# Picture 1

American International University-Bangladesh

**Final Term Report**

**Project Title:** Tour Guide Management System

**Course Name:** ADVANCE DATABASE MANAGEMENT SYSTEM

**Course Teacher:** Rezwan Ahmed

**Semester: Summer** 21-22

|  |  |  |
| --- | --- | --- |
| Name | ID | Section |
| Anik Sen | **20-42138-1** | **C** |
| Abdullah Yousuf Noman | **20-42130-1** | **C** |
| Musfiqur Rahman | **20-42558-1** | **C** |
| Ananna Rashid Kumkum | **20-43589-1** | **C** |

***Table of Contents***

Admin Panel 3

Interfaces 4

Searching Queries 6

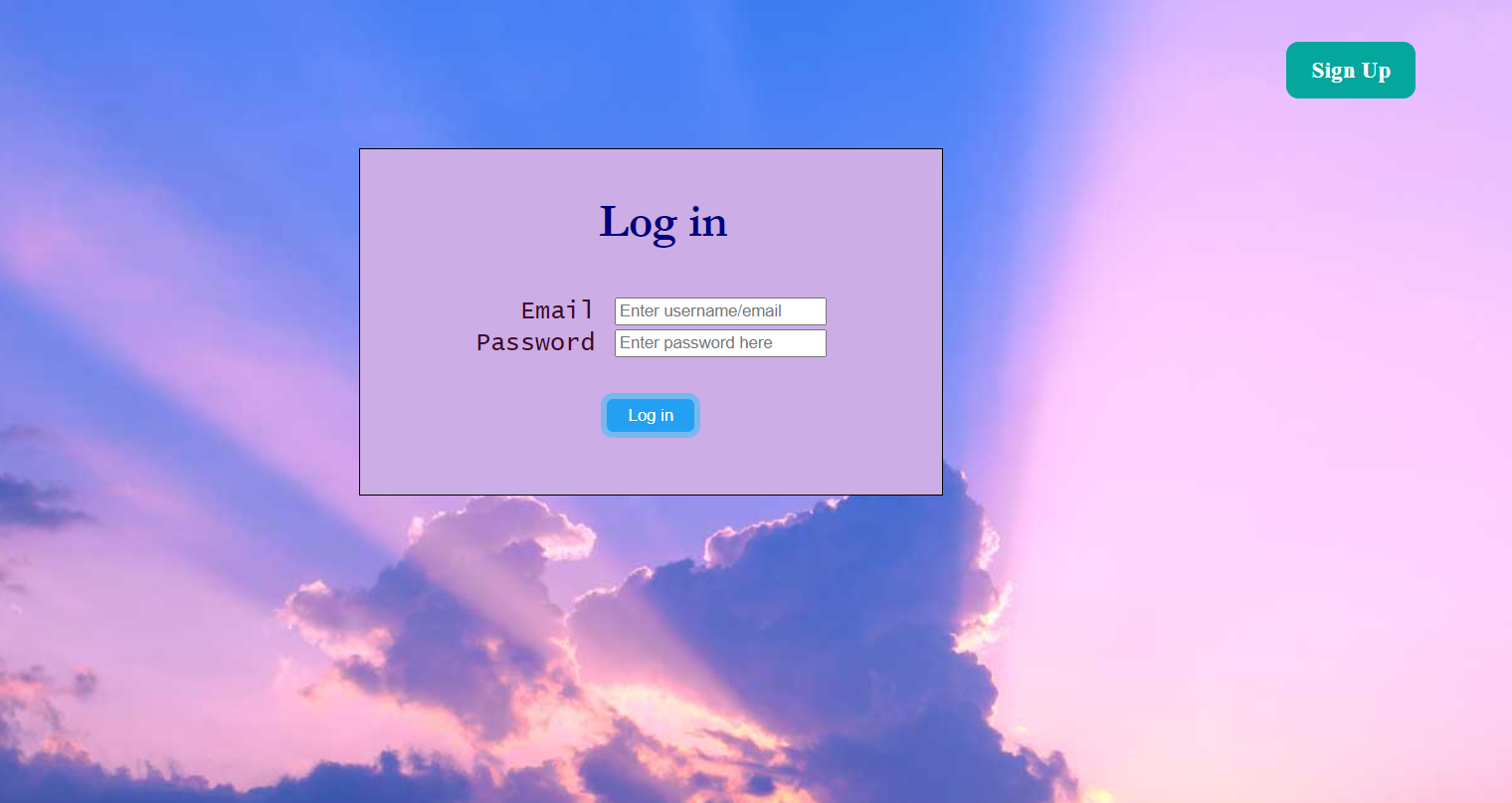
Table with Sequence 9

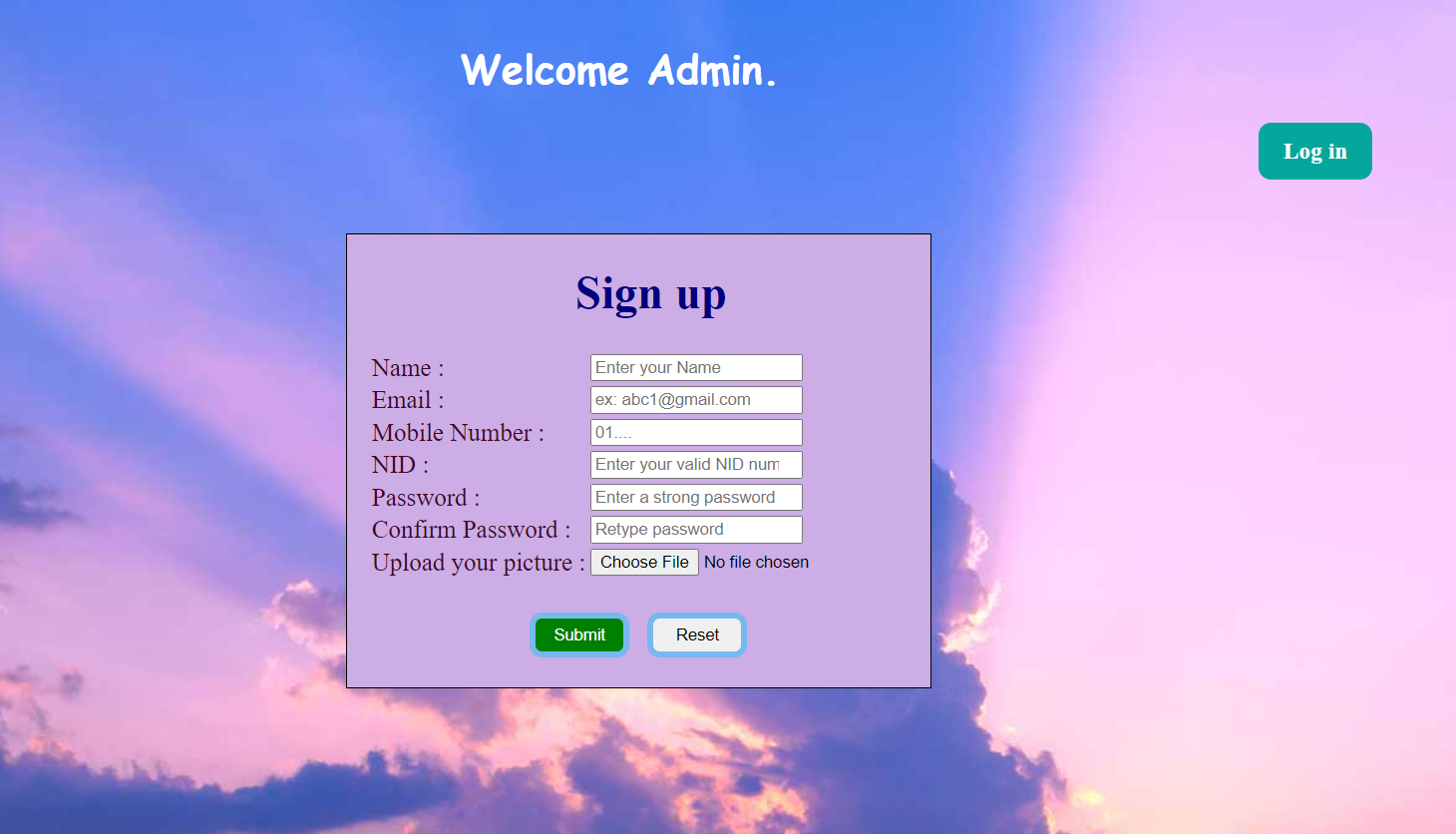
View 12

Procedures and Functions(Using Packages and Exceptions Handling) 14

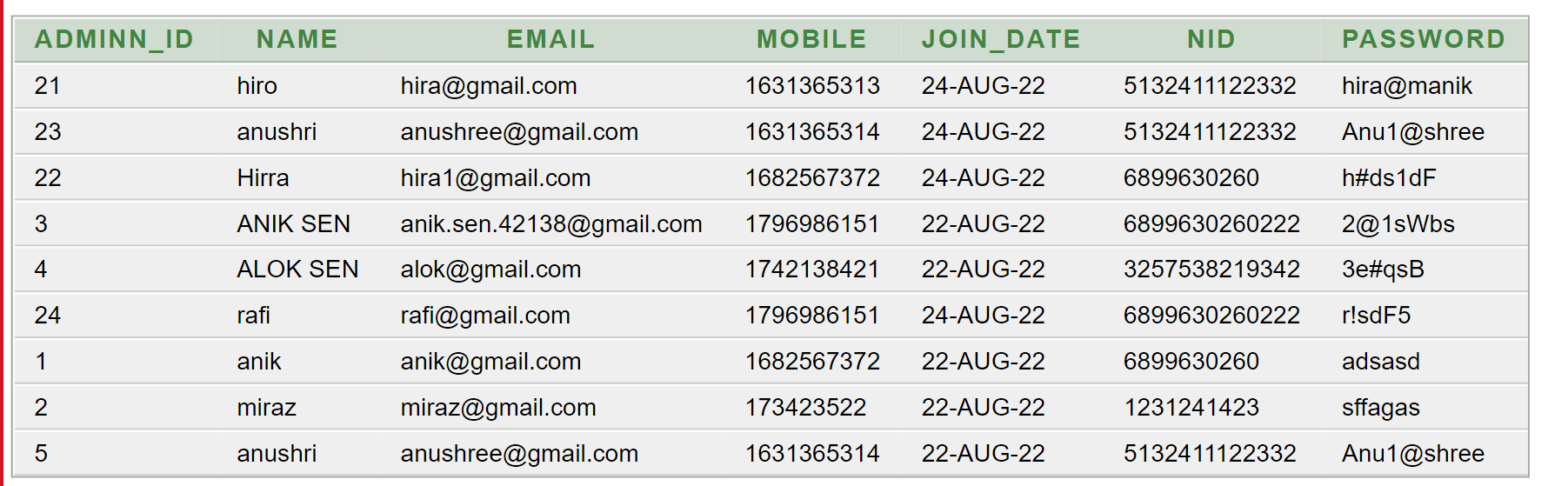
Triggers 18

* **We created the admin pannel page where an admin can log in to the system, if he doesn’t have an account he can easily create a new account from sign up option.**

