



# **PROJECT REPORT**

## **Backend Development**

**Of**

## **Railway-Reservation-System**

15.07.2022

**Submitted To-**

Dr. Viswanath Gunturi

**Submitted By-**

AMAN KUMAR

2020CEB1005

## Table Details

Tables	Attributes
admin	username, password
user_	username, name, email, address, password
train	train_no, date, ac_num, sleeper_num,
train_released	train_no, date, source, destination, ac_available, sleeper_available
ticket	pnr_no, train_no, date, coach, username, source, destination
passenger	pnr_no, name, age, gender, berth_no, berth_type, coach_no
seating_plan (additional)	coach_no, berth_no, berth_type, name, pnr_no, source, destination

*\*Bold attributes/their combinations denote Primary Keys*

- ◆ An additional table “seating\_plan” is created that is basically a helping table for the functions to store and print the output.

- I. **admin** table stores the username & password of admins.
- II. **user\_** table stores the basic information about all the users registered on the portal.
- III. **train** table has the fixed details of a train, including train number, date of journey, number of AC & Sleeper seats.
- IV. **train\_released** table stores the total number of seats released by admin on a day in each train for different source and destination in AC & Sleeper Coaches.

- v. ***ticket*** table stores the details of a ticket, namely PNR number, coach(AC/Sleeper) for which ticket is booked, train number, train date, name of the user who booked the ticket as well as source and destination.
- vi. ***passenger*** table stores the information related to all passengers, their basic info, including name, age & gender, and details of their booking, including their berth number, coach number, and berth type.

## Primary Keys

- I. admin  
PRIMARY KEY(username)
- II. user\_  
PRIMARY KEY(username)
- III. train  
PRIMARY KEY (train\_no,date)
- IV. train\_released  
PRIMARY KEY (train\_no,date,source,destination)
- v. ticket  
PRIMARY KEY (pnr\_no)
- VI. passenger  
PRIMARY KEY (pnr\_no,berth\_no,coach\_no)

## Foreign Key Relationships

- I. train\_released
  - FOREIGN KEY( train\_no, date) refer to table train (train\_no,date)
- II. ticket
  - FOREIGN KEY (username) refer to table user\_ (username)
  - FOREIGN KEY (train\_no, date, source, destination) refer to table train\_released (train\_no, date, source, destination)
- III. Passenger
  - FOREIGN KEY (pnr\_no) refer to table ticket (pnr\_no)

## Attribute Information

- 1) ac\_num : Total number of ac seats in a train before releasing.
- 2) sleeper\_num : Total number of sleeper seats in a train before releasing.
- 3) ac\_available : Total number of ac seats available in a train released into the system on a particular day.
- 4) sleeper\_available : Total number of sleeper seats available in a train released into the system on a particular day.
- 5) coach : Choice of coach (AC/SLEEPER).

## Stored Procedures

- I. assign\_berth
  - This is used to assign the coach\_no, berth\_no, berth\_type to the passengers and update the available seats for booking.
- II. seating\_plan
  - This is used to print all the details(name of passenger seat assigned for, berth\_no, berth\_type, coach\_no, pnr\_no, source and destination) of all the booked seats in a train on a day.

IN PLACE OF USING STORED PROCEDURES/FUNCTIONS/TRIGGERS AT VARIOUS INSTANCES, DIFFERENT CHECKS ARE ALREADY PROVIDED IN TABLES AS FOLLOWS -:

1. Constraint “NOT NULL” is provided to almost every attribute to ensure that any field does not remain blank.
2. Constraint “UNIQUE” is provided to prevent repeating details like email registered should not be repeated EXCEPT the primary key.
3. Constraint “CHECK” is provided to put some conditions like released seats should not be negative in number as well as should not be greater than the total ac/sleeper seats in a train.
4. Constraint “SERIAL” is provided for the random and non-repeating pnr number which is generated once a ticket is booked.

## Triggers

- I. check\_released\_seats  
Operation–After Insert

### Table–train\_released

- Check if the seats released are not negative as well as not zero in number in each coach (ac and sleeper) .
- Check if the seats released are not greater than the total number of seats in a train in each coach (sleeper/ ac).

## II. creating\_user

Operation–After Insert

### Table–user\_

- It creates a new user (user type) with the same username and password as one inserts in the user\_ table.
- It also grants the roles to the new user created.

## III. creating\_admin

Operation–After Insert

### Table–admin

- It creates a new user (admin type) with the same username and password as one inserts in the admin table.
- It also grants the roles to the new admin created.

## IV. getting\_seats

Operation–After Delete

### Table–ticket

- It deletes all the passengers details registered on a ticket when user(booker) cancels the booked ticket.
- It updates the seats available in the released trains after making the seats free which were booked before cancellation.

## Server Side Validations

- I. For Login, Register, Train Release, Ticket Booking : all field must be non-empty
- II. Register:
  - a. Password – Minimum of 5 characters.
  - b. Email–should be of actual email format (like [xyz@gmail.com](mailto:xyz@gmail.com)).

