DataBase Management Project On Railway Reservation System

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Gautam Sunuwar

Registration number: 12111066

Roll No:K21DPB40

Section:K21DP

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

1.INTRODUCTION

Database is an organized collection of data. The data is typically organized to model aspects of reality in a way that supports processes requiring information. A DBMS makes it possible for end users to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible. The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified and the database schema, which defines the database's logical structure. These three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. The DBMS can offer both logical and physical data independence. That means it can protect users and applications from needing to know where data is stored or having to be concerned about changes to the physical structure of data.

The main purpose of maintaining database for Railway Reservation System is to reduce the manual errors involved in the booking and cancelling of tickets and make it convenient for the customers and providers to maintain the data about their customers and also about the seats available at them. Due to automation many loopholes that exist in the manual maintenance of the records can be removed. The speed of obtaining and processing the data will be fast. For future expansion the proposed system can be web enabled so that clients can make various enquiries about trains between stations. Due to this, sometimes a lot of problems occur and they are facing many disputes with customers. To solve the above problem, we design a data base which includes customer details, availability of seats in trains
no of trains and their details.

1.1 OBJECTIVE OF PROJECT

The Railway Reservation System facilitates the passengers to enquire about the trains available on the basis of source and destination, Booking and Cancellation of tickets, enquire about the status of the booked ticket, etc. The aim of case study is to design and develop a database maintaining the records of different trains, train status, and passengers. .It is the computerized system of reserving the seats of train seats in advanced. It is mainly used for long route. On-line reservation has made the process for the reservation of seats very much easier than ever before.

In country like India, there are number of counters for the reservation of the seats and one can easily make reservations and get tickets. Then this project contains entity relationship model diagram based on railway reservation system and introduction to relation model. There is also design of the database of the railway reservation system based on relation model. Example of some SQL queries to retrieves data from rail management database.

This project explores how computer technology can be used to solve the problem of user. The main objectives provided by this software are as follows:

- ❖ To enquire about availability of trains
- ❖ To reserve and cancel their seats
- ❖ To modify the information related to trains details , time , tickets fare etc.

This project is dedicated to model existing railway reservation systems that aim at development of Railway Reservation System that facilitates the railway customer to manage their reservations and the railway administrator to modify the backend database in a user-friendly manner.

1.2 PROJECT DESCRIPTION

This project is about creating the database about Railway Reservation System. The railway reservation system facilitates the passengers to enquire about the trains available on the basis of source and destination, booking and cancellation of tickets, enquire about the status of the booked ticket, etc. The aim of case study is to design and develop a database maintaining the records of different trains, train status, and passengers. The record of train includes its number, name, source, destination, and days on which it is available, whereas record of train status includes dates for which tickets can be booked, total number of seats available, and number of seats already booked.

Passengers can book their tickets for the train in which seats are available. For this, passenger has to provide the desired train number and the date for which ticket is to be booked. Before booking a ticket for a passenger, the validity of train number and booking date is checked. Once the train number and booking date are validated, it is checked whether the seat is available. If yes, the ticket is booked with confirm status and corresponding ticket ID is generated which is stored along with other details of the passenger. The ticket once booked can be cancelled at any time. For this, the passenger has to provide the ticket ID (the unique key). The ticket ID is searched and the corresponding record is deleted.

List of Assumption Since the reservation system is very large in reality, it is not feasible to develop the case study to that extent and prepare documentation at that level. Therefore, a small sample case study has been created to demonstrate the working of the reservation system. To implement this sample case study, some assumptions have been made, which are as follows:

1. The number of trains has been restricted to 5.

2. The booking is open only for next seven days from the current date.

3. Only two categories of tickets can be booked, namely, AC and General.

4. The total number of tickets that can be booked in each category (AC and General) is 10.

5. The total number of tickets that can be given the status of waiting is 2.

6. The in- between stoppage stations and their bookings are not considered.

List of trains has to be maintained. Detailed Passenger information is to be maintained In the booking procedure, the train number, train date, and category are read from the passenger. On the basis of the values provided by the passenger, corresponding record is retrieved from the Train Status. If the desired category is AC, then total number of AC seats and number of booked AC seats are compared in order to find whether ticket can be booked or not. Similarly, it can be checked for the general category. If ticket can be booked, then passenger details are read and stored in the Passenger table. In the cancellation procedure, ticket ID is read from the passenger and corresponding record is searched in the Passenger. If the record exists, it is deleted. After deleting the record (if it is confirmed), first record with waiting status for the same train and same category are searched from the Passenger table and its status is changed to confirm.

1.3 SCOPE OF THE PROJECT

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing systems. The system provides proper security and reduces the manual work.

- ❖ Security of data.
- ❖ Ensure data accuracy's.
- ❖ Proper control of the higher officials.
- ❖ Minimize manual data entry.
- ❖ Minimum time needed for the various processing.
- ❖ Greater efficiency.
- ❖ Better service.
- ❖ User friendliness and interactive.
- ❖ Minimum time required.

