DBMS Project
IRCTC Clone

Ujjwal Godara	2020345
Arvind	**2020286**
Rahul Raj	2020322
Shivam Aggarwal	**2020123**

Scope of Project

IRCTC stands for Indian Railways Catering and Tourism Corporation. It is an Indian public sector that provides ticketing, catering and tourism services to Indian railways. Major portion of users use IRCTC app for ticketing only.

The system provides a functionality to passengers to add themselves to the view the train listings. The user can also update their personal details in the system. The system performs functions like providing the best options of the available trains from a specified source and the destination to the passengers. The train details has comprehensive information like train number, train name, train type, available class types along with respective fares, the frequency of the train during the week, seat availability, route and stoppage details and total distance and total time taken for the journey.

Passengers can book their tickets based on their preference and availability. The system also provides a functionality to the passenger to cancel their tickets.

The super admin of the user can add, update and delete new trains from the system, manage stations and route details and also manage passengers.

Entities

- **Passenger:** This table includes information about the passenger who would be travelling such as name, contact details, age and user id.
- **Ticket:** This table includes passenger details corresponding to the specific source and destination, date of journey, PNR, Train no and fare.
- **Station:** This table contains the station name, Location and station Code.
- **Train:** This table contains train name, train number, total seats, Departure date and time.
- **User:** This table contains User ID, Age, Name and Contact no.

- **Stakeholders:**

  Passengers, Govt. Officials of Railways (Super Admin)

- **Entities**

  User, Passenger, Train, Station, Ticket

- **Ternary Relationship**

  No ternary relationship

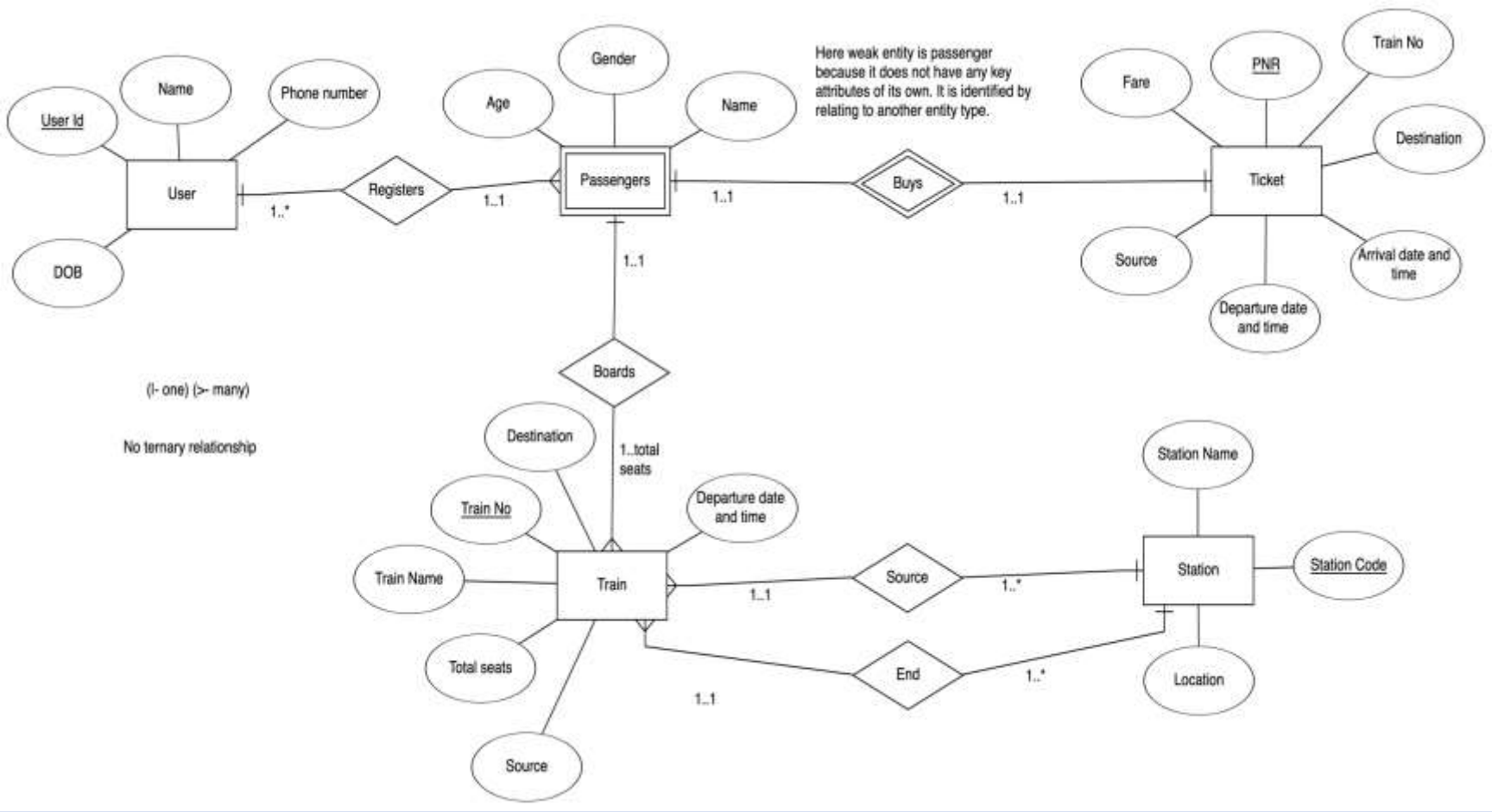
- **Weak Entity**

  Passenger because it does not have any key attributes of its own. Identifies by relation to another entity type