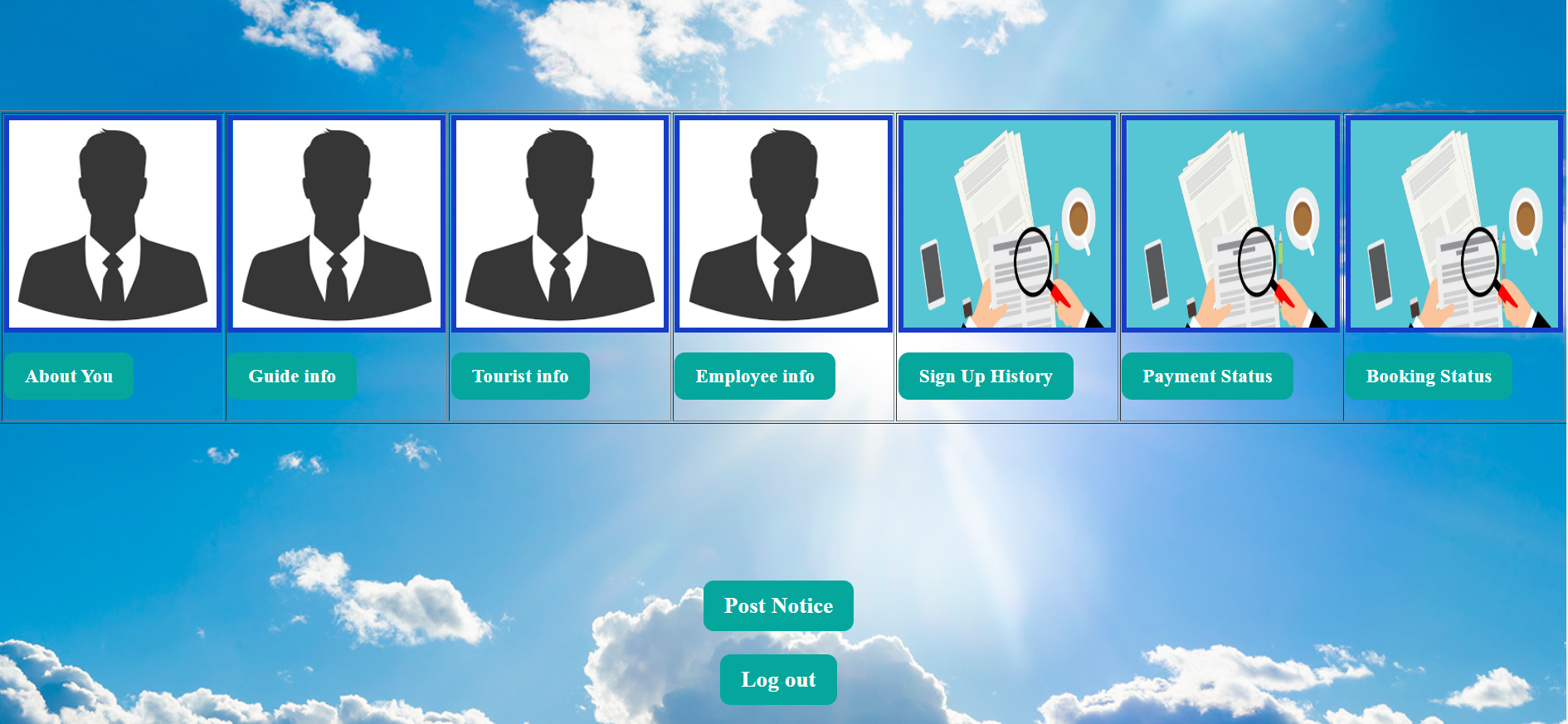




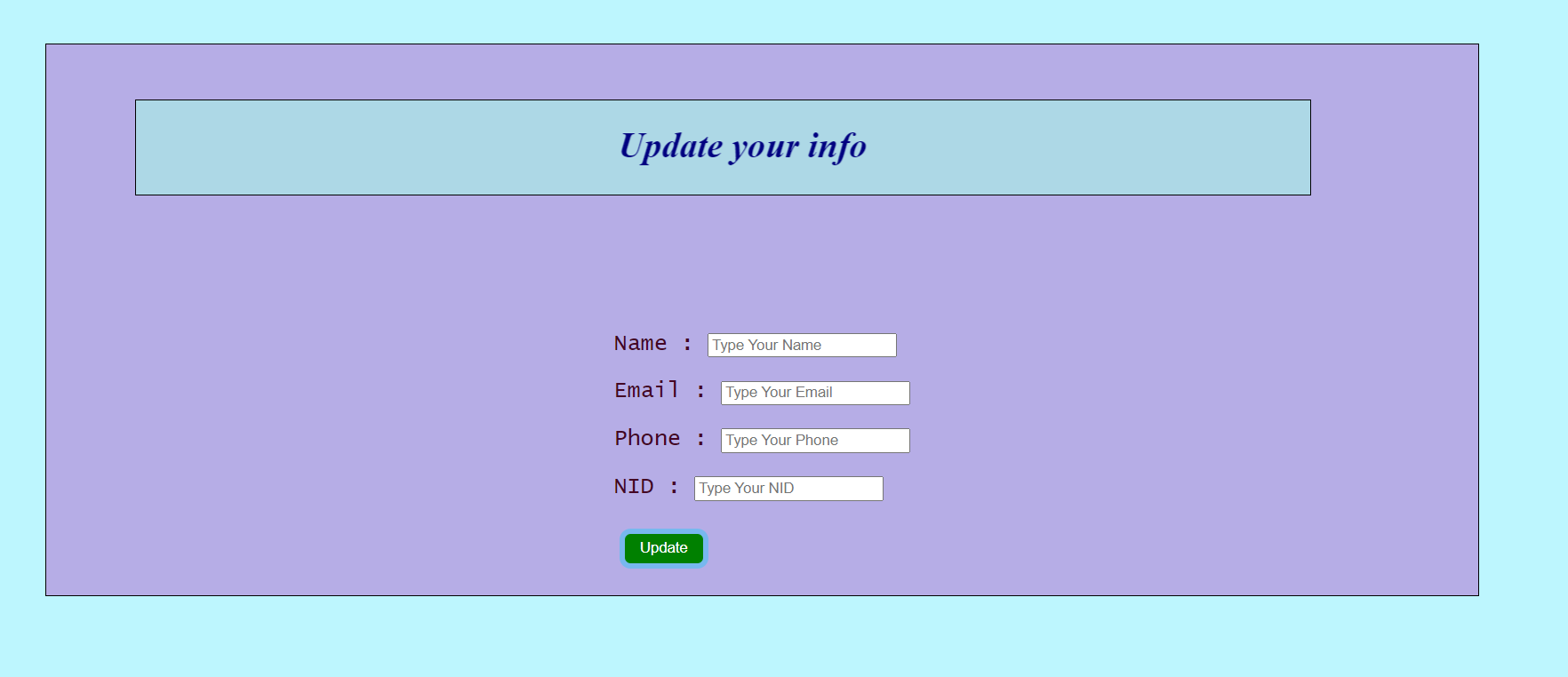
* Admin’s database where data stored after sign up.

