A report on

College Fest Event Management

By

2018103569

Niranjan.K

2018103626

Vivekananden

Submitted for the course

CS7411- Database Management Systems Laboratory

	Max	Assigned	
SQL	7		Evaluator Name and Signature
Views	3		
Procedures	5		
Functions	5		
Triggers	5		
Total			Date:

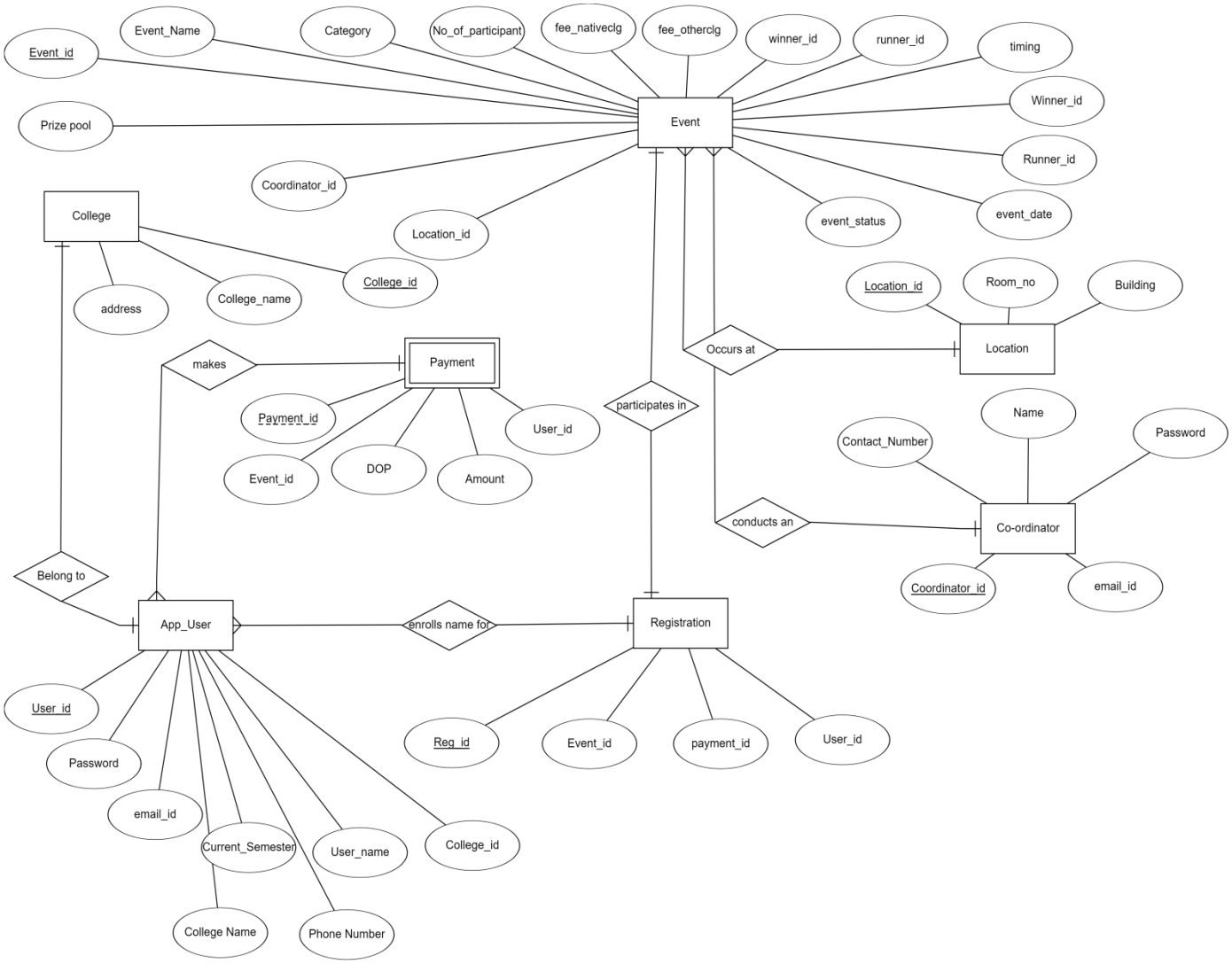
ER Diagram

(Use A3 sheet if necessary)

Note:

