FIT INDIA FITNESS ACTIVITY FARE MANAGEMENT SYSTEM

Prepared For

DATABASE MANAGEMENT SYSTEM

(CSE2004)

PROJECT COMPONENT

Submitted To

Dr. Nithya S

Associate Professor School of Computer Science and Engineering

Submitted by

NIRBHAY S GANDHI

19BCE0273

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

FIT INDIA, FITNESS ACTIVITY FARE MANAGEMENT SYSTEM

REVIEW REPORT
Submitted by

NIRBHAY S GANDHI 19BCE0273
SHAIK MOHAMMAD SUHAIL 19BCE0226
DEV BILASPURE 19BCI0139

Prepared For

DATABASE MANAGEMENT SYSTEM (CSE2004)
PROJECT COMPONENT

Submitted To
Dr. Nithya S
Associate Professor

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Table of Content

1. Introduction:

1.1 Background:

Our topic is " Fit India, Fitness activity fair Management system", this is been made using a Relational database Management system.

Our socity has been suffring from the a monotonus and a stressful life style and therefore they now they are struggling from many desieses such as knee cramps at an early age, neck pain, spondalitees, migrane, low immunity, etc.

So RED BULL beverages, DELL India, MICROSOFT India has taken an intitiative to make a change in their daily life, We have organized a fitness activity fair.

And our group has been given incharge to manage this fair.

1.2 Objective:

We have tried to come up with an application which is made to register for the events that are being held in our Fitness Activity fair. The user can be participant and administrator. The admin can update the information about the events and can retrieve transactions involved. The participant will be able to look for the events being held and can participate by making payments for the registered events.

We have several clubs some of them are sponcered by many big firms like M.G MOTORS , KIA, TATA MOTORS, GAP, MAHINDRA HEALTH LIFE, RED BULL.

But ofcourse for every event a participant has to play, but a very minimal amount!.

And these events are being organized at KALAPURAM PARTY PLOT.

There many subsections are created to conduct every event. Each event has been allocated a controller and he controlles his/her allocated event.

1.3 Motivation:

We have used the use of Relational database management system to deal with registration of the events and the transactions involved. Relational database management system (RDBMS) .A relational database

refers to a database that stores data in a structured format, using rows and columns. This makes it easy to locate and access specific values within the database. It is "relational" because the values within each table are related to each other. Tables may also be related to other tables. The relational structure makes it possible to run queries across multiple tables at once.

1.4 Contribution of the project:

We have used the concept of relational database, which will allow the user (may be participant or organizer) to store the details of the candidates.

They can be able to store the details of the organizing Clubs, sponcer for that club, which event they will organize, where the event will be held, time, date, duration, etc.

1.5 Organization of the project:

We have used a databases namely FAIR. The database “FAIR” has 7 relations mapped among each other namely, Club, Venue and Event_Details, Event, Participant, Card and Trans_history. The E-R diagram for the same is shown below.

2. Project Resource Requirements:

2.1 Software Requirements:

For backend:

Oracle 11g or MySQL or any online SQL editor will work.

For frontend:

Java, python, php.

2.2 Hardware Requirements:

As such no dedicated hardware is required, daily use hardware such as a basic pc/laptop, keyboard are enough.

3. Litrature Suryey:

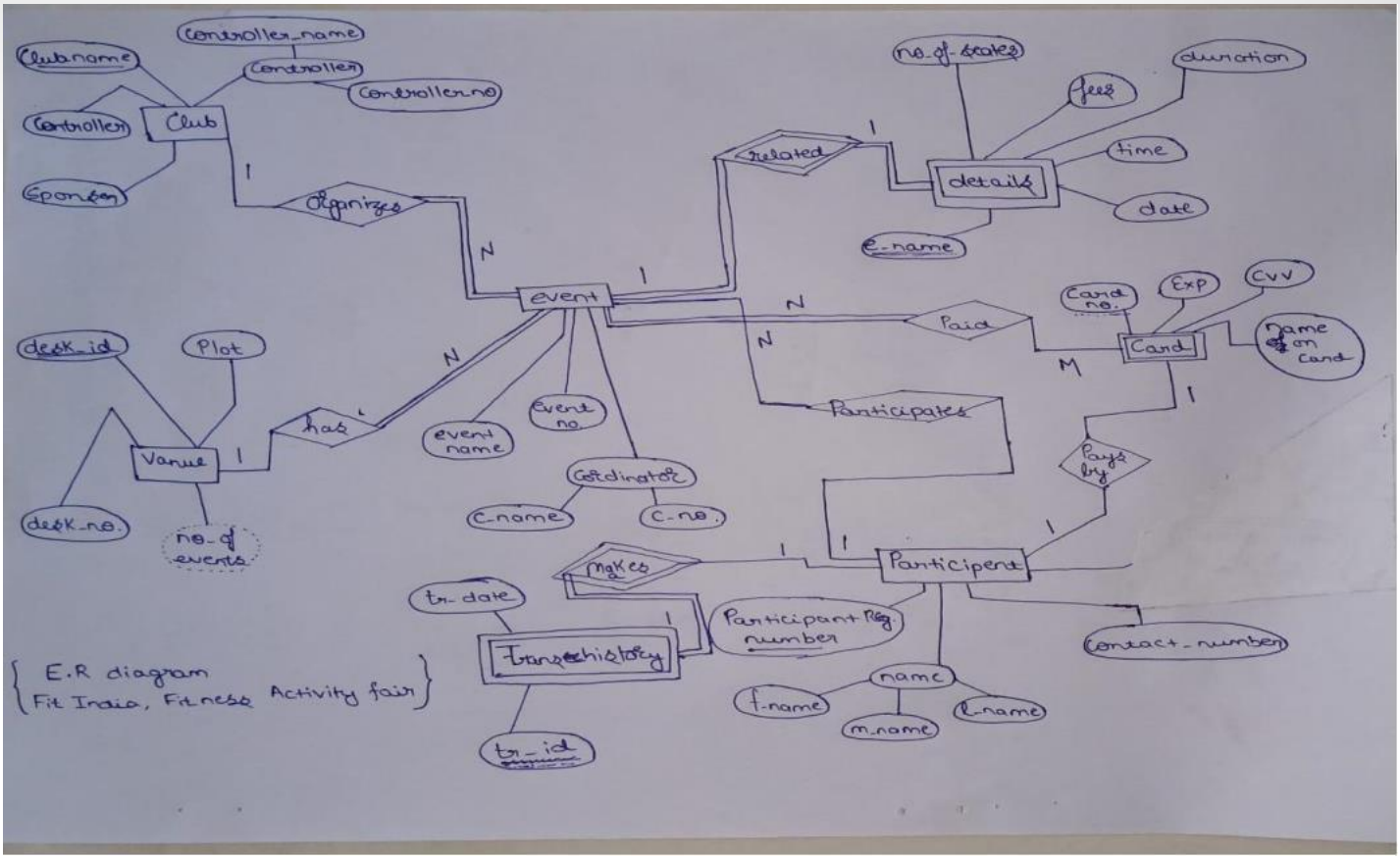
It is a client developer project, therefore does not require any survey. It works as per the requirement of the client.

3.1 Data Collection:

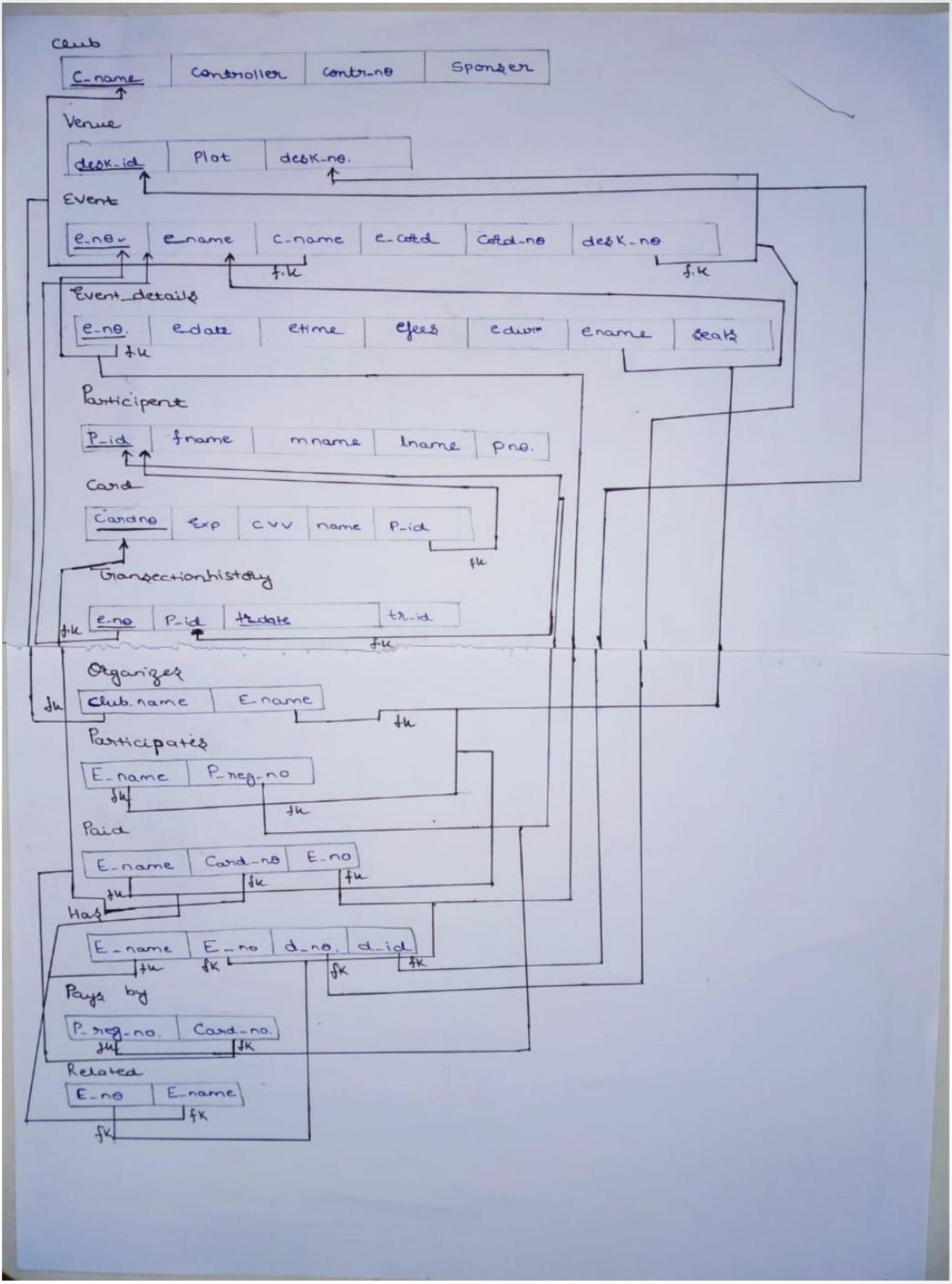
Data is collected from the client and we have collected data such as Club admin can store the details of the club from the member sponcers, Details of the event from the from the club like which event they are organizing and on which date as well where is the venue for that event. Then the event coordinaters have the data for the active enthusiast participients, participients will be given a unique Id, then we have the details of the payment that the participient has paid for each event, we have the data of their transection history and card. So, all these Data will be shown in the form of the Entities so as to construct the table. So, we have the entities as Club, Event, Event_Det, Participient, Card, Trand_history. So, now based on these data we will proceeding with the table creation and other expects of it.