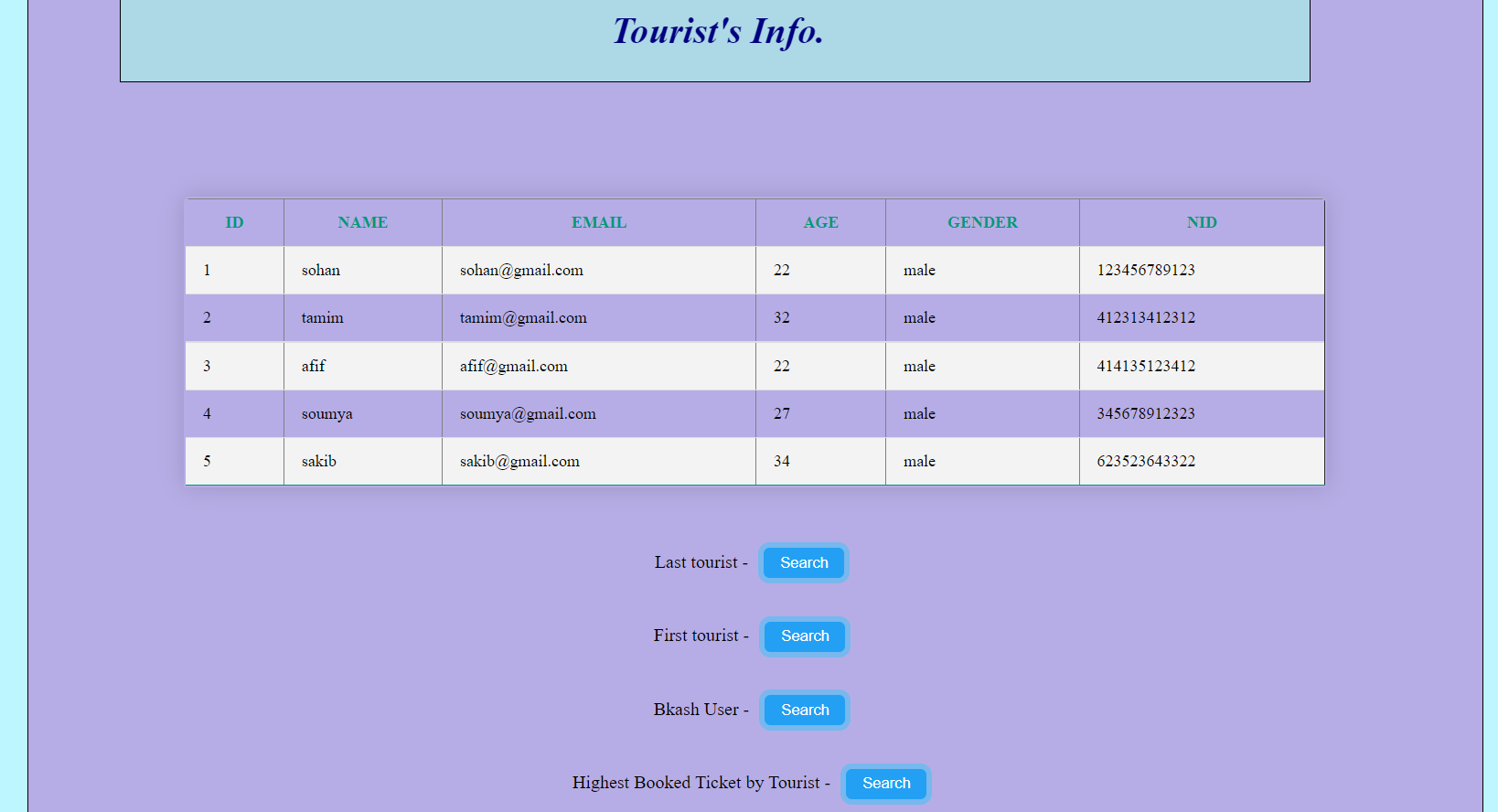


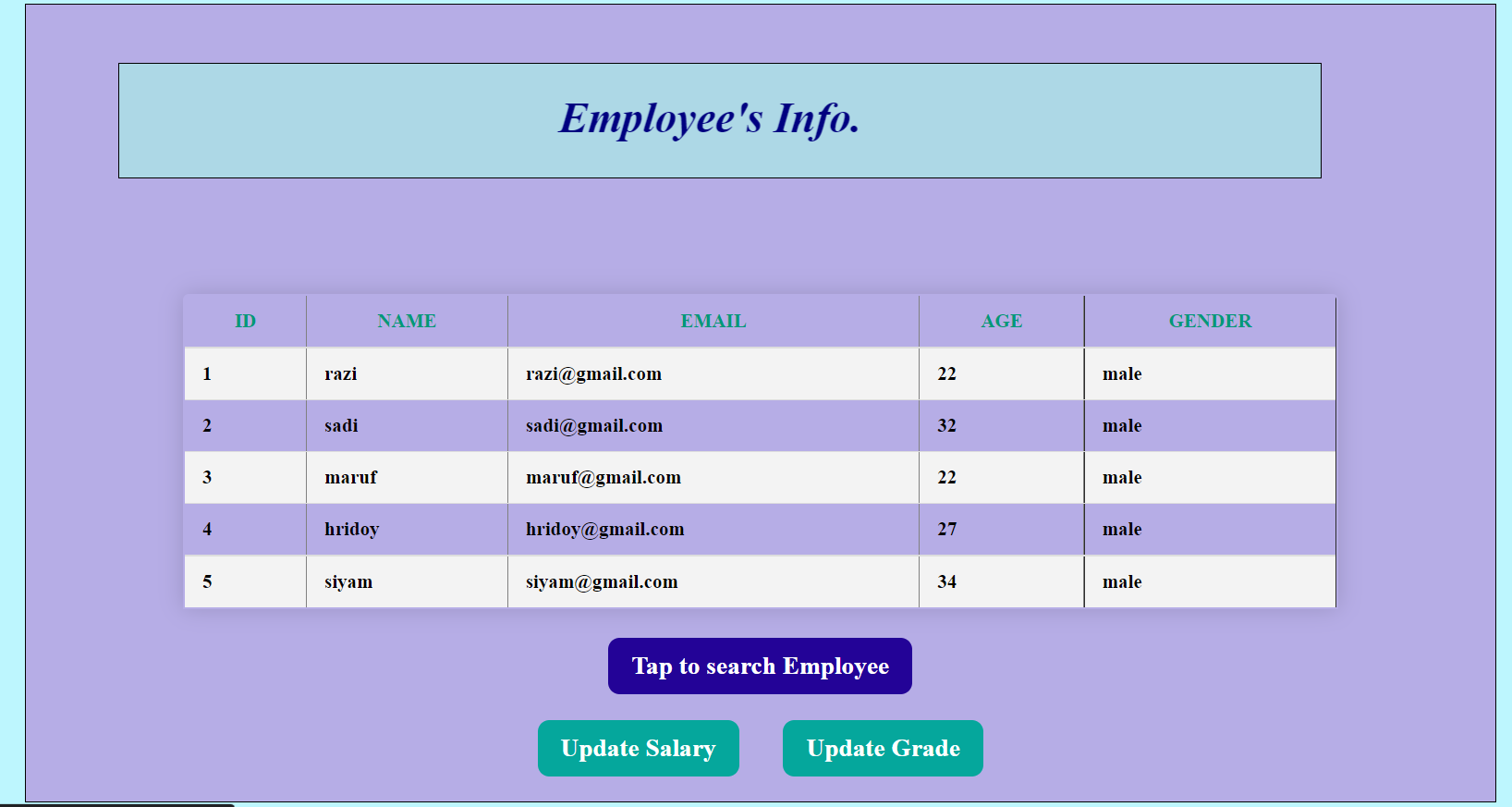
* **Interfaces to communicate with database.**

