IRCTC
DBMS PROJECT
Railway Management System

Group Members:
Ishan Pandey
Prakrit Garg
Sushil kumar Giri
Vishesh Rangwani

SCOPE

Scope OF Project

Railway reservation is a service available to passengers to book tickets to travel by train. It involves details such as the availability of seats in a particular train, cancellation of tickets if any, and booking of tickets.

The purpose of a case study is to design and develop a database to maintain the records of different trains, train status, and users. The train status includes its number, description, time, destination, source, and cost. Users can book their tickets for the train in which seats are available. To accomplish this, users must provide the desired train number and the date for which tickets are to be booked.

Before you can book a ticket for a user, you must check the validity of the train number. Once you’re sure the train is valid, you can check to see whether the seat is available. If the seat is available,then you can book a ticket with a status of confirmed. You will then generate a ticket ID and store it with the other information.

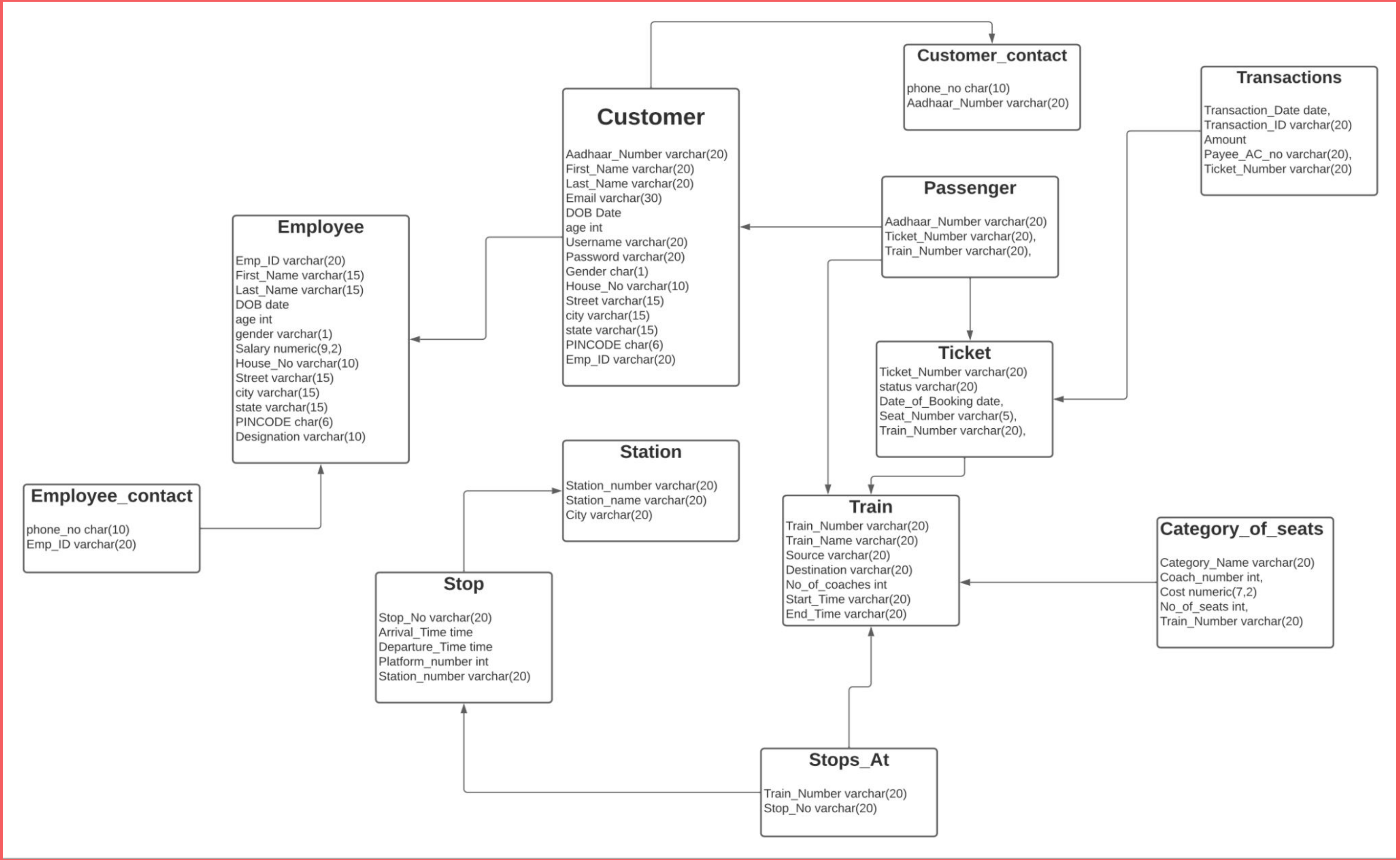
Stakeholders

- Passengers: Users or Passengers can book/modify/cancel tickets. They are the primary users of the system. They can book a ticket with their required seat type, date, destination etc.

- Station Masters: Can use this system to maintain and plan train timings at stations. They can see seat availability and arrival/departure times for trains.

- Railway Administration: Can use this system to gauge the sales of tickets, the popularity of routes and Seats used.

Relational Schema


Customer
Aadhaar_Number varchar(20)
First_Name varchar(20)
Last_Name varchar(20)
Email varchar(30)
DOB Date
age int
Username varchar(20)
Password varchar(20)
Gender char(1)
House_No varchar(10)
Street varchar(15)