4. Design of the Project:

4.1 E-R Diagram:



4.2 E-R Diagram to Relational Mapping:



4.3 Table and Constaints:

Club

Attribute	Datatype	Constraint
C_name	varchar	Primary key and Foreign key to Event (C_name) on delete cascade
Controller	varchar	not null
Controller_no	number	not null
Sponcer	varchar	

Venue

Attribute	Datatype	Constraint
Desk_id	number	Primary key
Plot_name	varchar	not null
Desk_no	varchar	not null

Event

Attribute	Datatype	Constraint
E_no	varchar	Primary key and Foreign key to Event_details (E_no) and Trans_history (E_no) on delete cascade
E_name	varchar	not null
E_cord	varchar	not null
Cord_no	number	not null

Event_details

Attribute	Datatype	Constraint
E_date	date	not null
E_time	timestamp	not null
E_fees	numeric	default 0
E_duration	varchar2	
E_name	varchar2	not null
seates	number	default 0

Participent

Attribute	Datatype	Constraint
P_id	varchar	Primary key and Foreign key to Trans_history (P_id) and Card (P_id) on delete cascade
f_name	varchar	not null
m_name	varchar	
l_name	varchar	

P_no	number	Foreign key to Card (P_id) and Trans_history (P_id) on delete cascade

Card

Attribute	Datatype	Constraint
Card_no	number	not null
EXP	date	not null
CVV	number	not null
Name_on_card	varchar	not null

Trans_history

Attribute	Datatype	Constraint
trans_id	varchar	not null
trans_date	date	not null
P_ID	varchar	foreign key references Participent
total	number	new value greater than the previous one

k

5.2 Implimentation:

Implimentation of the tables has been shown below in the snapshorts.

6.1:

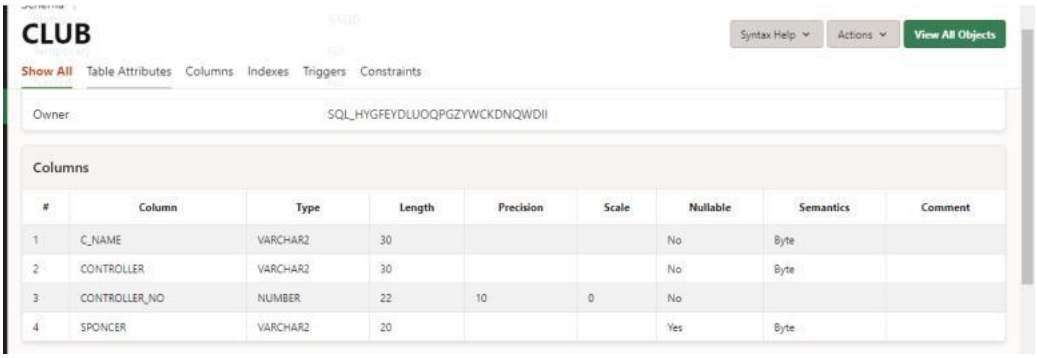
Club:

Creating Table without constraints:

```
create table Club(
C_name varchar(30) not null,
 Controller varchar(30) not null,
 Controller_no number(10) not null,
 Sponcer varchar(20),
primary key (C_name));
```

```
Table created.
```

View:



#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	C_NAME	VARCHAR2	30			No	Byte	
2	CONTROLLER	VARCHAR2	30			No	Byte	
3	CONTROLLER_NO	NUMBER	22	10	0	No		
4	SPONCER	VARCHAR2	20			Yes	Byte	

Adding constraints:

SQL Worksheet

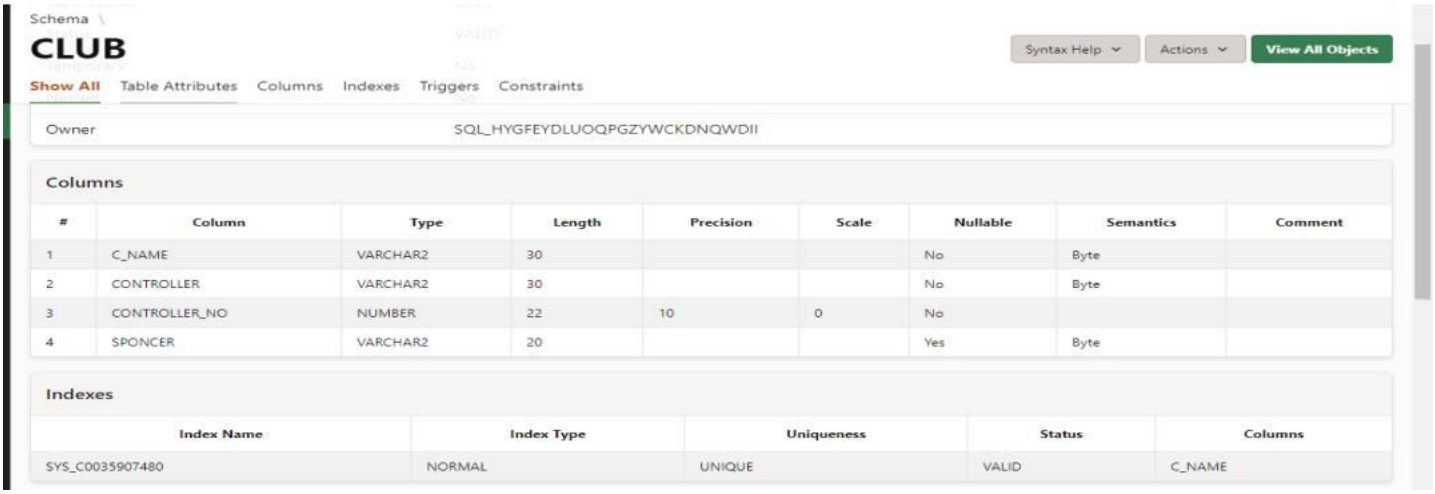
```
create table Club(
C_name varchar(30) not null,
 Controller varchar(30) not null,
 Controller_no number(10) not null,
 Sponcer varchar(20),
primary key (C_name));

alter table Club add constraint phone_no unique(Controller_no);
alter table club add constraint ctr_name unique(Controller);
alter table club modify Controller  null;
```

View:

Constraints

Constraint	Type	Condition	On Delete	Status	Last Change	Invalid?
SYS_C0035907477	Check	"C_NAME" IS NOT NULL	-	ENABLED	2 hours ago	-
SYS_C0035907478	Check	"CONTROLLER" IS NOT NULL	-	ENABLED	2 hours ago	-
SYS_C0035907479	Check	"CONTROLLER_NO" IS NOT NULL	-	ENABLED	2 hours ago	-
SYS_C0035907480	Primary Key	-	-	ENABLED	2 hours ago	-

Final table view with constraints:



Venue:

Creating Table without constraints:

```
create table Venue(
Desk_id number(4),
Plot_name varchar(10),
Desk_no number(4)
);
```

```
Table created.
```

View:

VENUE

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

Owner: SQL_HYGFEYDLUOQPGZYWCKDNQWDII

Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	DESK_ID	NUMBER	22	4	0	No		
2	PLOT_NAME	VARCHAR2	10			No	Byte	
3	DESK_NO	NUMBER	22	4	0	No		

Indexes

Index Name	Index Type	Uniqueness	Status	Columns
SYS_C0035907867	NORMAL	UNIQUE	VALID	DESK_ID

Adding constraints:

SQL Worksheet

Clear | Find

```
1  create table Venue(
2  Desk_id number(4),
3  Plot_name varchar(10) not null,
4  Desk_no number(4) not null,
5  primary key(Desk_id)
6  );
7
8
9  alter table venue add constraint v_pk primary key(desk_id);
10 alter table venue drop constraint v_pk;
11 alter table venue modify plot_name not null;
```

View:

Constraints

Constraint	Type	Condition	On Delete	Status	Last Change	Invalid?
SYS_C0035907865	Check	"PLOT_NAME" IS NOT NULL	-	ENABLED	116 minutes ago	-
SYS_C0035907866	Check	"DESK_NO" IS NOT NULL	-	ENABLED	116 minutes ago	-
SYS_C0035907867	Primary Key	-	-	ENABLED	116 minutes ago	-

Final table view with constraints:

Schema \

VENUE

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

Owner: SQL_HYGFEYDLUOQPGZYWCKDNQWDII

Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	DESK_ID	NUMBER	22	4	0	No		
2	PLOT_NAME	VARCHAR2	10			No	Byte	
3	DESK_NO	NUMBER	22	4	0	No		

Indexes

Index Name	Index Type	Uniqueness	Status	Columns
SYS_C0035907867	NORMAL	UNIQUE	VALID	DESK_ID

Events:

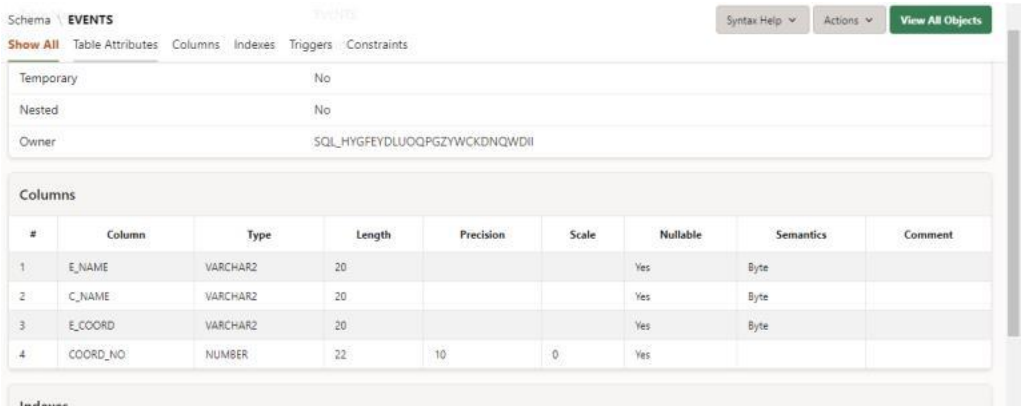
Creating Table without constraints:

L Worksheet

Clear Fin

```
1  create table Events(
2  E_name varchar(20),
3  C_name varchar(20),
4  E_coord varchar(20),
5  coord_no number(10)
6  );
7
```

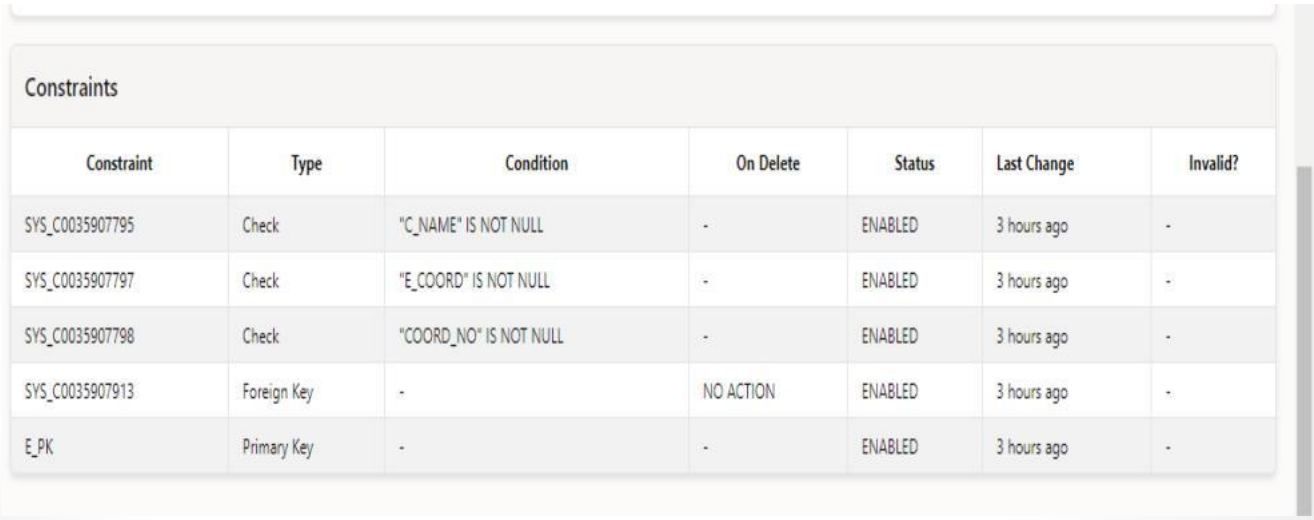
View:



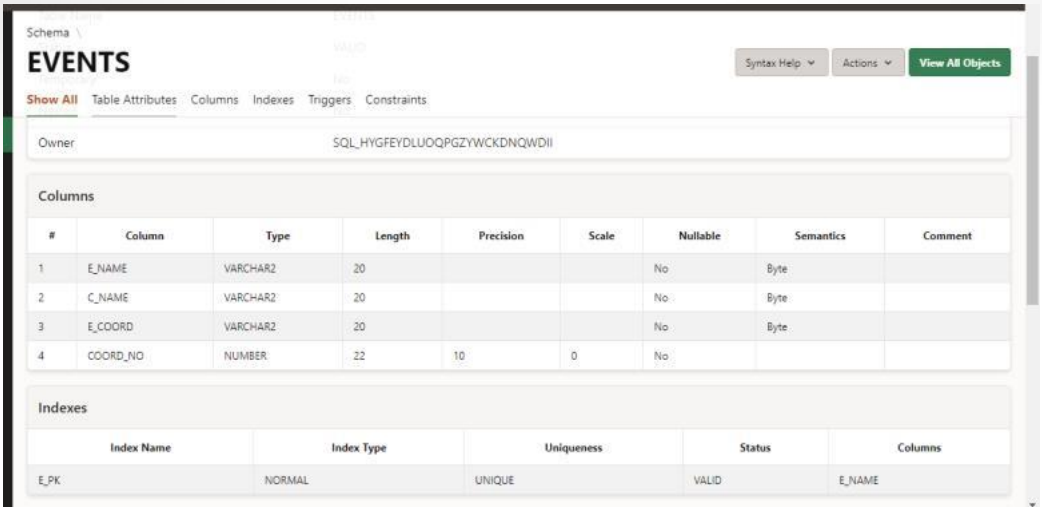
Adding constraints:

```
7
8  alter table events add constraint e_pk primary key(E_name);
9  alter table events modify C_name not null;
10
11 alter table events modify E_coord not null;
12
13 alter table Events add foreign key (C_name) references Club (C_name);
```

View:



Final table view with constraints


Schema

EVENTS

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

Owner	SQL_HYGFEYDLUOQPGZYWCKDNQWDII

Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	E_NAME	VARCHAR2	20			No	Byte	
2	C_NAME	VARCHAR2	20			No	Byte	
3	E_COORD	VARCHAR2	20			No	Byte	
4	COORD_NO	NUMBER	22	10	0	No		

Indexes

Index Name	Index Type	Uniqueness	Status	Columns
E_PK	NORMAL	UNIQUE	VALID	E_NAME

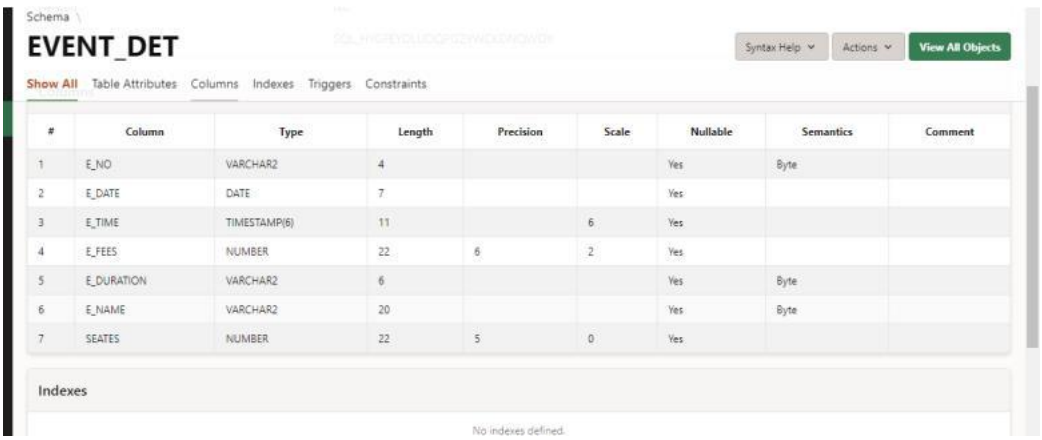
Event_Det:

Creating Table without constraints:

```
15
16
17  create table Event_Det(
18  E_no varchar(4),
19  E_date date,
20  E_time timestamp,
21  E_fees numeric(6,2) default 0,
22  E_duration varchar(6),
23  E_name varchar(20),
24  seates number(5) default 0,
25  foreign key (E_name) references Events (E_name)
26  );
27
28
29
```

```
Table created.
```

View:


Schema

EVENT_DET

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	E_NO	VARCHAR2	4			Yes	Byte	
2	E_DATE	DATE	7			Yes		
3	E_TIME	TIMESTAMP(6)	11		6	Yes		
4	E_FEES	NUMBER	22	6	2	Yes		
5	E_DURATION	VARCHAR2	6			Yes	Byte	
6	E_NAME	VARCHAR2	20			Yes	Byte	
7	SEATES	NUMBER	22	5	0	Yes		

Indexes

No indexes defined.

Adding constraints:

```
26 );
27 alter table event_det add constraint ed_pk primary key(E_no);
28 alter table event_det modify E_date not null;
29 alter table event_det modify E_time not null;
30 alter table event_det modify E_fees not null;
31 alter table event_det modify E_duration not null;
32 alter table event_det modify E_name not null;
33 alter table event_det modify E_date not null;
34 alter table event_t modify seates not null;
35
36
37 alter table event_Det add constraint ck_key check(seates >=0 and seates <=1500);
38 alter table event_det modify E_duration number(2);
39 alter table event_Det add constraint ck_dur check(E_duration >=0 and E_duration <=3);
40 alter table event_Det add constraint ck_date check(E_date>='01-Aug-2020' and E_date<='20-Aug-2020');
41 E_date 'YYYY-MM-DD'
```

View:

Schema \ EVENT_DET

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

Syntax Help | Actions | View All Objects

Constraints

Constraint	Type	Condition	On Delete	Status	Last Change	Invalid?
CK_DATE	Check	E_date>='01-Aug-2020' and E_date<='20-Aug-2020'	-	ENABLED	43 seconds ago	-
CK_DUR	Check	E_duration >=0 and E_duration <=3	-	ENABLED	23 minutes ago	-
CK_KEY	Check	seates >=0 and seates <=1500	-	ENABLED	28 minutes ago	-
SYS_C0035908534	Check	"E_TIME" IS NOT NULL	-	ENABLED	45 minutes ago	-
SYS_C0035908818	Check	"E_FEES" IS NOT NULL	-	ENABLED	34 minutes ago	-
SYS_C0035908819	Check	"E_DURATION" IS NOT NULL	-	ENABLED	34 minutes ago	-
SYS_C0035908820	Check	"E_NAME" IS NOT NULL	-	ENABLED	34 minutes ago	-
SYS_C0035908849	Check	"E_DATE" IS NOT NULL	-	ENABLED	32 minutes ago	-
SYS_C0035908850	Check	"SEATES" IS NOT NULL	-	ENABLED	32 minutes ago	-
SYS_C0035908425	Foreign Key	-	NO ACTION	ENABLED	50 minutes ago	-
ED_PK	Primary Key	-	-	ENABLED	47 minutes ago	-

Final table view with constraints

EVENT_DET

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	E_NO	VARCHAR2	4			No	Byte	
2	E_DATE	DATE	7			No		
3	E_TIME	TIMESTAMP(6)	11		6	No		
4	E_FEES	NUMBER	22	6	2	No		
5	E_DURATION	NUMBER	22	2	0	No		
6	E_NAME	VARCHAR2	20			No	Byte	
7	SEATES	NUMBER	22	5	0	No		

Indexes

Index Name	Index Type	Uniqueness	Status	Columns
ED_PK	NORMAL	UNIQUE	VALID	E_NO

Participient:

Creating Table without constraints:

```
41
42
43  create table Participient(
44  P_id varchar(5),
45  f_name varchar(20),
46  m_name varchar(20),
47  l_name varchar(20),
48  P_no number(15),
49  primary key (P_id)
50  );
51
```

View:

Schema \

PARTICIPIENT

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

Owner	SQL_HYGFEYDLUOQPGZYWCKDNQWDII

Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	P_ID	VARCHAR2	5			No	Byte	
2	F_NAME	VARCHAR2	20			Yes	Byte	
3	M_NAME	VARCHAR2	20			Yes	Byte	
4	L_NAME	VARCHAR2	20			Yes	Byte	
5	P_NO	NUMBER	22	15	0	Yes		

Adding constraints:

```
41
42
43  create table Participient(
44  P_id varchar(5),
45  f_name varchar(20),
46  m_name varchar(20),
47  l_name varchar(20),
48  P_no number(15),
49  primary key (P_id)
50  );
51
52  alter table participient modify f_name not null;
53  alter table participient add constraint uk1 unique(P_no);
54
55
56
57
58
59
60
```

```
Table altered.
```

View:

Schema \ **PARTICIPIENT**

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

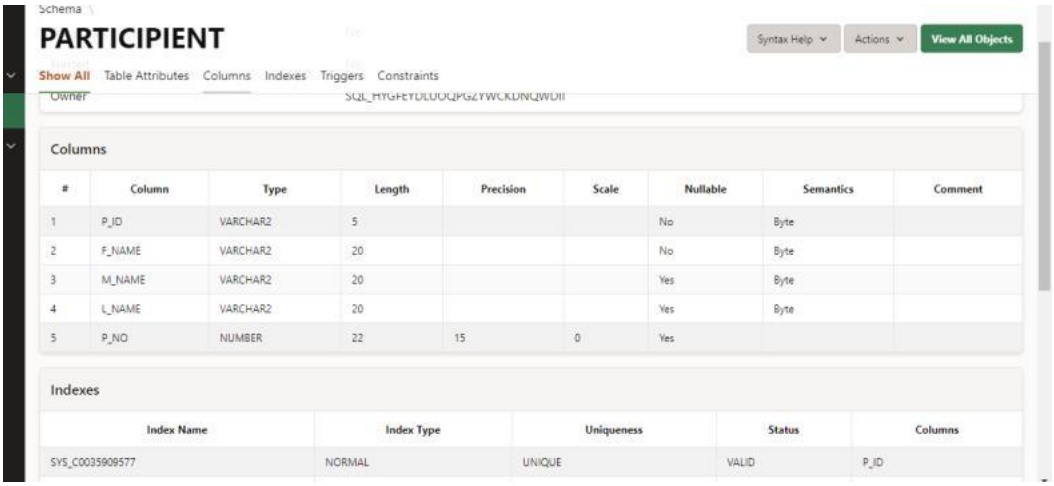
Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	P_ID	VARCHAR2	5			No	Byte	
2	F_NAME	VARCHAR2	20			No	Byte	
3	M_NAME	VARCHAR2	20			Yes	Byte	
4	L_NAME	VARCHAR2	20			Yes	Byte	
5	P_NO	NUMBER	22	15	0	Yes		

Indexes

Index Name	Index Type	Uniqueness	Status	Columns
SYS_C0035909577	NORMAL	UNIQUE	VALID	P_ID
UK1	NORMAL	UNIQUE	VALID	P_NO

Final table view with constraints



Card:

Creating Table without constraints:

```
54
55 create table Card(
56 card_no number(16),
57 EXP date ,
58 CVV number(3),
59 Name_on_card varchar(25),
60 P_id varchar(5)
61 );
62
63
```

View:

Schema \

CARD

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

Status	VALID
Temporary	No
Nested	No
Owner	SQL_HYGFEYDLUOQPGZYWCKDNQWDII

Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	CARD_NO	NUMBER	22	16	0	Yes		
2	EXP	DATE	7			Yes		
3	CVV	NUMBER	22	3	0	Yes		
4	NAME_ON_CARD	VARCHAR2	25			Yes	Byte	
5	P_ID	VARCHAR2	5			Yes	Byte	