1. Entities have to be depicted with Keys, Cardinality, Attribute types.

2. Go through the naming conventions/symbols

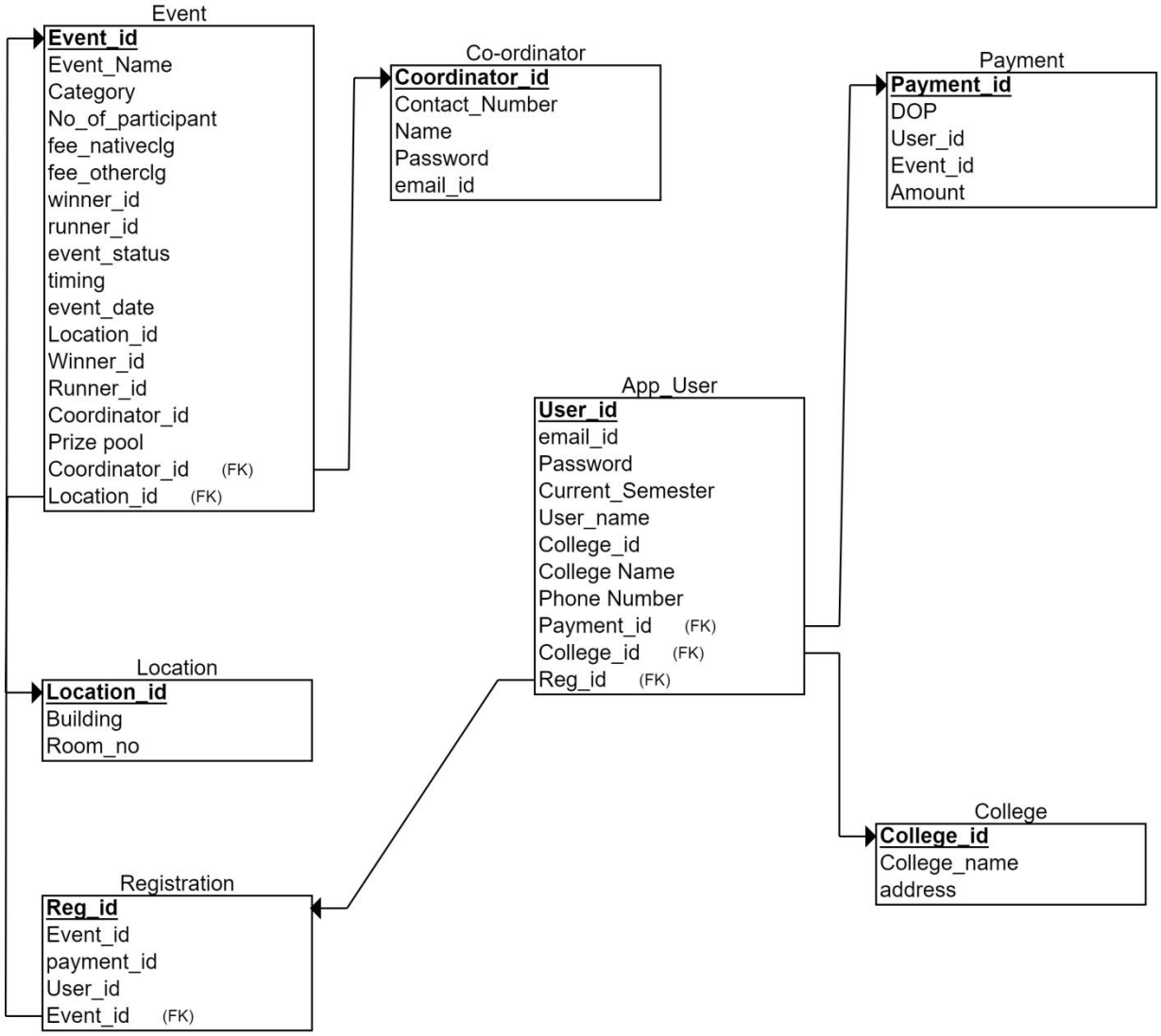


DB Schema

(Use A3 sheet if necessary)

Note:

1. Tables have to be depicted with Keys, Cardinality, Attribute types, relations.
2. Go through the naming conventions/symbols



Instances of each Relation (Snapshot)

COLLEGE

```
SQL> desc college
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COLLEGE_ID                                NOT NULL NUMBER(4)
 COLLEGE_NAME                              NOT NULL VARCHAR2(40)
 ADDRESS                                   NOT NULL VARCHAR2(40)

SQL>
```

```
SQL> select * from college;

COLLEGE_ID COLLEGE_NA ADDRESS
---------- ---------- ----------------------------------------
       501 CEG        Guindy
       502 MIT        Chromepet
       503 PSG        Coimbatore
       504 SSN        Chennai
       505 IIT        Madras
```

LOCATION

```
SQL> desc location
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LOCATION_ID                               NOT NULL NUMBER(4)
 BUILDING                                  NOT NULL VARCHAR2(20)
 ROOM_NO                                   NOT NULL NUMBER(4)

SQL>
```

```
SQL> select * from location;

LOCATION_ID BUILDING             ROOM_NO
----------- -------------------- ---------
        401 KP                       201
        402 KP                       202
        403 KP                       203
        404 Red Building              71
        405 Red Building              75
```

