Admin Table:

create table admin

(

aid number(5) constraint aid\_pk primary key,

name varchar2(50),

gender varchar2(10),

phone number(15),

email varchar2(50),

address varchar2(200),

password varchar2(50)

);

Retired\_Admin Table:

create table retired\_admin

(

aid number(5),

name varchar2(50),

gender varchar2(10),

phone number(15),

email varchar2(50),

address varchar2(200),

password varchar2(50)

);

Company Table:

create table company

(

coid number(5) constraint coid\_pk primary key,

name varchar2(50),

phone number(15),

email varchar2(50),

office varchar2(100),

tradelicense varchar2(30),

password varchar2(50),

status varchar2(10)

);

Customer Table:

create table customer

(

cuid number(5) constraint cuid\_pk primary key,

name varchar2(50),

gender varchar2(10),

phone number(15),

email varchar2(50),

address varchar2(200),

password varchar2(50)

);

Package Table:

create table package

(

pid number(5) constraint pid\_pk primary key,

name varchar2(50),

day number(2),

night number(2),

place varchar2(30),

cost number(10),

description varchar2(200),

company\_name varchar2(50)

);

Bookings Table:

create table bookings

(

bid number(5) constraint bid\_pk primary key,

package\_id number(5),

customer\_id number(5)

);

Payment Table:

create table payment

(

pyid number(5) constraint pyid\_pk primary key,

ammount number(10),

transition\_number varchar(30),

package\_id number(5),

customer\_id number(5),

status varchar2(10)

);

Tour Table:

create table tour

(

tid number(5) constraint tid\_pk primary key,

package\_id number(5),

customer\_id number(5),

company\_id number(5)

);

**Sequences:**

Admin Table:

Create Sequence a\_id

Start with 1

Increment by 1

Minvalue 0

Maxvalue 1000

Cycle;

Company Table:

Create Sequence co\_id

Start with 1

Increment by 1

Minvalue 0

Maxvalue 1000

Cycle;

Customer Table:

Create Sequence cu\_id

Start with 1

Increment by 1

Minvalue 0

Maxvalue 1000

Cycle;

Package Table:

Create Sequence p\_id

Start with 1

Increment by 1

Minvalue 0

Maxvalue 1000

Cycle;

Bookings Table:

Create Sequence b\_id

Start with 1

Increment by 1

Minvalue 0

Maxvalue 1000

Cycle;

Payment Table:

Create Sequence py\_id

Start with 1

Increment by 1

Minvalue 0

Maxvalue 1000

Cycle;

Tour Table:

Create Sequence t\_id

Start with 1

Increment by 1

Minvalue 0

Maxvalue 1000

Cycle;

select test\_id.nextval from test

**Function:**

create or replace function customer\_login(cu\_phone in number, cu\_password in varchar2)

return number

is

match\_count number;

begin

select count(\*) into match\_count

from customer

where phone=cu\_phone

and password=cu\_password;

if match\_count = 0 then

return 0;

elsif match\_count = 1 then

return 1;

else

return 2;

end if;

end customer\_login;

create or replace function admin\_login(a\_phone in number, a\_password in varchar2)

return number

is

match\_count number;

begin

select count(\*) into match\_count

from admin

where phone=a\_phone

and password=a\_password;

if match\_count = 0 then

return 0;

elsif match\_count = 1 then

return 1;

else

return 2;

end if;

end admin\_login;

create or replace function company\_login(co\_phone in number, co\_password in varchar2, co\_status in varchar2)

return number

is

match\_count number;

begin

select count(\*) into match\_count

from company

where phone=co\_phone

and password=co\_password

and status=co\_status;

if match\_count = 0 then

return 0;

elsif match\_count = 1 then

return 1;

else

return 2;

end if;

end company\_login;

**Procedure:**

**1.**

CREATE OR REPLACE PROCEDURE INSERTcustomer (

cu\_name customer.name%type,

cu\_gender customer.gender%type,

cu\_phone customer.phone%type,

cu\_email customer.email%type,

cu\_address customer.address%type,

cu\_password customer.password%type)

IS

BEGIN

INSERT INTO customer(CUID, NAME, GENDER, PHONE, EMAIL, ADDRESS, PASSWORD)

VALUES (cu\_id.nextval, cu\_name ,cu\_gender ,cu\_phone , cu\_email, cu\_address, cu\_password);

COMMIT;

END INSERTcustomer;

2.

