DBMS PROJECT REPORT – RailBot {THEME – IRCTC}

MENTOR – ASHMEET SINGH

ABOUT THE MEMBERS:

Name	Roll No	Year and Program
Abhishek Goyal	2019136	ECE, 3rd Year
Abhipriya Kumar	2019134	ECE, 3rd Year
Rahul Jaiswal	2020109	CSE, 2nd Year
Sahil Kumar Singh	2020115	CSE, 2nd Year

INDIVIDUAL CONTRIBUTION

Abhipriya Kumar – Python Command Line Scripting/Triggers/Embedded Queries/ Data Population

Abhishek Goyal – Documentation/Data Population/Relational Schema/ER Diagram/Query optimisation

Sahil Kumar Singh – SQL queries/Data Population

Rahul Jaiswal – Views and Grants/Indexing/SQL queries

PROBLEM STATEMENT AND SCOPE

IRCTC stands for Indian Railways Catering and Tourism Corporation. IRCTC comes under the Railways Ministry of the Central Government. Railways came into existence in India in early 19th Century and since then, has been the preferred mode of transportation for the common man. The Railways provide services across the length and breadth of the country and across various terrains. There are close to 23000 trains under the Indian Railways with an average of 24 million passengers using them on a daily basis.

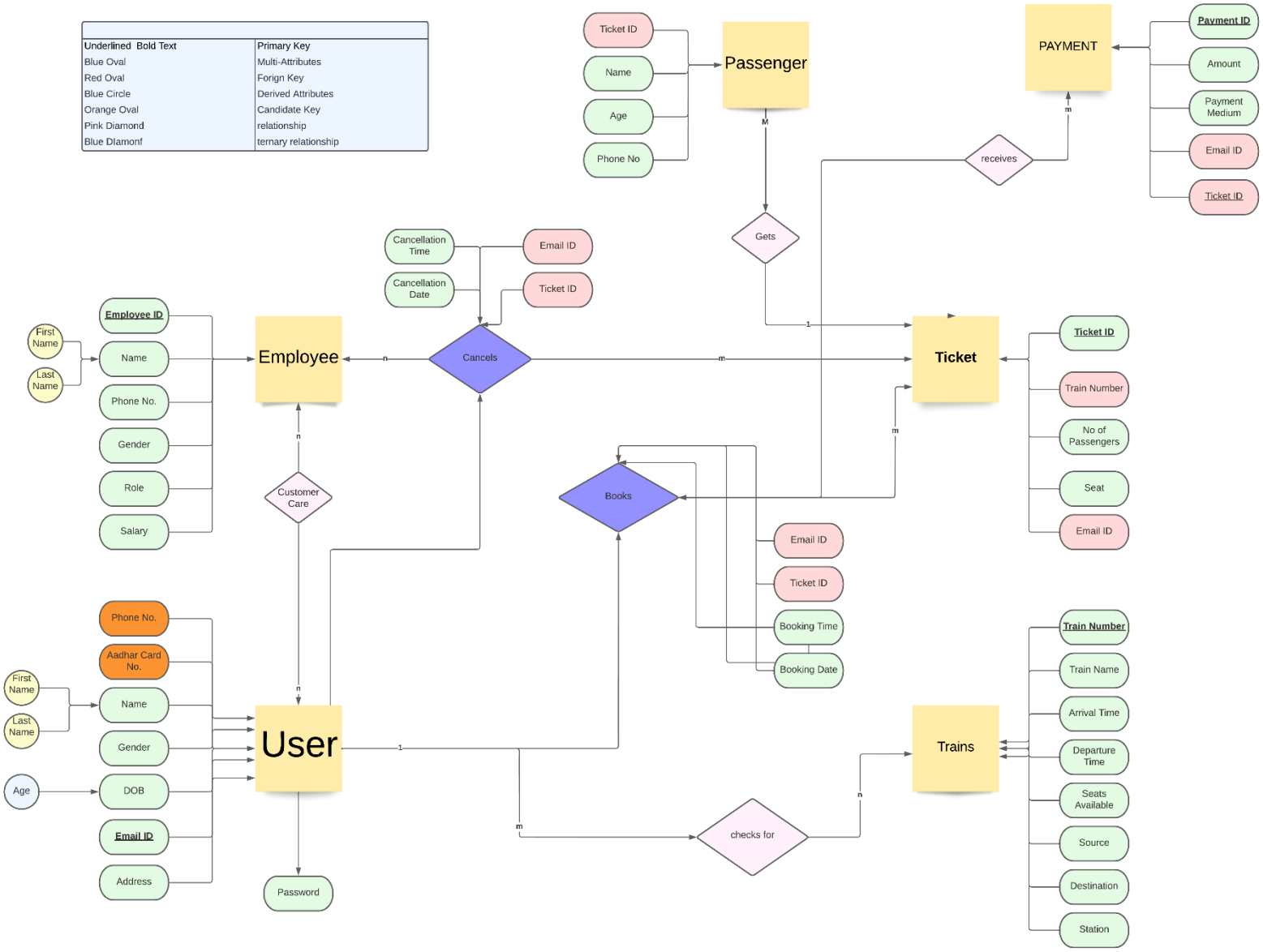
Currently, passengers can buy tickets at railway stations, and various online platforms like MakeMyTrip, Goibobo, and the official site of the IRCTC. This project can be construed as a miniature version of the IRCTC which serves as a medium between the users/passengers and Indians Railways, allowing members of the public to book tickets on an online platform. Since the onset of COVID in March 2020, there has been conscious efforts made by the Indian government to shift most operations from manual to digital/automated forms. This project is a small step in that very direction.