****

****

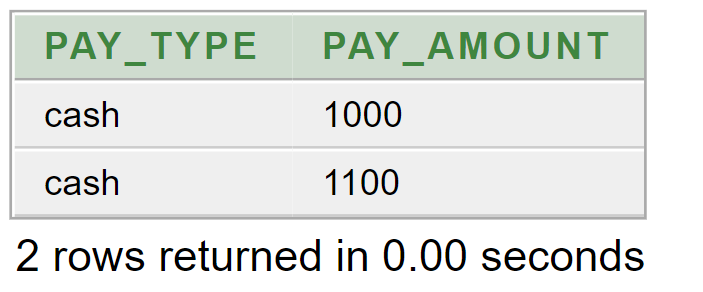






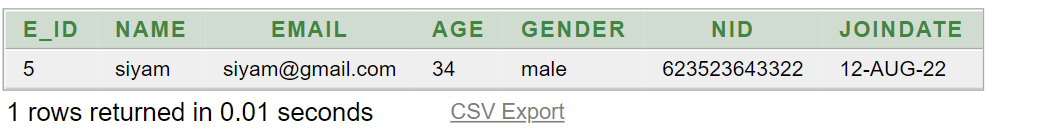
* **Different type of searching and advance searching. Some of them are attached from system.**
* **Display the payment method and amount considering the highest quantity of ticket.**

select pay\_type, pay\_amount from payment where quantity = (select max(quantity) from payment)

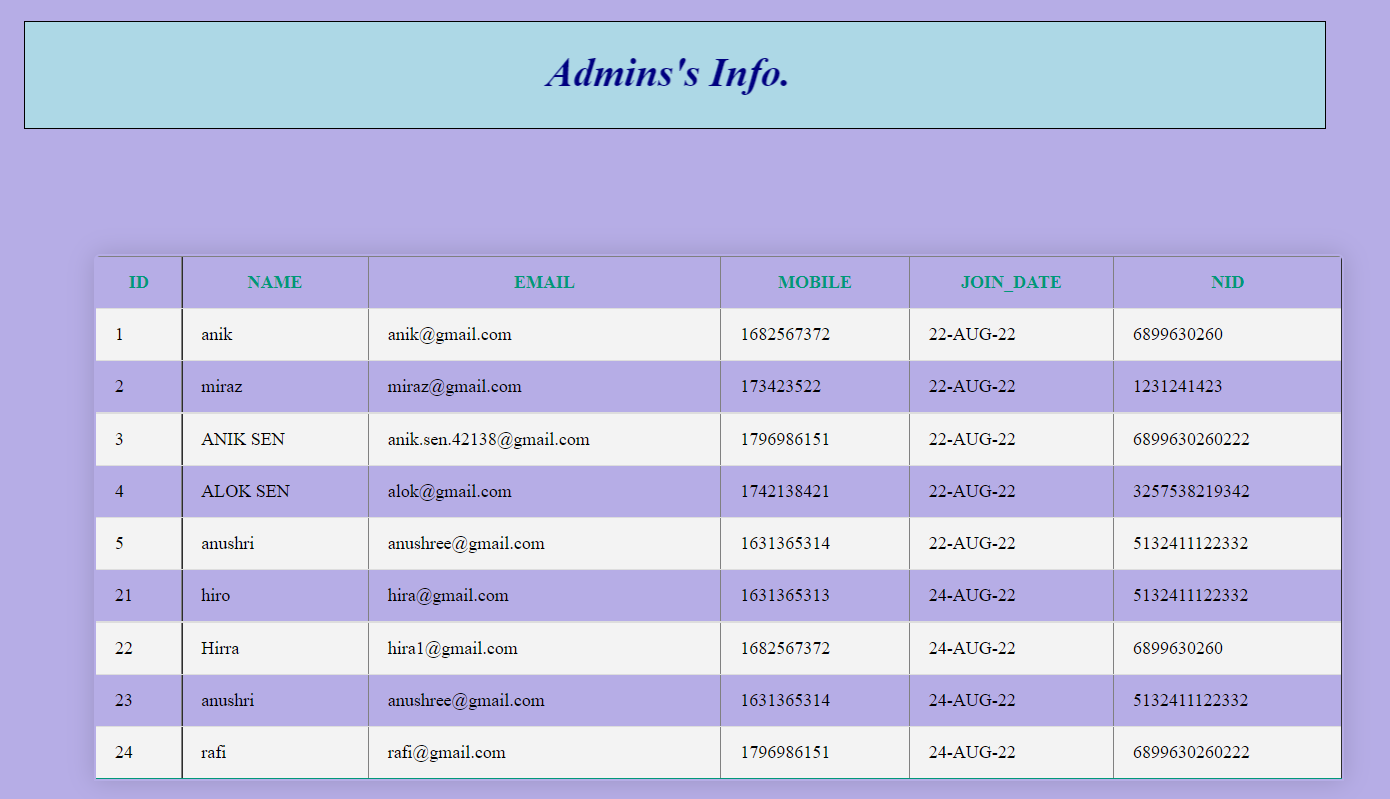


* **Display the employee details who joined after 'maruf' and age is more than 28**

select e.e\_id, e.name, e.email, e.age, e.gender, e.nid, e.joindate from emp e,emp m where m.name='maruf' and m.e\_id<e.e\_id and e.age>28;



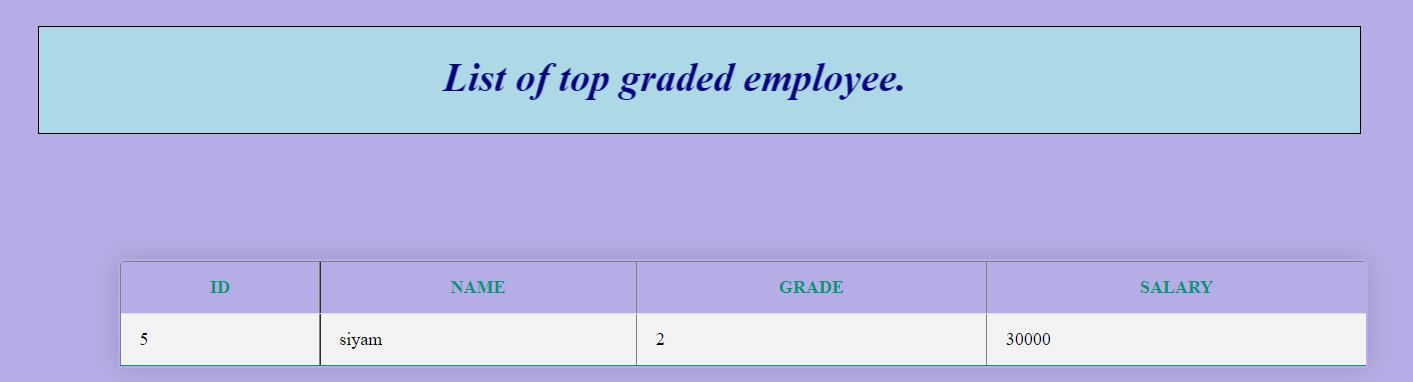
* **Display all the admins info order by admin id.**