Adding constraints:

```
62
63 alter table card add constraint c_pk primary key(card_no);
64 alter table card add foreign key (P_id) references Participient (P_id);
65 alter table card modify exp not null;
66 alter table card modify cvv not null;
67 alter table card modify name_on_card not null;
68 alter table card modify P_id not null;
69 alter table card add constraint cd_chk check(card_no >=0000000000000001 and card_no <=9999999999999999);
70
71
72
73
74
```

View:

Constraints

Constraint	Type	Condition	On Delete	Status	Last Change	Invalid?
CD_CHK	Check	card_no >=0000000000000001 and card_no <=9999999999999999	-	ENABLED	15 seconds ago	-
SYS_C0035909878	Check	"EXP" IS NOT NULL	-	ENABLED	3 minutes ago	-
SYS_C0035909879	Check	"CVV" IS NOT NULL	-	ENABLED	3 minutes ago	-
SYS_C0035909880	Check	"NAME_ON_CARD" IS NOT NULL	-	ENABLED	3 minutes ago	-
SYS_C0035909881	Check	"P_ID" IS NOT NULL	-	ENABLED	3 minutes ago	-
SYS_C0035909840	Foreign Key	-	NO ACTION	ENABLED	6 minutes ago	-
C_PK	Primary Key	-	-	ENABLED	7 minutes ago	-

Final table view with constraints:

Schema CARD

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	CARD_NO	NUMBER	22	16	0	No		
2	EXP	DATE	7			No		
3	CVV	NUMBER	22	3	0	No		
4	NAME_ON_CARD	VARCHAR2	25			No	Byte	
5	P_ID	VARCHAR2	5			No	Byte	

Indexes

Index Name	Index Type	Uniqueness	Status	Columns
C_PK	NORMAL	UNIQUE	VALID	CARD_NO

Triggers

Trans_history:

Creating Table without constraints:

```
70
71 create table Trans_history(
72 E_no varchar(4),
73 P_id varchar(5),
74 trans_date date,
75 trans_id varchar(5)
76 );
77
```

View:

Schema

TRANS_HISTORY

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

Owner SQL_HYGFEYDLUOQPGZYWCKDNQWDII

Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	E_NO	VARCHAR2	4			No	Byte	
2	P_ID	VARCHAR2	5			No	Byte	
3	TRANS_DATE	DATE	7			No		
4	TRANS_ID	VARCHAR2	5			No	Byte	

Indexes

Index Name	Index Type	Uniqueness	Status	Columns
T_PK	NORMAL	UNIQUE	VALID	TRANS_ID

Adding constraints:

```
76 );
77
78 alter table Trans_history add constraint t_pk primary key(trans_id);
79 alter table Trans_history add foreign key (P_id) references Participient (P_id);
80 alter table Trans_history add foreign key (E_no) references Event_det (E_no);
81 alter table Trans_history modify E_no not null;
82 alter table Trans_history modify P_id not null;
83 alter table Trans_history modify trans_date not null;
84 alter table Trans_history add constraint cd_date check(trans_date>='01-Aug-2020' and trans_date<='20-Aug-2020');
85
86
87
```

Table altered.

Resize Code Edi

View:

No triggers defined.

Constraints

Constraint	Type	Condition	On Delete	Status	Last Change	Invalid?
CD_DATE	Check	trans_date>='01-Aug-2020' and trans_date<='20-Aug-2020'	-	ENABLED	33 seconds ago	-
SYS_C0035910079	Check	"E_NO" IS NOT NULL	-	ENABLED	3 minutes ago	-
SYS_C0035910080	Check	"P_ID" IS NOT NULL	-	ENABLED	3 minutes ago	-
SYS_C0035910081	Check	"TRANS_DATE" IS NOT NULL	-	ENABLED	3 minutes ago	-
SYS_C0035910024	Foreign Key	-	NO ACTION	ENABLED	5 minutes ago	-
T_PK	Primary Key	-	-	ENABLED	6 minutes ago	-

Final table view with constraints:

Schema

TRANS_HISTORY

Syntax Help | Actions | View All Objects

Show All | Table Attributes | Columns | Indexes | Triggers | Constraints

Owner SQL_HYGFEYDLUOQPGZYWCKDNQWDII

Columns

#	Column	Type	Length	Precision	Scale	Nullable	Semantics	Comment
1	E_NO	VARCHAR2	4			No	Byte	
2	P_ID	VARCHAR2	5			No	Byte	
3	TRANS_DATE	DATE	7			No		
4	TRANS_ID	VARCHAR2	5			No	Byte	

Indexes

Index Name	Index Type	Uniqueness	Status	Columns
T_PK	NORMAL	UNIQUE	VALID	TRANS_ID

6.2 Inserting Values:

```
SQL> insert into Club values('Anokha','Raghav K',9610000011,'Red Bull');

1 row created.

SQL> insert into Club values('Augua','Zakir',9879562300,'Pepsico');

1 row created.

SQL> insert into Club values('Spotify','Kunal Verma', 9875252300,'Britania');

1 row created.

SQL> insert into Club values('TPK','Anand Mohan',9918989011,'Nestle');

1 row created.

SQL> insert into Club values('Mints','Kapil Mathew',9210033013,'Red Bull');

1 row created.

SQL> insert into Club values('Euros','Ishan Dutta',9712020799,'Red Bull');

1 row created.

SQL>
```

```
SQL> select * from Club;

C_NAME                         CONTROLLER                       CONTROLLER_NO
------------------------------ -------------------------------- -------------
SPONCER
----------------------
Anokha                         Raghav K                            9610000011
Red Bull

Augua                          Zakir                               9879562300
Pepsico

Spotify                        Kunal Verma                         9875252300
Britania

C_NAME                         CONTROLLER                       CONTROLLER_NO
------------------------------ -------------------------------- -------------
SPONCER
----------------------
TPK                            Anand Mohan                         9918989011
Nestle

Mints                          Kapil Mathew                        9210033013
Red Bull

Euros                          Ishan Dutta                         9712020799
Red Bull

6 rows selected.
```

```
SQL> insert into Venue values(1354,'PlotA',0003);

1 row created.

SQL> insert into Venue values(2457,'PlotB',0005);

1 row created.

SQL> insert into Venue values(4352,'PlotC',0007);

1 row created.

SQL> insert into Venue values(3121,'PlotD',0008);

1 row created.

SQL> insert into Venue values(1054,'PlotE',0009);

1 row created.

SQL> insert into Venue values(2304,'PlotF',0011);

1 row created.

SQL> select * from Venue;

    DESK_ID PLOT_NAME      DESK_NO
----------- ---------- -----------
       1354 PlotA                3
       2457 PlotB                5
       4352 PlotC                7
       3121 PlotD                8
       1054 PlotE                9
       2304 PlotF               11
```

```
E_NO E_NAME       C_NAME  E_COORD      COORD_NO   DESK_ID
-------------------------------------------------------
E123 100m Running Anokha  Varun Kumar  9879222330 0231
E121 Swimming     Spotify Akash        9979159188 0187
E119 Tracking     TPK     Davesh suman 8866202003 0141
E120 Shooting     Euros   Abhishek     9979252020 0169
```

Command Prompt - sqlplus

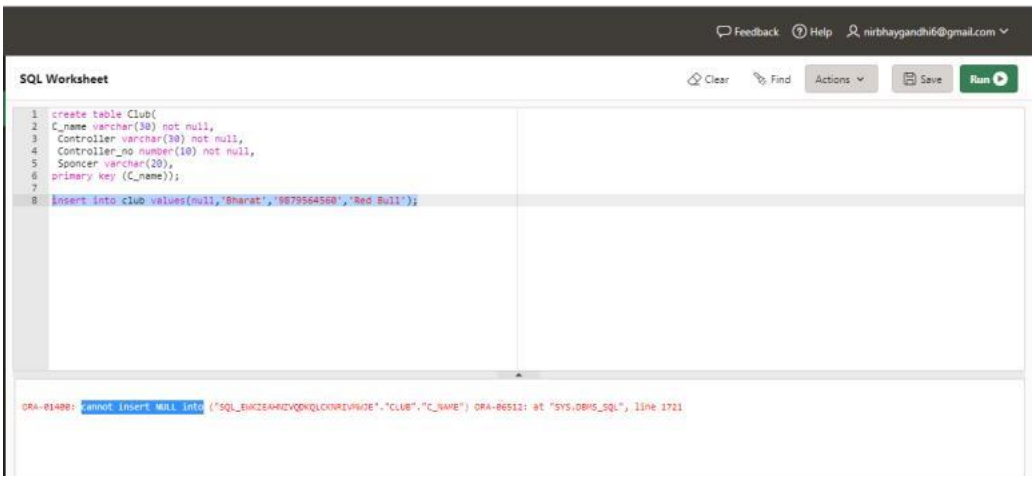
```
insert into Event_Det values('E102', '11-Aug-2020', '11-Aug-2020 09:50:00', 0.00, '2 HRS','Swimming',
*
ERROR at line 1:
ORA-02291: integrity constraint (SYSTEM.SYS_C0011080) violated - parent key not
found


SQL> insert into Participient values('P1010', 'NIraj', 'S', 'Negi', 9922445567);

1 row created.

SQL> insert into Participient values('P1020', 'Kushal', 'Kumar', 'Rastogi', 99224675444);

1 row created.

SQL> insert into Participient values('P3011', 'Arya', 'M', 'Nia', 9838220052);

1 row created.

SQL> insert into Participient values('P3011', 'Arya', 'M', 'Nia', 973838);
insert into Participient values('P3011', 'Arya', 'M', 'Nia', 973838)
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.SYS_C0011082) violated


SQL> insert into Participient values('P3021', 'Kiran', 'Muna', 'Mania', 9738383845);

1 row created.

SQL> insert into Participient values('P3011', 'Ragi', 'A', 'Zala', 7383554845);
insert into Participient values('P3011', 'Ragi', 'A', 'Zala', 7383554845)
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.SYS_C0011082) violated


SQL> insert into Participient values('P3071', 'Ragi', 'A', 'Zala', 7383554845);

1 row created.
```

```
SQL> select * from Participient;

P_ID  F_NAME               M_NAME               L_NAME                    P_NO
----- -------------------- -------------------- -------------------- ----------
P1010 NIraj                S                    Negi                 9922445567
P3011 Arya                 M                    Nia                  9838220052
P3021 Kiran                Muna                 Mania                9738383845
P3071 Ragi                 A                    Zala                 7383554845
```

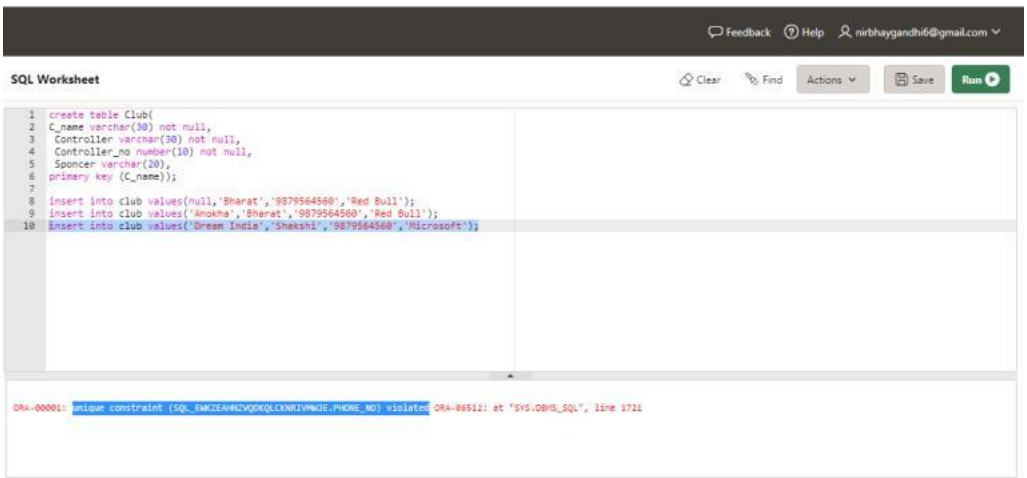
```
E_NO  P_ID   TRANS_DATE   TRANS_ID
_____________________________________

E101  P2951  11-Aug-2020  33205
E102  P3125  11-Aug-2020  37201
E203  P4117  12-Aug-2020  43439
E311  P5001  13-Aug-2020  63637
```