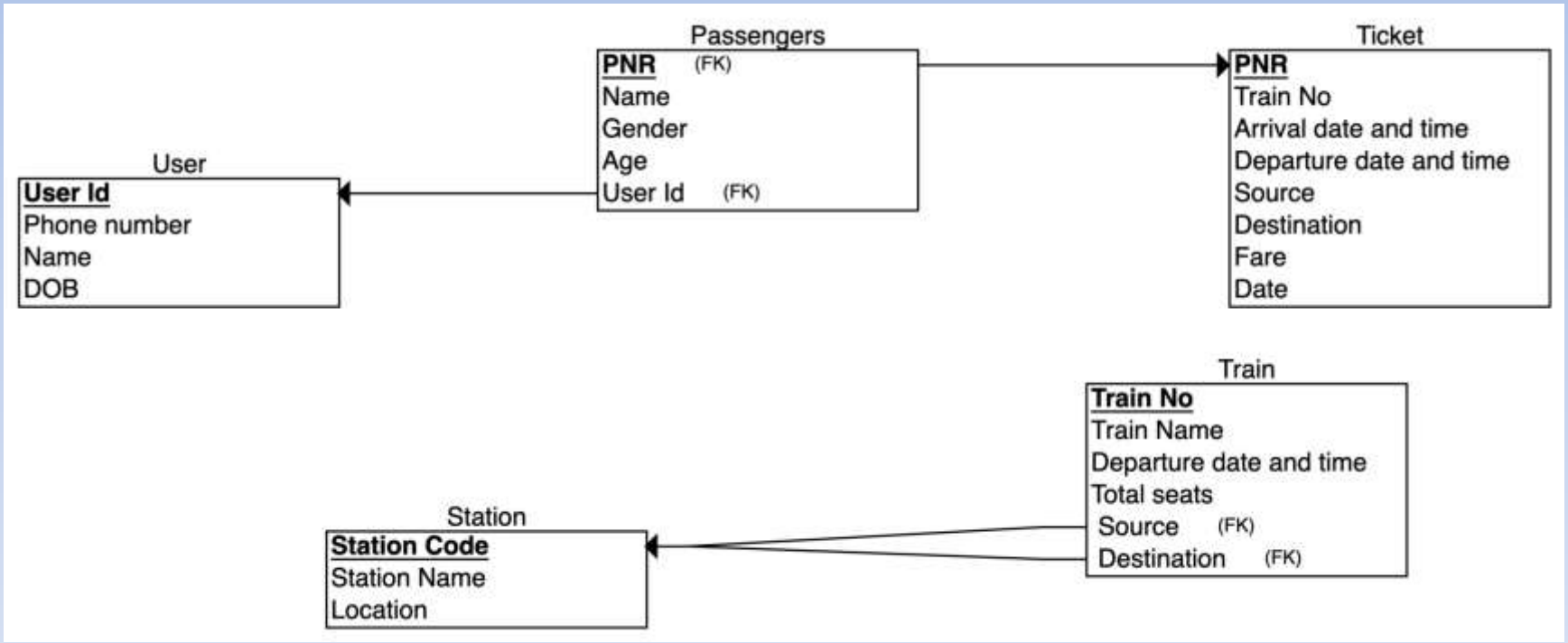
Relationship Between entities:

SR. No.	Relationship Type	Entities Involved	Entity Relationship
1	Registers	User and Passengers	One to many
2	Boards	Passengers and Trains	One to one
3	Buys	Passengers and Ticket	One to one
4	Source, End	Train and Station	One to one
5	Registers	Passengers And User	One to one
6	Boards	Trains and Passengers	One to total seats
7	Buys	Ticket and Passengers	One to one
8	Source, End	Station and Train	One to many

ER Diagram


Here weak entity is passenger because it does not have any key attributes of its own. It is identified by relating to another entity type.
User
User Id
Name
Phone number
DOB
Registers
1..*
1..1
Passengers
Gender
Age
Name
1..1
Buys
1..1
Ticket
Fare
PNR
Train No
Destination
Source
Arrival date and time
Departure date and time
Boards
(|- one) (>- many)
No ternary relationship
Destination
1..total seats
Train No
Departure date and time
Train Name
Train
Total seats
Source
1..1
Source
1..*
Station
Station Name
Station Code
End
1..*
1..1
Location

