CASE STUDY
OF
AIRLINE TICKET
RESERVATION
MANAGEMENT
SYSTEM
IN
RELATIONAL
DATABASE
DESIGN

CHITKARA
UNIVERSITY

CASE STUDY IN RELATIONAL DATABASE DESIGN

TITLE: AIRLINE TICKET RESERVATION MANAGEMENT SYSTEM

STUDENT’S NAME: KARAN AHUJA (2010990362)

ADISH JAIN (2010990028)

ADITI MAINI (2010990029)

AMIT GUPTA (2010990060)

GUIDE: Dr. SUSHEELA HOODA

ABSTRACT

One case study “Airline Ticket Reservation Management System” is presented. Input for this case study is taken from its informal specification to a relational schema using entity-relationship modeling and its translation to relational model, to database schema, to implementation of the database, to interactive SQL querying of the installed database (SQL/Oracle).Airline reservation System is a computerized system used to store and retrieve information and conduct transactions related to air travel. The project is aimed at exposing the relevance and importance of Airline Reservation Systems. It is projected towards enhancing the relationship between customers and airline agencies through the use of Airline Reservation System, and thereby making it convenient for the customers to book the flights as when they require such that they can utilize the software to make reservations.The software has two parts. First is user part and the administrator part. User part is used as a front end and administrator is the back end. Administrator is used by airline authority. It will allow the customers to access database and allow new customers to sign up for online access.The main purpose of the software is to reduce the manual errorsinvolved in the airline reservation process and make it convenient for the customers to book the flights as when they require such that they can utilize this software to make reservations, modify reservations or cancel a particular reservation.

ACKNOWLEDGEMENTS

I would like to express my gratitude to all of those who made it possible to complete this thesis, in particular to my Teacher Dr.Susheela Hooda. I would also like to thank my family for their understanding and continuous support.

CONTENTS

Chapter 1: INTRODUCTION

Database Management System:

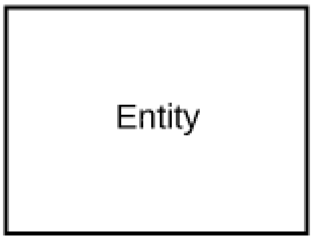
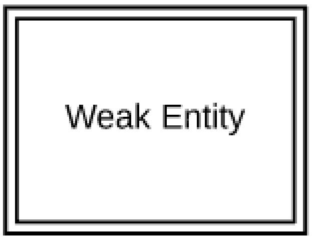
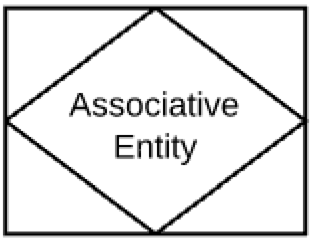
Database Management Systems (DBMS) are software systems used to store, retrieve, and run queries on data. A DBMS serves as an interface between an user and a database, allowing users to create, read, update, and delete data in the database. DBMS manages the data, and the database schema, allowing for data to be manipulated or extracted by users and other programs. This helps provide data security, data integrity, concurrency, and uniform data administrative procedures. DBMS optimizes the organization of data by following a database schema design technique called normalization, which splits a large table into smaller tables when any of its attributes have redundancy in values. DBMS offers many benefits over traditional file systems, including flexibility and a more complex backup system. Database management systems can be classified based on a variety of criteria such as the data model, the database distribution, or user numbers. The most widely used types of DBMS software are relational, distributed, hierarchical, and network. For Example, MySQL, Oracle etc. are popular commercial DBMS used in different applications. Relational database management systems (RDBMS) are the most popular data model because of its user-friendly interface. It is based on normalizing data in the rows and columns of the tables. This is a viable option when you need a data storage system that is scalable, flexible, and able to manage lots of information.

Relational Database Management System:

Relational database management systems (RDBMS) are the most popular data model because of its user-friendly interface. A relational database management system (RDBMS) is a collection of programs and properties that enables us and others to create, update, manages and interact with a relational database. RDBMS store data in the form of tables, with most commercial relational database management systems using Structured Query Language (SQL) to access the database. However, since SQL was invented after the initial development of the relational model, it is not necessary for RDBMS use. The RDBMS is the most popular database system among organizations across the world. It provides a dependable method of storing and retrieving large amounts of data while offering a combination of system performance , ease of implementation and better than the basic File system. An RDBMS is a type of database management system (DBMS) that stores data in a row-based table structure which connects related data elements. An RDBMS includes functions that maintain the security, accuracy, integrity and consistency of the data. This is different than the file storage used in a DBMS. In a DBMS, data is kept in a hierarchical form, whereas an RDBMS utilizes a table where the headers are used as column names and the rows contain the corresponding values.

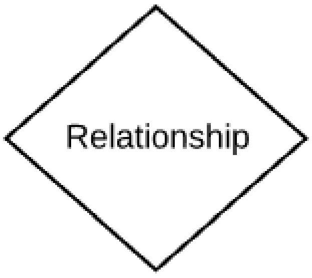
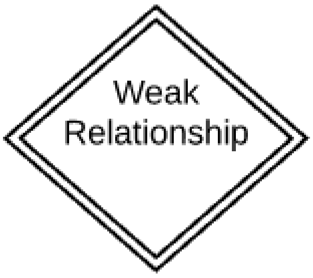
ER Diagram:

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design relational databases in the fields of database management, business information systems, education and research. Also known as ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

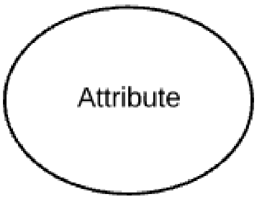
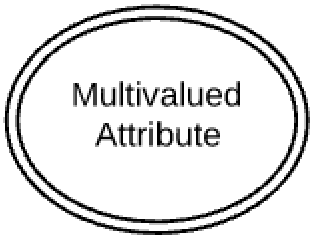
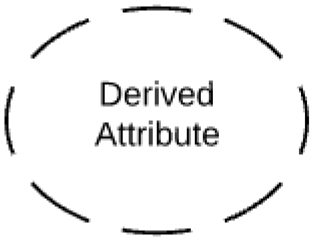
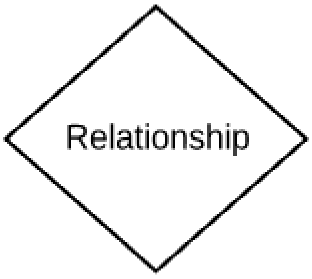
Entity Symbol	Name	Description
	Strong entity	These shapes are independent from other entities, and are often called parent entities, since they will often have weak entities that depend on them. They will also have a primary key, distinguishing each occurrence of the entity.
	Weak entity	Weak entities depend on some other entity type. They don't have primary keys, and have no meaning in the diagram without their parent entity.
	Associative entity	Associative entities relate the instances of several entity types. They also contain attributes specific to the relationship between those entity instances.