SCOPE: RailBot is a **Python script program** which takes prompt from the **command line** to manipulate the **DBMS** which allows the **user** to search trains depending on their preferred destinations, train schedules, seat availability, book and cancel tickets, avail services of the **customer care employees** at Indian Railways. Furthermore, the users can pay for the tickets through net banking, banking cards, UPI apps, and online wallets.

At the moment, the **stakeholders** at RailBot are given below:

1) RailBot Admins
2) Indian Railways and its employees
3) Members of the Public

ER DIAGRAM



Relational Schema

Entities:

1. User (First Name, Last Name, **Email ID**, Phone No, Gender, Address, Password, Aadhar Number, DOB)
   A. Email ID is primary key
   B. Phone no and Aadhar Number are candidate key
2. Employee (**Employee ID**, First Name, Last Name, Phone No, Role, Gender, Salary)
   A. Employee ID is primary key
   B. Role is a int varying from 1-4, from top most managerial post to customer care
3. Train (Train Name, Train Number, Arrival Time, Departure Time, Seats Available, Source, Destination. Station)
   A. Train Number is primary key
   B. Seats available is Yes/No
4. Ticket (**Ticket ID**, Train Number, Seats, No of Passengers, Email ID)
   A. Ticket ID is primary key

   B. Train Number and Email ID are foreign keys from Train and User entities respectively
5. Payment (**Payment ID**, Payment Medium, Amount, Email ID, Ticket ID)
   A. Payment ID is primary key
   B. Email ID and ticket ID are foreign keys from User and Ticket Entities respectively
6. Passenger (First Name, Last Name Phone Number, Age, Ticket_id)
   A. Passenger is a weak entity, ie no self primary key
   B. Ticket ID is foreign key from Ticket entity

Relationships

1. Cancel**:** (Cancellation Time, Cancellation Date, Ticket ID, Email ID)
   A. ternary relationship for User, Employee, and Ticket.
2. Book**:** (Booking Time, Booking Date, Ticket ID, Email ID)
   A. ternary relationship for User, Employee, and Ticket.
3. Unary Relationships:
   A. Checks – between User and Trains
   B. Customer Care – Between user and employee
   C. Gets – Between Passenger and Ticket
   D. Receives – Payment and Ticket

ASSUMPTIONS

1. Each ticket cost 300 rupees

SQL QUERIES

- Query to find all trains that go to Chennai and has seats available in them

```
SELECT *
FROM train
WHERE destinaition_station = 'Chennai' and seats_avail='YES';
```

- Query to find average age of passenger between 20 to 50

```
SELECT AVG(age)
AS Average_Age_passenger
FROM passenger
where 20<age<50;
```

QUERY OPTIMISATION: instead of storing age of every passenger in the given age group, and then summing them, we are directly doing the operation at once

- Query to find the gender ratio of employee

```
SELECT
   COUNT(IF(gender = 'Male', 1, NULL)) count_male,
   COUNT(IF(gender = 'Female', 1, NULL)) count_female,
   COUNT(IF(gender = 'Male', 1, NULL))/COUNT(IF(gender = 'Female', 1, NULL)) as ratio
FROM     employee;
```

- Query to find the difference in arrival and departure time

```
SELECT train_name,arrival_time,departure_time,
(departure_time - arrival_time)/100
As difference_in_time_minute
from train
```

- Query to update the phone number of a user

```
UPDATE userdata
SET phone_num ='111-222-333'
WHERE email_id ='acarslake1d@i2i.jp';
```

Query optimisation: directly phone number of the user in the userdata spreadsheet

- Query to find people who paid 600

```
select *
from passenger
where ticket_id in (
select ticket_id
from payment
where amount=600);
```

query optimisation: directly linking the ticket entity to the payment entity as there can be multiple passengers on a ticket but there is always only one person who buys the ticket, hence saving computation time

- Query to find the difference in book date and cancel date using inbuilt function

```
select b.email_id , b.ticket_id , DATEDIFF(c.cancellation_date , b.booking_date)
as days_gap from cancel c , book b
where b.email_id = c.email_id and b.ticket_id = c.ticket_id
```