Relational Schema



Tables & Data Population

User

PK: User ID

Name	Age	UserID	Phonenumber
George Rothwell	31	1	75842336525
Doug Rose	86	2	14382462843
Julian Khan	85	3	05354301450
Johnathan Stewart	76	4	58081225850
Ronald Grady	93	5	26161863331
Holly Jackson	39	6	58717306346
Tyson Grey	53	7	00152358168
Ron Andersson	22	8	04360047142
Alexander Coates	57	9	82483012088
Jules Hopkinson	95	10	73224256834
Angelique Webster	18	11	37557506188
Zara Wise	18	12	07315764712
Oliver Fox	96	13	70104130627
Penelope Jackson	63	14	03882587316
Rocco Clarkson	55	15	03111701275

Train

PK: Train No

TrainNo	Trainname	DepartureDateandtime	TotalSeats	Source	Destination
11084	MH35haEnUl	6419-05-30 17:21:41	533	1	4
13709	kjWy3QuRfd	8987-09-06 12:34:39	582	3	13
16048	VlpICiP8do	3552-09-24 02:43:12	535	2	10
16542	RjO8wOoOja	6992-01-19 19:16:46	563	12	10
18385	ZKK8hmXitp	2704-04-11 10:10:56	594	2	6
20991	lI7PZjMF4o	9276-10-20 07:53:08	598	7	8
21528	sjGNJNZL0q	2983-12-27 21:42:15	553	14	1
21772	HWjBfCyz1p	3082-08-10 02:05:55	565	9	6
25008	t6NcFWk45F	5309-04-19 08:33:51	563	2	6
25230	AcgLrNaefl	4700-11-27 21:11:01	509	1	8
31612	8iJxUgQAz6	4699-03-24 06:19:05	564	12	3
34474	MD7A17RQ…	8622-08-24 21:32:52	545	10	1
42059	b8lihjtVWH	5949-02-01 13:55:46	519	8	2
43719	jGq4vq0YE7	4535-10-02 00:55:57	529	5	3
45846	lhflHeDBTX	3310-09-09 05:53:36	554	3	10

Ticket PK: PNR

PNR	Fare	TrainNO	Source	Destination	DepartureDateTime
12344	115	11084	1	4	6419-05-30 17:21:41
12345	115	11084	1	4	6419-05-30 17:21:41
13370	1478	11084	1	4	6419-05-30 17:21:41
13556	427	13709	3	13	8987-09-06 12:34:39
23746	179	16048	2	10	3552-09-24 02:43:12
32180	1471	16542	12	10	6992-01-19 19:16:46
32290	1009	18385	2	6	2704-04-11 10:10:56
34122	205	20991	7	8	9276-10-20 07:53:08
36769	1245	21528	14	1	2983-12-27 21:42:15
37715	1826	21772	9	6	3082-08-10 02:05:55
39079	560	25008	2	6	5309-04-19 08:33:51
41550	1004	25230	1	8	4700-11-27 21:11:01
42510	1223	31612	12	3	4699-03-24 06:19:05
43407	168	34474	10	1	8622-08-24 21:32:52
44901	647	42059	8	2	5949-02-01 13:55:46
44944	1907	43719	5	3	4535-10-02 00:55:57
49429	299	45846	3	10	3310-09-09 05:53:36
49430	717	25008	2	6	5309-04-19 08:33:51
49431	348	20991	7	8	9276-10-20 07:53:08

Station

PK: Station Code

StationName	Stationcode	Location
Lancaster	1	Otawa
Lincoln	2	Oakland
Madison	3	Phoenix
Madison	4	Las Vegas
Oklahoma City	5	Richmond
Seattle	6	Jacksonville
Lincoln	7	Venice
Rochester	8	Jacksonville
Las Vegas	9	Amarillo
Bridgeport	10	Bellevue
Anaheim	11	Lancaster
Phoenix	12	San Antonio
Louisville	13	Richmond
Sacramento	14	Santa Ana
Anaheim	15	Louisville

Passenger

PK: PNR

Name	Gender	Age	PNR	UserId
Johnath…	male	76	12344	4
Doug R…	male	86	12345	2
arvind	Female	7	13370	1
Harvey	Male	16	23746	3
John	Male	65	32180	4
Bryce	Female	54	32290	5
Alexander	Male	46	34122	6
Ruth	Female	84	37715	8
Phillip	Male	79	39079	9
Julianna	Female	58	41550	10
Marjorie	Female	78	42510	11
Emery	Female	60	43407	12
Tony	Male	71	44901	13
Brad	Male	24	44944	14
Stacy	Female	66	49429	15

SQL Queries (for mid-sem)