Inserting values violating the intigrity constraints:

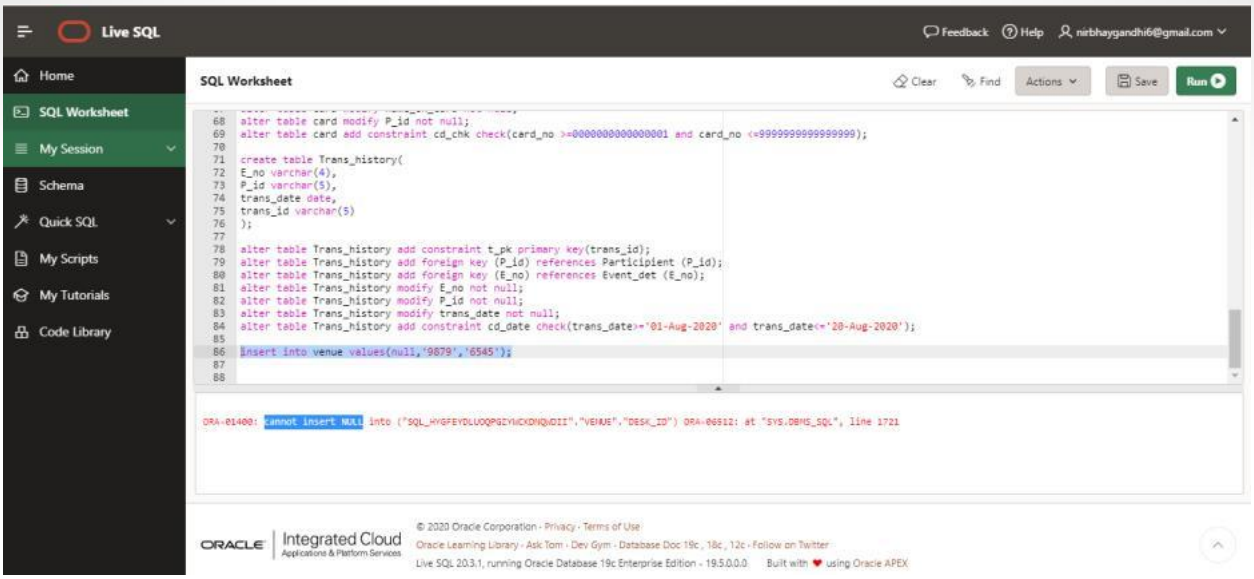
Club:



--



Venue:


Live SQL
Feedback
Help
nirbhaygandhi6@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
alter table card modify P_id not null;
alter table card add constraint cd_chk check(card_no >=0000000000000001 and card_no <=9999999999999999);
create table Trans_history(
E_no varchar(4),
P_id varchar(5),
trans_date date,
trans_id varchar(5)
);
alter table Trans_history add constraint t_pk primary key(trans_id);
alter table Trans_history add foreign key (P_id) references Participient (P_id);
alter table Trans_history add foreign key (E_no) references Event_det (E_no);
alter table Trans_history modify E_no not null;
alter table Trans_history modify P_id not null;
alter table Trans_history modify trans_date not null;
alter table Trans_history add constraint cd_date check(trans_date>='01-Aug-2020' and trans_date<='20-Aug-2020');
insert into venue values(null,'9879','6545');
ORA-01400: cannot insert NULL into ("SQL_HYGFEYDLUOQPGZYHCXDNQUII"."VENUE"."DESK_ID") ORA-06512: at "SYS.DBMS_SQL", line 1721
ORACLE
Integrated Cloud
Applications & Platform Services
© 2020 Oracle Corporation - Privacy - Terms of Use
Oracle Learning Library - Ask Tom - Dev Gym - Database Doc 19c, 18c, 12c - Follow on Twitter
Live SQL 20.3.1, running Oracle Database 19c Enterprise Edition - 19.5.0.0.0
Built with using Oracle APEX


Live SQL
Feedback
Help
nirbhaygandhi6@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
alter table card modify P_id not null;
alter table card add constraint cd_chk check(card_no >=0000000000000001 and card_no <=9999999999999999);
create table Trans_history(
E_no varchar(4),
P_id varchar(5),
trans_date date,
trans_id varchar(5)
);
alter table Trans_history add constraint t_pk primary key(trans_id);
alter table Trans_history add foreign key (P_id) references Participient (P_id);
alter table Trans_history add foreign key (E_no) references Event_det (E_no);
alter table Trans_history modify E_no not null;
alter table Trans_history modify P_id not null;
alter table Trans_history modify trans_date not null;
alter table Trans_history add constraint cd_date check(trans_date>='01-Aug-2020' and trans_date<='20-Aug-2020');
insert into venue values('D102','987909898','6545');
ORA-01722: invalid number ORA-06512: at "SYS.DBMS_SQL", line 1721

Event:


Live SQL
Feedback
Help
nirbhaygandhi6@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
alter table card modify P_id not null;
alter table card add constraint cd_chk check(card_no >=0000000000000001 and card_no <=9999999999999999);
create table Trans_history(
E_no varchar(4),
P_id varchar(5),
trans_date date,
trans_id varchar(5)
);
alter table Trans_history add constraint t_pk primary key(trans_id);
alter table Trans_history add foreign key (P_id) references Participient (P_id);
alter table Trans_history add foreign key (E_no) references Event_det (E_no);
alter table Trans_history modify E_no not null;
alter table Trans_history modify P_id not null;
alter table Trans_history modify trans_date not null;
alter table Trans_history add constraint cd_date check(trans_date>='01-Aug-2020' and trans_date<='20-Aug-2020');
insert into events values(null,'Augtd','Fjgsy','98756458451');
ORA-01400: cannot insert NULL into ("SQL_HYGFEYDLUOQPGZYHCXDNQUII"."EVENTS"."E_NAME") ORA-06512: at "SYS.DBMS_SQL", line 1721

--


Live SQL
Feedback
Help
nirbhaygandhi6@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
alter table card modify P_id not null;
alter table card add constraint cd_chk check(card_no >=0000000000000001 and card_no <=9999999999999999);
create table Trans_history(
E_no varchar(4),
P_id varchar(5),
trans_date date,
trans_id varchar(5)
);
alter table Trans_history add constraint t_pk primary key(trans_id);
alter table Trans_history add foreign key (P_id) references Participient (P_id);
alter table Trans_history add foreign key (E_no) references Event_det (E_no);
alter table Trans_history modify E_no not null;
alter table Trans_history modify P_id not null;
alter table Trans_history modify trans_date not null;
alter table Trans_history add constraint cd_date check(trans_date>='01-Aug-2020' and trans_date<='20-Aug-2020');
insert into events values(null,'Augtd','Fjgsy','987564584S1');
insert into events values('dsdsd',null,'Fjgsy','98756458451');
ORA-01400: cannot insert NULL into ("SQL_HYGFEYDLUOQPGZYWCKDNQWDII"."EVENTS"."C_NAME") ORA-06512: at "SYS.DBMS_SQL", line 1721

Event_Det:


Live SQL
Feedback
Help
nirbhaygandhi6@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
alter table card modify P_id not null;
alter table card add constraint cd_chk check(card_no >=0000000000000001 and card_no <=9999999999999999);
create table Trans_history(
E_no varchar(4),
P_id varchar(5),
trans_date date,
trans_id varchar(5)
);
alter table Trans_history add constraint t_pk primary key(trans_id);
alter table Trans_history add foreign key (P_id) references Participient (P_id);
alter table Trans_history add foreign key (E_no) references Event_det (E_no);
alter table Trans_history modify E_no not null;
alter table Trans_history modify P_id not null;
alter table Trans_history modify trans_date not null;
alter table Trans_history add constraint cd_date check(trans_date>='01-Aug-2020' and trans_date<='20-Aug-2020');
insert into events values(null,'Augtd','Fjgsy','987564584S1');
insert into event_det values('E656','25-Aug-2020','25-Aug-2020 00:05:15','1000','3 hrs','Fun fishing','56');
ORA-01849: hour must be between 1 and 12 ORA-06512: at "SYS.DBMS_SQL", line 1721

--


Live SQL
Feedback
Help
nirbhaygandhi6@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
insert into event_det values('E656','25-Aug-2020','25-Aug-2020 00:05:15','1000','3 hrs','Fun fishing','56');
insert into event_det values('E656','25-Aug-2020','25-Aug-2020 04:05:15','1000','3 hrs','Fun fishing','56');
ORA-01722: invalid number ORA-06512: at "SYS.DBMS_SQL", line 1721

--

--

--

Participient:


Live SQL
Feedback
Help
nirbhaygandhi6@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
insert into participient values('P1020', null, null, null, 9879332200);
insert into participient values('P1020', 'ninihjh', null, null, 9879332200);
insert into participient values('P1020', 'vinesh', null, null, 9879332200);
create table Participient(
P_id varchar(5),
f_name varchar(20) not null,
m_name varchar(20),
l_name varchar(20),
P_no number(15),
primary key (P_id)
);
ORA-01400: cannot insert NULL into ("SQL_JNEPXDHHQIXFHGAWLIGWJOTZB"."PARTICIPIENT"."F_NAME") ORA-06512: at "SYS.DBMS_SQL", line 1721

--


Live SQL
Feedback
Help
nirbhaygandhi6@gmail.com
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
Run
insert into participient values('P1020', null, null, null, 9879332200);
insert into participient values('P1020', 'ninihjh', null, null, 9879332200);
insert into participient values('P1020', 'vinesh', null, null, 9879332200);
create table Participient(
P_id varchar(5),
f_name varchar(20) not null,
m_name varchar(20),
l_name varchar(20),
P_no number(15),
primary key (P_id)
);
ORA-00001: unique constraint (SQL_JNEPXDHHQIXFHGAWLIGWJOTZB.SYS_C0035915210) violated ORA-06512: at "SYS.DBMS_SQL", line 1721

Card:


Live SQL
Feedback
Help
Home
SQL Worksheet
My Session
Schema
Quick SQL
My Scripts
My Tutorials
Code Library
SQL Worksheet
Clear
Find
Actions
Save
create table Card(
card_no number(16),
EXP date not null,
CVV number(3) not null,
Name_on_card varchar(25) not null,
P_id varchar(5),
primary key (card_no),
foreign key (P_id) references Participient (P_id)
);
insert into card values('1111222233334444', '20-jun-2023', '987', 'nikunnj', 'P1020');
insert into card values('8888222233335555', '08-jun-2023', '188', 'suresh', 'P2030');
ORA-02291: integrity constraint (SQL_JNEPXDHHQIXFHGAWLIGWJOTZB.SYS_C0035915290) violated - parent key not found ORA-06512: at "SYS.DBMS_SQL", line 1721

--

--

Trans_history:

--

--

7. Conclusion and future work

7.1 Conclusion:

The project started from just a simple step and thinking, so far has reached to a point where we collected the information from the event organizer that on what all things you need to organize and manage this fair, than we created E-R diagram than we proceded to make Relational Schema from the E-R diagram, after that we implimented the tables, we wrote codes for the tables and we implimented it.