2.System Description

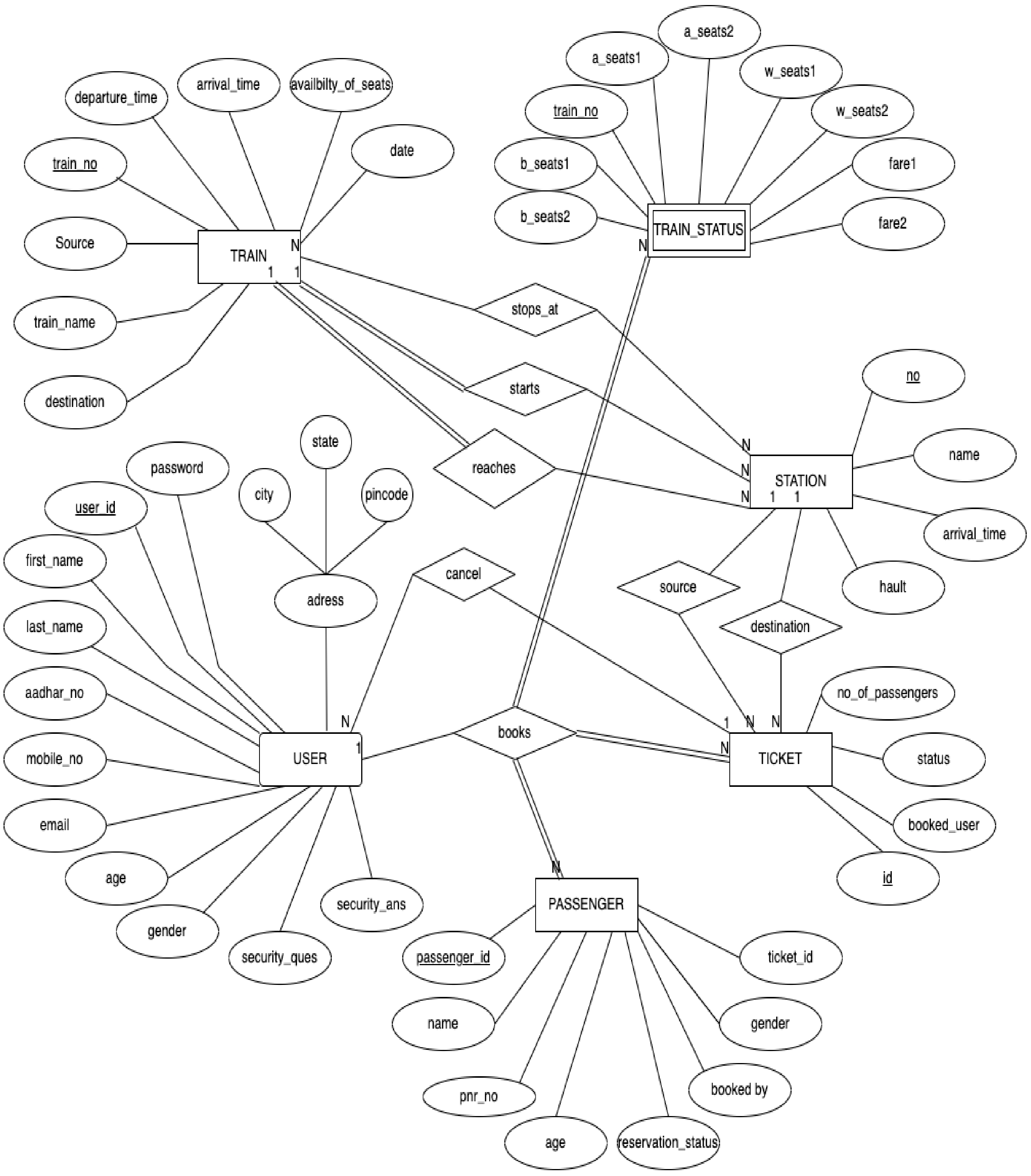
2.1 SCHEMA DESIGN

2.1.1 Listing of Entities & Attributes

ENTITIES	ATTRIBUTES
User	**User_id** Password First_name Last_name Gender Age Email Aadhar_no Mobile_no City State Pincode Security_ques Security_ans
passenger	**Passenger_id** Name Gender Age Pnr_no Seat_no Booked_by Reservation_status

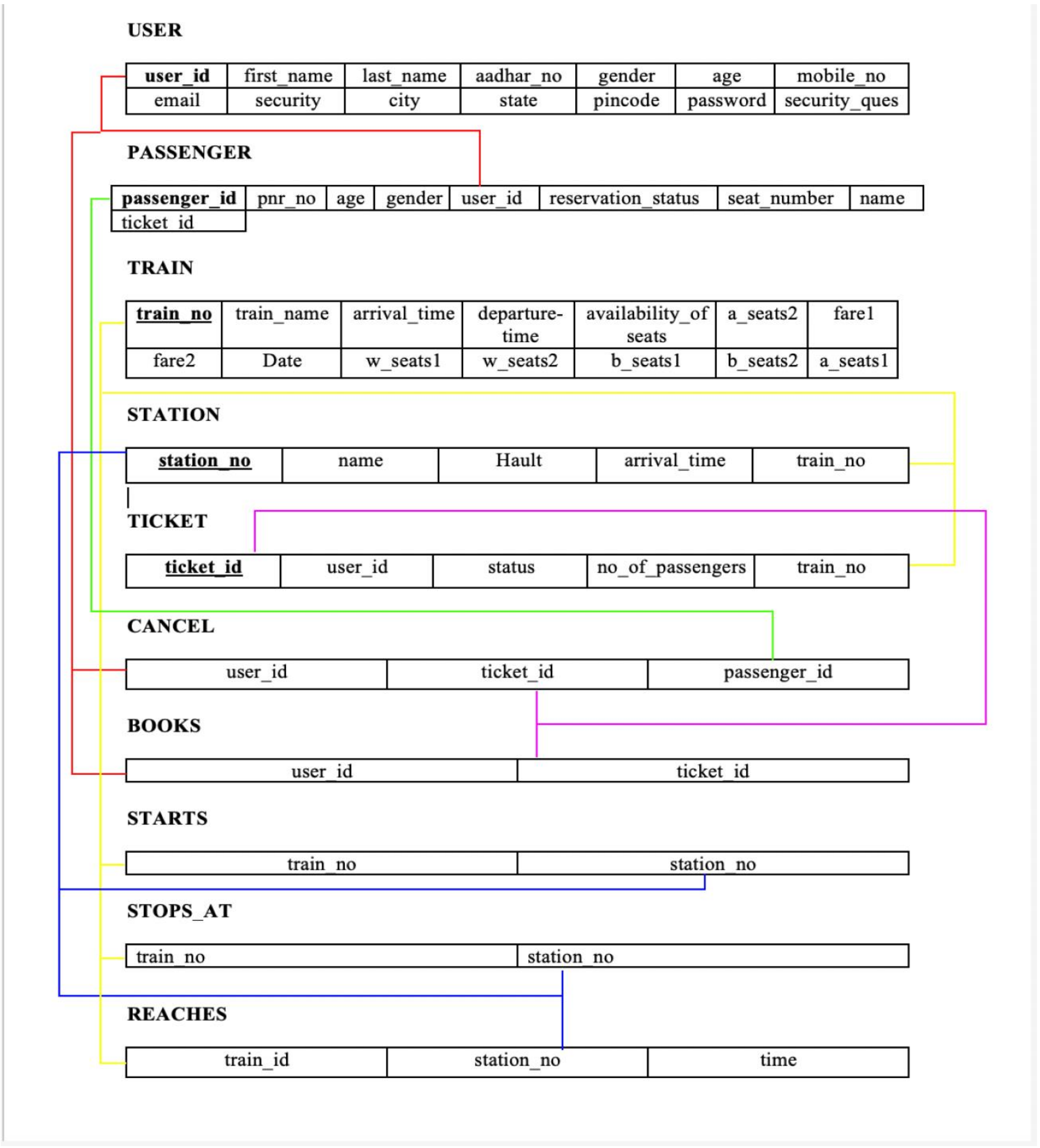
Train	**Train_no** Train_name Source Destination Arrival_time Departure_time Avalibility_of_seats Train_no A_seats1 A_seats2 A_seats3 B_seats1 B_seats2 B_seats3 W_Seats1 W_seats2 W_seats3
Station	**Station_No** Name Train_no Arrival_time Hault
Ticket	**Ticket_Id** Train_no Booked_user Status No_of_passengers

2.1.2 ER MODELS



2.3 Normalization & Final List of Relation

3.2.1 Normalization Table



USER

user_id	first_name	last_name	aadhar_no	gender	age	mobile_no
email	security	city	state	pincode	password	security_ques