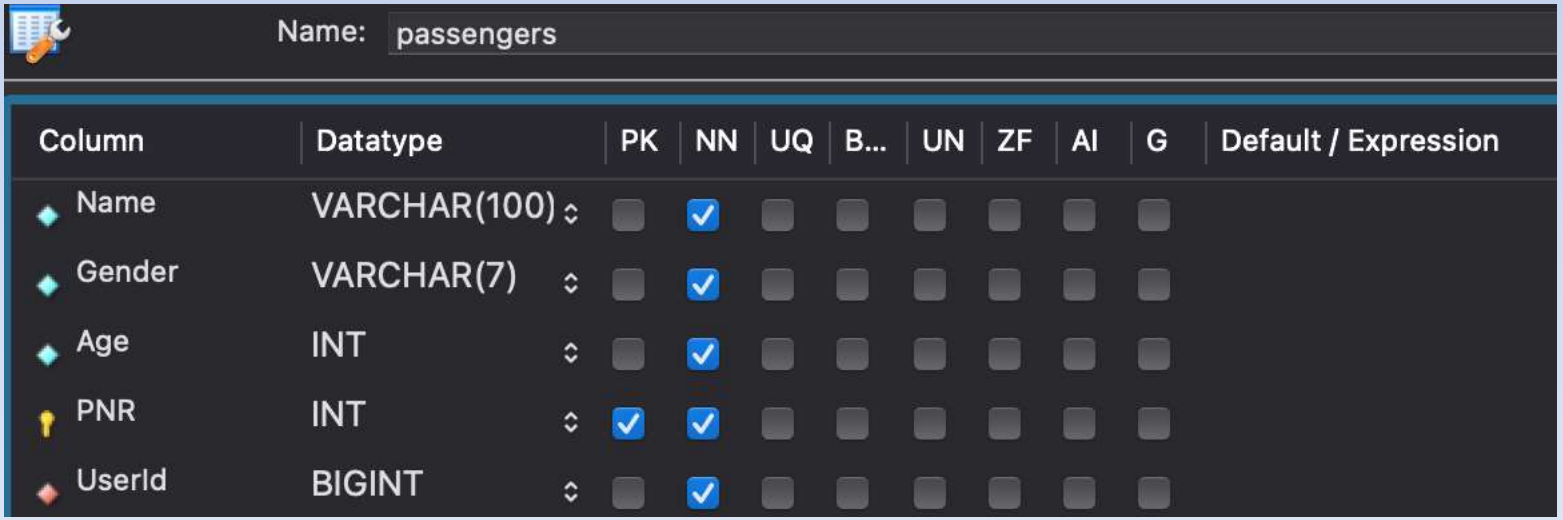
```sql
use irctc;
/*Select p.pnr,p.name,p.gender,p.age,p.userid from passengers as p where userid = '10' order by p.pnr;/*tickets booked by this user id */
/*1*/Select t.pnr,p.name,p.age,t.trainno,t.arrivaldatetime as "Arrival Date and Time", t.departuredatetime as "Departure Date And Time"
    from tickets as t,passengers as p where t.pnr = p.pnr and userid = 11;/* */

/*2*/Select ti.pnr,ti.fare,ti.trainno,ti.arrivaldatetime,ti.departuredatetime,t.totalseats,t.source,t.destination
    from tickets as ti ,trains as t where ti.trainno = t.trainno and t.departuredateandtime
    between '6992-01-19 19:16:46' and '9276-10-20 07:53:08' order by pnr;/* trains available between two dates*/

/*3*/Select t.source,s.stationname,count(*) as "NoOfTrainsLeaving"
    from trains as t , stations as s where t.source = s.stationcode group by source;/*no of trains leaving a source station*/

/*4*/Select t.pnr,p.name,t.fare from tickets as t,passengers as p where t.pnr = p.pnr order by t.pnr; /*passengers fare*/

/*5*/Select p.pnr,p.name,p.age,p.gender
    from tickets as t,passengers as p
    where t.pnr = p.pnr and t.trainno = '11084' order by t.pnr; /*list of passengers boarding a train*/
```

```
18
19 /*6*/Select p.userid,sum(fare),count(p.pnr) as 'tickets booked'
20     from tickets as t,passengers as p where t.pnr = p.pnr group by p.userid;/*no of tickets booked by a user and total fare*/
21
22 /*7*/create view pass as select name, pnr from passengers;
23
24 /*8*/update pass set name = 'arvind' where pnr = '36769';
25
26 /*9*/delete from pass where pnr = '36769';
27
28 /*10*/select * from trains where source = 2;/*list of trains leaving same station*/
29
30 /*11*/select p.pnr,p.name,p.age,t.fare,t.fare *0.9 as 'discounted price' from passengers as p join tickets as t where p.pnr=t.pnr and p.age>60;
31
```