ERD relationship symbols

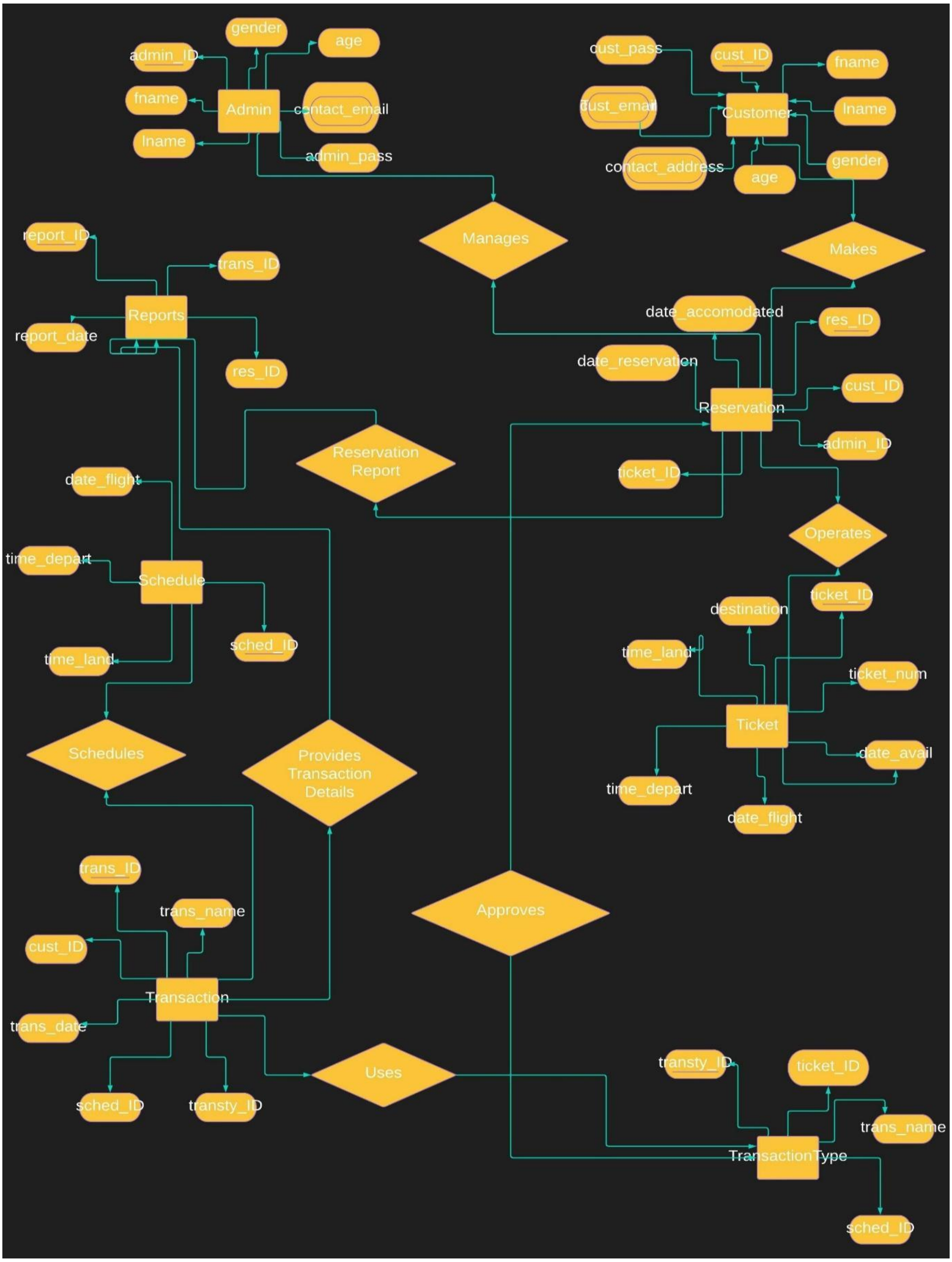
Within entity-relationship diagrams, relationships are used to document the interaction between two entities. Relationships are usually verbs such as assign, associate, or track and provide useful information that could not be discerned with just the entity types.

Relationship Symbol	Name	Description
	Relationship	Relationships are associations between or among entities.
	Weak relationship	Weak Relationships are connections between a weak entity and its owner.

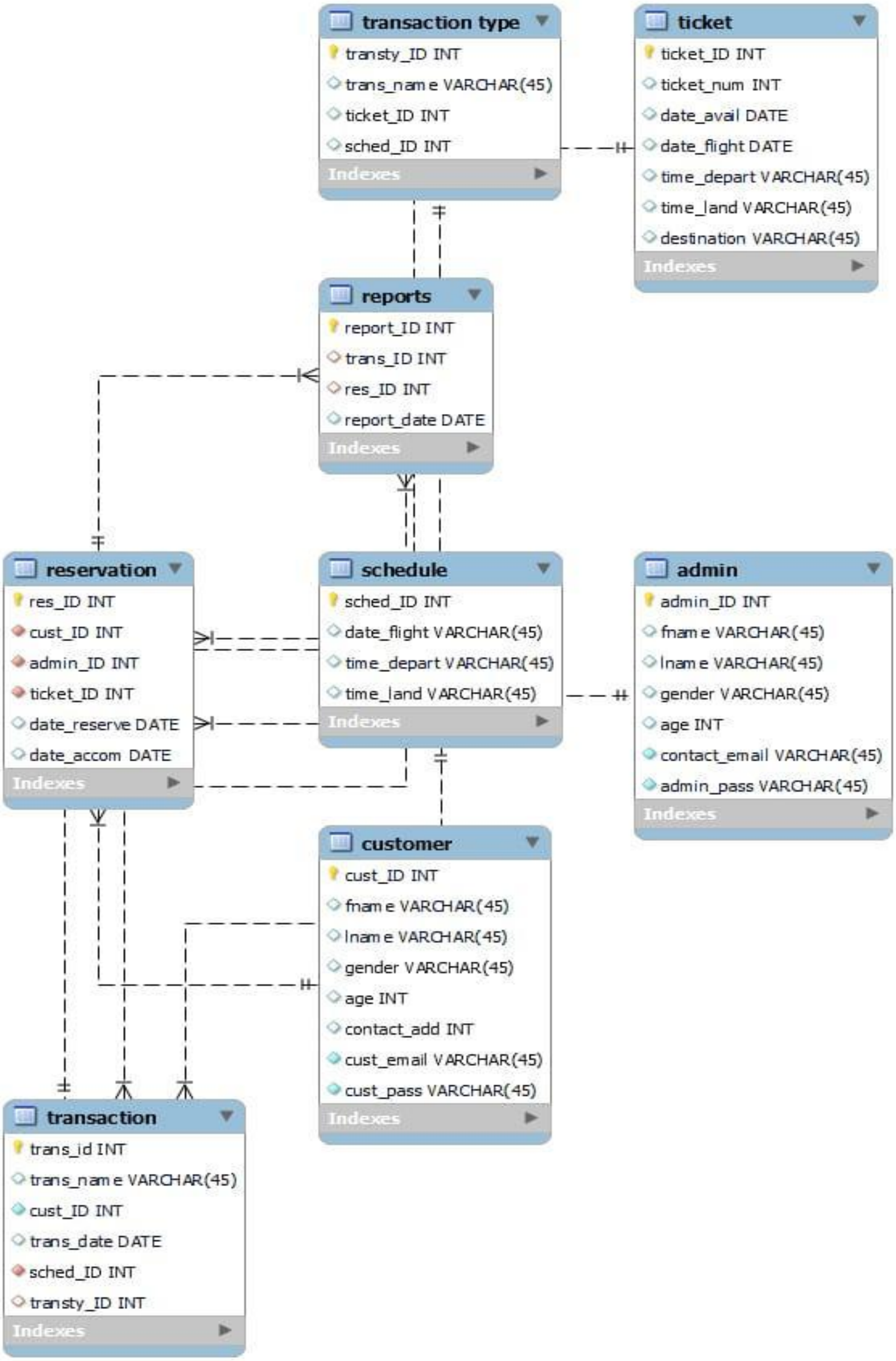
ERD attributes are characteristics of the entity that help users to better understand the database. Attributes are included to include details of the various entities that are highlighted in a conceptual ER diagram.

Attribute Symbol	Name	Description
	Attribute	Attributes are characteristics of an entity, a many-to-many relationship, or a one-to-one relationship.
	Multivalued attribute	Multivalued attributes are those that are can take on more than one value.
	Derived attribute	Derived attributes are attributes whose value can be calculated from related attribute values.
	Relationship	Relationships are associations between or among entities.

LOGICAL ER DIAGRAM:



PHYSICAL ER DIAGRAM:

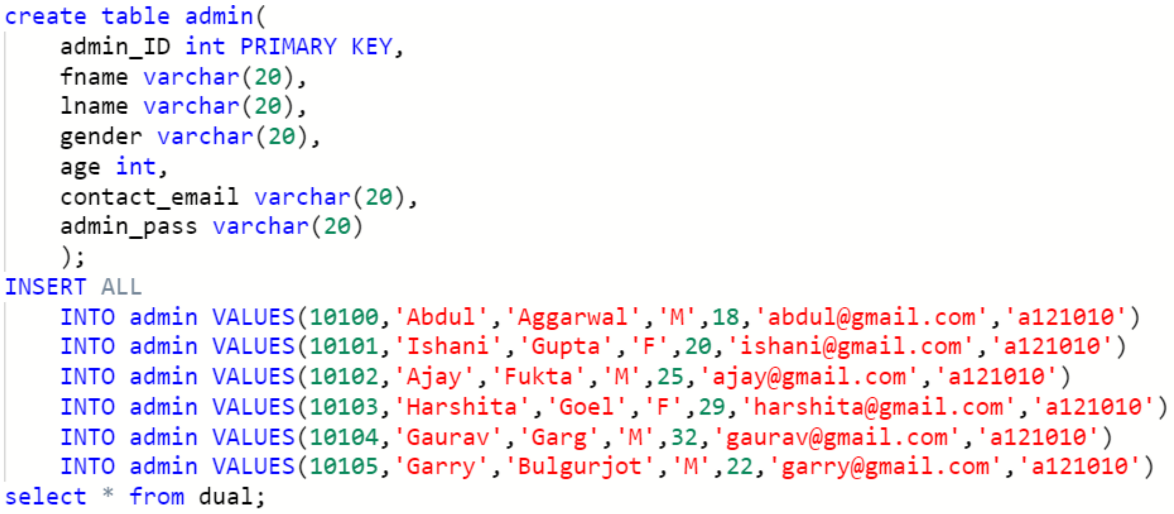


Brief Introduction:

Remember the days when to buy tickets for a vacation, you had to call up ticket agents, find the airlines, search for deals, and so on? Well, since the Internet took off, things haven't been the same. Now you have many Web sites from both third-party vendors and airlines offering anything from cars to cruises to vacation packages, with deal comparisons and what have you. Clearly, the consumer is the winner! As a part of your learning experience using Web Logic Server Internet- and Web-enabled applications have revolutionized the way businesses are carried out., We build a simple airline ticket booking system, which will model for a real-world. We'll begin by briefly outlining the features that will be provided by your airline ticket booking system. This Airline reservation helps users and the admin to access the details as they needed, as we the user wants the each and every detail of the our flight and all other things so this database will provide us with the same For example, the customer table will provide each and every necessary detail of customer and Ticket table will provide the details of the tickets and all other things related to it. For the transaction and billing problems to be sorted we have different tables such as Transaction and Transaction type table this will provide the details of each transaction and its type and all the details associated with it and the all type of data is managed by the admin and the report is created of each type of things happening on the web application. It contains information on schedules and fares and contains a database of reservations (or passenger name records) and of tickets issued. Airlines use databases for storing passenger information when tickets are booked and when passengers check-in at the airport.

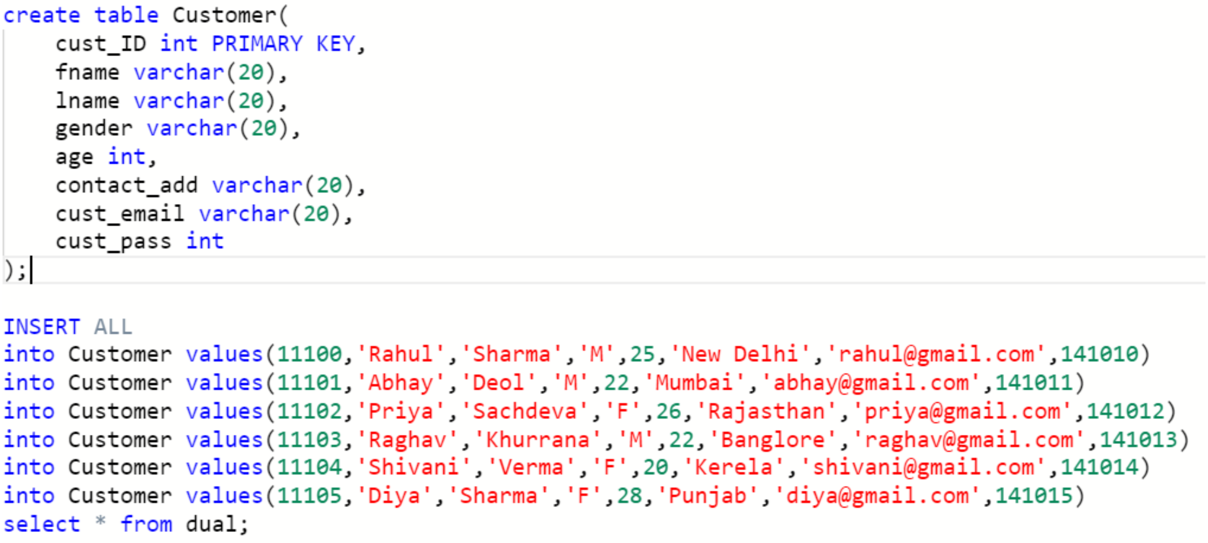
STRUCTURE OF CASE STUDY:

Admin table: All the details of the admin and the admin_ID is the primary key through which it can access all the details.



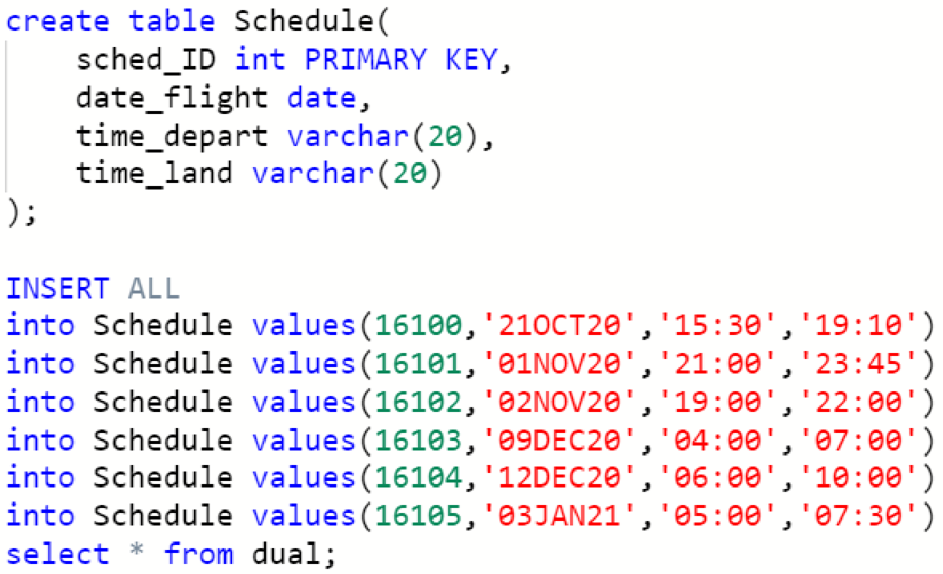
```
create table admin(
    admin_ID int PRIMARY KEY,
    fname varchar(20),
    lname varchar(20),
    gender varchar(20),
    age int,
    contact_email varchar(20),
    admin_pass varchar(20)
    );
INSERT ALL
    INTO admin VALUES(10100,'Abdul','Aggarwal','M',18,'abdul@gmail.com','a121010')
    INTO admin VALUES(10101,'Ishani','Gupta','F',20,'ishani@gmail.com','a121010')
    INTO admin VALUES(10102,'Ajay','Fukta','M',25,'ajay@gmail.com','a121010')
    INTO admin VALUES(10103,'Harshita','Goel','F',29,'harshita@gmail.com','a121010')
    INTO admin VALUES(10104,'Gaurav','Garg','M',32,'gaurav@gmail.com','a121010')
    INTO admin VALUES(10105,'Garry','Bulgurjot','M',22,'garry@gmail.com','a121010')
select * from dual;
```

Customer table: All the details of the customer in which cust_ID as the primary key is used to access all the customer details.



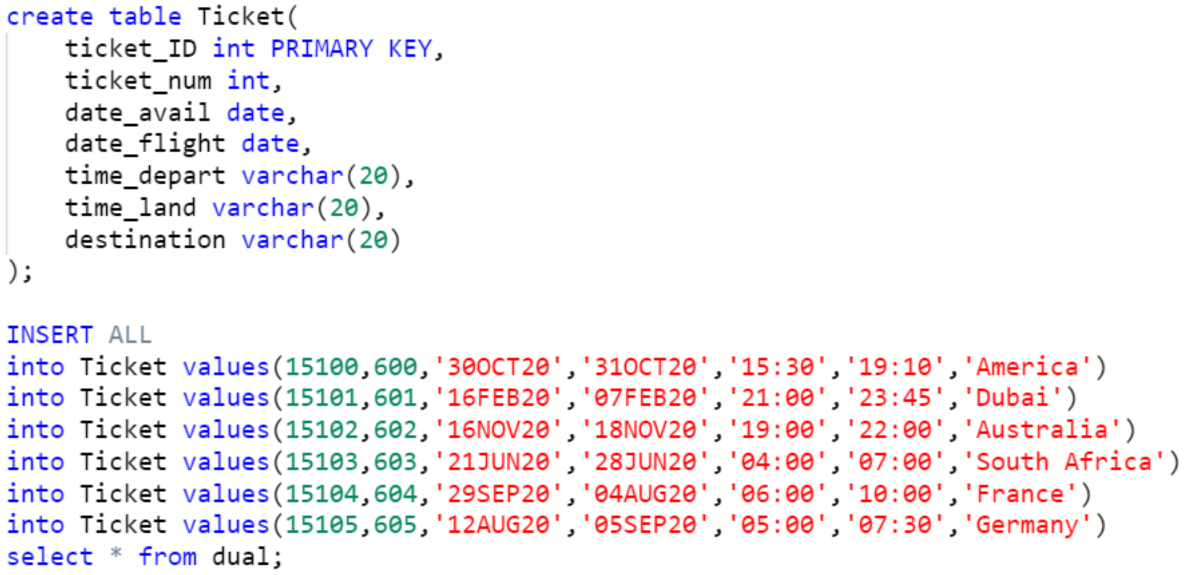
```
create table Customer(
    cust_ID int PRIMARY KEY,
    fname varchar(20),
    lname varchar(20),
    gender varchar(20),
    age int,
    contact_add varchar(20),
    cust_email varchar(20),
    cust_pass int
);

INSERT ALL
into Customer values(11100,'Rahul','Sharma','M',25,'New Delhi','rahul@gmail.com',141010)
into Customer values(11101,'Abhay','Deol','M',22,'Mumbai','abhay@gmail.com',141011)
into Customer values(11102,'Priya','Sachdeva','F',26,'Rajasthan','priya@gmail.com',141012)
into Customer values(11103,'Raghav','Khurrana','M',22,'Banglore','raghav@gmail.com',141013)
into Customer values(11104,'Shivani','Verma','F',20,'Kerela','shivani@gmail.com',141014)
into Customer values(11105,'Diya','Sharma','F',28,'Punjab','diya@gmail.com',141015)
select * from dual;
```