* **Tables using sequence to maintain primary key**

**Table : adminn**

create table adminn (

adminn\_id number(30),

name varchar(30) not null,

email varchar(30) not null,

mobile number(11) not null,

join\_date date not null,

nid number(13) not null,

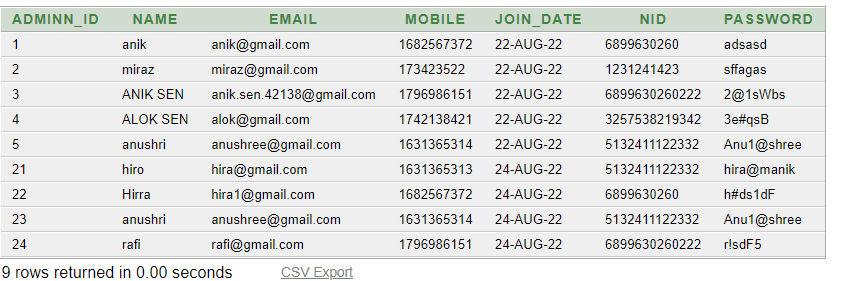
password varchar(20) not null,

primary key (adminn\_id));

CREATE SEQUENCE adminn\_id

START WITH 1

INCREMENT BY 1



**Table : tourist\_info**

create table tourist\_info(

t\_id number(20) not null,

name VARCHAR(50) not null,

email varchar(20) not null,

age number(30) not null,

gender varchar(20),

nid number(20) not null,

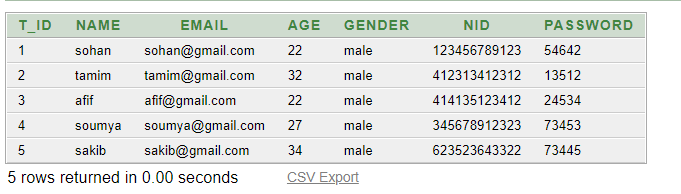
password varchar(8) not null,

primary key (t\_id));

CREATE SEQUENCE t\_id

START WITH 1

INCREMENT BY 1



**Table : emp**

create table emp(

e\_id number(20) not null,

name VARCHAR(50) not null,

email varchar(20) not null,

age number(30) not null,

gender varchar(20),

nid number(20) not null,

password varchar(8) not null,

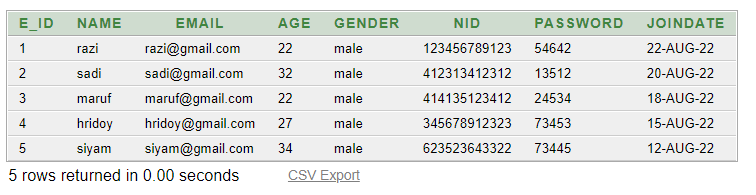
joindate date not null,

primary key (e\_id));

CREATE SEQUENCE e\_id

START WITH 1

INCREMENT BY 1



**Table : payment**

create table payment(

pay\_id number(20) not null,

quantity number(20) not null,

pay\_type varchar(20) not null,

pay\_amount number(20) not null,

t\_id number(20) not null,

primary key (pay\_id),

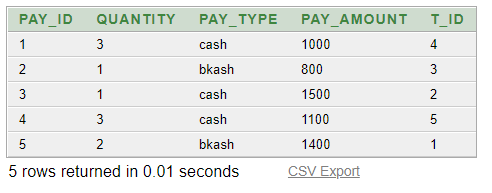
FOREIGN KEY (t\_id) REFERENCES tourist(t\_id)

);

CREATE SEQUENCE pay\_id

START WITH 1

INCREMENT BY 1



**Table : booking**

create table booking(

book\_id number(20) not null,

book\_type VARCHAR(20) not null,

pay\_id number(20) not null,

primary key (book\_id),

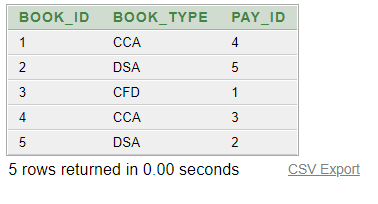
FOREIGN KEY (pay\_id) REFERENCES payment(pay\_id)

);

CREATE SEQUENCE book\_id

START WITH 1

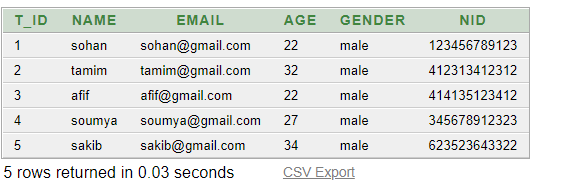
INCREMENT BY 1



* **View**
* CREATE OR REPLACE VIEW emp\_detail\_view AS

SELECT e\_id, NAME, email, age, gender

FROM emp

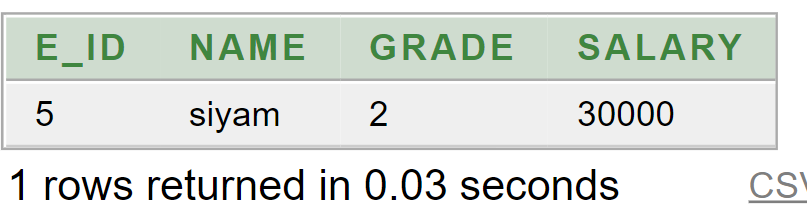


* CREATE or replace VIEW Hpaid\_emp\_view AS

select e.e\_id, e.name, d.grade, g.salary

from sal\_detail d, salgrade g, emp e

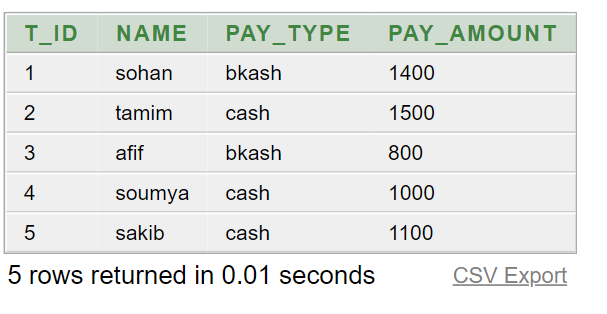
where d.e\_id = e.e\_id and d.grade = g.grade and g.grade<=2

