare

pass student.password%type;

login student.login\_id%type;

begin

login := :login;

pass := :pass;

--procedure call

stud\_update1(login,pass);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

ph student.phone%type;

login student.login\_id%type;

begin

login := :login;

ph := :phone;

--procedure call

stud\_update2(ph,login);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

st student.phone%type;

login student.login\_id%type;

begin

login := :login;

st := :street;

--procedure call

stud\_update3(st,login);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

pin student.pincode%type;

login student.login\_id%type;

village\_town student.village\_town%type;

begin

login := :login;

pin := :pincode;

village\_town := :village\_town;

--procedure call

stud\_update4(pin,village\_town,login);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

issue prtcbuspass.issue\_date%type;

valid prtcbuspass.valid\_date%type;

login prtcbuspass.login\_id%type;

fair prtcbuspass.fair\_id%type;

begin

login := :login\_id;

fair := :fair;

issue := :issue;

valid := :valid;

--procedure call

prtcbuspass\_update1(issue,valid,login,fair);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

pass institute.institute\_password%type;

login institute.institute\_id%type;

begin

login := :login;

pass := :password;

--procedure call

inst\_update1(login,pass);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

fair payment.fair%type;

login payment.fair\_id%type;

begin

fair := :fair;

fair\_id := :fair\_id;

--procedure call

payment\_update1(fair,fair\_id);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

<----declare delete----->

declare

login student.login\_id%type;

begin

login := :login\_id;

--procedure call

student\_delete(login);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

iid institute.institute\_id%type;

begin

iid := :institute\_id;

--procedure call

inst\_delete(iid);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

declare

fair payment.fair\_id%type;

begin

fair := :fair\_id;

--procedure call

payment\_delete(fair);

exception

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND');

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('DUPLICATE VALUE');

WHEN ACCESS\_INTO\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('ASSIGNMENT INTO AN ATTRIBUTE OF UNINITIALIZED OBJECT');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('ARITHMETIC ERROR OR SIZE ERROR');

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('A NUMBER DIVIDED BY ZERO');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

create or replace procedure stud\_insert(

login in student.login\_id%type,

pwd IN STUDENT.PASSWORD%TYPE,

iid in student.institute\_id%type,

name in student.name%type ,

roll in student.roll\_no%type,

ph in student.phone%type,

class in student.class%type,

street in student.street\_%type,

villto in student.village\_town%type,

pin in student.pincode%type ) is

begin

insert into student values (login,pwd,iid,name,roll,ph,class,street,villto,pin);

end;

create or replace procedure inst\_insert(

iid in institute.institute\_id%type,

pwd IN institute.institute\_password%TYPE,

name in institute.institute\_name%type ,

villto in institute.institute\_village\_town%type,

street in institute.institute\_street%type,

pin in institute.institute\_pincode%type ) is

begin

insert into institute values (iid,pwd,name,villto,street,pin);

end;

create or replace procedure prtc\_insert(

login in prtcbuspass.login\_id%type,

fair IN prtcbuspass.fair\_id%TYPE,

issue\_date in prtcbuspass.issue\_date%type ,

valid\_date in prtcbuspass.valid\_date%type

) is

begin

insert into prtcbuspass values (login,fair,issue\_date,valid\_date);

end;

create or replace procedure payment\_insert(

fair\_id in payment.fair\_id%type,

fair IN payment.fair%TYPE,

source in payment.source%type ,

desti in payment.destination%type

) is

begin

insert into payment values (fair\_id,fair,source,desti);

end;

<------delete----------->

create or replace procedure stud\_delete(

login in student.login\_id %type ) is

begin

delete student where login\_id = login;

commit;

end;

create or replace procedure inst\_delete(

inst in institute.institute\_id %type ) is

begin

delete institute where inst= institute\_id;

commit;

end;

create or replace procedure prtc\_delete(

login in prtcbuspass.login\_id %type,fair in prtcbuspass.fair\_id%type ) is

begin

delete prtcbuspass where login = login\_id and fair = fair\_id;

commit;

end;

create or replace procedure pay\_delete(

fair in payment.fair\_id %type) is

begin

delete prtcbuspass where fair= fair\_id ;

commit;

end;

<-----------update----------->

create or replace procedure stud\_update1(

login in student.login\_id%type,

pass in student.password%type) is begin

update student

set password = pass where login = login\_id;

commit;

end;

create or replace procedure stud\_update2(

ph in student.phone%type,

login in student.login\_id%type) is begin

update student

set phone=ph where login = login\_id;

commit;

end;

create or replace procedure stud\_update3(

st in student.street\_%type,

login in student.login\_id%type) is begin

update student

set street\_=st where login = login\_id;

commit;

end;

create or replace procedure stud\_update4(

pin in student.pincode%type,

vt in student.village\_town%type,

login in student.login\_id%type) is begin

update student

set pincode=pin,village\_town=vt where login = login\_id;

commit;

end;

create or replace procedure inst\_update1(

login in institute.institute\_id%type,

pass in institute.institute\_password%type) is begin

update institute

set institute\_password = pass where login = institute\_id;

commit;

end;

create or replace procedure prtcbuspass\_update1(

issue in prtcbuspass.issue\_date%type,

valid in prtcbuspass.valid\_date%type,

login in prtcbuspass.login\_id%type,

fair in prtcbuspass.fair\_id%type) is begin

update prtcbuspass

set issue\_date=issue,valid\_date=valid where login = login\_id and fair = fair\_id;

commit;

end;

create or replace procedure payment\_update1(

fa in payment.fair%type,

fa\_id in payment.fair\_id%type) is begin

update payment

set fair=fa where fair\_id = fa\_id;

commit;

end;

create or replace procedure insti\_update1(

insti\_pass in institute.institute\_password%type,

insti\_id in institute.institute\_id%type) is begin

update institute

set institute\_password=insti\_pass where insti\_id = institute\_id;

commit;

end;

<---------functions------------>

create or replace function total\_stud(

login in student.login\_id%type

) return number

is c number ;

begin

select count(login\_id) into c from student;

return (c);

end;

create or replace function total\_inst(

inst in institute.institute\_id%type

) return number

is c1 number ;

begin

select count(inst) into c1 from institute;

return (c1);

end;

<-------------cursor-------------->

create or replace procedure correspond1(

stud\_log1 in student.login\_id%type,

inst\_id1 in student.institute\_id%type)is

s number;

l varchar2(10);

cursor d is select institute\_id from student;

begin

open d;

loop

fetch d into l ;

exit when d%notfound;

select login\_id into s from student where institute\_id = l;

dbms\_output.put\_line(s);

end loop;

close d;

commit;

end;

<---------TRIGGER----------->

CREATE OR REPLACE TRIGGER BUS BEFORE INSERT OR UPDATE OF NAME ON STUDENT FOR EACH ROW BEGIN

:NEW.NAME := UPPER(:NEW.NAME);

END;