- Query to Merge the passenger and ticket table

```
select p.first_name, p.last_name , t.ticket_id ,  t.email_id from passenger p , ticket t where p.ticket_id = t.ticket_id;
```

- Query to find the relation between gender

```
create view employee_male as
select count(gender) as total_male from employee where gender = 'Male';
select * from employee_male;
create view employee_female as
select count(gender) as total_female from employee where gender = 'Female';
select * from employee_female;
select m.total_male/f.total_female from employee_male m, employee_female f;
```

- Query to find whose and how many ticket are booked under a specific email id

```
select * from (passenger p natural join ticket t) where email_id = 'dcobby0@amazon.co.uk';
```

- Query to find seats of all the passengers

```
elect first_name, last_name,seat from
ticket ti inner join passenger p
on ti.ticket_id=p.ticket_id;
```

- Query to find names and aadhar number of all the passengers of all trains starting from delhi

```
select first_name, last_name, aadhar_num, train_name
from userdata u
inner join ticket ti on u.email_id=ti.email_id
inner join train tr
on ti.train_num=tr.train_num
where tr.source_station='DELHI';
```

- Query to find names,gender,amount paid and seat of the passengers

```
select first_name, last_name,gender,amount,seat
from userdata u natural join (payment p natural join ticket t);
```

INDEXING

- create index idx_trainId_trainNum

  on ticket(ticket_id,train_num);

- create index idx_trainSearch

  on train(arrival_time,train_date,seats_avail);

- create index idx_userdata_list

  on userdata(first_name,last_name,aadhar_num);

- create index idx_employee_list

  on employee(first_name,last_name,roles);

- create index idx_passenger_list

  on passenger(first_name,last_name);

VIEWS AND GRANTS

```
create view passengers_train_num
as
select first_name, last_name,train_num from
```

```
ticket ti ,passenger p
where ti.ticket_id=p.ticket_id;

select * from passengers_train_num;

create view passengers_list
as
select first_name, last_name, aadhar_num
from userdata u,ticket ti, train tr
where ti.train_num=tr.train_num and
u.email_id=ti.email_id;

select * from passengers_list;

CREATE USER 'test_employee'@'localhost' IDENTIFIED BY 'testpassword';
GRANT SELECT, INSERT, DELETE, UPDATE ON cancel TO 'test_employee'@'localhost';
SHOW GRANTS FOR 'test_employee'@'localhost';


CREATE USER 'test_user'@'localhost' IDENTIFIED BY 'testpassword';
GRANT SELECT ON employee TO 'test_user'@'localhost';
SHOW GRANTS FOR 'test_user'@'localhost';
```

TRIGGERS

- Trigger to Assign Employees' gender during data input/sign up

```
delimiter //
create trigger tr before insert on employee for each row
begin
if new.gender = 'M' then set new.gender = 'Male';
end if;
if new.gender = 'F' then set new.gender = 'Female';
end if;
end //
```

- Trigger to make the payment amount 0 if a negative amount is entered as input

create **trigger** p1 **before insert** on payment for each row

delimiter //

begin

if new.amount < 0 then set new.amount = 0;

end if;

end //

- Trigger to assign value to booking date under book relationship

create **trigger** b1 **before insert** on book for each row

set new.booking_date = curdate();

- Trigger to assign value to booking time under book relationship

create **trigger** b2 **before insert** on book for each row

set new.booking_time = curtime();

- Trigger to assign value to cancellation date under book relationship

create **trigger** c1 **before insert** on cancel for each row

set new.cancellation_date = curdate();

- Trigger to assign value to cancellation time under book relationship

create **trigger** c2 **before insert** on cancel for each row

set new.cancellation_time = curtime();

- Trigger to assign value to salary table for every new employee entry
  create trigger e1 before insert on employee for each row
  set new.salary = new.roles*10000;
- Trigger to update employee's salary table upon promotion/demotion
  create trigger e2 before update on employee for each row
  set new.salary = new.roles*10000;