7.2 Future Work:

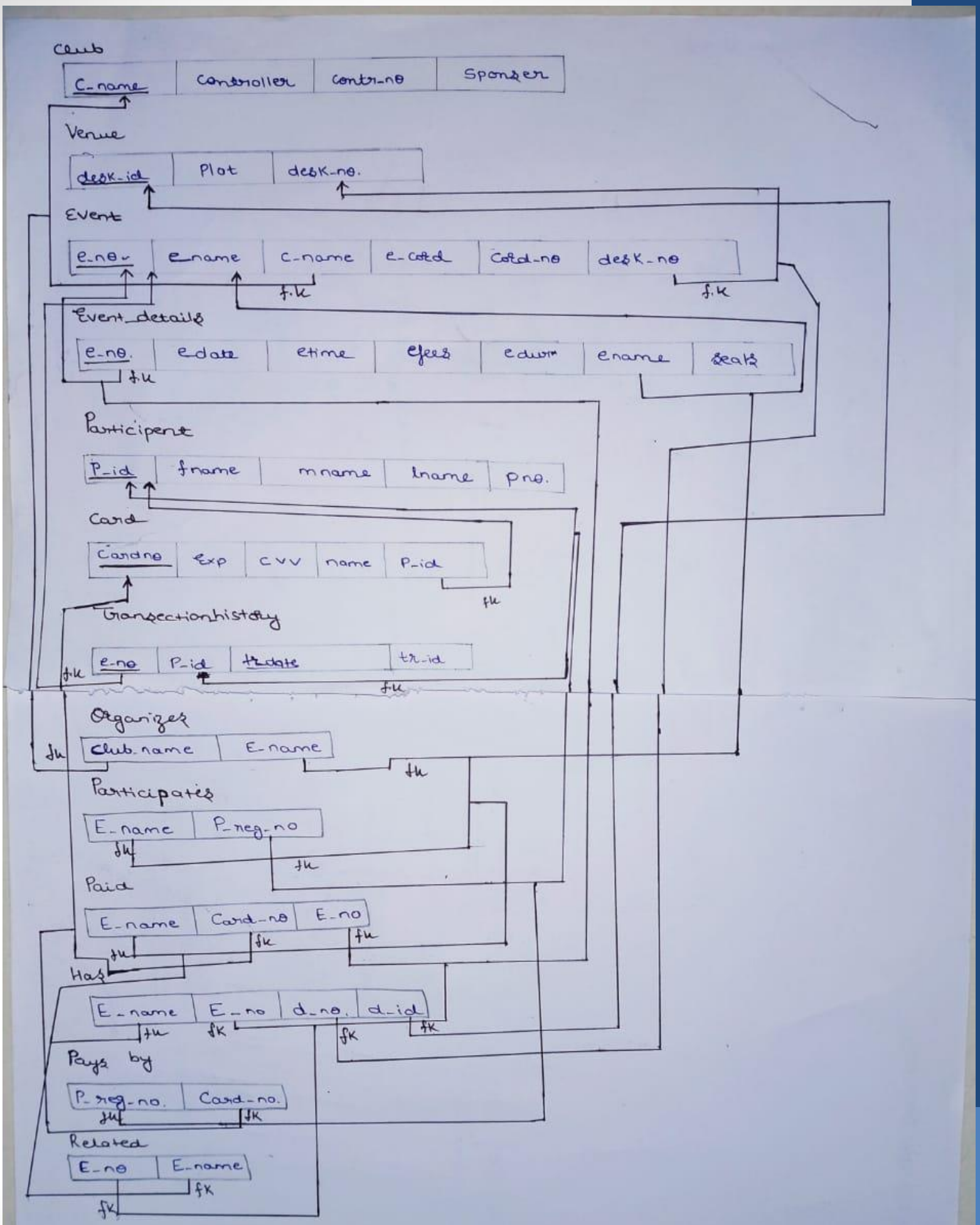
Still we have to achive a great mile stone to present this project to our teacher, still we have to normalize the table, resolve the functional dependence, then work fot the front end to get a good and presentable U.I and than we need to propogate the table with real world data, as the fare will start, definetly enteries will be done!.

NORMALIZATION- OVERVIEW

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

- There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly.

- most commonly used normal forms:
  - First normal form(1NF)
  - Second normal form(2NF)
  - Third normal form(3NF)
  - Boyce & Codd normal form (BCNF)


Club
C_name
Controller
Contr_no
Sponser
Venue
desk_id
Plot
desk_no.
Event
e_no
e_name
C_name
e_catd
Catd_no
desk_no
f.k
f.k
Event_details
e_no.
e_date
e_time
e_fees
e_dur
e_name
seats
f.k
Participent
P_id
fname
mname
lname
pno.
Card
Cardno
Exp
CVV
name
P_id
fk
Transectionhistory
e_no
P_id
tr_date
tr_id
fk
fk
Organizes
fk
Club_name
E_name
fk
Participates
E_name
P_reg_no
fk
fk
Paid
E_name
Card_no
E_no
fk
fk
fk
Has
E_name
E_no
d_no.
d_id
fk
fk
fk
fk
Pays by
P_reg_no.
Card_no.
fk
fk
Related
E_no
E_name
fk
fk

- First Normal Form

A relation is in first normal form if every attribute in that relation is **singled valued attribute**.

- Second Normal Form

Relation should be 1st normal form.

A relation is in 2NF if it has **No Partial Dependency.**

Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.

- Third Normal Form

Relation should be 2nd normal form.

A relation is in third normal form, if there is **no transitive dependency** for non-prime attributes.

A relation is in 3NF if **at least one of the following condition holds** in every non-trivial function dependency X –> Y

- X is a super key.
- Y is a prime attribute (each element of Y is part of some candidate key).

- First Normal Form

A relation is in first normal form if every attribute in that relation is **singled valued attribute**.

- Second Normal Form

Relation should be 1st normal form.

A relation is in 2NF if it has **No Partial Dependency.**

Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.

- Third Normal Form

Relation should be 2nd normal form.

A relation is in third normal form, if there is **no transitive dependency** for non-prime attributes.

A relation is in 3NF if **at least one of the following condition holds** in every non-trivial function dependency X –> Y

- X is a super key.
- Y is a prime attribute (each element of Y is part of some candidate key).

- Boyce & Codd normal form (BCNF)

A relation R is in BCNF if R is in Third Normal Form and for every FD, LHS is super key. A relation is in BCNF iff in every non-trivial functional dependency X –> Y, X is a super key.

NORMALIZATION- PROCESS

1. CLUB:

R1:

C_NAME	CONTROLLER	CONTROLLER_NO	SPONCER

Functional Dependencies:

C_name --> Controller

C_name --> Sponcer

Controller_no --> Controller

Candidate Keys:

c.k={ C_name, Controller_no }

Checking Normal Form:

1ST Normal Form 

- Yes because every attribute here is of atomic value.

2nd Normal Form 

for each FD, checking whether the LHS is a proper subset of some candidate key or the RHS are not all attributes of c.k.

NORMALIZATION- PROCESS

3rd Normal Form ✗

Therefore, If a Table is not in 3rd N.F, then by definition we can say that it will not be in it's higher normal form.

BCNF ✗

Normalizing to 2 N.F

finding the **minimal cover of the FDs**, which includes the FDs

C_name --> Controller

C_name --> Sponcer

Controller_no --> Controller

- The FD [C_name --> Controller] is a partial dependency (i.e., LHS is a proper subset of some c.k), the table is split into:

R2:

C_NAME	CONTROLLER	SPONCER

C_name --> Controller, Sponcer

R3:

C_NAME	CONTROLLER_NO

Normalizing to 3 N.F

C_name --> Controller_no

- Finding the minimal cover of the FDs, which includes the above FDs.

- Find all candidate keys.

C_name --> Controller, Sponcer
Controller_no --> Controller

- Checking each of the above FD for violation of 3NF, and splitting table accordingly.
- The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).
  The following 3NF table is obtained:

R4:

C_NAME	SPONCER

C_name --> Sponcer

R5:

CONTROLLER	CONTROLLER_NO

Controller_no --> Controller

R6:

C_NAME	CONTROLLER

Controller_name --> Controller

- A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.
- After Normalization we have achieved:-

1 N.F **2 N.F** **3 N.F** **B.C.N.F**

NORMALIZATION- PROCESS

2. VENUE:

R1:

DESK_ID	PLOT_NAME	DESK_NO	E_NO

Functional Dependencies:

E_no --> desk_no, desk_id

Plot_name --> desk_no, desk_id, E_no

Desk_id --> Desk_no

Candidate Keys:

c.k={ Plot_name }

Checking Normal Form:

1ST Normal Form ✓

- Yes because every attribute here is of atomic value.

2nd Normal Form ✓

for each FD, checking whether the LHS is a proper subset of some candidate key or the RHS are not all attributes of c.k.

NORMALIZATION- PROCESS

3rd Normal Form

for each FD, checking whether the LHS is superkey or the RHS are all key attributes

on checking functional dependency E_no --> Desk_id,Desk_no

The above FD violates definition of 3NF: LHS is not superkey, RHS contains a non-prime attribute .

If a Table is not in 2nd N.F, then by definition we can say that it will not be in it's higher normal form.

BCNF

Normalizing to 3 N.F

- Finding the minimal cover of the FDs, which includes the above FDs.
- Find all candidate keys.

- **R2:**

E_NO	DESK_ID

E_no --> Desk_id

R3:

PLOT_NAME	E_NO

E_no --> Plot_name

R4:

DESK_ID	DESK_NO

Desk_id --> Desk_no

- A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.
  After Normalization:-

1 N.F ✓ 2 N.F ✓ 3 N.F ✓ B.C.N.F ✓

3. EVENT:

R1:

E_NO	E_NAME	E_CORD	CORD_NO

Functional Dependencies:

E_no --> e_name, e_cord

cord_no --> E_cord

E_cord, cord_no --> e_name

Candidate Keys:

c.k={ e_no, cord_no}

Checking Normal Form:

1ST Normal Form ✓

- Yes because every attribute here is of atomic value.

NORMALIZATION- PROCESS

2nd Normal Form ✖

for each FD, checking whether the LHS is a proper subset of some candidate key or the RHS are not all attributes of c.k.

FD: e_no --> e_name, e_cord
The FD: violates definition of 2NF -- LHS is a proper subset of some CK

- If a Table is not in 2nd N.F, then by definition we can say that it will not be in it's higher normal form.

3rd Normal Form ✖

BCNF ✖

Normalizing to 2 N.F

finding the **minimal cover of the FDs**, which includes the FDs
e_no --> e_cord

E_no --> e_name
Cord_no --> e_cord
cord_no --> e_name

- The FD [e_no --> e_name] is a partial dependency (i.e., LHS is a proper subset of some c.k), the table is split into:

R2:

E_NO	E_NAME	E_CORD

e_no --> e_name, e_cord

e_no --> cord_no

R3:

E_NO	CORD_NO

NORMALIZATION- PROCESS

R4:

CORD_NO	E_NAME	E_CORD

Cord_no --> e_name, e_cord

After Normalization we have achieved:-

1 N.F ✓ 2 N.F ✓ 3 N.F ✓ B.C.N.F ✓

4. EVENT DETAILS:

R1:

E_DATE	E_TIME	E_FEES	E_DURATION	E_NAME	SEATES

Functional Dependencies:

E_name --> e_date, e_time, e_fees, e_duration, seates

E_date, E_time --> E_duration

Candidate Keys:

c.k={ e_name}

Checking Normal Form:

1ST Normal Form ✓

- Yes because every attribute here is of atomic value.

NORMALIZATION- PROCESS

2nd Normal Form ✓

for each FD, checking whether the LHS is a proper subset of some candidate key or the RHS are not all attributes of c.k.

3rd Normal Form ✗

for each FD, checking whether the LHS is superkey or the RHS are all key attributes

on checking functional dependency E_name --> e_date, e_time, e_fees, e_duration, seates, E_date, E_time --> E_duration

The above FD violates definition of 3NF: LHS is not superkey, RHS contains a non-prime attribute .

If a Table is not in 3rd N.F, then by definition we can say that it will not be in it's higher normal form.

BCNF ✗

Normalizing to 3 N.F

- Find all candidate keys.
- finding the **minimal cover of the FDs**, which includes the FDs

e_name --> e_date

E_name --> e_time

E_name --> e_fees

E_date, e_time --> e_duration

NORMALIZATION- PROCESS

- Checking with each functional dependency that violates 3rd N.F.