PASSENGER

passenger_id	pnr_no	age	gender	user_id	reservation_status	seat_number	name
ticket_id							

TRAIN

train_no	train_name	arrival_time	departure-time	availability_of seats	a_seats2	fare1
fare2	Date	w_seats1	w_seats2	b_seats1	b_seats2	a_seats1

STATION

station_no	name	Hault	arrival_time	train_no

TICKET

ticket_id	user_id	status	no_of_passengers	train_no

CANCEL

user_id	ticket_id	passenger_id

BOOKS

user_id	ticket_id

STARTS

train_no	station_no

STOPS_AT

train_no	station_no

REACHES

train_id	station_no	time

User_id: Red **train_no: Yellow** **station_no: Blue**
passenger_id: Green **ticket_id: Pink**

3.2.2 FINAL LIST OF RELATION SHIPS:

books - Ternary relation ship between USER,TRAIN,PASSENGER and TICKET.

starts - Between TRAIN and STATION

reaches - Between TRAIN and STATION

cancel - Between USER and TICKET

stops_at - Between TRAIN and STATION

4.PL-SQL

4.1 Create table and insert

4.1.1 Data Definition Language(DDL) :

4.1.1.1 CREATE USESR1

```
create table  USER1
(
user_id int primary key,
first_name varchar(50),
last_name varchar(50),
adhar_no varchar(20),
gender char,
age int,
mobile_no varchar(50),
email varchar(50),
city varchar(50),
state varchar(50),
pincode varchar(20),
password varchar(50),
security_ques varchar(50)
security_ans varchar(50)
);
```

4.1.1.2 CREATE TRAIN

```
create table  TRAIN
(
train_no int primary key,
train_name
varchar(50),
arrival_time timestamp,
departure_time timestamp,
availability_of_seats char,
date1 date
);
```

4.1.1.3 CREATE TRAIN

```
create table  STATION
(
station_no int ,
name varchar(50),
hault int,
arrival_time int,
train_no int
);
```

4.1.1.4 CREATE TRAIN_STATUS

```
create table TRAIN_STATUS
(
 train_no int primary key,
 b_seats1 int,
 b_seats2 int,
 a_seats1 int,
 a_seats2 int,
 w_seats1 int,
 w_seats2 int,
 fare1 float,
 fare2 float
);
```

4.1.1.5 CREATE TICKET

```
create table  TICKET
(
ticket_id int primary key,
user_id int,
status char,
no_of_passengers int,
train_no int
);
```

4.1.1.6 CREATE PASSENGER

```
create table PASSENGER
(passenger_id int primary key,
pnr_no int,
age int,
gender char,
user_id int,
reservation_status char,
seat_number varchar(5),
name varchar(50),
ticket_id int
);
```

4.1.1.7 CREATE STARTS

```
create table STARTS
(
train_no int primary key,
station_no int,
);
```

4.1.1.8 CREATE STOPS_AT

```
create table STOPS_AT
(
train_no int,
station_no int
);
```

4.1.1.9 CREATE REACHES

```
create table  REACHES

(
train_no int,
station_no int,
time1 timestamp,
);
```

4.1.1.10 CREATE BOOKING

```
create table BOOKING
(
passenger_id int primary key,
pnr_no int,
age int,
gender char,
user_id int,
reservation_status char,
seat_number varchar(5),
pname varchar(50),
ticket_id int,
ticket_price int,
ticket_status varchar(20)
);
```

4.1.1.11 CREATE CANCEL

```
create table CANCEL1
(
user_id int,
id int,
passenger_id int
);
```

4.1.2 DML (Data manipulation language) :

4.1.2.1 Insert in USER1

```
insert into USER1 values
(1701,'vijay', 'sharma','309887340843','M',34,'9887786655','vijay1@gmail.com','vijayawada',
'andhrapradesh','520001','12345@#','favouritecolour','red');
```

```
insert into USER1 values
(1702,'rohith','kumar','456709871234,','M',45,'9809666555','rohith1kumar@gmail.com','gunt
ur','andhrapradesh','522004','12@#345','favouritebike','bmw');
```

```
insert into USER1 values
(1703,'manasvi','sree','765843210987','F',20,'9995550666','manasvi57@gmail.com','guntur','a
ndhrapradesh','522004','0987hii', 'favourite flower','rose');
```

4.1.2.2 Insert in TRAIN

```
insert into TRAIN values(12711,'pinakini exp','113000','114000','A',20170410);
insert into TRAIN values(12315,'cormandel exp','124500','125000','B',20170410);
```

4.1.2.3 Insert in TRAIN_STATUS

```
insert into TRAIN_STATUS values(12711,10,4,0,1,1,0,100,450);
insert into TRAIN_STATUS values(12315,10,5,0,0,2,1,300,600);
```

4.1.2.4 Insert in TICKET

```
insert into TICKET values(4001,1701,'C',1,12711);
insert into TICKET values(4002,1702,'N',1,12315);
```

4.1.2.5 Insert in PESSENGER

```
insert into PASSENGER values(5001,78965,45, 'M',1701,'C','B6-45','ramesh',4001);
insert into PASSENGER values(5002,54523,54,'F',1701,'W','B3-21','surekha',4002);
```

4.1.2.6 Insert in STARTS

```
insert into STARTS values(12711,111);
insert into STARTS values(12315,222);
```

4.1.2.7 Insert in STOPS_AT

```
insert into STOPS_AT values(12711,222);
insert into STOPS_AT values(12315,111);
```

4.1.2.8 Insert in REACHE

```
insert into REACHE values(12711,222,'040000');
insert into REACHE values(12315,111,'053500');
```

4.1.2.9 Insert in BOOKING

```
insert into BOOKING values(5001,78965,45, 'M',1701,'C','B6-45','ramesh',4001,
10000,'sleeperClass');
insert into BOOKING values(5002,54523,54,'F',1702,'W','B3-21','surekha',4002, 5400,
'GeneralClass');
insert into BOOKING values(5003,89233,64,'F',1703,'W','B3-25','surekha',4002, 4500,
'GeneralClass');
```

4.1.2.10 Insert in CANCEL1

```
insert into CANCEL1 values (1701,4001,5001);
insert into CANCEL11 values (1702,4002,5002);
insert into CANCEL1 values (1703,4003,5003);
```