EVENT

```
SQL> desc event
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EVENT_ID                                  NOT NULL NUMBER(4)
 EVENT_NAME                                NOT NULL VARCHAR2(40)
 CATEGORY                                  NOT NULL VARCHAR2(40)
 NO_OF_PARTICIPANT                         NOT NULL NUMBER(2)
 FEE_NATIVECLG                             NOT NULL NUMBER(6)
 FEE_OTHERCLG                              NOT NULL NUMBER(6)
 WINNER_ID                                          NUMBER(6)
 RUNNER_ID                                          NUMBER(6)
 EVENT_STATUS                              NOT NULL VARCHAR2(40)
 TIMING                                    NOT NULL VARCHAR2(40)
 EVENT_DATE                                NOT NULL DATE
 PRIZE_POOL                                NOT NULL VARCHAR2(40)
 LOCATION_ID                               NOT NULL NUMBER(4)
 COORDINATOR_ID                            NOT NULL NUMBER(4)
```

```
SQL> select * from event;

  EVENT_ID EVENT_NAME                          CATEGORY                          NO_OF_PARTICIPANT FEE_NATIVECLG FEE_OTHERCLG  WINNER_ID  RUNNER_ID EVENT_STATUS
           TIMING         EVENT_DAT PRIZE_POOL          LOCATION_ID COORDINATOR_ID
---------- ---------------------------------------- ------------------------------------ ----------------- ------------- ------------ ---------- ---------- ------------
-------- --------------- --------- -------------------- ----------- --------------
       601 Ninja Coding                         Coding                                              15           100          200       2004       2001 completed
           14:00 to 16:00  01-FEB-20 400                        401           203
       602 OSPC                                 Coding                                              20           200          400       2003       2004 completed
           15:00 to 16:00  10-FEB-20 5000                       402           205
       603 Circuit Craze                        Electronics                                         50            50          100       2010       2009 completed
           10:00 to 14:00  07-FEB-20 500                        403           201
       604 Chaos Theory                         Management                                          20           100          300       2003       2001 completed
           11:00 to 19:00  10-FEB-20 2000                       403           202
       605 God Speed                            Flagship                                            50         10000        30000       2001       2003 completed
           12:00 to 18:00  10-FEB-20 3000                       405           204
```

CO - ORDINATOR

```
SQL> desc co_ordinator
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COORDINATOR_ID                            NOT NULL NUMBER(4)
 NAME                                      NOT NULL VARCHAR2(40)
 EMAIL_ID                                  NOT NULL VARCHAR2(40)
 PASSWORD                                  NOT NULL VARCHAR2(40)
 CONTACT_NUMBER                            NOT NULL NUMBER(10)

SQL>
```

```
SQL> select * from co_ordinator;

COORDINATOR_ID NAME                 EMAIL_ID             PASSWORD                             CONTACT_NUMBER
-------------- -------------------- -------------------- ------------------------------------ --------------
           201 Baktha               baktha@gmail.com     baktha123                               9000000000
           202 Lucha                lucha@gmail.com      lucha123                                8000000000
           203 Ucha                 ucha@hotmail.com     ucha123                                 7000000000
           204 Kacha                kacha@gmail.com      kacha123                                6000000000
           205 Bacha                bacha@yahoo.com      bacha123                                5000000000
```

APP USER

```
SQL> select * from app_user;

   USER_ID EMAIL_ID                             PASSWORD                             COLLEGE_NAME                         CURRENT_SEMESTER PHONE_NUMBER USER_NAM
E              COLLEGE_ID
---------- ---------------------------------- ------------------------------------ ------------------------------------ ---------------- ------------ --------
------------ ----------
       101 good@mail.com                      bhsbj                                CEG                                                 3   9987898348 Pallavan
                     501
       102 hbcw@hsds.com                      sjns                                 MIT                                                 5   8247924329 Madhavan
                     502
       103 fhfbe@ehbfew.com                   bewcbewk213                          SSN                                                 5   2734847234 Roman
                     504
       104 whdewe@kjjbdw.com                  ewgdwey                              IIT                                                 7   9374913481 Undertak
er                   505
       105 fwjefw@gmail.com                   ugwe324                              PSG                                                 3   8736284424 Batista
                     503
```

```
SQL> desc app_user
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                   NOT NULL NUMBER(4)
 EMAIL_ID                                  NOT NULL VARCHAR2(40)
 PASSWORD                                  NOT NULL VARCHAR2(40)
 COLLEGE_NAME                              NOT NULL VARCHAR2(40)
 CURRENT_SEMESTER                          NOT NULL NUMBER(4)
 PHONE_NUMBER                              NOT NULL NUMBER(10)
 USER_NAME                                 NOT NULL VARCHAR2(20)
 COLLEGE_ID                                NOT NULL NUMBER(4)
```

PAYMENT

```
SQL> select * from payment;

PAYMENT_ID    AMOUNT    USER_ID   EVENT_ID DOP
---------- ---------- ---------- ---------- ---------
       701        400        103        602 12-JAN-20
       702        500        105        602 10-JAN-20
       703        300        104        603 06-JAN-20
       704        200        101        605 08-JAN-20
       705        342        102        601 17-JAN-20
```

```
SQL> desc payment
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PAYMENT_ID                                NOT NULL NUMBER(4)
 AMOUNT                                    NOT NULL NUMBER(7)
 USER_ID                                   NOT NULL NUMBER(4)
 EVENT_ID                                  NOT NULL NUMBER(4)
 DOP                                       NOT NULL DATE

SQL> _
```

REGISTRATION

```
SQL> select * from registration;

   REG_ID  EVENT_ID PAYMENT_ID   USER_ID
--------- ---------- ---------- ----------
      301        601        701        101
      302        602        702        102
      303        603        703        103
      304        604        704        104
      305        605        705        105
```

```
SQL> desc registration
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REG_ID                                    NOT NULL NUMBER(4)
 EVENT_ID                                  NOT NULL NUMBER(4)
 PAYMENT_ID                                NOT NULL NUMBER(4)
 USER_ID                                   NOT NULL NUMBER(4)
```