EMBEDDED QUERIES

```
from pandasql import sqldf
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="Abhi@12345",
  database = "irctc"
)

mycursor = mydb.cursor()
mycursor.execute("select * from (passenger p natural join ticket t) where email_id = 'dcobby0@amazon.co.uk';")
for x in mycursor:
  print(x)

print()
mycursor.execute("select u.email_id from userdata u where u.email_id in (select email_id from ticket);")

for x in mycursor:
  print(x)
print()

mycursor.execute("select m.total_male/f.total_female from employee_male m, employee_female f;")

for x in mycursor:
  print(x)
print()

mycursor.execute("""SELECT train_name,arrival_time,departure_time,
(departure_time - arrival_time)/100
As difference_in_time_minute
from train;""")

for x in mycursor:
  print(x)
print()

mycursor.execute("""select * from (passenger p natural join ticket t) where email_id = 'dcobby0@amazon.co.uk';""")

for x in mycursor:
  print(x)
print()
```

```
(1, 'Megumi', 'Sharma', 28, '206-527-3085', 1, 2, '3 SL & sno. 22', 'dcobby0@amazon.co.uk')
(16, 'asd', 'asdfg', 21, '213-675-7864', 21, 1, ' 1 AC & sno. 25', 'dcobby0@amazon.co.uk')

('acastaignet2@digg.com',)
('dcobby0@amazon.co.uk',)
('dcullivang@springer.com',)
('easpin6@bing.com',)
('iarkcollm@answers.com',)
('ksaintepauls@edublogs.org',)
('neburneq@imageshack.us',)
('nreaveleya@abc.net.au',)
('pbrockman4@princeton.edu',)
('tblakero@telegraph.co.uk',)

(Decimal('1.3077'),)

('A', datetime.timedelta(seconds=61200), datetime.timedelta(seconds=62400), Decimal('20.0000'))
('B', datetime.timedelta(seconds=66600), datetime.timedelta(seconds=67200), Decimal('10.0000'))
('C', datetime.timedelta(seconds=46800), datetime.timedelta(seconds=48000), Decimal('20.0000'))
('D', datetime.timedelta(seconds=39600), datetime.timedelta(seconds=40200), Decimal('10.0000'))
('E', datetime.timedelta(seconds=50400), datetime.timedelta(seconds=51600), Decimal('20.0000'))
('F', datetime.timedelta(seconds=54000), datetime.timedelta(seconds=54600), Decimal('10.0000'))
('G', datetime.timedelta(seconds=64800), datetime.timedelta(seconds=66000), Decimal('20.0000'))
('H', datetime.timedelta(seconds=61200), datetime.timedelta(seconds=62400), Decimal('20.0000'))
('I', datetime.timedelta(seconds=66600), datetime.timedelta(seconds=67200), Decimal('10.0000'))
('J', datetime.timedelta(seconds=46800), datetime.timedelta(seconds=48000), Decimal('20.0000'))
('K', datetime.timedelta(seconds=39600), datetime.timedelta(seconds=40200), Decimal('10.0000'))
('L', datetime.timedelta(seconds=50400), datetime.timedelta(seconds=51600), Decimal('20.0000'))
('M', datetime.timedelta(seconds=54000), datetime.timedelta(seconds=54600), Decimal('10.0000'))
('N', datetime.timedelta(seconds=64800), datetime.timedelta(seconds=66000), Decimal('20.0000'))
('O', datetime.timedelta(seconds=61200), datetime.timedelta(seconds=62400), Decimal('20.0000'))
('P', datetime.timedelta(seconds=66600), datetime.timedelta(seconds=67200), Decimal('10.0000'))
('Q', datetime.timedelta(seconds=46800), datetime.timedelta(seconds=48000), Decimal('20.0000'))
('R', datetime.timedelta(seconds=39600), datetime.timedelta(seconds=40200), Decimal('10.0000'))
('S', datetime.timedelta(seconds=50400), datetime.timedelta(seconds=51600), Decimal('20.0000'))
('T', datetime.timedelta(seconds=54000), datetime.timedelta(seconds=54600), Decimal('10.0000'))
('U', datetime.timedelta(seconds=64800), datetime.timedelta(seconds=65400), Decimal('10.0000'))
('V', datetime.timedelta(seconds=66600), datetime.timedelta(seconds=67200), Decimal('10.0000'))
('W', datetime.timedelta(seconds=46800), datetime.timedelta(seconds=48000), Decimal('20.0000'))
('X', datetime.timedelta(seconds=39600), datetime.timedelta(seconds=40200), Decimal('10.0000'))
('Y', datetime.timedelta(seconds=50400), datetime.timedelta(seconds=51600), Decimal('20.0000'))

(1, 'Abhishekh', 'Goyal', 34, '688-403-1732', 1, 2, '3 SL & sno. 22', 'dcobby0@amazon.co.uk')
(1, 'Megumi', 'Sharma', 28, '206-527-3085', 1, 2, '3 SL & sno. 22', 'dcobby0@amazon.co.uk')
(16, 'asd', 'asdfg', 21, '213-675-7864', 21, 1, ' 1 AC & sno. 25', 'dcobby0@amazon.co.uk')
```

FUTURE SCOPE

Indian railways run various operations many of which affects millions of people. A more detailed and widely implemented bot will provide space to other stakeholders associated with Indian Railways like vendors at Railway Stations, the Railways Ministry, private entities owning operational trains like Megha Engineering and Infrastructures Limited, Sainath sales and Services Pvt Ltd, IRB Infrastructure Developers Limited etc.

THANK YOU