city varchar(15)
state varchar(15)
PINCODE char(6)
Emp_ID varchar(20)
Customer_contact
phone_no char(10)
Aadhaar_Number varchar(20)
Transactions
Transaction_Date date,
Transaction_ID varchar(20)
Amount
Payee_AC_no varchar(20),
Ticket_Number varchar(20)
Passenger
Aadhaar_Number varchar(20)
Ticket_Number varchar(20),
Train_Number varchar(20),
Employee
Emp_ID varchar(20)
First_Name varchar(15)
Last_Name varchar(15)
DOB date
age int
gender varchar(1)
Salary numeric(9,2)
House_No varchar(10)
Street varchar(15)
city varchar(15)
state varchar(15)
PINCODE char(6)
Designation varchar(10)
Ticket
Ticket_Number varchar(20)
status varchar(20)
Date_of_Booking date,
Seat_Number varchar(5),
Train_Number varchar(20),
Station
Station_number varchar(20)
Station_name varchar(20)
City varchar(20)
Employee_contact
phone_no char(10)
Emp_ID varchar(20)
Train
Train_Number varchar(20)
Train_Name varchar(20)
Source varchar(20)
Destination varchar(20)
No_of_coaches int
Start_Time varchar(20)
End_Time varchar(20)
Category_of_seats
Category_Name varchar(20)
Coach_number int,
Cost numeric(7,2)
No_of_seats int,
Train_Number varchar(20)
Stop
Stop_No varchar(20)
Arrival_Time time
Departure_Time time
Platform_number int
Station_number varchar(20)
Stops_At
Train_Number varchar(20)
Stop_No varchar(20)

RELATIONAL SCHEMA

Customer(
Aadhaar_Number ,First_Name ,Last_Name, Email,DOB,age,Username,Password,Gender , House_No , Street , city , state , PINCODE ,Emp_ID);

Customer_contact (phone_no,   Aadhaar_Number);

Passenger(Aadhaar_Number ,Ticket_Number ,Train_Number);

Ticket(Ticket_Number ,status,Date_of_Booking,Seat_Number,Train_Number);

Transactions (Transaction_Date,Transaction_ID,Amount,Payee_AC_no,Ticket_Number);

Train (Train_Number,Train_Name ,Source ,Destination,No_of_coaches ,Start_Time ,End_Time);

Category_of_seats (Category_Name,Coach_number,Cost,No_of_seats ,Train_Number);