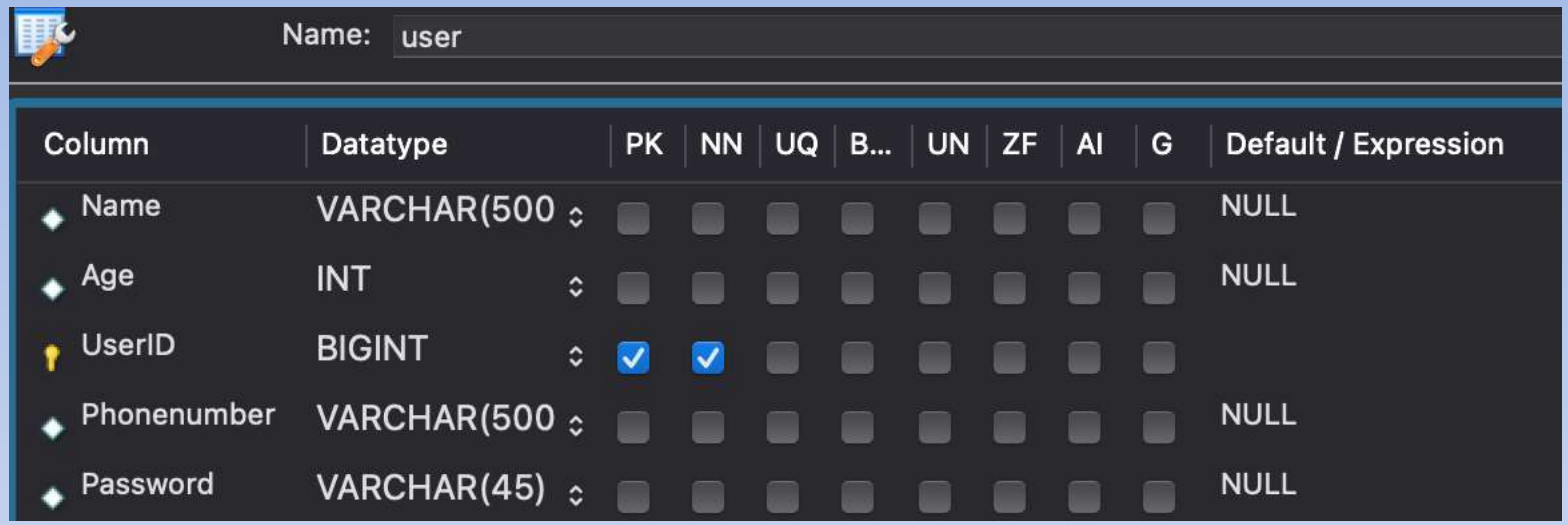
Constraints

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table.

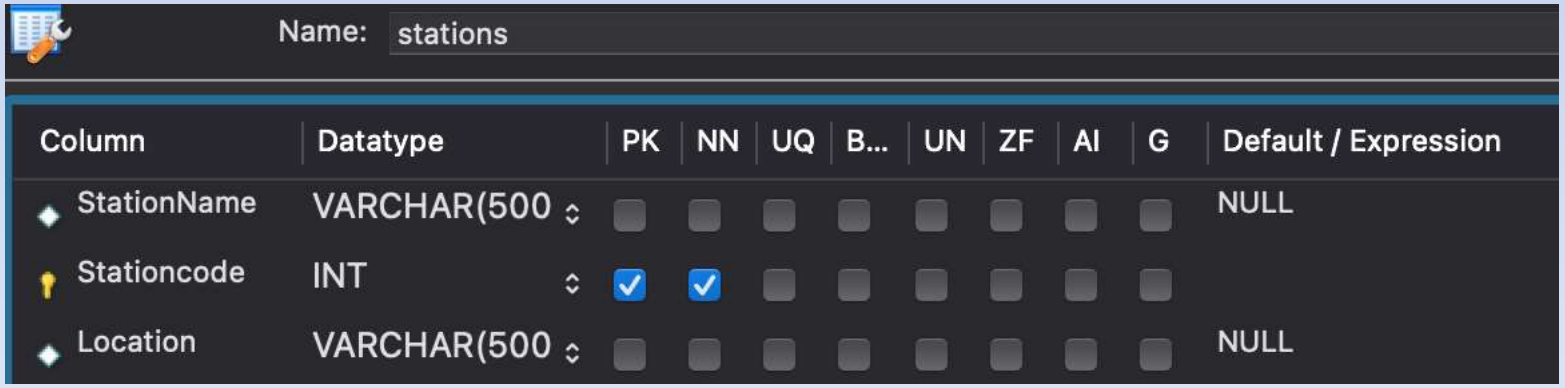
- Passengers



-User



-Stations



-Trains

Name: trains

Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G	Default / Expression
TrainNo	INT	☑	☑	☐	☐	☐	☐	☐	☐	
Trainname	VARCHAR(500	☐	☑	☐	☐	☐	☐	☐	☐	
DepartureDat...	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	
TotalSeats	INT	☐	☑	☐	☐	☐	☐	☐	☐	
Source	INT	☐	☑	☐	☐	☐	☐	☐	☐	
Destination	INT	☐	☑	☐	☐	☐	☐	☐	☐	

-Tickets

Name: tickets

Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G	Default / Expression
PNR	INT	☑	☑	☐	☐	☐	☐	☐	☐	
Fare	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
TrainNO	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Source	VARCHAR(200	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Destination	VARCHAR(200	☐	☐	☐	☐	☐	☐	☐	☐	NULL
DepartureDat...	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Views and Grants

Views- View is a virtual table based on the result-set of an SQL statement.

Grants - SQL Grant is used to provide permissions like Select, All, Execute to user on the database objects like Tables, Views, Databases and other objects in a SQL Server.

There are **2 views – Passenger and Admin**

Passengers can book their tickets based on their preference and availability. The system also provides a functionality to the passenger to cancel their tickets.

The super admin of the user can add, update and delete new trains from the system, manage stations and route details and also manage passengers.