Schedule table: It contains all the schedules of the flight in which we used sched_ID as primary key to get all the details.



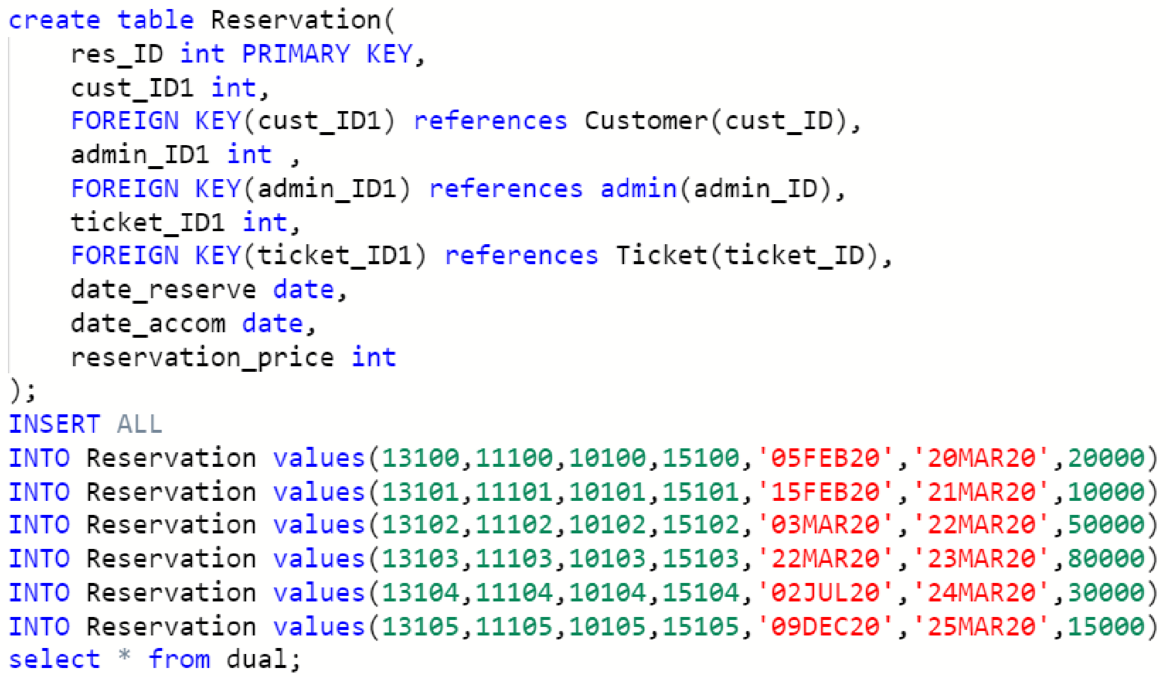
```
create table Schedule(
    sched_ID int PRIMARY KEY,
    date_flight date,
    time_depart varchar(20),
    time_land varchar(20)
);

INSERT ALL
into Schedule values(16100,'21OCT20','15:30','19:10')
into Schedule values(16101,'01NOV20','21:00','23:45')
into Schedule values(16102,'02NOV20','19:00','22:00')
into Schedule values(16103,'09DEC20','04:00','07:00')
into Schedule values(16104,'12DEC20','06:00','10:00')
into Schedule values(16105,'03JAN21','05:00','07:30')
select * from dual;
```

Ticket table: It tells all the ticket details and we used ticket_ID as primary key to fetch all the details on the particular ticket



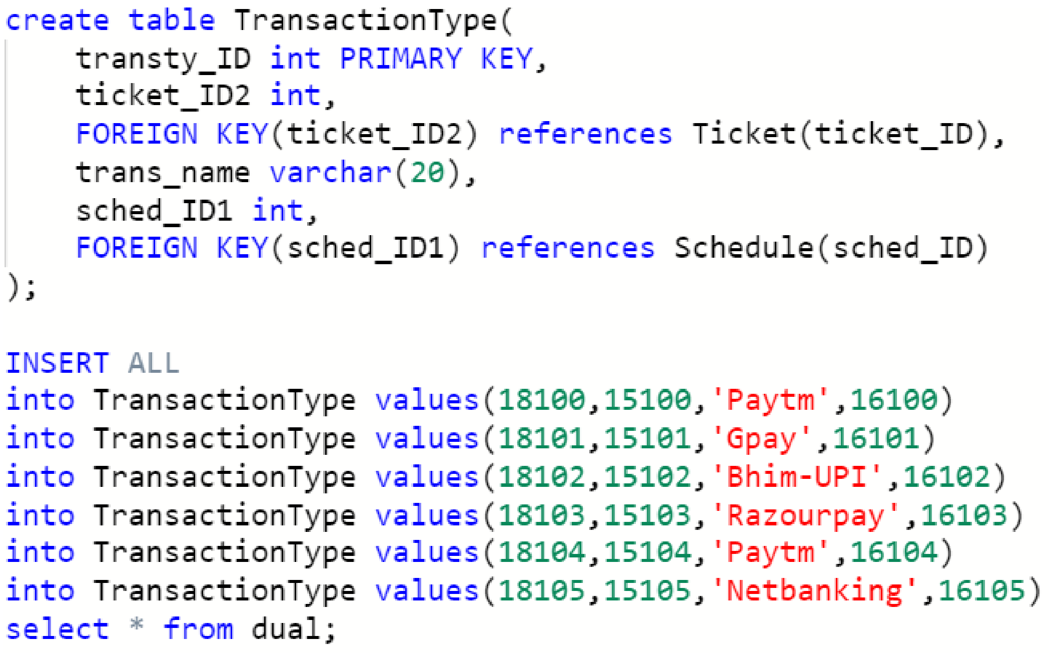
```
create table Ticket(
    ticket_ID int PRIMARY KEY,
    ticket_num int,
    date_avail date,
    date_flight date,
    time_depart varchar(20),
    time_land varchar(20),
    destination varchar(20)
);

INSERT ALL
into Ticket values(15100,600,'30OCT20','31OCT20','15:30','19:10','America')
into Ticket values(15101,601,'16FEB20','07FEB20','21:00','23:45','Dubai')
into Ticket values(15102,602,'16NOV20','18NOV20','19:00','22:00','Australia')
into Ticket values(15103,603,'21JUN20','28JUN20','04:00','07:00','South Africa')
into Ticket values(15104,604,'29SEP20','04AUG20','06:00','10:00','France')
into Ticket values(15105,605,'12AUG20','05SEP20','05:00','07:30','Germany')
select * from dual;
```

Reservation table: It tells about the reservations made by the customer by using the res_ID we can fetch all the reservation details.