## Ticket Booking Procedure


### Step -I Registration Of New User(booker)

Insert new user details into the user\_ table who will book the ticket. On this same ticket , all the co-passengers will be registered .It is because on a single ticket, more than one passenger can be registered.

```
postgres=# INSERT INTO user_ VALUES('IRCTC','new user','irctc@gmail.com','railway','irctc');
INSERT 0 1
postgres=# SELECT * FROM user_;
```

username	name	email	address	password
postgres	postgres	postgres@gmail.com	postgres	12345
akash	akashkumar	akash@gmail.com	chharba	akash0
aman	aman	aman@gmail.com	chharba	aman0
nitin	nitin	nitin@gmail.com	chh	nitin
lavi	lavi	lavi@gmail.com	chharba	lavidon
aryan	arya	lav@gmail.com	chharba	aryan
cseproff	cse	cse@gmail.com	ropar	cse12345
IRCTC	new user	irctc@gmail.com	railway	irctc

(8 rows)




## Step -II Checking Train Availability

Using query, look into the table “table\_released” the date scheduled, train available with required seats in required coach.

```
postgres=# SELECT * FROM train_released;
```

train_no	date	source	destination	ac_available	sleeper_available
2	2022-07-15	delhi	mumbai	130	54
1	2022-07-17	punjab	dehradun	95	80
1	2022-07-17	delhi	mumbai	130	54
1	2022-07-15	punjab	dehradun	100	80
2	2022-07-15	punjab	dehradun	93	80
1	2022-07-15	dehradun	delhi	60	90
1	2022-07-15	delhi	mumbai	130	60
1	2022-07-16	punjab	dehradun	100	80
1	2022-07-16	dehradun	delhi	60	90
1	2022-07-16	delhi	mumbai	130	60
1	2022-07-17	dehradun	delhi	60	90
1	2022-07-18	punjab	dehradun	100	80
1	2022-07-18	dehradun	delhi	60	90
1	2022-07-18	delhi	mumbai	130	60
1	2022-07-19	punjab	dehradun	100	80
1	2022-07-19	dehradun	delhi	60	90
1	2022-07-19	delhi	mumbai	130	60
2	2022-07-15	dehradun	delhi	60	90
2	2022-07-16	punjab	dehradun	100	80
2	2022-07-16	dehradun	delhi	60	90
2	2022-07-16	delhi	mumbai	130	60
2	2022-07-17	punjab	dehradun	100	80
2	2022-07-17	dehradun	delhi	60	90
2	2022-07-17	delhi	mumbai	130	60
2	2022-07-18	punjab	dehradun	100	80
2	2022-07-18	dehradun	delhi	60	90




## Step -III Booking A Ticket(PNR Generation)

Insert all attribute values in the ticket table except in “pnr\_no” attribute which is set as “SERIAL” .This will generate pnr number.

```
postgres=# INSERT INTO ticket(train_no,date,coach,username,source,destination) VALUES(2,'18-07-2022','sleeper','IRCTC','punjab','dehradun');
INSERT 0 1
postgres=# SELECT * FROM ticket;
```

pnr_no	train_no	date	coach	username	source	destination
16	2	2022-07-15	ac	nitin	punjab	dehradun
17	1	2022-07-17	ac	cseproff	punjab	dehradun
18	2	2022-07-15	sleeper	akash	delhi	mumbai
20	2	2022-07-15	sleeper	aryan	delhi	mumbai
21	1	2022-07-17	sleeper	aman	delhi	mumbai
22	1	2022-07-17	sleeper	lavi	delhi	mumbai
23	2	2022-07-18	sleeper	IRCTC	punjab	dehradun

(7 rows)





## Step -IV Getting Passenger Details And Assigning Berth

Call procedure “assign\_berth” which takes passenger details and previously generated pnr\_no as input.

```
postgres=# Call assign_berth('LION',52,'MALE',23);
CALL
postgres=# Call assign_berth('LIONESS',40,'FEMALE',23);
CALL
postgres=# Call assign_berth('TIGER',25,'MALE',23);
CALL
postgres=# SELECT * FROM passenger;
```

pnr_no	name	age	gender	berth_no	berth_type	coach_no
16	nitin	10	male	15	UB	3
16	nitin	10	male	16	UB	3
17	csproject	20	male	7	LB	4
16	rajesh	25	male	17	SL	3
16	raja	29	male	0	SU	3
17	nitin	20	male	8	LB	4
17	ajay	25	male	9	UB	4
17	shalu	20	female	10	UB	4
17	resmi	25	female	11	SL	4
21	nitin	20	male	23	SL	3
21	ajay	25	male	0	SU	3
21	shalu	20	female	1	LB	4
22	nitin	20	male	2	MB	4
22	ajay	25	male	3	UB	4
22	shalu	20	female	4	LB	4
16	nitin	20	male	1	LB	4
16	ajay	25	male	2	LB	4
16	resmi	25	female	3	UB	4
18	nitin	20	male	9	LB	4
18	ajay	25	male	10	MB	4
18	shalu	20	female	11	UB	4
20	nitin	20	male	12	LB	4
20	ajay	25	male	13	MB	4
20	shalu	20	female	14	UB	4
23	LION	52	MALE	9	LB	4
23	LIONESS	40	FEMALE	10	MB	4
23	TIGER	25	MALE	11	UB	4

(27 rows)

## Step -V Printing Entire Seating Arrangement Of A Train For A Date

Call procedure “seating\_plan” which takes train number and date as input parameters. After that , data will be printed in the additional table “seating\_plan”.