CREATE OR REPLACE PROCEDURE INSERTcompanydata(

co\_name company.name%type,

co\_phone company.phone%type,

co\_email company.email%type,

co\_office company.office%type,

co\_tradelicense company.tradelicense%type,

co\_password company.password%type,

co\_status company.status%type)

IS

BEGIN

INSERT INTO company(COID, NAME, PHONE, EMAIL, OFFICE, TRADELICENSE, PASSWORD, STATUS)

VALUES (co\_id.nextval, co\_name, co\_phone, co\_email, co\_office, co\_tradelicense, co\_password, co\_status);

COMMIT;

END INSERTcompanydata;

3.

CREATE OR REPLACE PROCEDURE INSERTadmin(

a\_name admin.name%type,

a\_gender admin.gender%type,

a\_phone admin.phone%type,

a\_email admin.email%type,

a\_address admin.address%type,

a\_password admin.password%type)

IS

BEGIN

INSERT INTO admin(AID, NAME, GENDER, PHONE, EMAIL, ADDRESS, PASSWORD)

VALUES (a\_id.nextval, a\_name ,a\_gender ,a\_phone , a\_email, a\_address, a\_password);

COMMIT;

END INSERTadmin;

4.

CREATE OR REPLACE PROCEDURE INSERTpackage(

p\_name package.name%type,

p\_day package.day%type,

p\_night package.night%type,

p\_place package.place%type,

p\_cost package.cost%type,

p\_description package.description%type,

p\_company\_name package.company\_name%type)

IS

BEGIN

INSERT INTO package(PID, NAME, DAY, NIGHT, PLACE, COST, DESCRIPTION, COMPANY\_NAME)

VALUES (p\_id.nextval, p\_name, p\_day, p\_night, p\_place, p\_cost, p\_description, p\_company\_name);

COMMIT;

COMMIT;

5.

CREATE OR REPLACE PROCEDURE delete\_admin(a\_aid IN admin.aid%TYPE)

IS

BEGIN

DELETE admin where aid=a\_aid;

COMMIT;

END delete\_admin;

6.

CREATE OR REPLACE PROCEDURE UPDATEpackage(

p\_id in package.pid%type,

p\_name in package.name%type,

p\_day in package.day%type,

p\_night in package.night%type,

p\_place in package.place%type,

p\_cost in package.cost%type,

p\_description in package.description%type

)

IS

BEGIN

UPDATE Package SET name=p\_name, day=p\_day, night=p\_night, place=p\_place, cost=p\_cost, description=p\_description

where pid= p\_id;

COMMIT;

END;

7.

CREATE OR REPLACE PROCEDURE UPDATEstatus(

co\_id IN company.coid%TYPE)

IS

BEGIN

UPDATE company SET status = 'Valid' where coid = co\_id;

COMMIT;

END;

8.

CREATE OR REPLACE PROCEDURE DELETEcompany(

co\_id IN company.coid%TYPE)

IS

BEGIN

Delete from company where coid = co\_id;

COMMIT;

END;

**Trigger:**

**1.**

create or replace trigger delete\_admin

BEFORE delete on admin

for each row

when (old.aid>0)

begin

insert into retired\_admin values(:old.aid, :old.name, :old.gender, :old.phone, :old.email, :old.address, :old.password);

end;

**2.**

create or replace trigger cus\_update

before insert or update or delete on customer

begin

if to\_char(sysdate,'h24:mi') between '10:00' and '17:00' and

to\_char(sysdate,'day') not between 'SATDAY' and 'FRIDAY' then

raise\_application\_error(-20201,'you may only make changes during normal

office hours except saturday and friday');

end if;

end;

alter trigger cus\_update disable;

**3.**

create or replace trigger customer\_insert

after insert on customer

begin

dbms\_output.put\_line('customer record inserted');

end;

**4.**

create or replace trigger admin\_update

after insert on admin

begin

dbms\_output.put\_line('admin table updated.');

end;

**5.**

create or replace trigger com\_update

before insert or update or delete on company

begin

if to\_char(sysdate,'h24:mi') between '10:00' and '17:00' and

to\_char(sysdate,'day') not between 'SATDAY' and 'FRIDAY' then

raise\_application\_error(-20202,'you may only make changes during normal

office hours except saturday and friday');

end if;

end;

**6.**

create or replace trigger admin\_insert

before insert on admin

begin

if to\_char(sysdate,'h24:mi') between '10:00' and '17:00' and

to\_char(sysdate,'day') not between 'SATDAY' and 'FRIDAY' then

raise\_application\_error(-20203,'You can only add admin during working hours except saturday and friday ');

end if;

end;