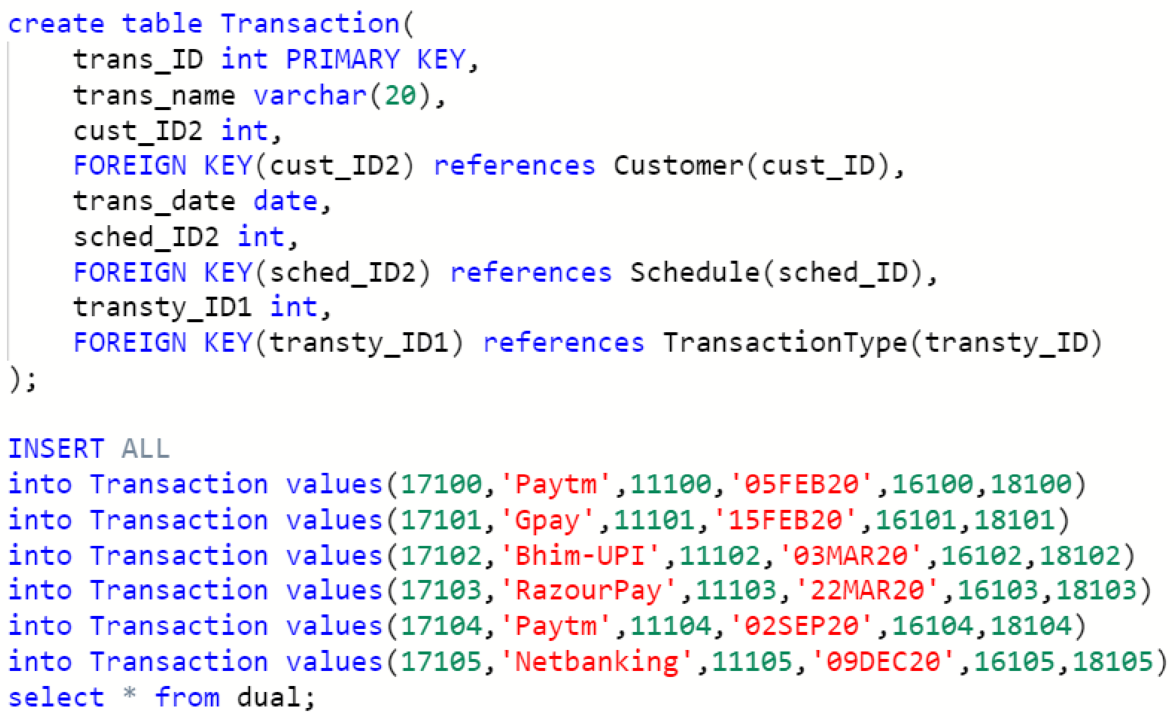
```
create table Reservation(
    res_ID int PRIMARY KEY,
    cust_ID1 int,
    FOREIGN KEY(cust_ID1) references Customer(cust_ID),
    admin_ID1 int ,
    FOREIGN KEY(admin_ID1) references admin(admin_ID),
    ticket_ID1 int,
    FOREIGN KEY(ticket_ID1) references Ticket(ticket_ID),
    date_reserve date,
    date_accom date,
    reservation_price int
);
INSERT ALL
INTO Reservation values(13100,11100,10100,15100,'05FEB20','20MAR20',20000)
INTO Reservation values(13101,11101,10101,15101,'15FEB20','21MAR20',10000)
INTO Reservation values(13102,11102,10102,15102,'03MAR20','22MAR20',50000)
INTO Reservation values(13103,11103,10103,15103,'22MAR20','23MAR20',80000)
INTO Reservation values(13104,11104,10104,15104,'02JUL20','24MAR20',30000)
INTO Reservation values(13105,11105,10105,15105,'09DEC20','25MAR20',15000)
select * from dual;
```

Transaction Type table: It tells about the type of the transaction made for the reserving the ticket and we can use transty_ID as primary key to get transaction type details.



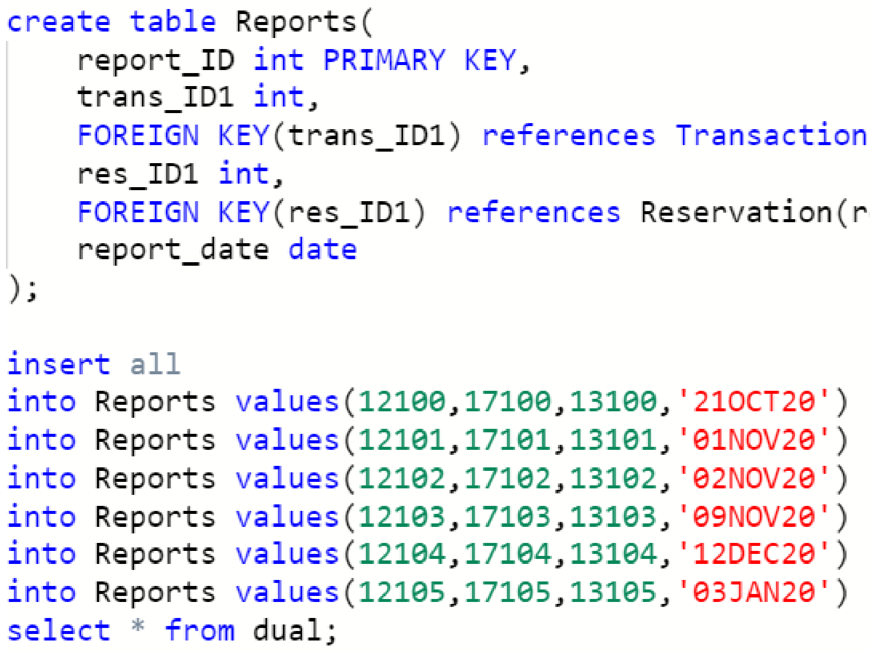
```
create table TransactionType(
    transty_ID int PRIMARY KEY,
    ticket_ID2 int,
    FOREIGN KEY(ticket_ID2) references Ticket(ticket_ID),
    trans_name varchar(20),
    sched_ID1 int,
    FOREIGN KEY(sched_ID1) references Schedule(sched_ID)
);

INSERT ALL
into TransactionType values(18100,15100,'Paytm',16100)
into TransactionType values(18101,15101,'Gpay',16101)
into TransactionType values(18102,15102,'Bhim-UPI',16102)
into TransactionType values(18103,15103,'Razourpay',16103)
into TransactionType values(18104,15104,'Paytm',16104)
into TransactionType values(18105,15105,'Netbanking',16105)
select * from dual;
```

Transaction table: It tells about the transaction made by using trans_ID as primary key to fetch transaction details.



```
create table Transaction(
    trans_ID int PRIMARY KEY,
    trans_name varchar(20),
    cust_ID2 int,
    FOREIGN KEY(cust_ID2) references Customer(cust_ID),
    trans_date date,
    sched_ID2 int,
    FOREIGN KEY(sched_ID2) references Schedule(sched_ID),
    transty_ID1 int,
    FOREIGN KEY(transty_ID1) references TransactionType(transty_ID)
);

INSERT ALL
into Transaction values(17100,'Paytm',11100,'05FEB20',16100,18100)
into Transaction values(17101,'Gpay',11101,'15FEB20',16101,18101)
into Transaction values(17102,'Bhim-UPI',11102,'03MAR20',16102,18102)
into Transaction values(17103,'RazourPay',11103,'22MAR20',16103,18103)
into Transaction values(17104,'Paytm',11104,'02SEP20',16104,18104)
into Transaction values(17105,'Netbanking',11105,'09DEC20',16105,18105)
select * from dual;
```

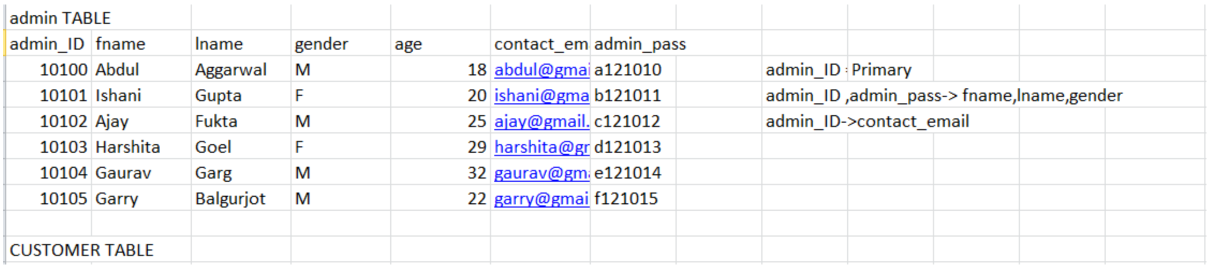
Reports table: It makes the reports for the reservation and transaction made by the admin .he uses report_ID to get all the report.



```
create table Reports(
    report_ID int PRIMARY KEY,
    trans_ID1 int,
    FOREIGN KEY(trans_ID1) references Transaction(trans_ID),
    res_ID1 int,
    FOREIGN KEY(res_ID1) references Reservation(res_ID),
    report_date date
);

insert all
into Reports values(12100,17100,13100,'21OCT20')
into Reports values(12101,17101,13101,'01NOV20')
into Reports values(12102,17102,13102,'02NOV20')
into Reports values(12103,17103,13103,'09NOV20')
into Reports values(12104,17104,13104,'12DEC20')
into Reports values(12105,17105,13105,'03JAN20')
select * from dual;
```