SQL Queries

Note: You may use more than one construct in a question

#	Name	Tables Involved	Constructs Used (Mention accordingly)	Output
1	Participants from a particular college	College,App_user	Subquery	See SQL section below
2	Particulars of an event	Event,Location	Subquery	See SQL Section below
3	Details of students enrolled to a particular event	App_user,event ,payment	Joins	See SQL Section below
4	Event with maximum number of participants	event	Aggregate function	See SQL Section below
5	Building with how many rooms allocated for events	Location	Grouping ,subquery	See SQL function below
6	Total cash spent for festival	Event	Aggregate functions	See SQL function below

Views

#	Name	Tables Involved	Description	Output
1	Event_location_date	Event,location	Display the event name,building ,room no. and the date.	Go to views section below.
2	Max_payment	Payment,app_user	Display the user name who has made the max. payment	Go to views section below
3	Event_coords	Event,co_ordinator	Display the event name,coordinator name and phone number	Go to views section below
4	Prize_for_event	event	Display the prize pool for each event.	Go to views section below

Procedures

#	Name & Description	Tables Involved	IN & OUT Parameter(s)	Output
1	FEES A procedure to display all registration fee with event as input.	Event Registr ation	Event name Fee	See procedures section
2	LOC A procedure to display the location where event occurs	Location and Event	IN : Event name OUT:Location details	See procedures section
3	List_of_coordinator A procedure to display the list of all co-ordinators and phone number	Co-ordinator	NIL	See procedures section below
4	Event_status A procedure to display the status of all events	Event	NIL	See procedures section below

Functions

#	Name & Description	Tables Involved	Parameter(s)	Return Type	Output
1	Winner_id a function that gets winner_id of an event given as input the event_id	Event	Winner_id(event _id) := 603	number	105
2	No_of_events A function that returns number of events a coordinator conducts	Event	No_of_events(coord_id) := 201	number	1
3	Login A function that authorizes login_id and password	App_User	Login(user_id,password) := (101,’bhshj”)	varchar	Authorisation success
4	Reg_details A function that gives registration details of a particular reg_no	Registration	Reg_details(reg_id)	Rowtype%registration	See Functions Section below

Triggers

#	Name & Description	Trigger Type	Tables Involved	Output
1	No_payment_delete Deletion not allowed on payment table.	After Delete	Payment	Deletion on payment not allowed
2	Coord_id Auto increment coordinator id when inserted	Before Insert	Cordinator	Before insertion id field is incremented
3	Event_id Auto increment event id when inserted	Before insert	Event	Before insertion id field is incremented
4	Location_id Auto increment location id when inserted	Before insert	Location	Before insertion id field is incremented

VIEWS

1.Create a view to display an event's location and the date.

QUERY:

create or replace view event_location_date as select s.event_name,s.event_date,s.timing,ss.building,ss.room_no from event s natural join location ss;

```
SQL> create or replace view event_location_date as select s.event_name,s.event_date,s.timing,ss.building,ss.room_no from event  s natural join location  ss;

View created.

SQL> select * from event_location_date;

EVENT_NAME                             EVENT_DAT TIMING          BUILDING               ROOM_NO
-------------------------------------- --------- --------------- -------------------- ----------
Ninja Coding                           01-FEB-20 14:00 to 16:00  KP                        201
OSPC                                   10-FEB-20 15:00 to 16:00  KP                        202
Circuit Craze                          07-FEB-20 10:00 to 14:00  KP                        203
Chaos Theory                           10-FEB-20 11:00 to 19:00  KP                        203
God Speed                              10-FEB-20 12:00 to 18:00  Red Building               75
```

2.Create a view to display the name of the user who has made the maximum payment.

QUERY:

create or replace view max_payment as select user_name from app_user where user_id =(select user_id from payment where amount = (select max(amount) from payment))

```
SQL> create or replace view max_payment as select user_name from app_user where user_id =(select user_id from payment where amount = (select max(amount) from payment))
  2  ;

View created.

SQL> select * from max_payment;

USER_NAME
-------------------
Batista
```

3.Display the event name,co-ordinator name and phone number for each event through views

QUERY:

create or replace view event_coords as select event_name,name,contact_number from event natural join co_ordinator;

```
SQL> create or replace view event_coords as select event_name,name,contact_number from event natural join co_ordinator;

View created.

SQL> select * from event_coords;

EVENT_NAME                             NAME                 CONTACT_NUMBER
-------------------------------------- -------------------- --------------
Circuit Craze                          Baktha                   9000000000
Chaos Theory                           Lucha                    8000000000
Ninja Coding                           Ucha                     7000000000
God Speed                              Kacha                    6000000000
OSPC                                   Bacha                    5000000000
```

4.Create a view to display the prize amount for all events

QUERY:

create or replace view prize_for_event as select event_name,prize_pool from event

```
SQL> create or replace  view prize_for_event as select event_name,prize_pool from event;

View created.

SQL> select * from prize_for_event;

EVENT_NAME                                PRIZE_POOL
----------------------------------------- --------------------
Ninja Coding                              400
OSPC                                      5000
Circuit Craze                             500
Chaos Theory                              2000
God Speed                                 3000
```

FUNCTIONS

1.Create a function to get the winner of an event taking as input the event id.