Views

```
/*passenger view*/
create view passenger_view as select passengers.name, passengers.gender, passengers.age, passengers.pnr,
tickets.trainno, tickets.fare, tickets.departuredatetime, tickets.arrivaldatetime from passengers
inner join tickets where passengers.pnr = tickets.pnr;

select * from passenger_view;

/* admin view */

create view admin_view as select passengers.name, passengers.gender, passengers.age, passengers.pnr,
tickets.trainno, tickets.fare, tickets.departuredatetime, tickets.arrivaldatetime, station.stationname, station.code, station.location,
 trains.trainno, trains.trainname,trains.departuredateandtime,trains.totalseats,trains.source,trains.destination,
 user.name, user.age,user.userid,user.ponenumber,user.password from passengers
inner join tickets
inner join trains
inner join user
inner join stations
where passengers.pnr = tickets.pnr &
tickets.trainno = trains.trainno;
```

Grants

```
/*grant for user*/
GRANT DELETE, SELECT
ON passengers
TO customer@localhost;

GRANT DELETE, SELECT
ON tickets
TO customer@localhost;

GRANT SELECT
ON stations
TO customer@localhost;

GRANT SELECT
ON trains
TO customer@localhost;
```

```
/*grant for admin*/
GRANT INSERT, UPDATE, DELETE, SELECT
ON passengers
TO admin@localhost;

GRANT INSERT, UPDATE, DELETE, SELECT
ON stations
TO admin@localhost;

GRANT INSERT, UPDATE, DELETE, SELECT
ON tickets
TO admin@localhost;

GRANT INSERT, UPDATE, DELETE, SELECT
ON trains
TO admin@localhost;

GRANT INSERT, UPDATE, DELETE, SELECT
ON user
TO admin@localhost;
```

Embedded SQL Queries

Embedded SQL is a method of combining the computing power of a programming language and the database manipulation capabilities of SQL

```
query 1
//calculate the total fare collected

declare
a int;
begin
        a=0;
        for item In(
                select fare from passengers
        )loop
        a=a+item.fare
        end loop;
        dbms.output.put_line('Total fare collected is '||a);
end;
```

```
//query 2
//to increase or decrease the fare for 18+. also tells the no of rows in which fare updated
//also has a trigger in it that tells that who updated the trigger after each update

SET SERVEROUTPUT ON;

Create or replace trigger fare_increase
before update on tickets
enable
declare
        v_user VARCHAR2(20);
BEGIN
  UPDATE tickets SET fare = fare * 1.05 WHERE age<18;
  select user INTO v_user from dual
  dbms_output.put_line('You just updated a fare mr. '\\v_user);
  dbms_output.put_line('Updated ' || SQL%ROWCOUNT || ' fare.');
END;
```

```
//query 3
//It does a query to get the name and pnr of every passenger that has aa in its name and also uu not in its name
.

BEGIN
  FOR item IN
  (
    SELECT name, pnr FROM passengers WHERE name=%aa%
       AND name NOT LIKE '%uu%'
  )
  LOOP
    dbms_output.put_line('name = ' || item.name ||
      ', pnr = ' || item.pnr);
  END LOOP;
END;
```

```
query 4
DECLARE
-- we find the average fare.
-- Then we find all the passenger paying
-- more than that average fare.
   CURSOR c1 IS
     SELECT name,pnr,fare
       FROM passengers t
     WHERE
       fare >
       (
         SELECT AVG(salary)
           FROM passengers
       )
BEGIN
   FOR person IN c1
   LOOP
     dbms_output.put_line('paying above-average fare = ' ||
       person.name person.fare);
   END LOOP;
END;
```

```
query 5
returns the train no and passenger id if train has more than 2 passeneger

DECLARE
  CURSOR c1 IS
    SELECT t1.name, trainno
      FROM train t1,
      (
        SELECT name, COUNT(*) as passenger
          FROM passengers
          GROUP BY pnr
      ) t2
    WHERE
      t1.trainno = t2.trainn
      AND passenger >= 2;
BEGIN
   FOR dept IN c1
   LOOP
     dbms_output.put_line('train no = ' || dept.name ||
       ', passenger = ' || dept.passenger);
   END LOOP;
END;
```

```python
@app.route('/', methods = ['GET', 'POST'])
def index(): # function returns what to print on screen
    error = None
    if request.method == 'POST':
        userDetails = request.form
        userId = userDetails['userid']
        temp1 = userId
        session['userid'] = userId;
        password = userDetails['password']
        cur = mysql.get_db().cursor()
        result = cur.execute("Select name,password from user where userid = " + userId)
        if(result == 0):
            error = 'Invalid username or password. Please try again!'
        else:
            temp = cur.fetchall()
            cur.close()
            print(temp)
            if(password == temp[0][1]):
                flash("UserID = "+userId)
                return redirect('/book')
            else:
                error = 'Invalid username or password. Please try again!'
    return render_template('index.html', error=error)
```

```
def register():
    if request.method == 'POST':
        details = request.form
        name = details['name']
        age = details['age']
        userid = details['userid']
        phonenumber = details['phonenumber']
        password = details['password']
        con = mysql.connect()
        cur = con.cursor()
        result = cur.execute("Select * from user where userid = " + userid)
        print(result)
        if result == 0:

            cur.execute("insert into user values('" + name + "'," + age + "," + userid + ",'" + phonenumber
        con.commit()
        cur.close()
```

```
result = cur.execute("Select * from user where userid = " + temp1)
temp = cur.fetchall()
print(temp)
cur.execute("Select source,destination,departuredateandtime from trains where trainno = " + trainNumber)
sd = cur.fetchall()
print(sd)
print(type(sd[0][2]))
date = sd[0][2].strftime("%Y-%m-%d %H:%M:%S")
cur.execute("select pnr from tickets")
pnr = cur.fetchall()
leng = len(pnr)
fare = random.randrange(100, 1000, 1)
print("Insert into tickets values (" + str(pnr[leng-1][0] + 1)+","+str(fare)+"," + trainNumber + "," + str(sd[0][0]) + "," + str(sd[0][1]) + ",'"+date+"'")
cur.execute("Insert into tickets values (" + str(pnr[leng-1][0] + 1)+","+str(fare)+"," + trainNumber + "," + str(sd[0][0]) + "," + str(sd[0][1]) + ",'"+date+"')")
cur.execute("INSERT INTO passengers VALUES ('" + temp[0][0]+ "','male'," + str(temp[0][1]) + ","+str(pnr[leng-1][0] + 1)+"," + str(temp[0][2])+")")
ticket = [pnr[leng-1][0] + 1, fare, trainNumber, sd[0][0],sd[0][1],date,temp[0][0],'male',temp[0][1], temp[0][2]]
con.commit()
cur.close()
return render_template('ticket.html', tickets = ticket)
```

```
def searchtrain():
    if request.method == 'POST':
        details = request.form
        trainno = details['trainno']
        con = mysql.connect()
        cur = con.cursor()
        temp2 = cur.execute("Select * from trains where trainno = " + trainno)
        trains = cur.fetchall()
        result = cur.execute("Select stationname from stations where stationcode = " + str(trains[0][4]))
        stations = cur.fetchall()
        s = stations[0][0]
        result = cur.execute("Select stationname from stations where stationcode = " + str(trains[0][5]))
        stations = cur.fetchall()
        d = stations[0][0]
        t = [trains[0][0],trains[0][1], trains[0][2].strftime("%Y-%m-%d %H:%M:%S"), s, d]
        return render_template('showtrains.html', trains = t)
    return render_template('usertrainsearc.html')
```