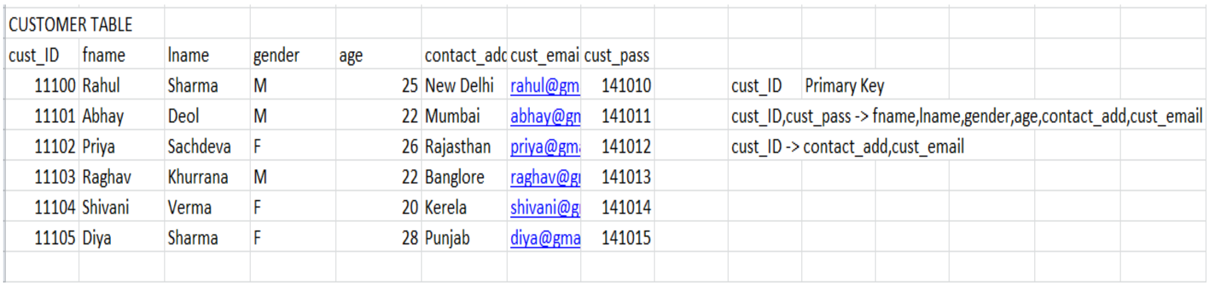
KEYS AND FUNCTIONAL DEPENDENCIES:

Admin table:



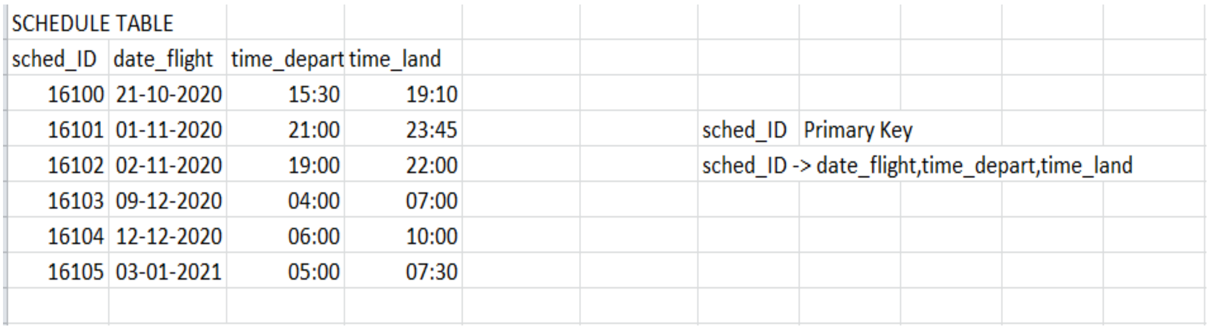
admin TABLE								
admin_ID	fname	lname	gender	age	contact_em	admin_pass		
10100	Abdul	Aggarwal	M	18	abdul@gmai	a121010	admin_ID	Primary
10101	Ishani	Gupta	F	20	ishani@gma	b121011	admin_ID ,admin_pass-> fname,lname,gender	
10102	Ajay	Fukta	M	25	ajay@gmail.	c121012	admin_ID->contact_email	
10103	Harshita	Goel	F	29	harshita@gr	d121013		
10104	Gaurav	Garg	M	32	gaurav@gm	e121014		
10105	Garry	Balgurjot	M	22	garry@gmai	f121015		
CUSTOMER TABLE								

Customer table:



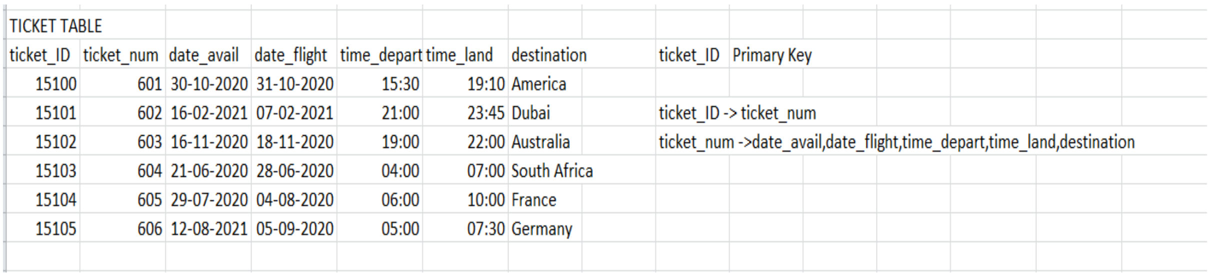
CUSTOMER TABLE									
cust_ID	fname	lname	gender	age	contact_adc	cust_emai	cust_pass		
11100	Rahul	Sharma	M	25	New Delhi	rahul@gm	141010	cust_ID	Primary Key
11101	Abhay	Deol	M	22	Mumbai	abhay@gn	141011	cust_ID,cust_pass -> fname,lname,gender,age,contact_add,cust_email	
11102	Priya	Sachdeva	F	26	Rajasthan	priya@gm	141012	cust_ID -> contact_add,cust_email	
11103	Raghav	Khurrana	M	22	Banglore	raghav@gı	141013		
11104	Shivani	Verma	F	20	Kerela	shivani@g	141014		
11105	Diya	Sharma	F	28	Punjab	diya@gma	141015		

Schedule table:



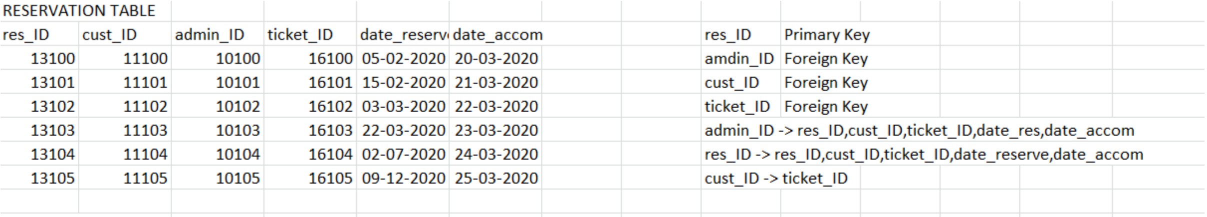
SCHEDULE TABLE					
sched_ID	date_flight	time_depart	time_land		
16100	21-10-2020	15:30	19:10		
16101	01-11-2020	21:00	23:45	sched_ID	Primary Key
16102	02-11-2020	19:00	22:00	sched_ID -> date_flight,time_depart,time_land	
16103	09-12-2020	04:00	07:00		
16104	12-12-2020	06:00	10:00		
16105	03-01-2021	05:00	07:30		

Ticket table:



TICKET TABLE								
ticket_ID	ticket_num	date_avail	date_flight	time_depart	time_land	destination	ticket_ID	Primary Key
15100	601	30-10-2020	31-10-2020	15:30	19:10	America		
15101	602	16-02-2021	07-02-2021	21:00	23:45	Dubai	ticket_ID -> ticket_num	
15102	603	16-11-2020	18-11-2020	19:00	22:00	Australia	ticket_num ->date_avail,date_flight,time_depart,time_land,destination	
15103	604	21-06-2020	28-06-2020	04:00	07:00	South Africa		
15104	605	29-07-2020	04-08-2020	06:00	10:00	France		
15105	606	12-08-2021	05-09-2020	05:00	07:30	Germany		

Reservation table:



RESERVATION TABLE

res_ID	cust_ID	admin_ID	ticket_ID	date_reserv	date_accom
13100	11100	10100	16100	05-02-2020	20-03-2020
13101	11101	10101	16101	15-02-2020	21-03-2020
13102	11102	10102	16102	03-03-2020	22-03-2020
13103	11103	10103	16103	22-03-2020	23-03-2020
13104	11104	10104	16104	02-07-2020	24-03-2020
13105	11105	10105	16105	09-12-2020	25-03-2020