FUNCTION:

```
create or replace function winner_id (eid in number)
return number
is winid number;
num number ;
begin
select winner_id into winid from event where event_id = eid;
return winid;
end;
/
```

FUNCTION CALL:

```
DECLARE
no number :=&no;
res number;
BEGIN
res := winner_id(no);
dbms_output.put_line('Winner id is:'||' '||res);
end;
```

/

```
SQL> create or replace function winner_id (eid in number)
  2  return number
  3  is winid number;
  4  num number ;
  5  begin
  6  select winner_id into winid from event where event_id = eid;
  7  return winid;
  8  end;
  9  /

Function created.

SQL>
SQL> DECLARE
  2  no number :=&no;
  3  res number;
  4  BEGIN
  5  res := winner_id(no);
  6  dbms_output.put_line('Winner id is:'||' '||res);
  7  end;
  8  /
Enter value for no: 603
old   2: no number :=&no;
new   2: no number :=603;
Winner id is: 105

PL/SQL procedure successfully completed.
```

2.Create a function to display the number of co-ordinators an event has.

FUNCTION:

create or replace function no_of_events(coord in number)
return number
is
counts number;
even_id number;
cursor NOE is

```
select event_id from event where
coordinator_id=coord;
begin
counts := 0;
open NOE;
LOOP
fetch NOE into even_id;
exit when NOE%NOTFOUND;
counts := counts + 1;
end loop;
close NOE;
return counts;
end;
/
```

FUNCTION CALL:

```
declare    coord number := &coord_id;
 chk number;
 begin
 chk := no_of_events(coord);
 dbms_output.put_line(coord || ' ' || 'has'|| ' ' ||chk|| ' ' ||'events');
 end;
 /
```

```
SQL> create or replace function no_of_events(coord in number)
  2  return number
  3  is
  4  counts number;
  5  even_id number;
  6  cursor NOE is
  7  select event_id from event where
  8  coordinator_id=coord;
  9  begin
 10  counts := 0;
 11  open NOE;
 12  LOOP
 13  fetch NOE into even_id;
 14  exit when NOE%NOTFOUND;
 15  counts := counts + 1;
 16  end loop;
 17  close NOE;
 18  return counts;
 19  end;
 20  /

Function created.

SQL>
SQL>  declare    coord number := &coord_id;
  2   chk number;
  3   begin
  4   chk := no_of_events(coord);
  5   dbms_output.put_line(coord || ' ' || 'has'|| ' ' ||chk|| ' ' ||'events');
  6   end;
  7   /
Enter value for coord_id: 202
old   1:  declare    coord number := &coord_id;
new   1:  declare    coord number := 202;
202 has 1 events

PL/SQL procedure successfully completed.
```

3.Create a function to authorize login for app users.

<u>FUNCTION DECLARATION:</u>

```
create or replace function login1(id in number,psd in varchar)
return number
is
psd_chk varchar2(20);
begin
select password into psd_chk from app_user where user_id =id;
if psd_chk = psd then
dbms_output.put_line('Correct password');
```

```
return 1;
else
dbms_output.put_line('Incorrect password');
return 0;
end if;
end;
/
```

FUNCTION CALL:

```
declare
user_id number := &user_id;
psd varchar2(20) := '&psd';
res number;
begin
res := login1(user_id,psd);
If res = 1 then
dbms_output.put_line('Success');
else
dbms_output.put_line('Failed');
end if;
end;
/
```

SQL Plus

```
SQL> create or replace function login1(id in number,psd in varchar)
  2  return number
  3  is
  4  psd_chk varchar2(20);
  5  begin
  6  select password into psd_chk from app_user where user_id =id;
  7  if psd_chk = psd then
  8  dbms_output.put_line('Correct password');
  9  return 1;
 10  else
 11  dbms_output.put_line('Incorrect password');
 12  return 0;
 13  end if;
 14
 15  end;
 16  /

Function created.

SQL>
SQL> declare
  2  user_id number := &user_id;
  3  psd varchar2(20) := '&psd';
  4  res number;
  5  begin
  6  res := login1(user_id,psd);
  7  If res = 1 then dbms_output.put_line('Success');
  8  else dbms_output.put_line('Failed');
  9  end if;
 10  end;
 11  /
Enter value for user_id: 103
old   2: user_id number := &user_id;
new   2: user_id number := 103;
Enter value for psd: bewcbewk213
old   3: psd varchar2(20) := '&psd';
new   3: psd varchar2(20) := 'bewcbewk213';
Correct password
Success

PL/SQL procedure successfully completed.
```

4.Create a function to display the registration details given the reg. number

FUNCTION:

```
create or replace function reg_details(regid in number)
return registration%rowtype
is
details registration%rowtype;
begin
select * into details from registration where reg_id = regid;return details;
```

```
end;
/
```