Stops_At (Train_Number,Stop_No);

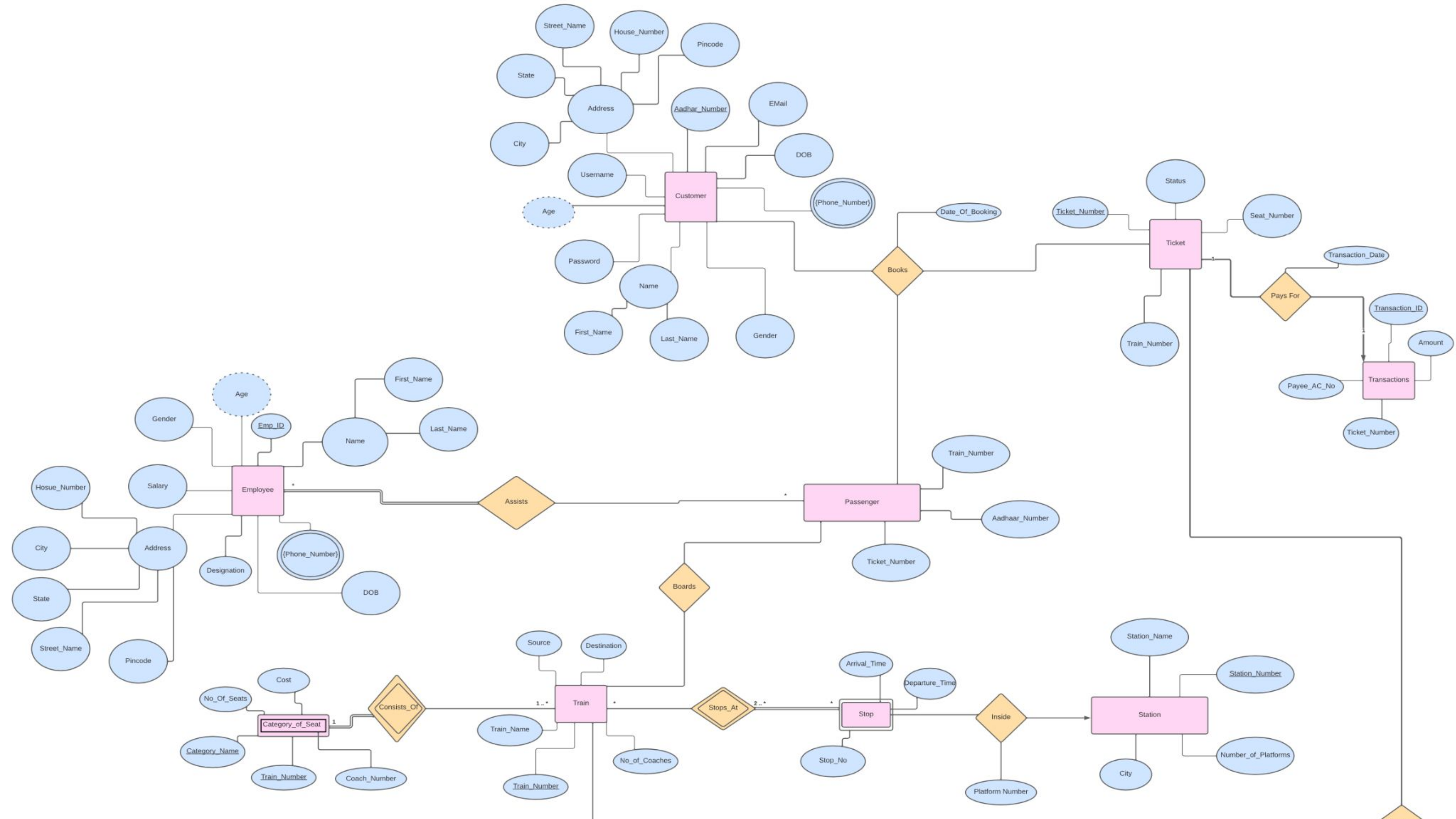
Stop (Stop_No,Arrival_Time,Departure_Time,Platform_number,Station_number);