The FD [e_date, e_time --> e_duration] The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).The following 3NF table is obtained:

R2:

E_NAME	E_DATE	E_TIME	E_FEES	SEATES

e_name --> e_date, e_time, e_fees, seates

e_date, e_time --> e_duration

R3:

E_DATE	E_TIME	E_DURATION

Both R2 and R3 are in 3 N.F and in B.C.N.F as well.

After Normalization we have achieved:-

1 N.F **2 N.F** **3 N.F** **B.C.N.F**

5. PARTICIPIENT:

R1:

P_ID	F_NAME	M_NAME	L_NAME	P_NO

Functional Dependencies:

P_id--> f_name, m_name, l_name,P_no

P_id, P_no --> f_name, m_name, l_name

F_name, P_no --> l_name, m_name, P_id

NORMALIZATION- PROCESS

Candidate Keys:

c.k={ {P_id}, {f_name, P_no} }

finding the **minimal cover of the FDs**, which includes the FDs
f_name, p_no --> p_id

p_id --> l_name
p_id --> p_no
p_id --> m_name

p_id --> f_name

Checking Normal Form:

1ST Normal Form ✓

- Yes because every attribute here is of atomic value.

2nd Normal Form ✓

- for each non-trivial FD, checking whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

Here every F.D satisfies this property.

3rd Normal Form ✓

- for each FD, checking whether the LHS is superkey or the RHS are all key attribute.
- Here every F.D satisfies this property.

BCNF ✓

- A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.
- Here every F.D satisfies this property.

NORMALIZATION- PROCESS

6. CARD:

R1:

CARD_NO	EXP	CVV	NAME_ON_CARD

Functional Dependencies:

Card_no--> exp, name_on_card, cvv

cvv, name_on_card --> card_no, exp

Candidate Keys:

c.k={ {card_no}, {name_on_card, cvv} }

finding the **minimal cover of the FDs**, which includes the FDs
card_no --> Exp
card_no --> CVV
card_no --> name_on_card
CVV, name_on_card --> card_no

Checking Normal Form:

1ST Normal Form ✔

- Yes because every attribute here is of atomic value.

NORMALIZATION- PROCESS

2nd Normal For

- for each non-trivial FD, checking whether the LHS is a proper subset of some candidate key or the RHS are not all key attributes.

Here every F.D satisfies this property.

3rd Normal Form

- for each FD, checking whether the LHS is superkey or the RHS are all key attribute.
- Here every F.D satisfies this property.

BCNF

- A table is in BCNF if and only if for every non-trivial FD, the LHS is a superkey.
- Here every F.D satisfies this property.

7. PAID:

R1:

CARD_NO	E_NAME	E_NO	P_REG_ID

Functional Dependencies:

P_reg_id--> card_no, e_name, e_no

Card_no --> e_name, e_no

e_no --> e_name

NORMALIZATION- PROCESS

Candidate Keys:

c.k={P_reg_id}

finding the **minimal cover of the FDs**, which includes the FDs
P_reg_id --> card_no
card_no --> e_no
e_no --> e_name

Checking Normal Form:

1ST Normal Form 

- Yes because every attribute here is of atomic value.

2nd Normal Form 

- for each non-trivial FD, checking whether the LHS is a proper subset of some candidate key or the RHS are not all prime attributes.

3rd Normal Form 

for each FD, checking whether the LHS is superkey or the RHS are all key attributes
checking functional dependency P_reg_id --> card_no, e_no, e_name
checking functional dependency card_no --> e_name, e_no
The above FD violates definition of 3NF: it is non-trivial, LHS is not superkey, RHS contains a non-key attribute.

NORMALIZATION- PROCESS

If a Table is not in 3rd N.F, then by definition we can say that it will not be in it's higher normal form.

BCNF 

Normalizing to 3 N.F

- Checking with each functional dependency that violates 3rd N.F.
- The FD [P_reg_id --> card_no, e_no, e_name, card_no --> e_name, e_no] The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes).The following 3NF table is obtained:

R2:

CARD_NO	E_NO

card_no --> e_no

P_reg_id --> card_no **R3:**

CARD_NO	P_REG_ID

R4:

E_NAME	E_NO

e_no --> e_name

Both R2, R3 and R4 are in 3 N.F and in B.C.N.F as well.

After Normalization we have achieved:-

1 N.F **2 N.F** **3 N.F** **B.C.N.F**

QUERIES

Q.1. Find the name on the card of all transection that included an event named ‘Sparkling Speakers’ using nested subqueries.

select name_on_card from card where e_name in (select e_name from events where e_name = ‘Sparkling Speakers');

```
SQL> select name_on_card from card where e_name in (select e_name from events where desk_id = 1354);
NAME_ON_CARD
--------------------
Kunal Mehta
Jayanti Patel
Minal Shah

3 rows selected.
```

QUERIES

Q.2. Find the list of participients who paid their bill through the card number starting with either ‘5680 (Master card)’ or ‘7900 (visa card)’ using subquery.

select f_name from Participient where p_id in (select p_id from participient, card where card_no like '[5860]%' or card_no like '[7900]%');

```
SQL> select f_name from Participient where p_id in (select p_id from participient, card where card_no  like '[5860]%' or card_no like
'[7900]%');
select p_id from participient
                *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> select f_name from Participant where p_id in (select p_id from Participant, card where card_no  like '[5860]%' or card_no like
'[7900]%');
F_NAME
------------------
Punam
Kunal
Nitin
Bheem
Sanjay
Rakesh
Shivendra
```

QUERIES

Q.3. Display Participent id, event name, transection id and transection date of a participent enrolled in every event.

Using nested query.

```
select distinct event_details.p_id, e_name, trans_id, trans_date from trans_history,event_details where event_details.p_id in( select distinct p_id from participent);
```

```
SQL> SQL> select distinct event_details.p_id, e_name, trans_id, trans_date from trans_history,event_details where event_details.p_id in(
select distinct p_id from participent);

P_ID E_NAME               TRANS_ID TRANS_DAT
---- -------------------- ---------- ---------
P207 Brain fight          7.9001E+15 19-AUG-20
P102 Cute cuples          5.8610E+15 21-AUG-20
P109 D.J night            5.8600E+15 03-AUG-20
P207 Brain fight          5.8600E+15 03-AUG-20
P101 Swimming             7.9001E+15 19-AUG-20
P102 Cute cuples          5.8607E+15 15-AUG-20
P101 Swimming             7.9001E+15 12-AUG-20
P109 D.J night            7.9001E+15 12-AUG-20

P_ID E_NAME               TRANS_ID TRANS_DAT
---- -------------------- ---------- ---------
P109 D.J night            5.8607E+15 15-AUG-20
P106 Blast Badminton      5.8600E+15 03-AUG-20
P101 Swimming             5.8600E+15 03-AUG-20
P106 Blast Badminton      7.9001E+15 19-AUG-20
P207 Brain fight          5.8607E+15 15-AUG-20
P102 Cute cuples          7.9001E+15 19-AUG-20
P109 D.J night            7.9001E+15 19-AUG-20
P106 Blast Badminton      7.9001E+15 12-AUG-20

P_ID E_NAME               TRANS_ID TRANS_DAT
---- -------------------- ---------- ---------
P207 Brain fight          5.8610E+15 21-AUG-20
P102 Cute cuples          5.8600E+15 03-AUG-20

18 rows selected.
```

QUERIES

Q.4. Display the event name and number of all events who were held between ’01-Aug-2020’ and ’20-Aug-2020’.

Select distinct e_name, e_no from Events, Event_det e where e.E_date > '01-Aug-2020' and e.E_date < '20-Aug-2020';

```
Flash Racing
Javallian Throw

SQL> Select distinct E_name, e_no from Events, Event_det e where e.E_date > '01-Aug-2020' and e.E_date < '20-Aug-2020';
E_NAME          E_NO
--------------- ----
Blast Badminton E132
Dolphin Race    E005
Cute Cuples     E897
Sparkling Speakers E332
Brain Fight     E086
Swimming        E135
D.J night       E045
```

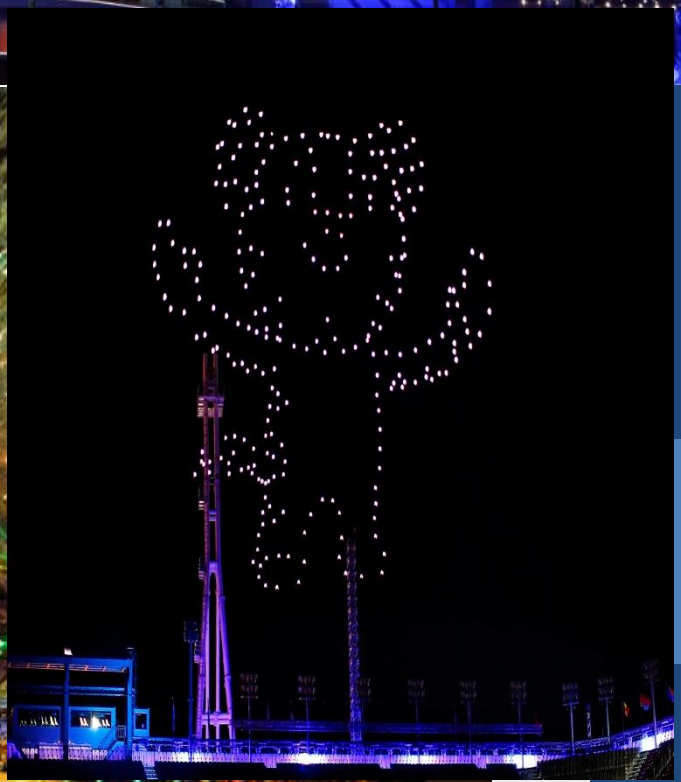
QUERIES

Q.5.Find the last date of month from the start date of an event with event id ‘E121’;

select last_day(e_date) from Event_det, Event where E_no = 'E121';

```
SQL> select last_day(e_date) from Event_det, Event where E_no = 'E121';
 LAST_DAY(E_DATE)
 ------------------
 19-AUG-20
```

CLOSING CEREMONY OF FIT INDIA FITNESS ACTIVITY FAIR



PL/SQL QUERIES

1. LUCKY DRAW WINNER PROGRAM

```
declare
type bonenza is varray(20) of varchar(4);
lucky_id bonenza;
x number(2) := &sponcer_key;
i number(2);
j number(10);
begin
lucky_id :=
bonenza('P001','P005','P008','P011','P016','P018','P020','P023','P025','P026','P031','P035','
P045','P051','P058','P059','P062','P063','P068','P113' );
i := 1;
j := 1;
dbms_output.put_line(' . ');
dbms_output.put_line(' who is the lucky '||x||' ?');
dbms_output.put_line(' who is the lucky '||x||' ?');
dbms_output.put_line(' who is the lucky '||x||' ?');
dbms_output.put_line('Calculating.....');
dbms_output.put_line('.....Calculating.....');
dbms_output.put_line('.....Calculating.....');
dbms_output.put_line('.....Calculating');
for j in 1..200000 loop
dbms_output.put_line('   ');
end loop;
loop
dbms_output.put_line('.....lucky draw.....'||lucky_id(i));
i := i+1;
exit when i>20;
end loop;
i :=1;
loop
dbms_output.put_line('.....lucky draw.....'||lucky_id(i));
i := i+1;
exit when i>20;
end loop;
dbms_output.put_line(' . ');
dbms_output.put_line(' . ');
dbms_output.put_line(' . ');
dbms_output.put_line('lucky draw completed.');
dbms_output.put_line('  !!!The Winner is!!!  '||lucky_id(x));
end;
```