FUNCTION CALL:

```
declare
detailss registration%rowtype;
regid number := '&reg_id';
begin
detailss := reg_details(regid);
dbms_output.put_line(' ');
dbms_output.put_line('Registration id :'||detailss.reg_id);dbms_output.put_line('Event id:
'||detailss.event_id);
dbms_output.put_line('Payment id : '||detailss.payment_id);
dbms_output.put_line('User id :'||detailss.user_id);
end;
/
```

```
SQL> create or replace function reg_details(regid in number)
  2  return registration%rowtype
  3  is
  4  details registration%rowtype;
  5  begin
  6  select * into details from registration where reg_id = regid;return details;
  7  end;
  8  /

Function created.

SQL>
SQL> declare
  2  detailss registration%rowtype;
  3  regid number := '&reg_id';
  4  begin
  5  detailss := reg_details(regid);
  6  dbms_output.put_line(' ');
  7  dbms_output.put_line('Registration id :'||detailss.reg_id);dbms_output.put_line('Event id: '||detailss.event_id);
  8  dbms_output.put_line('Payment id : '||detailss.payment_id);
  9  dbms_output.put_line('User id :'||detailss.user_id);
 10  end;
 11  /
Enter value for reg_id: 303
old   3: regid number := '&reg_id';
new   3: regid number := '303';
Registration id :303
Event id: 603
Payment id : 703
User id :103

PL/SQL procedure successfully completed.
```

5.Create a function to display the runner id given as input the event id.

FUNCTION:

```
create or replace function runner_id (eid in number)
return number
is runid number;
num number ;
begin
select runner_id into runid from event where event_id = eid;
return runid;
end;
```

/

FUNCTION CALL:

```
DECLARE
no number :=&no;
res number;
BEGIN
res := runner_id(no);
dbms_output.put_line('Runner id is:'||' '||res);
end;
/
```

```
SQL> create or replace function runner_id (eid in number)
  2  return number
  3  is runid number;
  4  num number ;
  5  begin
  6  select runner_id into runid from event where event_id = eid;
  7  return runid;
  8  end;
  9  /

Function created.

SQL>
SQL> DECLARE
  2  no number :=&no;
  3  res number;
  4  BEGIN
  5  res := runner_id(no);
  6  dbms_output.put_line('Runner id is:'||' '||res);
  7  end;
  8  /
Enter value for no: 605
old   2: no number :=&no;
new   2: no number :=605;
Runner id is: 102

PL/SQL procedure successfully completed.
```

TRIGGERS

1.Create a trigger which prohibits the deletion on payment table.

Code:

```
create or replace trigger no_payment_delete
before delete
on payment
begin
raise_application_error(-20001,'You cant delete a Payment detail');
end;
/
```

```
SQL> create or replace trigger no_payment_delete
  2  before delete
  3  on payment
  4  begin
  5  raise_application_error(-20001,'You cant delete a Payment detail');
  6  end;
  7  /

Trigger created.
```

2.Create a trigger to automatically increment co_ordinator id when values are inserted into table.

CODE:

```
CREATE SEQUENCE coord_id START WITH 201;
CREATE OR REPLACE TRIGGER coord_id
BEFORE INSERT ON co_ordinator
FOR EACH ROW

BEGIN
  SELECT coord_id.NEXTVAL
  INTO   :new.coordinator_id
  FROM   dual;
END;
/
```

SQL Plus

```
SQL> CREATE SEQUENCE coord_id START WITH 201;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER coord_id
  2  BEFORE INSERT ON co_ordinator
  3  FOR EACH ROW
  4
  5  BEGIN
  6    SELECT coord_id.NEXTVAL
  7    INTO   :new.coordinator_id
  8    FROM   dual;
  9  END;
 10  /

Trigger created.
```

3.Create a trigger to automatically increment event id when values are inserted into table.

CODE:

```
CREATE SEQUENCE event_id START WITH 601;
CREATE OR REPLACE TRIGGER event_id
BEFORE INSERT ON event
FOR EACH ROW

BEGIN
  SELECT even_id.NEXTVAL
  INTO   :new.event_id
  FROM   dual;
END;
/
```

```
SQL> CREATE SEQUENCE event_id START WITH 601;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER event_id
  2  BEFORE INSERT ON event
  3  FOR EACH ROW
  4
  5  BEGIN
  6    SELECT even_id.NEXTVAL
  7    INTO   :new.event_id
  8    FROM   dual;
  9  END;
 10  /
```

4.Create a trigger to automatically increment location id when values are inserted into table.

CODE:

```
CREATE SEQUENCE location_id START WITH 401;
CREATE OR REPLACE TRIGGER locs_id
BEFORE INSERT ON location
FOR EACH ROW

BEGIN
  SELECT location_id.NEXTVAL
  INTO   :new.location_id
  FROM   dual;
END;
/
```

```
SQL> CREATE SEQUENCE location_id START WITH 401;

Sequence created.

SQL> CREATE OR REPLACE TRIGGER locs_id
  2  BEFORE INSERT ON location
  3  FOR EACH ROW
  4
  5  BEGIN
  6    SELECT location_id.NEXTVAL
  7    INTO   :new.location_id
  8    FROM   dual;
  9  END;
 10  /

Trigger created.
```

PROCEDURES

1.Create a procedure that gives the registration fee with event as input.