****

* CREATE or Replace VIEW payment\_by\_user AS

SELECT t.t\_id, t.NAME, p.pay\_type, p.pay\_amount

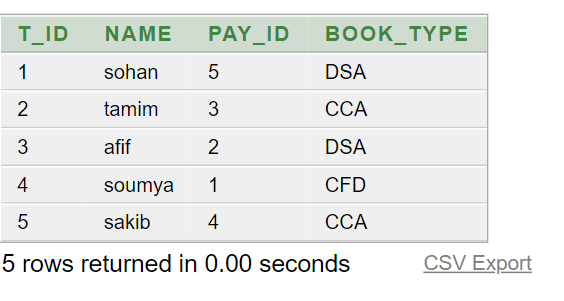
FROM tourist\_info t, payment p where t.t\_id = p.t\_id order by t.t\_id



* CREATE or Replace VIEW booked\_by\_user AS

SELECT t.t\_id, t.name, p.pay\_id, b.book\_type

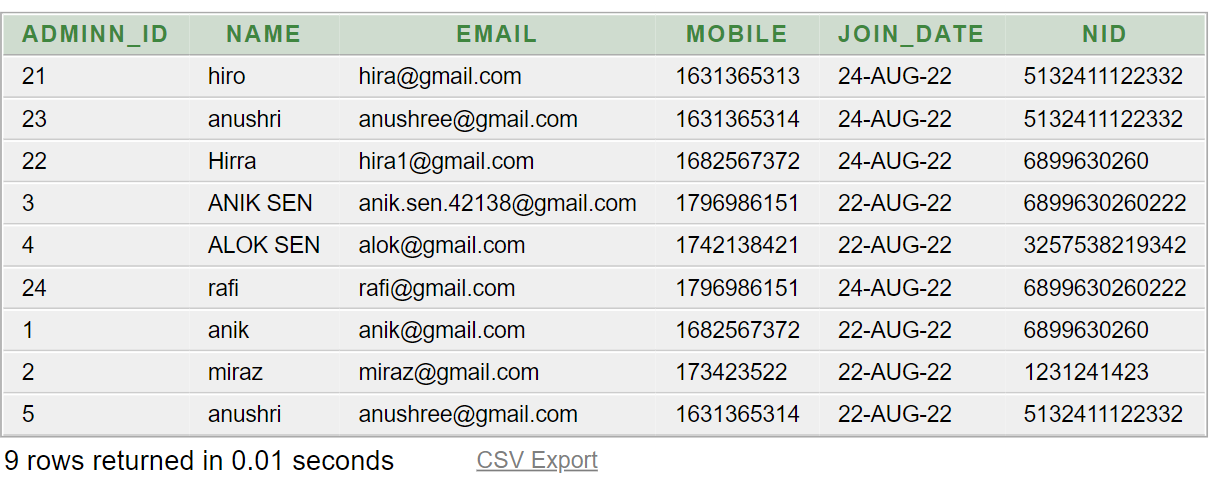
FROM tourist\_info t, payment p, booking b where t.t\_id = p.t\_id and p.pay\_id = b.pay\_id order by t.t\_id



* CREATE or replace VIEW admin\_detail\_view AS

SELECT adminn\_id, NAME, email, mobile, join\_date, nid

FROM adminn



* **Procedures and Functions (Using Package and Exception handling)**
* This package is used for admin’s registration and update admin’s info. There are two procedures under this package. We use some exception handling approach here

create or replace package pack\_admin\_add\_update

as

procedure proc\_add\_admin(aname adminn.name%type, mail adminn.email%type, phone adminn.mobile%type, nidd adminn.nid%type, pass adminn.password%type);

procedure proc\_update\_admin(aname adminn.name%type, mail adminn.email%type, phone adminn.mobile%type, nidd adminn.nid%type, pass adminn.password%type);

end pack\_admin\_add\_update;

create or replace package body pack\_admin\_add\_update

is

procedure proc\_add\_admin(aname adminn.name%type, mail adminn.email%type, phone adminn.mobile%type, nidd adminn.nid%type, pass adminn.password%type)

is

begin

insert into adminn (ADMINN\_ID,NAME,EMAIL,MOBILE,JOIN\_DATE,NID,PASSWORD) values (adminn\_id.nextval, aname, mail, phone, sysdate, nidd, pass);

end proc\_add\_admin;

procedure proc\_update\_admin(aname adminn.name%type, mail adminn.email%type, phone adminn.mobile%type, nidd adminn.nid%type, pass adminn.password%type)

is

begin

update adminn set name = aname, email = mail, mobile =phone, nid=nidd where password = pass;

end proc\_update\_admin;

end pack\_admin\_add\_update;

* This package is used for employee’s grade update, delete and insert purpose. There are three procedures and two functions under this package. We use some exception handling approach here

create or replace package emp\_grade

as

procedure emp\_grade\_up(id emp.e\_id%type, grd sal\_detail.grade%type);

procedure emp\_grade\_delete(id emp.e\_id%type);

procedure emp\_grade\_insert(id emp.e\_id%type, grd sal\_detail.grade%type);

end emp\_grade;

create or replace package body emp\_grade

is

function VALID\_EMPNO(id emp.e\_id%type)

return boolean

is

chk emp.e\_id%type;

begin

select e\_id into chk from emp where id = e\_id;

return true;

exception

when no\_data\_found then

return false;

end VALID\_EMPNO;

function VALID\_GRADE(grd salgrade.grade%type)

return boolean

is

chk salgrade.grade%type;

begin

select grade into chk from salgrade where grade = grd;

return true;

exception

when no\_data\_found then

return false;

end VALID\_GRADE;

procedure emp\_grade\_up(id emp.e\_id%type, grd sal\_detail.grade%type)

is

begin

if (VALID\_EMPNO(id)) then

if (VALID\_GRADE(grd)) then

update sal\_detail set grade=grd where e\_id=id;

dbms\_output.put\_line('Grade Updated');

else

dbms\_output.put\_line('Invalid Grade');

end if;

else

dbms\_output.put\_line('Employee ID does not exist');

end if;

end emp\_grade\_up;

procedure emp\_grade\_delete(id emp.e\_id%type)

is

begin

if (VALID\_EMPNO(id)) then

delete from sal\_detail where e\_id=id;

else

dbms\_output.put\_line('Employee ID does not exist');

end if;

end emp\_grade\_delete;

procedure emp\_grade\_insert(id emp.e\_id%type, grd sal\_detail.grade%type)

is

begin

if (VALID\_GRADE(grd)) then

insert into sal\_detail values(id, grd);

else

dbms\_output.put\_line('Invalid Grade');

end if;

end emp\_grade\_insert;

end emp\_grade;