Station (Station_number ,Station_name,City,Number_of_Platforms);

Employee (Emp_ID ,;0First_Name ,Last_Name ,DOB ,age ,gender ,Salary,House_No, Street , city , state, PINCODE,Designation);

Employee_contact (phone_no ,Emp_ID);

E-R Diagram


Street_Name
House_Number
Pincode
State
Address
City
Aadhar_Number
EMail
DOB
Username
Customer
Age
{Phone_Number}
Password
Name
First_Name
Last_Name
Gender
Date_Of_Booking
Books
Ticket_Number
Status
Ticket
Seat_Number
Train_Number
1
Transaction_Date
Pays For
Transaction_ID
Amount
Payee_AC_No
Transactions
Ticket_Number
Age
Gender
Emp_ID
First_Name
Name
Last_Name
Hosue_Number
Salary
Employee
*
Assists
*
Passenger
Train_Number
Aadhaar_Number
Ticket_Number
City
Address
{Phone_Number}
Designation
DOB
State
Street_Name
Pincode
Boards
Source
Destination
Arrival_Time
Departure_Time
Station_Name
Station_Number
Cost
No_Of_Seats
Category_of_Seat
1
Consists_Of
1 .. *
Train
*
Stops_At
2 .. *
*
Stop
Inside
Station
Train_Name
No_of_Coaches
Stop_No
Number_of_Platforms
Category_Name
Train_Number
Coach_Number
Train_Number
Platform Number
City

Functionalities

The Railway Database provides User a Wide variety of functionalities which are as follows:

1)To register Customer as Passenger

2)To book a ticket

3)To print ticket details

4)To login as an employee and analyze and control the backend services

5)To cancel passenger’s ticket

6)Payment Portal

7)Passenger and Employee Portals with vast Functionalities

8) Check schedules

VIEWS

1)CREATE VIEW Customer_Contact_Details AS SELECT cc.Aadhaar_Number, cc.Phone_no, c.Email FROM Customer c, Customer_contact cc WHERE c.Aadhaar_Number=cc.Aadhaar_Number;

2)CREATE VIEW train_schedule AS SELECT Train_Name, Source, Destination, Start_Time, End_Time FROM Train;

3)CREATE VIEW Employee_Comm_Details AS SELECT ec.Emp_ID, ec.Phone_no, e.House_No, e.Street, e.city, e.state, e.PINCODE FROM Employee e, Employee_contact ec WHERE ec.Emp_ID = e.Emp_ID;

These below are the queries for the implementation of the above created views:

1)sql = "SELECT Phone_no, Email FROM Customer_Contact_Details WHERE Aadhaar_Number = %s"

2)sql = "SELECT * FROM train_schedule WHERE lower(Source) = %s and LOWER(Destination) = %s"

GRANTS

GRANT SELECT, INSERT, DELETE, UPDATE ON Employee TO 'root'@localhost :
This grant provides functionalities including fetching of data, deletion of data, insertion of data and updation of data b y the administrator of database.Here ‘root’ signifies the administrator of the main database.

GRANT SELECT, INSERT, DELETE ON Ticket TO 'Customer1'@localhost:
This grant provides access to data of ticket to the passenger for booking of new ticket,deletion of an existing ticket and fetching data of booked ticket here ‘Customer1’@localhost is the general name for the passenger.

GRANT SELECT, INSERT, DELETE, UPDATE ON Employee TO 'Admin1'@localhost:
This grant provides access of data of employees to the login employee and gives control of fetching the data,deletion of record,insertion of new record and updation of previous record.Here ‘Admin1’@localhost depict the general acronym for Employee.