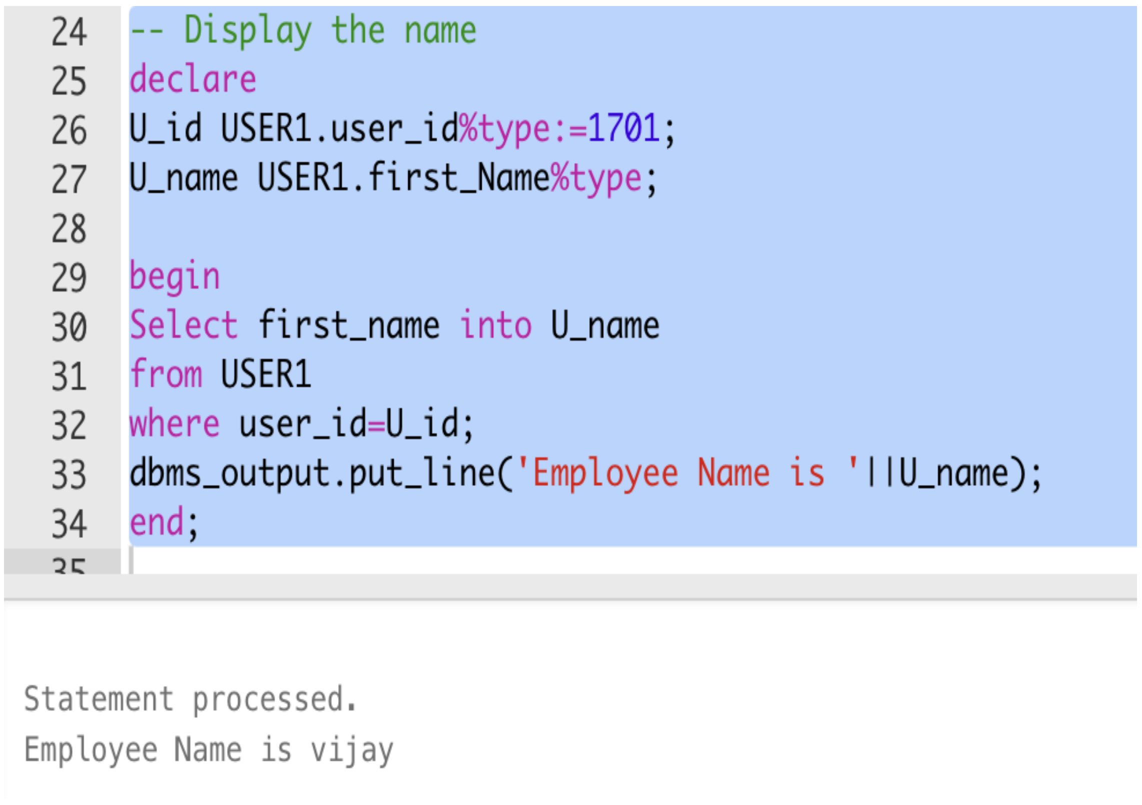
4.1.2.11 Insert in CANCEL1

```
insert into STATION values(12321,'pinakini exp','143100','114000',20170410);
insert into STATION values(16543,'cormandel exp','1214500','125000',20170410);
insert into STATION values(12243,'KANDU exp','1214500','125000',20172160);
```

4.2 PL-SQL QUERIES RELATED TO REPORT GENERATION

4.2.1. Print name of person whose user ID 1701

```
declare
U_id USER1.user_id%type:=1701;
U_name USER1.first_Name%type;
begin
Select first_name into U_name
from USER1
where user_id=U_id;
dbms_output.put_line('Employee Name is '||U_name);
end;
```

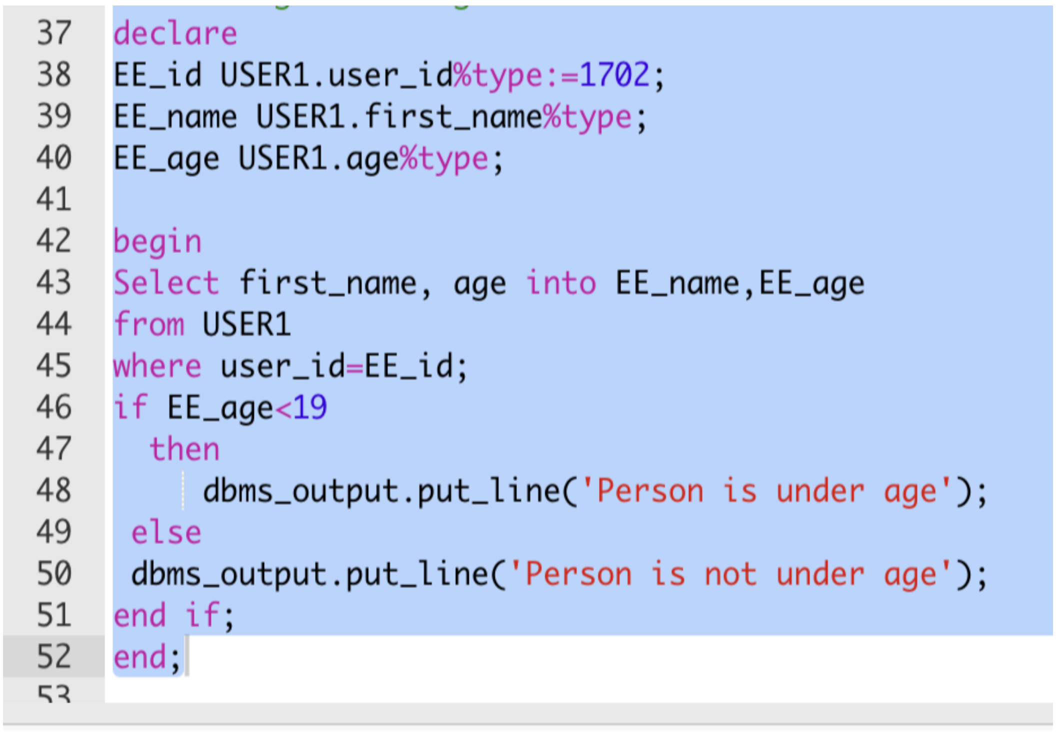


```
-- Display the name
declare
U_id USER1.user_id%type:=1701;
U_name USER1.first_Name%type;

begin
Select first_name into U_name
from USER1
where user_id=U_id;
dbms_output.put_line('Employee Name is '||U_name);
end;
```

```
Statement processed.
Employee Name is vijay
```

4.2.2 Check if the passengers travelling is under age or not

```
EE_id USER1.user_id%type:=1702;
EE_name USER1.first_name%type;
EE_age USER1.age%type;

begin
Select first_name, age into EE_name,EE_age
from USER1
where user_id=EE_id;
if EE_age<19
  then
     dbms_output.put_line('Person is under age');
 else
 dbms_output.put_line('Person is not under age');
end if;
end;
```