CODE:

```
create or replace procedure fees(name in event.event_name % type)
is
begin
declare
native event.fee_nativeclg%type;
other event.fee_otherclg%type;
cursor amount is
select fee_nativeclg,fee_otherclg from event where event_name=name;
begin
open amount;
loop
fetch amount into native,other;
exit when amount % notfound;
dbms_output.put_line(native||'                                    '||other);
end loop;
end;
end;
/
```

CALL:

```
declare
name varchar(15):= '&name';
begin
dbms_output.put_line('_____________________________________________________________________
_____________');
dbms_output.put_line('Native fees' || ' ' || 'Other clg fees');
fees(name);
end;
/
```

SQL Plus

```
SQL> set serveroutput on;
SQL> create or replace procedure fees(name in event.event_name % type)
  2  is
  3  begin
  4  declare
  5  native event.fee_nativeclg%type;
  6  other event.fee_otherclg%type;
  7  cursor amount is
  8  select fee_nativeclg,fee_otherclg from event where event_name=name;
  9  begin
 10  open amount;
 11  loop
 12  fetch amount into native,other;
 13  exit when amount % notfound;
 14  dbms_output.put_line(native||'          '||other);
 15  end loop;
 16  end;
 17  end;
 18  /

Procedure created.

SQL>
SQL> declare
  2  name varchar(15):= '&name';
  3  begin
  4  dbms_output.put_line('_____________________________________________________________');
  5  dbms_output.put_line('Native fees' || ' ' || 'Other clg fees');
  6  fees(name);
  7  end;
  8  /
Enter value for name: OSPC
old   2: name varchar(15):= '&name';
new   2: name varchar(15):= 'OSPC';
_____________________________________________________________
Native fees Other clg fees
200         400

PL/SQL procedure successfully completed.
```

2.Create a procedure to give the place where event occurs.

PROCEDURE:

```
create or replace procedure loc(name in event.event_name % type)
is
begin
declare
event_location location.location_id%type;
event_building location.building%type;
event_room_no location.room_no%type;
cursor one is
select l.location_id,building,room_no from location l,event e where l.location_id=e.location_id
and e.event_name=name;
begin
open one;
loop
fetch one into event_location,event_building,event_room_no;
exit when one % notfound;
dbms_output.put_line(event_location||'                    '||event_building||'
        '||event_room_no);
end loop;

end;
end;
/
```

CALL:

```
declare
name varchar(15):= '&name';
begin
dbms_output.put_line('_____________________________________________________________
_____________');
loc(name);
end;
/
```

```
SQL Plus
SQL> create or replace procedure loc(name in event.event_name % type)
  2  is
  3  begin
  4  declare
  5  event_location location.location_id%type;
  6  event_building location.building%type;
  7  event_room_no location.room_no%type;
  8  cursor one is
  9  select l.location_id,building,room_no from location l,event e where l.location_id=e.location_id and e.event_name=name;
 10  begin
 11  open one;
 12  loop
 13  fetch one into event_location,event_building,event_room_no;
 14  exit when one % notfound;
 15  dbms_output.put_line(event_location||'             '||event_building||'                '||event_room_no);
 16  end loop;
 17
 18  end;
 19  end;
 20  /

Procedure created.

SQL>
SQL>
SQL>
SQL> declare
  2  name varchar(15):= '&name';
  3  begin
  4  dbms_output.put_line('_____________________________________________________');
  5  loc(name);
  6  end;
  7  /
Enter value for name: Ninja Coding
old   2: name varchar(15):= '&name';
new   2: name varchar(15):= 'Ninja Coding';
_____________________________________________________
401             KP                 201

PL/SQL procedure successfully completed.
```

3.List the name of all co-ordinators and their phone numbers.

PROCEDURE:

```
create or replace procedure list_of_coordinator
is
begin
declare
v_name co_ordinator.name%type;
v_contact co_ordinator.contact_number%type;
cursor coord is
select name,contact_number from co_ordinator;
begin
open coord;
loop
fetch coord into v_name,v_contact;
exit when coord % notfound;
dbms_output.put_line(v_name||'              '||v_contact);
end loop;
end;
end;
/
```

CALL:

```
begin
list_of_coordinator();
end;
/
```

```
SQL> set serveroutput on;
SQL> create or replace procedure list_of_coordinator
  2  is
  3  begin
  4  declare
  5  v_name co_ordinator.name%type;
  6  v_contact co_ordinator.contact_number%type;
  7  cursor coord is
  8  select name,contact_number from co_ordinator;
  9  begin
 10  open coord;
 11  loop
 12  fetch coord into v_name,v_contact;
 13  exit when coord % notfound;
 14  dbms_output.put_line(v_name||' '||v_contact);
 15  end loop;
 16  end;
 17  end;
 18  /

Procedure created.

SQL>
SQL> set serveroutput on;
SQL> begin
  2  list_of_coordinator();
  3  end;
  4  /
Baktha 9000000000
Lucha 8000000000
Ucha 7000000000
Kacha 6000000000
Bacha 5000000000

PL/SQL procedure successfully completed.
```

4. Create a procedure to list the status of all events.

PROCEDURE:

```
create or replace procedure participant
is
begin
declare
v_name event.event_name%type;
v_event event.event_status%type;
cursor parti is
```

```
select event_name,event_status from event where event_status='completed' or
event_status='underway' or event_status='yet to start';
begin
open parti;
loop
fetch parti into v_name,v_event;
exit when parti % notfound;
dbms_output.put_line(v_name||' '||v_event);
end loop;
end;
end;
/
```