SQL QUERIES

Embedded Queries

```
insert into Ticket values (ticketNo, status, today, seatNo, selected_train)

SELECT Aadhaar_Number FROM Customer WHERE Username = username

SELECT Train_Number, s FROM(SELECT Train_Number, SUM(No_of_seats) as s FROM
Category_of_seats GROUP BY Train_Number) as T WHERE s = (SELECT MIN(s) FROM (SELECT
Train_Number, SUM(No_of_seats) s FROM Category_of_seats GROUP BY Train_Number) as T1)


SELECT * FROM Train WHERE LOWER(Source) = input_source.lower()  and LOWER(Destination) =
input_destination.lower()

UPDATE Category_of_seats SET No_of_seats = noOfAvailableSeats-1 WHERE Train_Number =
selected_train and Coach_number = coachNumber

 SELECT COUNT(*) FROM Employee

DELETE FROM Employee_contact WHERE Emp_ID =empid
```

General Queries

1)"SELECT Category_Name FROM Category_of_seats C, Transactions T WHERE T.Ticket_number = %s and C.Train_Number = %s and C.Cost = T.Amount"

2)SELECT First_Name, Last_Name FROM Customer C, Passenger P WHERE P.Aadhaar_Number = C.Aadhaar_Number and P.Ticket_number = %s"

3)DELETE FROM Category_of_seats WHERE Train_Number = %s"

4)SELECT Aadhaar_Number FROM Customer WHERE Username = %s

5)SELECT * FROM Category_of_seats WHERE Train_Number = %s and Coach_number = %s

6)UPDATE Employee SET Designation = %s WHERE Emp_ID = %s

7)UPDATE Train SET Start_Time = %s, End_Time = %s WHERE Train_Number = %s

8)SELECT avg(Amount) FROM Transactions

10)SELECT Category_Name, k FROM(SELECT Category_Name, SUM(No_of_seats) k FROM Category_of_seats GROUP BY Category_Name) as T1 WHERE k = (SELECT MAX(s) FROM (SELECT Category_Name, SUM(No_of_seats) s FROM Category_of_seats GROUP BY Category_Name) AS T)

11) SELECT COUNT(*) FROM Employee"

12)SELECT e.First_Name, e.Last_Name FROM Employee e, (SELECT ecc.Emp_ID From Employee_contact ecc GROUP BY ecc.Emp_ID having Count(ecc.phone_no)>=2) ec WHERE e.Emp_ID==ec.Emp_ID

Query Optimization

1) With the use of indexing, the faster access in made possible with the help of following in this query:

SELECT * FROM Customer WHERE Username = %s and Password = %s
Indexing used:CREATE INDEX username_password ON Customer (Username, Password)

2)Instead of fetching all the records for Employee, fetching only necessary data fastens processing of query:

Inefficient:

Select * from Employee

Efficient:

Select empid,first_name from Employee

3)Inefficient:

SELECT Category_Name FROM Category_of_seats C, Transactions T WHERE T.Ticket_number = %s and C.Train_Number = %s and C.Cost = T.Amount

Efficient:

```
SELECT Category_Name FROM
(SELECT * FROM Category_of_seats WHERE Train_Number=%s)std,
(SELECT * FROM Transactions WHERE Ticket_Number=%s)std
WHERE std.Category_of_seats=std.Transactions
```

4) Inefficient:

```
SELECT First_Name, Last_Name FROM Customer C, Passenger P WHERE P.Aadhaar_Number = C.Aadhaar_Number
```

Efficient:

```
SELECT First_name,Last_Name FROM Customer C INNER JOIN Passenger P ON
P.Aadhaar_Number=C.Aadhaar_Number
```

5) **With usage of LIMIT ,we can limit the records being displayed**

Inefficient:

```
SELECT Emp_ID, Designation FROM Employee e WHERE e.Salary=100000
```

Efficient:

```
SELECT Emp_ID, Designation FROM Employee e WHERE e.Salary=100000 LIMIT 1000
```

INDEXING

CREATE INDEX username_password ON Customer (Username, Password):
This index is created for accessing the customer login credentials (username and password) frequently.