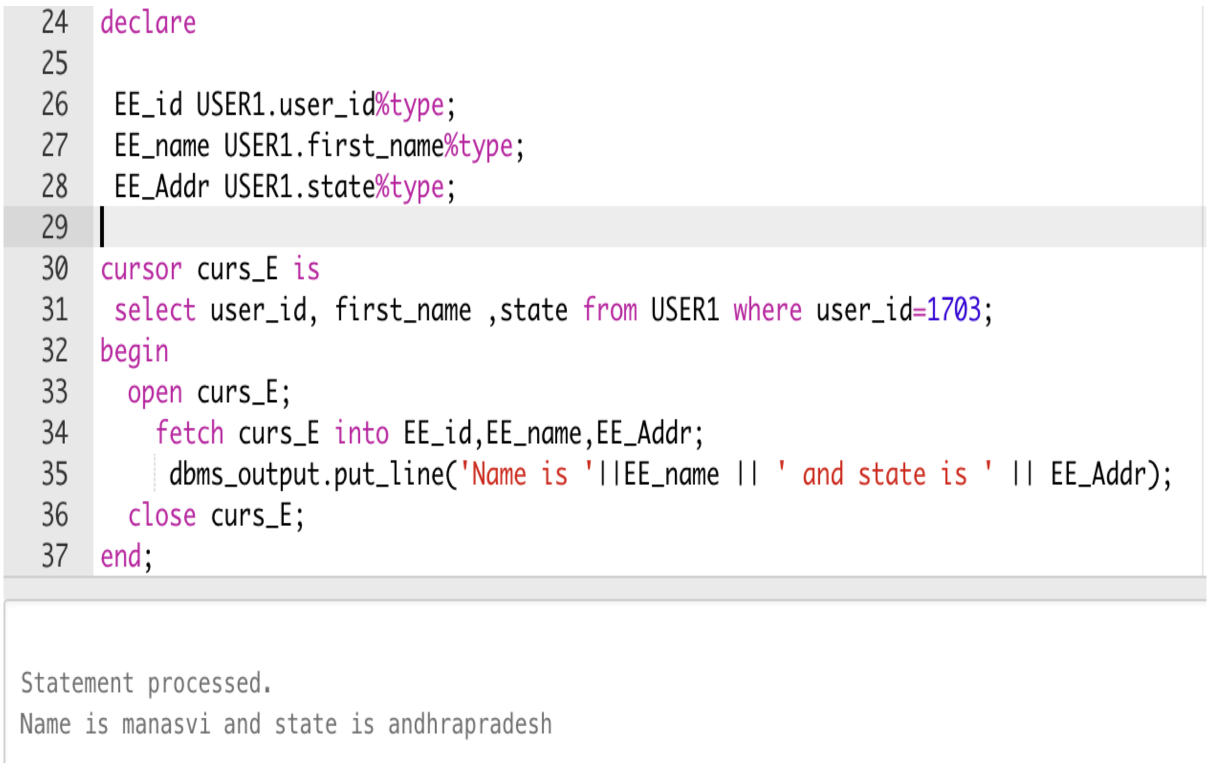


```
declare
EE_id USER1.user_id%type:=1702;
EE_name USER1.first_name%type;
EE_age USER1.age%type;

begin
Select first_name, age into EE_name,EE_age
from USER1
where user_id=EE_id;
if EE_age<19
  then
     dbms_output.put_line('Person is under age');
 else
 dbms_output.put_line('Person is not under age');
end if;
end;
```

```
Statement processed.
Person is not under age
```

4.2.3 Display State and pessanger's name of user ID user (cursor)

```
declare
 EE_id USER1.user_id%type;
 EE_name USER1.first_name%type;
 EE_Addr USER1.state%type;

cursor curs_E is
 select user_id, first_name ,state from USER1 where user_id=1703;
begin
  open curs_E;
    fetch curs_E into EE_id,EE_name,EE_Addr;
     dbms_output.put_line('Name is '||EE_name || ' and state is ' || EE_Addr);
  close curs_E;
end;
```

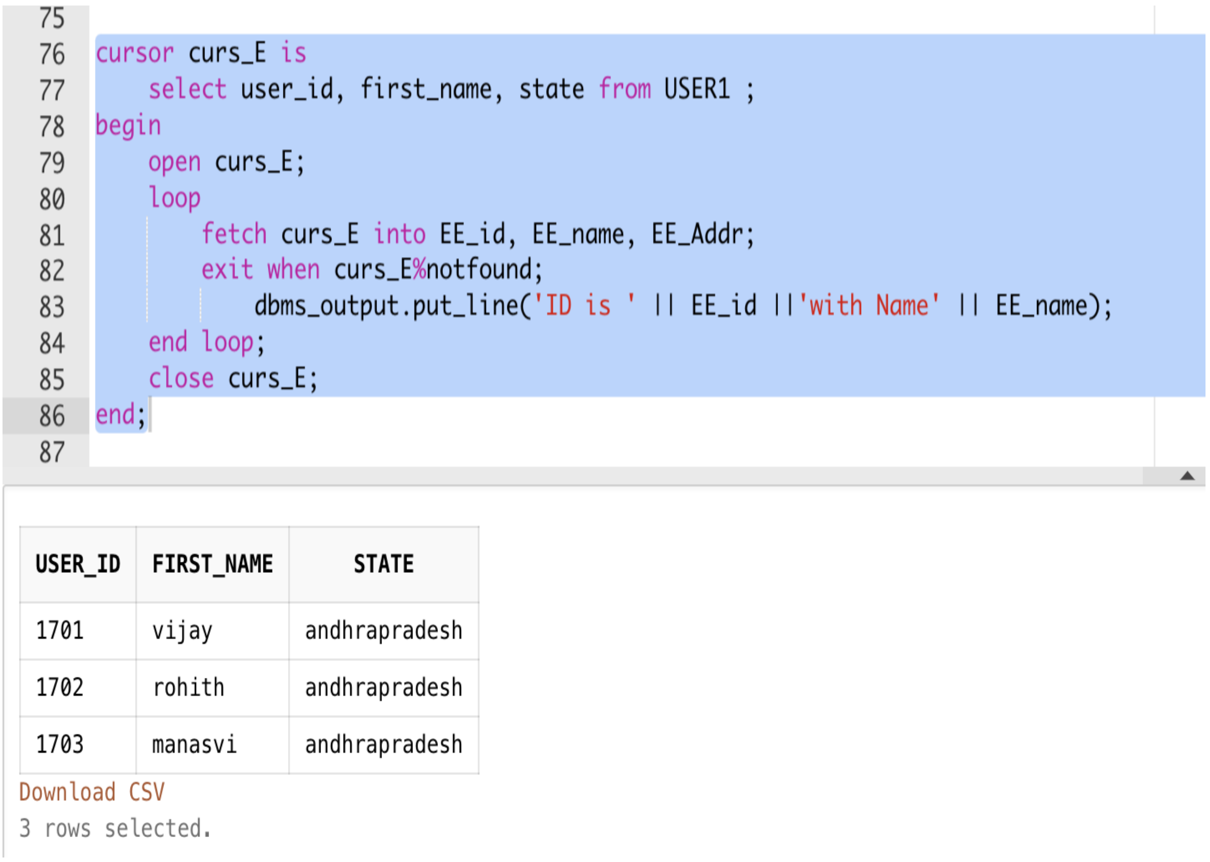


```
declare

 EE_id USER1.user_id%type;
 EE_name USER1.first_name%type;
 EE_Addr USER1.state%type;

cursor curs_E is
 select user_id, first_name ,state from USER1 where user_id=1703;
begin
  open curs_E;
    fetch curs_E into EE_id,EE_name,EE_Addr;
     dbms_output.put_line('Name is '||EE_name || ' and state is ' || EE_Addr);
  close curs_E;
end;
```

```
Statement processed.
Name is manasvi and state is andhrapradesh
```