CALL:

```
begin
participant();
end;
/
```

```
SQL> create or replace procedure participant
  2  is
  3  begin
  4  declare
  5  v_name event.event_name%type;
  6  v_event event.event_status%type;
  7  cursor parti is
  8  select event_name,event_status from event where event_status='completed' or event_status='underway' or event_status='yet to start';
  9  begin
 10  open parti;
 11  loop
 12  fetch parti into v_name,v_event;
 13  exit when parti % notfound;
 14  dbms_output.put_line(v_name||' '||v_event);
 15  end loop;
 16  end;
 17  end;
 18  /

Procedure created.

SQL>
SQL> set serveroutput on;
SQL> begin
  2  participant();
  3  end;
  4  /
Ninja Coding completed
OSPC completed
Circuit Craze completed
Chaos Theory completed
God Speed completed

PL/SQL procedure successfully completed.
```

SQL

1.List all the participants from a particular college.

QUERY:

select user_name,email_id,user_id,current_semester,phone_number from app_user a,college c where a.college_id=c.college_id and c.college_name='&College_Name';

```
SQL> select user_name,email_id,user_id,current_semester,phone_number from app_user a,college c where a.college_id=c.college_id and c.college_name='&College_Name';
Enter value for college_name: MIT
old   1: select user_name,email_id,user_id,current_semester,phone_number from app_user a,college c where a.college_id=c.college_id and c.college_name='&College_Name'
new   1: select user_name,email_id,user_id,current_semester,phone_number from app_user a,college c where a.college_id=c.college_id and c.college_name='MIT'

USER_NAME            EMAIL_ID                                 USER_ID CURRENT_SEMESTER PHONE_NUMBER
-------------------- ---------------------------------------- ---------- ---------------- ------------
Madhavan             hbcw@hsds.com                               102                5   8247924329
```

2.Given a date diplay the event name,building,roomno,registration fee.

QUERY:

select event_name,building,room_no,fee_nativeclg,fee_otherclg from event e,location l where e.location_id=l.location_id and event_date='&date';

```
SQL> select event_name,building,room_no,fee_nativeclg,fee_otherclg from event e,location l where e.location_id=l.location_id and event_date='&date';
Enter value for date: 10-feb-2020
old   1: select event_name,building,room_no,fee_nativeclg,fee_otherclg from event e,location l where e.location_id=l.location_id and event_date='&date'
new   1: select event_name,building,room_no,fee_nativeclg,fee_otherclg from event e,location l where e.location_id=l.location_id and event_date='10-feb-2020'

EVENT_NAME                             BUILDING               ROOM_NO FEE_NATIVECLG FEE_OTHERCLG
---------------------------------------- -------------------- ---------- ------------- ------------
OSPC                                   KP                        202           200          400
Chaos Theory                           KP                        203           100          300
God Speed                              Red Building               75         10000        30000
```

3. List all the participants registered for a particular event(Take event name as input)

QUERY:

select user_name,email_id,a.user_id,current_semester,phone_number,event_name from app_user a,event e,payment p where a.user_id=p.user_id and e.event_id=p.event_id and e.event_name='&Event_Name';

```
SQL> select user_name,email_id,a.user_id,current_semester,phone_number,event_name from app_user a,event e,payment p where a.user_id=p.user_id and e.event_id=p.event_id and
e.event_name='&Event_Name';
Enter value for event_name: OSPC
old   1: select user_name,email_id,a.user_id,current_semester,phone_number,event_name from app_user a,event e,payment p where a.user_id=p.user_id and e.event_id=p.event_id
and e.event_name='&Event_Name'
new   1: select user_name,email_id,a.user_id,current_semester,phone_number,event_name from app_user a,event e,payment p where a.user_id=p.user_id and e.event_id=p.event_id
and e.event_name='OSPC'

USER_NAME            EMAIL_ID                                 USER_ID CURRENT_SEMESTER PHONE_NUMBER EVENT_NAME
-------------------- ---------------------------------------- ---------- ---------------- ------------ ----------------------------------------
Roman                fhfbe@ehbfew.com                             103                5   2734847234 OSPC
Batista              fwjefw@gmail.com                             105                3   8736284424 OSPC
```

4. Give which event can have maximum number of participants

select no_of_participant,event_name from event where no_of_participant in (select max(no_of_participant) from event);

```
SQL> select no_of_participant,event_name from event where no_of_participant in (select max(no_of_participant) from event
);

NO_OF_PARTICIPANT EVENT_NAME
----------------- ------------------------------------
               50 Circuit Craze
               50 God Speed
```

4. Give which building has how many rooms allocated for the event

select count(room_no),building from location group by building;

```
SQL> select count(room_no),building from location group by building;

COUNT(ROOM_NO) BUILDING
-------------- --------------------
             3 KP
             2 Red Building
```

5. give the total prize pool of the event

select sum(prize_pool) from event;

```
SQL> select sum(prize_pool) from event;

SUM(PRIZE_POOL)
---------------
          10900
```