### SEATING ARRANGEMENT FOR TRAIN NUMBER=2 AND DATE=18<sup>th</sup> OF JULY

```
postgres=# Call seating_plan(2,'18-07-2022');
CALL
postgres=# SELECT * FROM seating_plan;
 coach_no | berth_no | berth_type | name   | pnr_no | source | destination
-----+-----+-----+-----+-----+-----+-----
      4 |      9 | LB         | LION   | 23     | punjab | dehradun
      4 |     10 | MB         | LIONESS| 23     | punjab | dehradun
      4 |     11 | UB         | TIGER  | 23     | punjab | dehradun
(3 rows)
```

### SEATING ARRANGEMENT FOR TRAIN NUMBER=2 AND DATE=15<sup>th</sup> OF JULY

```
postgres=# Call seating_plan(2,'15-07-2022');
CALL
postgres=# SELECT * FROM seating_plan order by coach_no,berth_no;
 coach_no | berth_no | berth_type | name   | pnr_no | source | destination
-----+-----+-----+-----+-----+-----+-----
      3 |      0 | SU         | raja   | 16     | punjab | dehradun
      3 |     15 | UB         | nitin  | 16     | punjab | dehradun
      3 |     16 | UB         | nitin  | 16     | punjab | dehradun
      3 |     17 | SL         | rajesh | 16     | punjab | dehradun
      4 |      1 | LB         | nitin  | 16     | punjab | dehradun
      4 |      2 | LB         | ajay   | 16     | punjab | dehradun
      4 |      3 | UB         | resmi  | 16     | punjab | dehradun
      4 |      9 | LB         | nitin  | 18     | delhi  | mumbai
      4 |     10 | MB         | ajay   | 18     | delhi  | mumbai
      4 |     11 | UB         | shalu  | 18     | delhi  | mumbai
      4 |     12 | LB         | nitin  | 20     | delhi  | mumbai
      4 |     13 | MB         | ajay   | 20     | delhi  | mumbai
      4 |     14 | UB         | shalu  | 20     | delhi  | mumbai
(13 rows)
```

## Step -VI Booking Cancellation

To cancel a booked ticket, just delete the tuple from the table “ticket” according to the pnr number which is generated on the ticket which we want to cancel. After that, triggers and function will update the free seats and delete the passenger details too registered on the ticket.

```
postgres=# DELETE FROM ticket where pnr_no=23;
DELETE 1
postgres=# SELECT * FROM ticket;
```

pnr_no	train_no	date	coach	username	source	destination
16	2	2022-07-15	ac	nitin	punjab	dehradun
17	1	2022-07-17	ac	cseproff	punjab	dehradun
18	2	2022-07-15	sleeper	akash	delhi	mumbai
20	2	2022-07-15	sleeper	aryan	delhi	mumbai
21	1	2022-07-17	sleeper	aman	delhi	mumbai
22	1	2022-07-17	sleeper	lavi	delhi	mumbai

(6 rows)

```
postgres=# SELECT * FROM passenger;
```

pnr_no	name	age	gender	berth_no	berth_type	coach_no
16	nitin	10	male	15	UB	3
16	nitin	10	male	16	UB	3
17	csproject	20	male	7	LB	4
16	rajesh	25	male	17	SL	3
16	raja	29	male	0	SU	3
17	nitin	20	male	8	LB	4
17	ajay	25	male	9	UB	4
17	shalu	20	female	10	UB	4
17	resmi	25	female	11	SL	4
21	nitin	20	male	23	SL	3
21	ajay	25	male	0	SU	3
21	shalu	20	female	1	LB	4
22	nitin	20	male	2	MB	4
22	ajay	25	male	3	UB	4
22	shalu	20	female	4	LB	4
16	nitin	20	male	1	LB	4
16	ajay	25	male	2	LB	4
16	resmi	25	female	3	UB	4
18	nitin	20	male	9	LB	4
18	ajay	25	male	10	MB	4
18	shalu	20	female	11	UB	4
20	nitin	20	male	12	LB	4
20	ajay	25	male	13	MB	4
20	shalu	20	female	14	UB	4

(24 rows)

## Technology Stack

Database–PostgreSQL

## Steps To Run

- Install PostgreSQL on your system.
- Here database name is “postgres”, its username is “postgres” and its password is “12345”.
- Import the database file named as “postgres” (uploaded in SQL folder) in PostgreSQL(pgAdmin).

OR

- Copy all the scripts of tables, functions, triggers and that of procedures too into PostgreSQL ,in a newly created database, and run them once.
- Run queries as required in SQL Shell(psql).