SQL Queries (for end-sem)

```sql
/*1 find 2nd highest ticket fare*/
select distinct fare
    from tickets t1
    where 2 = (select count(distinct fare)
    from tickets t2
    where t1.fare <= t2.fare);

/*2 max fares of trains grouped by sources*/
select source, max(fare) from tickets group by source having max(fare) > (select avg(fare) from tickets);

-- 3 free seats in a train
select trainno,totalseats ,
    (totalseats - (select count(trainno) from tickets ti where ti.trainno = t.trainno)) as free_seats
    from trains t where t.trainno = "11084";
```

Query 4-8

```sql
-- 4 total capacity of a station
select sum(totalseats) as total_capacity from trains where source = 2 or destination = 3;


-- 5 details of the passenger having highest fare
select * from passengers where pnr=(
select pnr from tickets where fare=(
select max(fare) from tickets));

-- 6 avg fare of persons having age > 18 and fare < 1000
select avg(fare)
from tickets where pnr in (
select pnr from passengers where age >18 and fare <1000);

-- 7 passengers having age > avg(age)
select *
from passengers where age >(
select avg(age) from passengers);

-- 8
create view ticketinfo as
    select t.pnr , p.name as passengers_name , u.name as user_name, u.userid
    from tickets t, passengers p , user u
    where u.userid= p.userid and p.pnr = t.pnr;
```

Query 8-12

```sql
-- 9
Select ti.pnr,ti.fare,ti.trainno,ti.departuredatetime,
    t.totalseats,t.source,t.destination
    from tickets as ti ,trains as t
    where ti.trainno = t.trainno and t.departuredateandtime
    between '6992-01-19 19:16:46' and '9276-10-20 07:53:08'
    order by pnr;/* trains available between two dates*/

-- 10
Select t.source,s.stationname,count(*) as "NoOfTrainsLeaving"
    from trains as t , stations as s where t.source = s.stationcode
    group by source;/*no of trains leaving a source station*/

-- 11
Select p.userid,sum(fare),count(p.pnr) as 'tickets booked'
    from tickets as t,passengers as p where t.pnr = p.pnr group by p.userid;/*no of tickets booked by a user and total fare*/

-- 12
Select t.pnr,p.name,p.age,t.trainno, t.departuredatetime as "Departure Date And Time"
    from tickets as t,passengers as p where t.pnr = p.pnr and userid = 11;/* */
```

Query optimization

SQL Query optimization is defined as **the iterative process of enhancing the performance of a query in terms of execution time, the number of disk accesses, and many more cost measuring criteria**