CREATE UNIQUE INDEX ticket_number_index ON Ticket(Ticket_Number):
This index is created for searches based on the ticket number for accessing of data.

CREATE INDEX train_src_dest ON Train(Source, Destination):
This index is created for frequent access of source and destinations of trains in booking of ticket and its scheduling.

CREATE UNIQUE INDEX train_number_index ON Train(Train_Number):
This index is created on a foreign key train number for scheduling and accessing of train records.

CREATE UNIQUE INDEX Emp_ID_index ON Employee(Emp_ID):
This index is created for the searches based on the employee id of employee records frequently.

TRIGGERS

These triggers are created for the automatic updation of certain database fields whenever insertion , deletion and updation of data is done.

There are also testing mechanism for these triggers that can be seen below for its direct implementations.

```
-- TRIGGER 1

CREATE TRIGGER calc_age after INSERT on Customer_contact for each row update CUSTOMER set
CUSTOMER.age = YEAR(CURDATE()) - YEAR(Customer.DOB);
```

This trigger is for storing the age of the Customer in the database as a derived attribute.

```
-- TRIGGER 2

CREATE TRIGGER calc_age_emp after INSERT on Employee_contact for each row update Employee set
Employee.age = YEAR(CURDATE()) - YEAR(Employee.DOB);
```

This trigger is for storing the age of the Employee in the database as a derived attribute.

This trigger helps in checking if the phone number of Customer is valid if more than 10 digits are present then the phione number is replaced with ‘ XXXXXXXXXX’

```
-- TRIGGER 3

delimiter #

create trigger check_phn_no before insert on Customer_contact

for each row

BEGIN

IF(length(NEW.Phone_no) <> 10) THEN

   SET NEW.Phone_no = "XXXXXXXXXX";

END IF;

END#

delimiter ;

-- Query to check direct implementation of  Trigger 3

INSERT into Customer_contact values('123123123', '9999991999');
```

This trigger helps in checking if the phone number of Employee is valid if more than 10 digits are present then the phione number is replaced with ‘ XXXXXXXXXX’

```
-- Trigger 4

delimiter #

create trigger check_phn_no_employee before insert on Employee_contact

for each row

BEGIN

IF(length(NEW.phone_no) <> 10) THEN

   SET NEW.phone_no = "XXXXXXXXXX";

END IF;

END#

delimiter ;

-- Query to check direct implementation of  Trigger 4

INSERT into Employee_contact(Emp_ID, phone_no) values('99998888', '99');
```

This trigger is for checking the validity of pincode , if not valid then replaced in the database with “XXXXXX”

```
-- Trigger 5

delimiter #

create trigger check_PINCODE before insert on Customer

for each row

BEGIN

IF(length(NEW.PINCODE) <> 6) THEN

   SET NEW.PINCODE = "XXXXXX";

END IF;

END#

delimiter ;
```

Contribution

VISHESH RANGWANI:

- Entire Command Line interface connected to workbench using python with embedded SQL queries
- SQL Queries with aggregation functions
- Indexes
- Triggers
- Views
- Grants
- Created the flow of the application
- Created tables with test data in MySQL WorkBench

ISHAAN PANDEY

- GUI

PRAKRIT GARG:

- Initialized the GUI for the application
- Created GUI interface to workbench using python with embedded SQL queries
- Created the main menu GUI using tkinter and implemented respective queries
- Created the customer registration GUI using tkinter, stored data and performed operations on the data using respective queries
- Created the ticket GUI using tkinter, stored data and performed operations on the data using respective SQL queries
- Controlled the flow for the above 3 parts

SUSHIL KUMAR GIRI:

- Query Optimization
- Project documentation -compilation of all things into ppt
- GUI

Limitations And Assumptions

- No Modification of Trains’ route allowed
- Only 3 categories exist: AC and General
- Single Transaction can only process one ticket at a time
- The Time between stoppage stations and booking is not considered
- Only a single source and destination station is allowed per booking
- Simple UI