PL/SQL QUERIES

```
 31  /
Enter value for sponcer_key: 15
old   4: x number(2) := &sponcer_key;
new   4: x number(2) := 15;
.
who is the lucky 15 ?
who is the lucky 15 ?
who is the lucky 15 ?
..........lucky draw..........P001
..........lucky draw..........P005
..........lucky draw..........P008
..........lucky draw..........P011
..........lucky draw..........P016
..........lucky draw..........P018
..........lucky draw..........P020
..........lucky draw..........P023
..........lucky draw..........P025
..........lucky draw..........P026
..........lucky draw..........P031
..........lucky draw..........P035
..........lucky draw..........P045
..........lucky draw..........P051
..........lucky draw..........P058
..........lucky draw..........P059
..........lucky draw..........P062
..........lucky draw..........P063
..........lucky draw..........P068
..........lucky draw..........P113
..........lucky draw..........P001
..........lucky draw..........P005
..........lucky draw..........P008
..........lucky draw..........P011
..........lucky draw..........P016
..........lucky draw..........P018
..........lucky draw..........P020
..........lucky draw..........P023
..........lucky draw..........P025
..........lucky draw..........P025
..........lucky draw..........P026
..........lucky draw..........P031
..........lucky draw..........P035
..........lucky draw..........P045
..........lucky draw..........P051
..........lucky draw..........P058
..........lucky draw..........P059
..........lucky draw..........P062
..........lucky draw..........P063
..........lucky draw..........P068
..........lucky draw..........P113
..........lucky draw..........P001
..........lucky draw..........P005
..........lucky draw..........P008
..........lucky draw..........P011
..........lucky draw..........P016
..........lucky draw..........P018
..........lucky draw..........P020
..........lucky draw..........P023
..........lucky draw..........P025
..........lucky draw..........P026
..........lucky draw..........P031
..........lucky draw..........P035
..........lucky draw..........P045
..........lucky draw..........P051
..........lucky draw..........P058
..........lucky draw..........P059
..........lucky draw..........P062
..........lucky draw..........P063
..........lucky draw..........P068
..........lucky draw..........P113
.
.
.
lucky draw completed.
!!!The Winner is!!!  P058

PL/SQL procedure successfully completed.

SQL>
```

2. **Cursor Program** to combine the required number of tables and to show the participient details to the end usr.

```
declare
v_pid participent.p_id%type;
v_pname participent.p_name%type;
v_pno participent.p_no%type;
v_ename event_details.e_name%type;
cursor participent_details is
select participent.p_id, p_name, p_no, e_name from
participent, event_details where
participent.p_id=event_details.p_id;
begin
dbms_output.put_line(' -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_');
open participent_details;
loop
fetch participent_details into v_pid, v_pname, v_pno,
v_ename;
dbms_output.put_line('I.D: '||v_pid||'   Name: '||v_pname||'
Contact No: '||v_pno||'   Registered Event: '||v_ename);
dbms_output.put_line(' -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_');
exit when participent_details%NotFound;
end loop;
close participent_details;
end;
```

PL/SQL QUERIES (CURSOR PROGRAM)

```
 19  /
-----------------------------------------------------------------------------------
I.D: P106   Name: Sanjay   Contact No: 9090312010   Registered Event: Blast
Badminton
-----------------------------------------------------------------------------------
I.D: P207   Name: Nitin   Contact No: 7652031201   Registered Event: Brain fight
-----------------------------------------------------------------------------------
I.D: P102   Name: Kunal   Contact No: 9502130202   Registered Event: Cute cuples
-----------------------------------------------------------------------------------
I.D: P101   Name: Punam   Contact No: 9876565656   Registered Event: Swimming
-----------------------------------------------------------------------------------
I.D: P109   Name: Rakesh   Contact No: 7878956450   Registered Event: D.J night
-----------------------------------------------------------------------------------
I.D: P109   Name: Rakesh   Contact No: 7878956450   Registered Event: D.J night
-----------------------------------------------------------------------------------

PL/SQL procedure successfully completed.
```

PL/SQL QUERIES (CURSOR + PROCEDURE PROGRAM)

3. Implement a **procedure** to get the combined name of club concatinated with Sponcer.

```
CREATE OR REPLACE PROCEDURE
format_associate_sponcer(format_name1 IN OUT varchar2,
format_name2 IN OUT varchar2) IS
begin
format_name1 := SUBSTR(format_name1, 1, 3) ||
SUBSTR(format_name2, 5, 3);
end;

declare
leter1 varchar(20);
leter2 varchar(20);
cursor cure is select c_name, sponcer from club;
begin
open cure;
loop
fetch cure into leter1, leter2;
format_associate_sponcer(leter1, leter2);
dbms_output.put_line(leter1);
exit when cure%NotFound;
end loop;
close cure;
end;
```

PL/SQL QUERIES (CURSOR + PROCEDURE PROGRAM)

```
L/SQL procedure successfully completed.

QL> CREATE OR REPLACE PROCEDURE format_associate_sponcer(format_name1 IN OUT varchar2, format_name2 IN OUT varchar2) IS
 2  begin
 3  format_name1 := SUBSTR(format_name1, 1, 3) || SUBSTR(format_name2, 5, 3);
 4  end;
 5  /

rocedure created.
```

```
SQL> declare
  2  leter1 varchar(20);
  3  leter2 varchar(20);
  4  cursor cure is select c_name, sponcer from club;
  5  begin
  6  open cure;
  7  loop
  8  fetch cure into leter1, leter2;
  9  format_associate_sponcer(leter1, leter2);
 10  dbms_output.put_line(leter1);
 11  exit when cure%NotFound;
 12  end loop;
 13  close cure;
 14  end;
 15  /
SubBul
Augico
Spoani
Infle
Minorw
Eurnda
Eurnda
```

PL/SQL QUERIES(TRIGGERS)

4. Create a **Trigger** that will fire when the new total amount entered is less than the previous amount during the transection.

```
CREATE OR REPLACE TRIGGER total_update_check
BEFORE UPDATE OF total ON trans_history
FOR EACH ROW
BEGIN
if :NEW.total < :OLD.total then
raise_application_error(-20125, '::::::Total should be greater than the previous::::::');
end if;
end;
```

```
SQL> CREATE OR REPLACE TRIGGER total_update_check
  2  BEFORE UPDATE OF total ON trans_history
  3  FOR EACH ROW
  4  BEGIN
  5  if :NEW.total < :OLD.total then
  6  raise_application_error(-20125, '::::::Total should be greater than the previous::::::');
  7  end if;
  8  end;
  9  /

Trigger created.

SQL>  update trans_history set total=400 where p_id='P102';
 update trans_history set total=400 where p_id='P102'
        *
ERROR at line 1:
ORA-20125: ::::::Total should be greater than the previous::::::
ORA-06512: at "SYSTEM.TOTAL_UPDATE_CHECK", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TOTAL_UPDATE_CHECK'


SQL>
```

PL/SQL QUERIES(TRIGGERS)

5. Create a **Trigger** that will fire when the event date entered is after 21st Aug or before 01st Aug 2020

```
CREATE OR REPLACE TRIGGER evnt_date_check
BEFORE INSERT OR UPDATE OF e_date ON event_details
FOR EACH ROW
BEGIN
if ( :NEW.e_date NOT BETWEEN '01-Aug-20' AND '20-Aug-20' ) then
raise_application_error(-20125, ':::::::Event date should between 1st to 20th August:::::::');
end if;
end;
```

```
SQL> CREATE OR REPLACE TRIGGER evnt_date_check
  2  BEFORE INSERT OR UPDATE OF e_date ON event_details
  3  FOR EACH ROW
  4  BEGIN
  5  if ( :NEW.e_date NOT BETWEEN '01-Aug-20' AND '20-Aug-20' ) then
  6  raise_application_error(-20125, ':::::::Event date should between 1st to 20th August:::::::');
  7  end if;
  8  end;
  9  /

Trigger created.

SQL> update event_details set e_date='23-Aug-20' where p_id='P102';
update event_details set e_date='23-Aug-20' where p_id='P102'
       *
ERROR at line 1:
ORA-20125: :::::::Event date should between 1st to 20th August:::::::
ORA-06512: at "SYSTEM.EVNT_DATE_CHECK", line 3
ORA-04088: error during execution of trigger 'SYSTEM.EVNT_DATE_CHECK'
```

PL/SQL QUERIES(TRIGGERS)

5. Create a **Trigger** that will fire when the transection happens after 21st Aug or before 01st Aug 2020.

```
CREATE OR REPLACE TRIGGER trans_date_check
BEFORE INSERT OR UPDATE OF trans_date ON trans_history
FOR EACH ROW
BEGIN
if ( :NEW.trans_date NOT BETWEEN '01-Aug-20' AND '20-Aug-20' ) then
raise_application_error(-20125, ':::::::Event date should between 1st to 20th August:::::::');
end if;
end;
```

```
SQL> CREATE OR REPLACE TRIGGER trans_date_check
  2  BEFORE INSERT OR UPDATE OF trans_date ON trans_history
  3  FOR EACH ROW
  4  BEGIN
  5  if ( :NEW.trans_date NOT BETWEEN '01-Aug-20' AND '20-Aug-20' ) then
  6  raise_application_error(-20125, ':::::::Event date should between 1st to 20th August:::::::');
  7  end if;
  8  end;
  9  /

Trigger created.

SQL>  update trans_history set trans_date='23-Aug-20' where p_id='P102';
 update trans_history set trans_date='23-Aug-20' where p_id='P102'
        *
ERROR at line 1:
ORA-20125: :::::::Event date should between 1st to 20th August:::::::
ORA-06512: at "SYSTEM.TRANS_DATE_CHECK", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRANS_DATE_CHECK'


SQL>
```

PL/SQL QUERIES (CURSOR + PROCEDURE PROGRAM)

6. End user wants to associate every event to where it is being held. Create a **procedure** that can do this task.

```
CREATE OR REPLACE PROCEDURE avanue(e_name in out varchar, plot_name in out varchar) IS
begin
e_name :=  e_name||'/*/*/*/*/*\*\*\*\*\ '||plot_name;
end;

declare
v_ename varchar(500);
v_plotname varchar(500);
cursor newor is select e_name, plot_name from event, venue where event.desk_id=venue.desk_id;
begin
open newor;
loop
fetch newor into v_ename, v_plotname;
avanue(v_ename, v_plotname);
dbms_output.put_line(v_ename);
exit when newor%NotFound;
end loop;
close newor;
end;
```

PL/SQL QUERIES (CURSOR + PROCEDURE PROGRAM)

```
SQL> declare
  2  v_ename varchar(500);
  3  v_plotname varchar(500);
  4  cursor newor is select e_name, plot_name from event, venue where event.desk_id=venue.desk_id;
  5  begin
  6  open newor;
  7  loop
  8  fetch newor into v_ename, v_plotname;
  9  avanue(v_ename, v_plotname);
 10  dbms_output.put_line(v_ename);
 11  exit when newor%NotFound;
 12  end loop;
 13  close newor;
 14  end;
 15  /
```

```
 15  /
Blast Badminton/*/*/*/*/*\*\*\*\*\*\ Margdarshika
Cute cuples/*/*/*/*/*\*\*\*\*\*\ Smriti
Brain Fight/*/*/*/*/*\*\*\*\*\*\ Ashta van
Swimming/*/*/*/*/*\*\*\*\*\*\ Margdarshika
D.J night/*/*/*/*/*\*\*\*\*\*\ Law garden
D.J night/*/*/*/*/*\*\*\*\*\*\ Law garden/*/*/*/*/*\*\*\*\*\*\ Law garden

PL/SQL procedure successfully completed.

SQL>
```

PL/SQL QUERIES(FUNCTIONS)

7. Create a **Function** that will calculate the G.S.T over each transection.

```
CREATE OR REPLACE FUNCTION tax_cal(fees number)
RETURN number IS
begin
return (fees*18/100);
end;

select E_name,  tax_cal(e_fees) as fees_gst from event_details;
```

```
SQL> CREATE OR REPLACE FUNCTION tax_cal(fees number) RETURN number IS
  2  begin
  3  return (fees*18/100);
  4  end;
  5  /

Function created.

SQL>
```

```
SQL> select E_name, tax_cal(e_fees) as fees_gst from event_details;

E_NAME               FEES_GST
-------------------- ----------
Blast Badminton             270
Brain fight                 180
Cute cuples                 216
Swimming                    333
D.J night                   450
```

- FRONT END IN DOC.