4.2.4 Display details of all the passengers with downdable CSV (cursor)

```
cursor curs_E is
    select user_id, first_name, state from USER1 ;
begin
    open curs_E;
    loop
        fetch curs_E into EE_id, EE_name, EE_Addr;
        exit when curs_E%notfound;
            dbms_output.put_line('ID is ' || EE_id ||'with Name' || EE_name);
    end loop;
    close curs_E;
end;
```



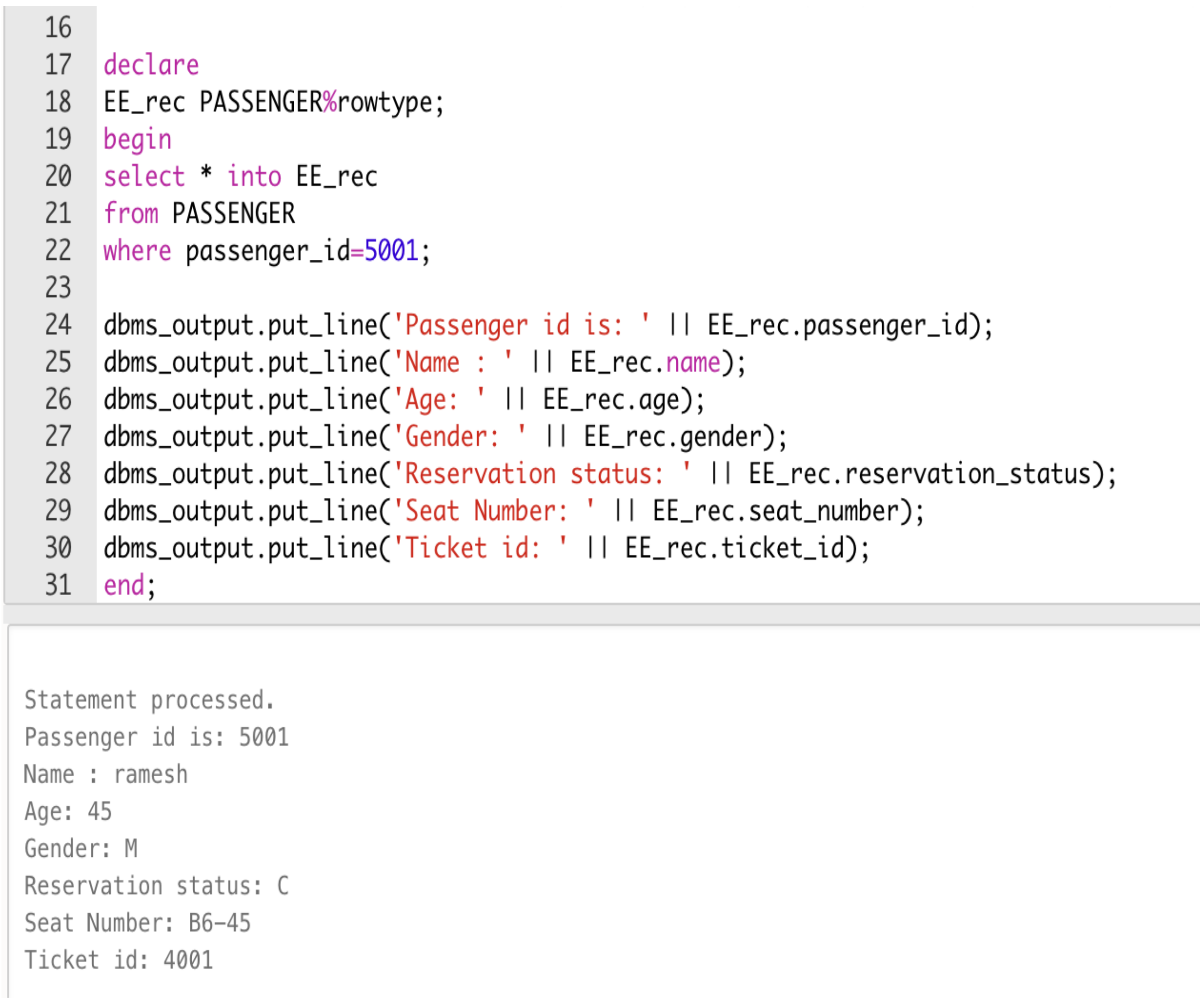
```
cursor curs_E is
    select user_id, first_name, state from USER1 ;
begin
    open curs_E;
    loop
        fetch curs_E into EE_id, EE_name, EE_Addr;
        exit when curs_E%notfound;
            dbms_output.put_line('ID is ' || EE_id ||'with Name' || EE_name);
    end loop;
    close curs_E;
end;
```

USER_ID	FIRST_NAME	STATE
1701	vijay	andhrapradesh
1702	rohith	andhrapradesh
1703	manasvi	andhrapradesh

```
Download CSV
3 rows selected.
```

4.2.5 Display the Detail Recipt of pessenger with pessenger ID=5001

```
declare
EE_rec PASSENGER%rowtype;
begin
select * into EE_rec
from PASSENGER
where passenger_id=5001;

dbms_output.put_line('Passenger id is: ' || EE_rec.passenger_id);
dbms_output.put_line('Name : ' || EE_rec.name);
dbms_output.put_line('Age: ' || EE_rec.age);
dbms_output.put_line('Gender: ' || EE_rec.gender);
dbms_output.put_line('Reservation status: ' || EE_rec.reservation_status);
dbms_output.put_line('Seat Number: ' || EE_rec.seat_number);
dbms_output.put_line('Ticket id: ' || EE_rec.ticket_id);
end;
```