* This package is used for employee’s salary update purpose. There is a procedure and a function under this package. We use some exception handling approach here

create package pack\_sal\_update

as

procedure proc\_sal\_update(grad salgrade.grade%type, sal salgrade.salary%type);

end pack\_sal\_update;

create or replace package body pack\_sal\_update

is

function VALID\_GRADE(grad salgrade.grade%type)

return boolean

is

chk salgrade.grade%type;

begin

select grade into chk from salgrade where grade = grad;

return true;

exception

when no\_data\_found then

return false;

end VALID\_GRADE;

procedure proc\_sal\_update(grad salgrade.grade%type, sal salgrade.salary%type)

is

begin

if (VALID\_GRADE(grad)) then

update salgrade set salary = sal where grade = grad;

dbms\_output.put\_line('Salary Updated.');

else

dbms\_output.put\_line('This Grade does not exist');

end if;

end proc\_sal\_update;

end pack\_sal\_update;

* This package is used for search some tourist information. There is a procedure named proc\_bkash\_user . We use some exception handling approach here

create package pack\_under\_tourist

as

procedure proc\_bkash\_user(id OUT tourist\_info.t\_id%type, tname OUT tourist\_info.name%type, temail OUT tourist\_info.email%type, tage OUT tourist\_info.age%type, tgender OUT tourist\_info.gender%type, tnid OUT tourist\_info.nid%type);

end pack\_under\_tourist;

create or replace package body pack\_under\_tourist

is

procedure proc\_bkash\_user(id OUT tourist\_info.t\_id%type, tname OUT tourist\_info.name%type, temail OUT tourist\_info.email%type, tage OUT tourist\_info.age%type, tgender OUT tourist\_info.gender%type, tnid OUT tourist\_info.nid%type)

is

cursor c1

is

select t\_id, name, email, age, gender, nid from tourist\_info where t\_id in (select t\_id from payment where pay\_type = 'bkash');

i number(25);

begin

for i in c1 loop

id := i.t\_id;

tname := i.name;

temail := i.email;

tage := i.age;

tgender := i.gender;

tnid := i.nid;

dbms\_output.put\_line(id||' '|| tname ||' '|| temail ||' '|| tage ||' '|| tgender||' '|| tnid);

end loop;

end proc\_bkash\_user;

end pack\_under\_tourist;

declare

id tourist\_info.t\_id%type;

tname tourist\_info.name%type;

temail tourist\_info.email%type;

tage tourist\_info.age%type;

tgender tourist\_info.gender%type;

tnid tourist\_info.nid%type;

begin

pack\_under\_tourist.proc\_bkash\_user(id, tname , temail , tage , tgender, tnid);

end;

* **Triggers**
* This trigger is used to keep the log of admin registration.
* create table admin\_reg\_log ( user\_name varchar(20), opt\_name varchar(20), opt\_date date);

create or replace trigger admin\_reg\_log

after insert on adminn

declare

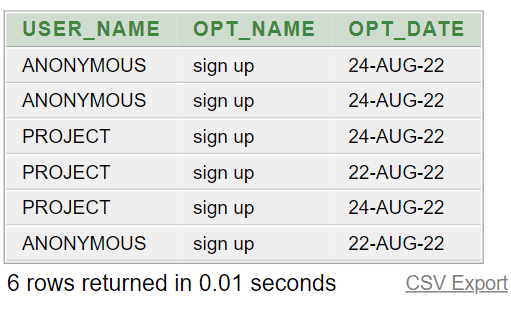
opt varchar(30);

begin

opt := 'sign up';

insert into admin\_reg\_log values(user,opt,sysdate);

end;



* This trigger is used to keep secure the employee registration process.
* create or replace trigger emp\_reg\_secure

before insert on emp

declare

begin

if to\_char(sysdate,'HH24') not between '8' and '17' or to\_char(sysdate,'DAY') in('FRIDAY', 'SATURDAY') then

raise\_application\_error(-20202, 'Please come in regular working working hour');

END IF;

End;

* This trigger is used to keep the log of employee registration.
* create table emp\_reg\_log ( user\_name varchar(20), opt\_name varchar(20), opt\_date date);

create or replace trigger emp\_reg\_log

after insert on emp

declare

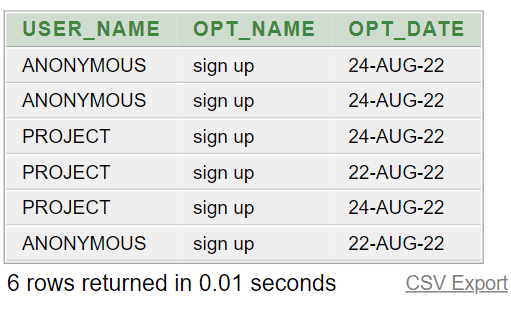
opt varchar(30);

begin

opt := 'sign up';

insert into emp\_reg\_log values(user,opt,sysdate);

end;



* This trigger is used to keep the log of employee’s grade update.
* create table emp\_grade\_uplog ( e\_id number(20), old\_grade number(20), up\_grade number(20), up\_date date);

create or replace trigger emp\_grade\_up

after update of grade on sal\_detail

for each row

begin

insert into emp\_grade\_uplog values (:old.e\_id, :old.grade, :new.grade, sysdate);

end;

* create or replace trigger sal\_alart

before update on salgrade

for each row

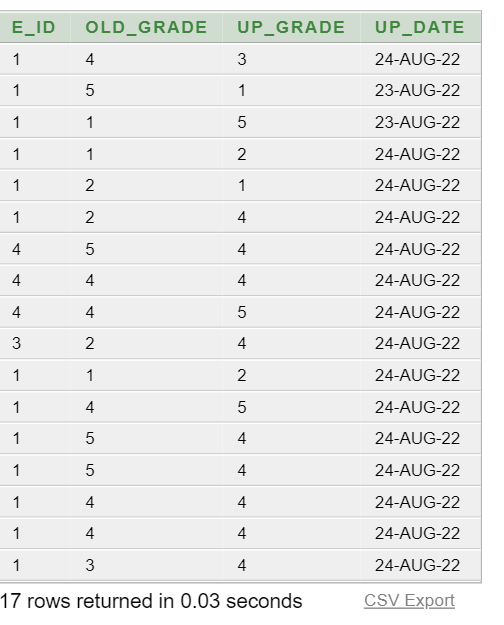
begin

if :new.salary < :old.salary then

raise\_application\_error(-20202, 'Warning: Updated salary must be greater then previous salary.');

END IF;

End;



* This trigger is used to keep the log of employee’s salary update.
* create table emp\_sal\_uplog ( grade number(20), old\_salary number(20), updated\_salary number(20), update\_date date);

create or replace trigger emp\_sal\_up

after update of SALARY on salgrade

for each row

begin

insert into emp\_sal\_uplog values (:old.grade, :old.salary, :new.salary, sysdate);

end;