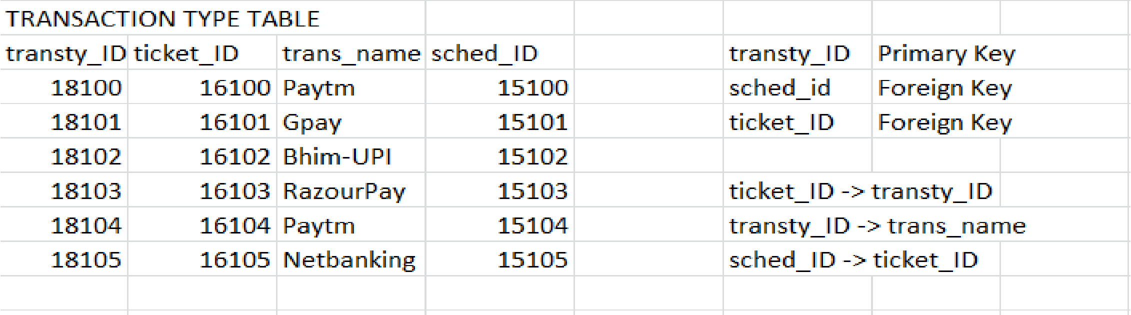
res_ID	Primary Key
amdin_ID	Foreign Key
cust_ID	Foreign Key
ticket_ID	Foreign Key

admin_ID -> res_ID,cust_ID,ticket_ID,date_res,date_accom

res_ID -> res_ID,cust_ID,ticket_ID,date_reserve,date_accom

cust_ID -> ticket_ID

Transaction Type table:



TRANSACTION TYPE TABLE

transty_ID	ticket_ID	trans_name	sched_ID
18100	16100	Paytm	15100
18101	16101	Gpay	15101
18102	16102	Bhim-UPI	15102
18103	16103	RazourPay	15103
18104	16104	Paytm	15104
18105	16105	Netbanking	15105

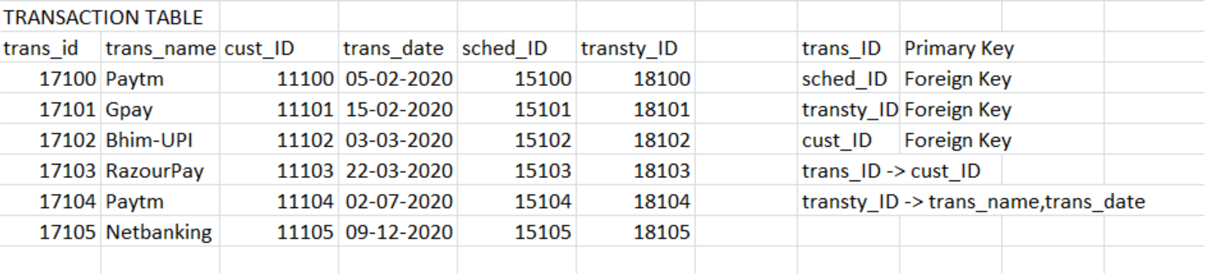
transty_ID	Primary Key
sched_id	Foreign Key
ticket_ID	Foreign Key

ticket_ID -> transty_ID

transty_ID -> trans_name

sched_ID -> ticket_ID

Transaction table:



TRANSACTION TABLE

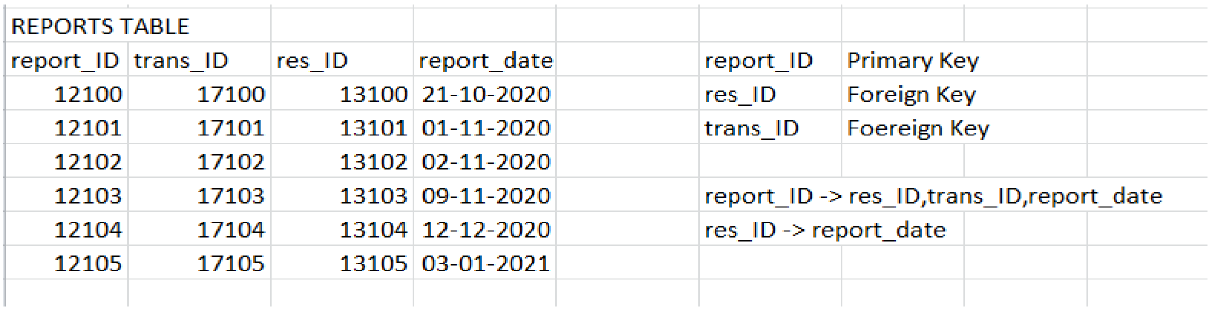
trans_id	trans_name	cust_ID	trans_date	sched_ID	transty_ID
17100	Paytm	11100	05-02-2020	15100	18100
17101	Gpay	11101	15-02-2020	15101	18101
17102	Bhim-UPI	11102	03-03-2020	15102	18102
17103	RazourPay	11103	22-03-2020	15103	18103
17104	Paytm	11104	02-07-2020	15104	18104
17105	Netbanking	11105	09-12-2020	15105	18105

trans_ID	Primary Key
sched_ID	Foreign Key
transty_ID	Foreign Key
cust_ID	Foreign Key

trans_ID -> cust_ID

transty_ID -> trans_name,trans_date

Reports table:



REPORTS TABLE

report_ID	trans_ID	res_ID	report_date
12100	17100	13100	21-10-2020
12101	17101	13101	01-11-2020
12102	17102	13102	02-11-2020
12103	17103	13103	09-11-2020
12104	17104	13104	12-12-2020
12105	17105	13105	03-01-2021

report_ID	Primary Key
res_ID	Foreign Key
trans_ID	Foereign Key

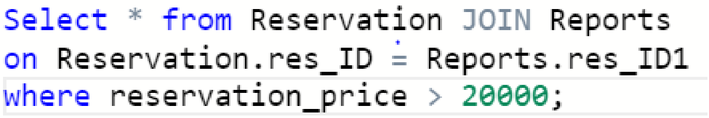
report_ID -> res_ID,trans_ID,report_date

res_ID -> report_date

INTERACTIVE QUERIES:

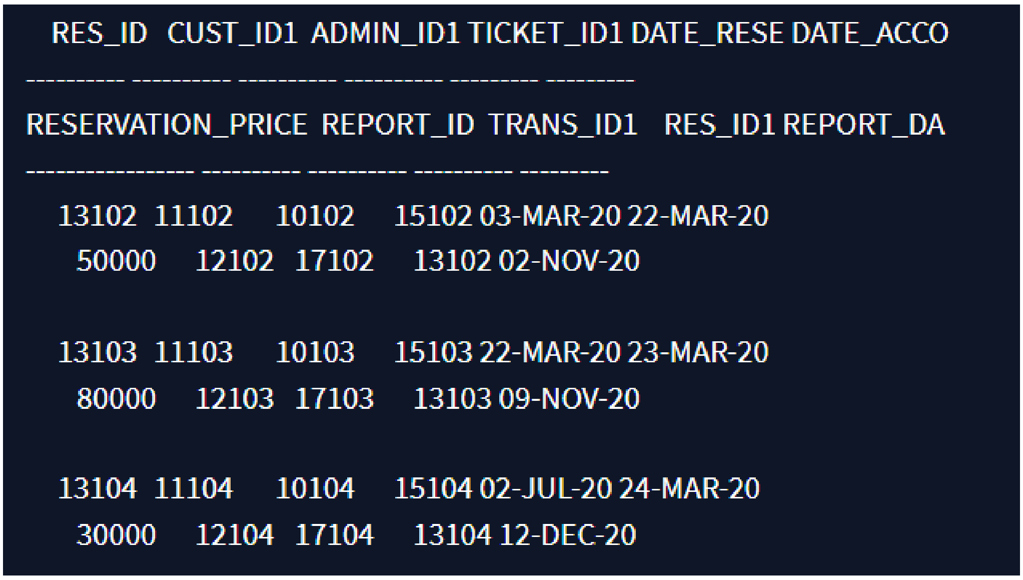
QUERY1:

To get the report where reservation price is greater than 20000



```
Select * from Reservation JOIN Reports
on Reservation.res_ID = Reports.res_ID1
where reservation_price > 20000;
```

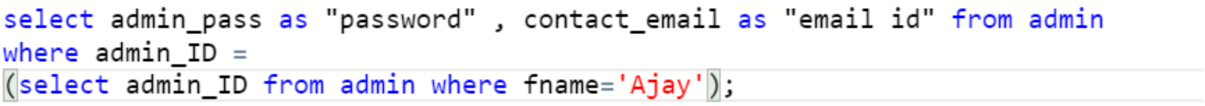
OUTPUT:



```
  RES_ID  CUST_ID1  ADMIN_ID1 TICKET_ID1 DATE_RESE DATE_ACCO
---------- ---------- ---------- ---------- --------- ---------
RESERVATION_PRICE  REPORT_ID  TRANS_ID1   RES_ID1 REPORT_DA
----------------- ---------- ---------- ---------- ---------
    13102  11102      10102      15102 03-MAR-20 22-MAR-20
      50000      12102    17102      13102 02-NOV-20

    13103  11103      10103      15103 22-MAR-20 23-MAR-20
      80000      12103    17103      13103 09-NOV-20

    13104  11104      10104      15104 02-JUL-20 24-MAR-20
      30000      12104    17104      13104 12-DEC-20
```