- Adding indexes : Table indexes in databases help retrieve information faster and more efficiently

```
create index pnr_index on tickets(pnr);
create index fare_index on tickets(fare);
create index user_index on user(userid);
create index trainno_index on trains(trainno);
create index pass_pnr_index on passengers(pnr);
create index station_index on stations(stationcode);
```

- Avoid using multiple OR: It evaluates each component of the OR which, in turn, may lead to poor performance. Use Union instead.

```
select sum(totalseats) as total_capacity
    from trains where source = 2
    union
    select sum(totalseats) as total_capacity
    from trains where destination = 3;
```

- Use SELECT fields instead of SELECT *: Selecting limited data increases the performance

```
-- 2
select trainno,totalseats ,
    (totalseats - (select count(trainno) from tickets ti where ti.trainno = t.trainno)) as free_seats
    from trains t where t.trainno = "11084";
```

- **LIMIT command:** The limit command is used to control the number of rows to be displayed from the result set.

```
-- 4
select *
from passengers where age >18 limit 10;
```

- Use wildcard wisely: Having the wildcard (%) at the end of the string when searching on uncertain characters is not as challenging as having the wildcard at the beginning of the search string.

```
-- 3
select passengers.pnr, passengers.name from passengers where passengers.name like 'J%';
```

Triggers

- A SQL trigger is a database object which fires when an event occurs in a database

```
delimiter $$
DROP TRIGGER IF EXISTS t_passenger_delete;
GO
CREATE TRIGGER t_passenger_delete ON passenger INSTEAD OF DELETE
AS BEGIN
    DECLARE @id INT;
    DECLARE @count INT;
    SELECT @id = id FROM DELETED;
    SELECT @count = COUNT(*) FROM tickets WHERE pnr = @id;
    IF @count = 0
        DELETE FROM passengers WHERE id = @id;
    ELSE
        THROW 51000, 'can not delete - passenger is referenced in other tables', 1;
END $$
delimiter ;
```

```
delimiter $$
create trigger for_tickets1 before insert on tickets
for each row
begin
 if new.source = new.destination then
 signal sqlstate '45000' set message_text = 'Source and destination should be different.';
 end if;
end$$
delimiter ;
```

```
delimiter $$
create trigger for_passenger1 before insert on passengers
for each row
begin
 if new.name = '' then
 signal sqlstate '45000' set message_text = 'Name should not be null';
 end if;
end$$
delimiter ;
```

```
delimiter $$
CREATE TRIGGER before_update_dare
BEFORE UPDATE ON tickets
FOR EACH ROW
BEGIN
    IF NEW.fare <> OLD.fare THEN
     INSERT INTO fare_changes(pnr,old_fare,new_fare)
         VALUES(NEW.pnr,OLD.fare,NEW.fare);
     END IF;
END$$
delimiter ;
```

```
delimiter //
create trigger age_verify
before insert on passengers
for each row
if new.age<0 then set new.age=20;
end if; //
```

```
delimiter //
create trigger check_null_depdate
after insert
on tickets
for each row
begin
if new.departuredatetime is null then
insert into message(messageID,message)
values (new.pnr, concat('hi',new.pnr,',please update your departure date and time'));
end if ;
end //
delimiter ;
```

Indexing

An index on an attribute of a relation is a data structure that allows the database system to find those tuples in the relation that have a specified value for that attribute efficiently, without scanning through all the tuples of the relation.

```
create index pnr_index on tickets(pnr);
create index fare_index on tickets(fare);
create index user_index on user(userid);
create index trainno_index on trains(trainno);
create index pass_pnr_index on passengers(pnr);
create index station_index on stations(stationcode);
```

Website UI

Home

Indian Railway Catering and Tourism Corporation
[ITRTC]

IRCTC

Registration Page

SEARCH TRAINS | Home Login

Name:

UserID:

Age:

Phone No:

Password:

SUBMIT

Home

Indian Railway Catering and Tourism Corporation
[ITRTC]

IRCTC

Hey,User! Welcome to our new ITRTC Website

Book trains

TrainNumber:

Book Now

Home

IRCTC

TICKET BOOKED

PNR	FARE	TRAIN NUMBER	SOURCE	DESTINATION	DEPARTURE DATE AND TIME	Name	GENDER	AGE	USER ID
49431	348	20991	7	8	9276-10-20 07:53:08	douglas	male	44	30

Home Search trains

IRCTC

Indian Railway Catering and Tourism Corporation

UserID:

Password:

LOGIN

Register

IRCTC

Hey,User! Welcome to our new ITRTC Website

Search Trains

TrainNumber:

SEARCH TRAIN

Home

IRCTC

TRAIN NO	TRAIN NAME	DEPARTURE DATE AND TIME	SOURCE	DESTINATION
20991	II7PZjMF4o	9276-10-20 07:53:08	Lincoln	Rochester

Link to DRIVE (Contains all the code)

https://drive.google.com/drive/folders/1osAiKpk6x4FJX068MKg8hX_EztaqIndb?usp=sharing