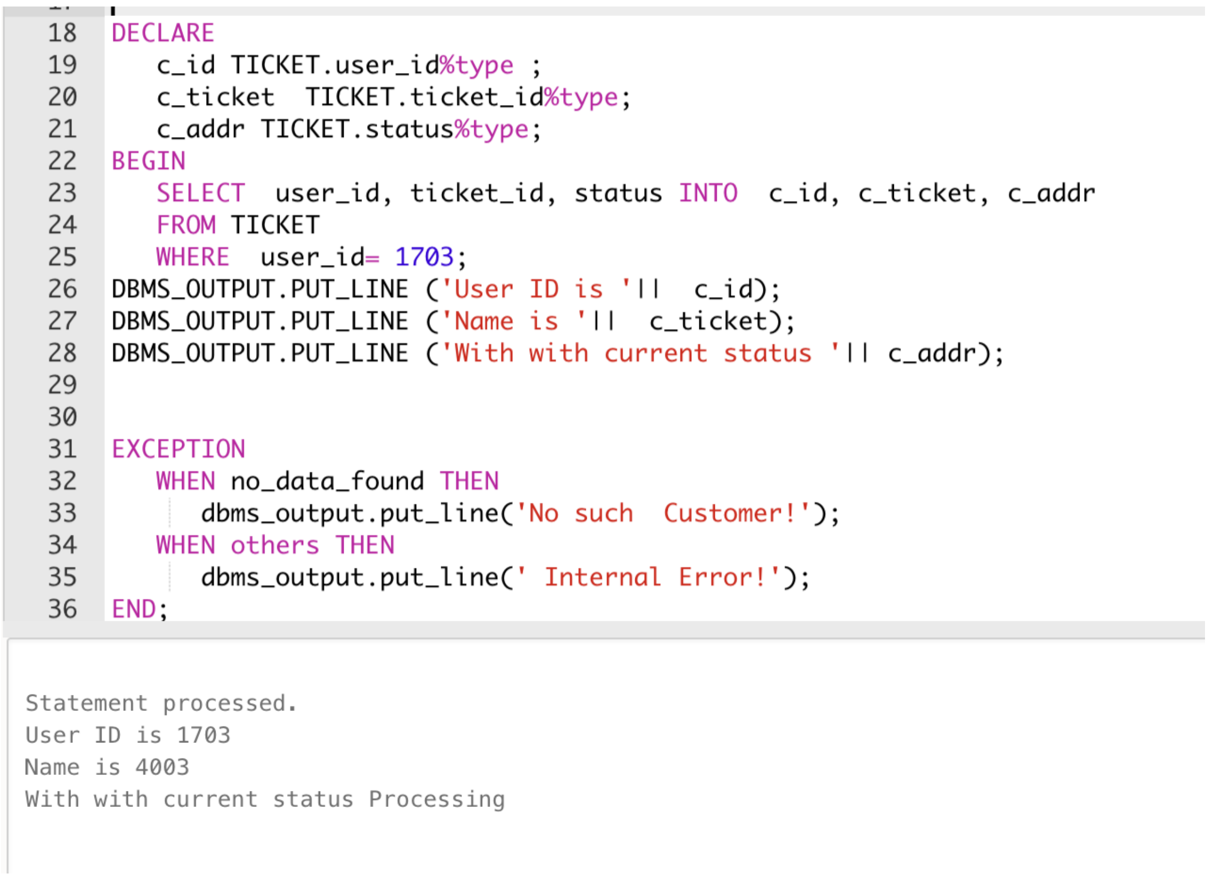


```

declare
EE_rec PASSENGER%rowtype;
begin
select * into EE_rec
from PASSENGER
where passenger_id=5001;

dbms_output.put_line('Passenger id is: ' || EE_rec.passenger_id);
dbms_output.put_line('Name : ' || EE_rec.name);
dbms_output.put_line('Age: ' || EE_rec.age);
dbms_output.put_line('Gender: ' || EE_rec.gender);
dbms_output.put_line('Reservation status: ' || EE_rec.reservation_status);
dbms_output.put_line('Seat Number: ' || EE_rec.seat_number);
dbms_output.put_line('Ticket id: ' || EE_rec.ticket_id);
end;
```

```
Statement processed.
Passenger id is: 5001
Name : ramesh
Age: 45
Gender: M
Reservation status: C
Seat Number: B6-45
Ticket id: 4001
```

4.2.6 Add Exception and handle it while displaying the ticket status of user id=1703 (Exception handle)

```
DECLARE
   c_id TICKET.user_id%type ;
   c_ticket  TICKET.ticket_id%type;
   c_addr TICKET.status%type;
BEGIN
   SELECT  user_id, ticket_id, status INTO  c_id, c_ticket, c_addr
   FROM TICKET
   WHERE  user_id= 1703;
DBMS_OUTPUT.PUT_LINE ('User ID is '||  c_id);
DBMS_OUTPUT.PUT_LINE ('Name is '||  c_ticket);
DBMS_OUTPUT.PUT_LINE ('With with current status '|| c_addr);


EXCEPTION
   WHEN no_data_found THEN
      dbms_output.put_line('No such  Customer!');
   WHEN others THEN
      dbms_output.put_line(' Internal Error!');
END;
```

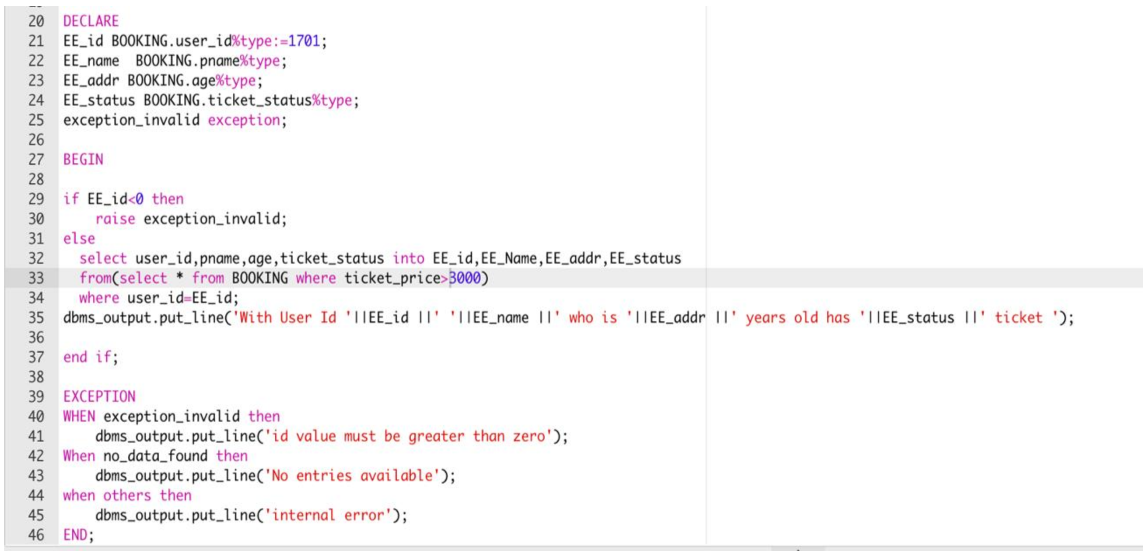


```
DECLARE
   c_id TICKET.user_id%type ;
   c_ticket  TICKET.ticket_id%type;
   c_addr TICKET.status%type;
BEGIN
   SELECT  user_id, ticket_id, status INTO  c_id, c_ticket, c_addr
   FROM TICKET
   WHERE  user_id= 1703;
DBMS_OUTPUT.PUT_LINE ('User ID is '||  c_id);
DBMS_OUTPUT.PUT_LINE ('Name is '||  c_ticket);
DBMS_OUTPUT.PUT_LINE ('With with current status '|| c_addr);


EXCEPTION
   WHEN no_data_found THEN
      dbms_output.put_line('No such  Customer!');
   WHEN others THEN
      dbms_output.put_line(' Internal Error!');
END;
```

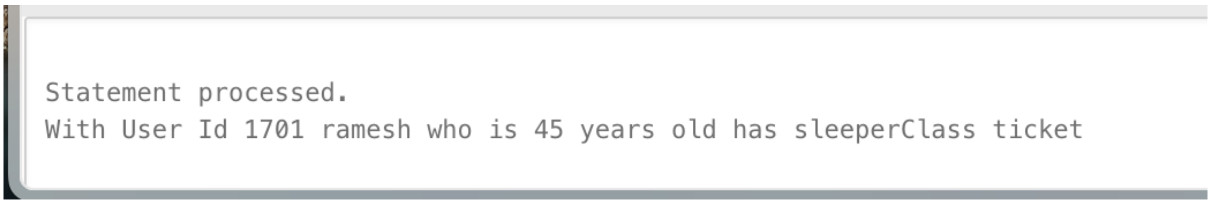
```
Statement processed.
User ID is 1703
Name is 4003
With with current status Processing
```

4.2.7 Display the type of Ticket passenger with user id=1701 is having. Add exception to handle the invild data.

```
DECLARE
EE_id BOOKING.user_id%type:=1701;
EE_name  BOOKING.pname%type;
EE_addr BOOKING.age%type;
EE_status BOOKING.ticket_status%type;
exception_invalid exception;
BEGIN
if EE_id<0 then
    raise exception_invalid;
else
  select user_id,pname,age,ticket_status into EE_id,EE_Name,EE_addr,EE_status
  from(select * from BOOKING where ticket_price>3000)
  where user_id=EE_id;
dbms_output.put_line('With User Id '||EE_id ||' '||EE_name ||' who is '||EE_addr ||' years old
has '||EE_status ||' ticket ');
end if;

EXCEPTION
WHEN exception_invalid then
    dbms_output.put_line('id value must be greater than zero');
When no_data_found then
    dbms_output.put_line('No entries available');
when others then
    dbms_output.put_line('internal error');
end;
```



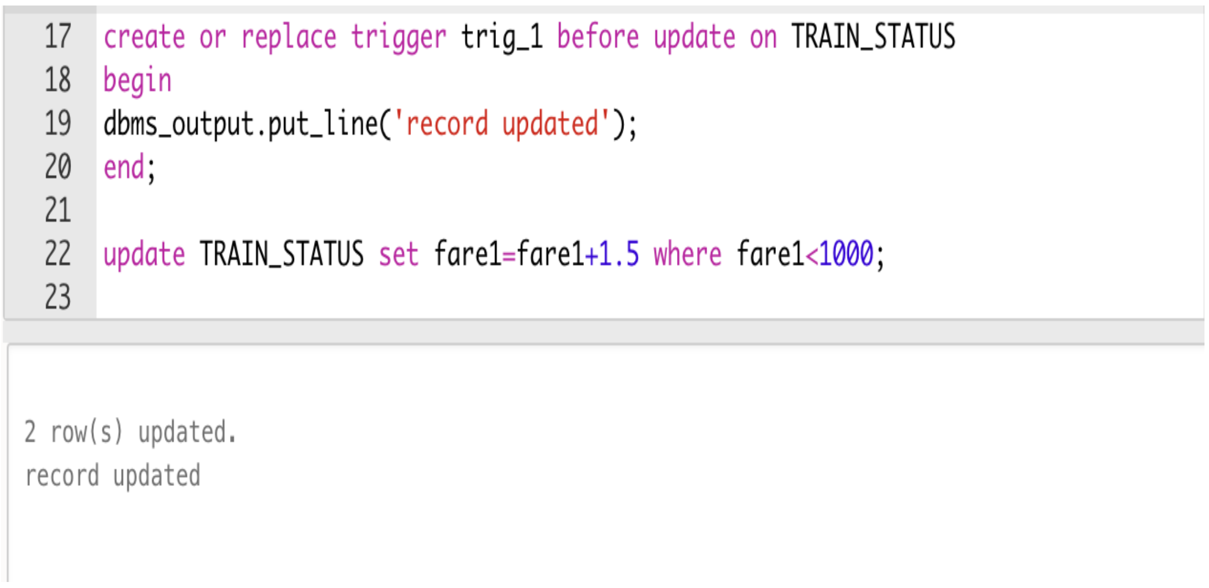
```
DECLARE
EE_id BOOKING.user_id%type:=1701;
EE_name  BOOKING.pname%type;
EE_addr BOOKING.age%type;
EE_status BOOKING.ticket_status%type;
exception_invalid exception;

BEGIN

if EE_id<0 then
    raise exception_invalid;
else
  select user_id,pname,age,ticket_status into EE_id,EE_Name,EE_addr,EE_status
  from(select * from BOOKING where ticket_price>3000)
  where user_id=EE_id;
dbms_output.put_line('With User Id '||EE_id ||' '||EE_name ||' who is '||EE_addr ||' years old has '||EE_status ||' ticket ');

end if;

EXCEPTION
WHEN exception_invalid then
    dbms_output.put_line('id value must be greater than zero');
When no_data_found then
    dbms_output.put_line('No entries available');
when others then
    dbms_output.put_line('internal error');
END;
```



```
Statement processed.
With User Id 1701 ramesh who is 45 years old has sleeperClass ticket
```

4.2.8 Trigger and update the value by adding GST on Fare1 and tigger the massge 'record updated'

```
create or replace trigger trig_1 before update on TRAIN_STATUS
begin
dbms_output.put_line('record updated');
end;

update TRAIN_STATUS set fare1=fare1+1.5 where fare1<1000;
```



```
create or replace trigger trig_1 before update on TRAIN_STATUS
begin
dbms_output.put_line('record updated');
end;

update TRAIN_STATUS set fare1=fare1+1.5 where fare1<1000;
```

```
2 row(s) updated.
record updated
```

4. Conclusion

In our project Railway reservation system we have stored all the information about the trains scheduled and the users booking tickets and even status of trains, seats etc. This data base is helpful for the applications which facilitate passengers to book the train tickets and check the details of trains and their status from their place itself it avoids inconveniences of going to railway station for each and every query they get. We had considered the most important requirements only, many more features and details cand be added to our project in order to obtain even more user friendly applications. These applications are already in progress and in future they can be upgraded and may become part of amazing technology.