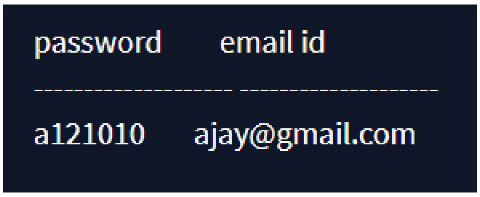
QUERY 2:

To get the email id and password of admin Ajay



```
select admin_pass as "password" , contact_email as "email id" from admin
where admin_ID =
(select admin_ID from admin where fname='Ajay');
```

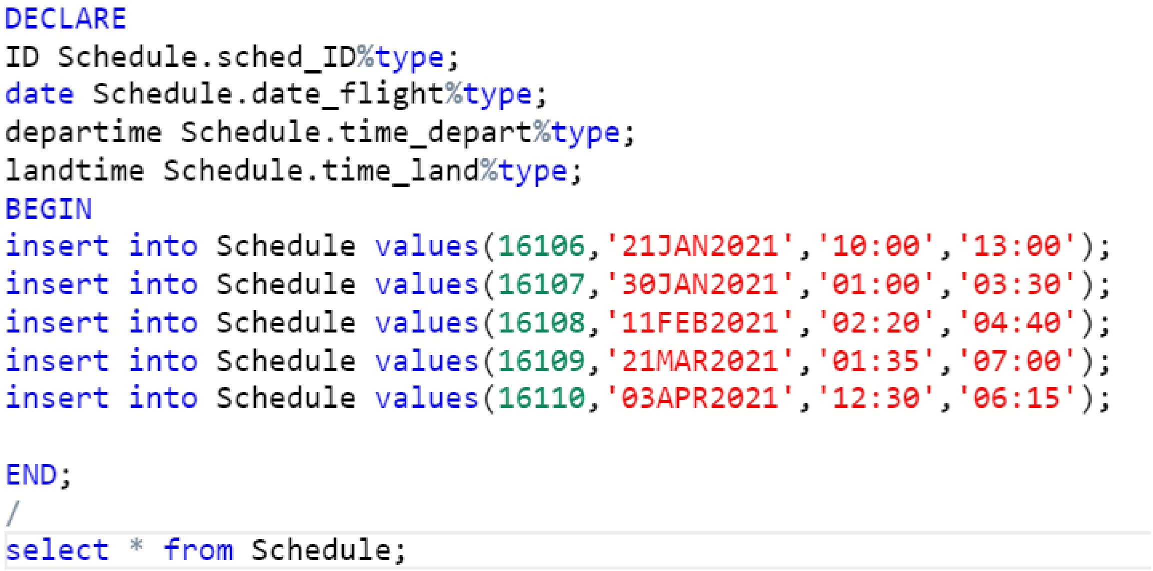
Output:



```
password      email id
-------------------- --------------------
a121010     ajay@gmail.com
```

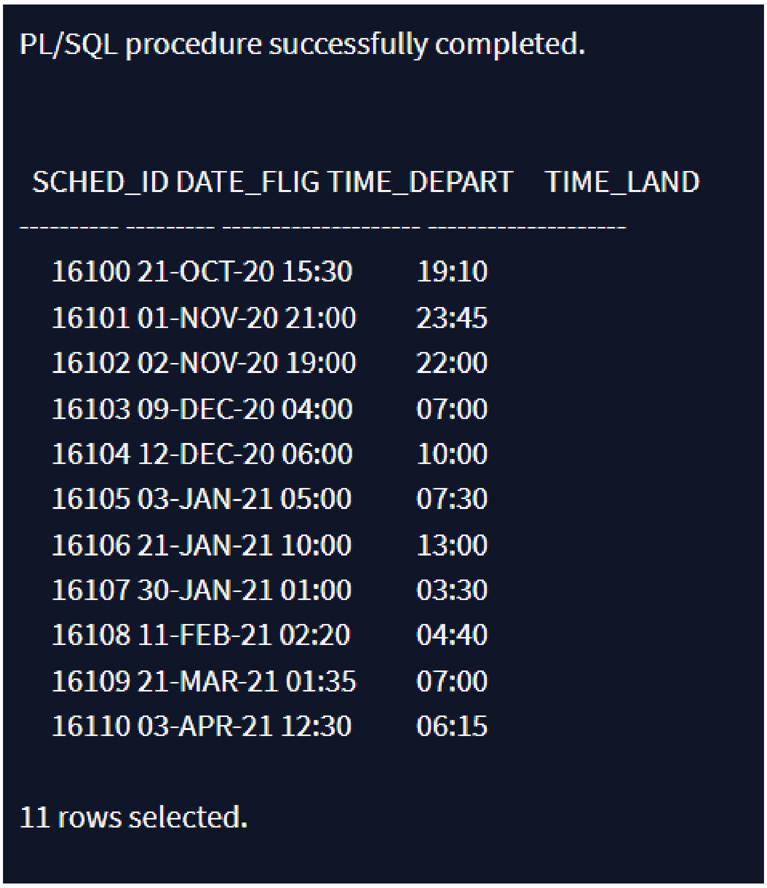
QUERY 3:

To insert more flight schedule using PL/SQL queries in Schedule table.



```
DECLARE
ID Schedule.sched_ID%type;
date Schedule.date_flight%type;
departime Schedule.time_depart%type;
landtime Schedule.time_land%type;
BEGIN
insert into Schedule values(16106,'21JAN2021','10:00','13:00');
insert into Schedule values(16107,'30JAN2021','01:00','03:30');
insert into Schedule values(16108,'11FEB2021','02:20','04:40');
insert into Schedule values(16109,'21MAR2021','01:35','07:00');
insert into Schedule values(16110,'03APR2021','12:30','06:15');

END;
/
select * from Schedule;
```

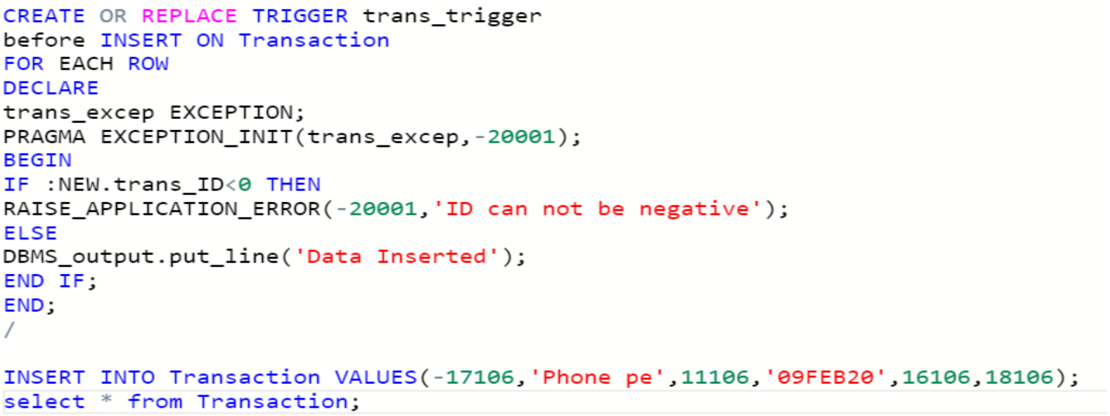
Output:



```
PL/SQL procedure successfully completed.


 SCHED_ID DATE_FLIG TIME_DEPART    TIME_LAND
---------- --------- -------------------- --------------------
    16100 21-OCT-20 15:30        19:10
    16101 01-NOV-20 21:00        23:45
    16102 02-NOV-20 19:00        22:00
    16103 09-DEC-20 04:00        07:00
    16104 12-DEC-20 06:00        10:00
    16105 03-JAN-21 05:00        07:30
    16106 21-JAN-21 10:00        13:00
    16107 30-JAN-21 01:00        03:30
    16108 11-FEB-21 02:20        04:40
    16109 21-MAR-21 01:35        07:00
    16110 03-APR-21 12:30        06:15

11 rows selected.
```

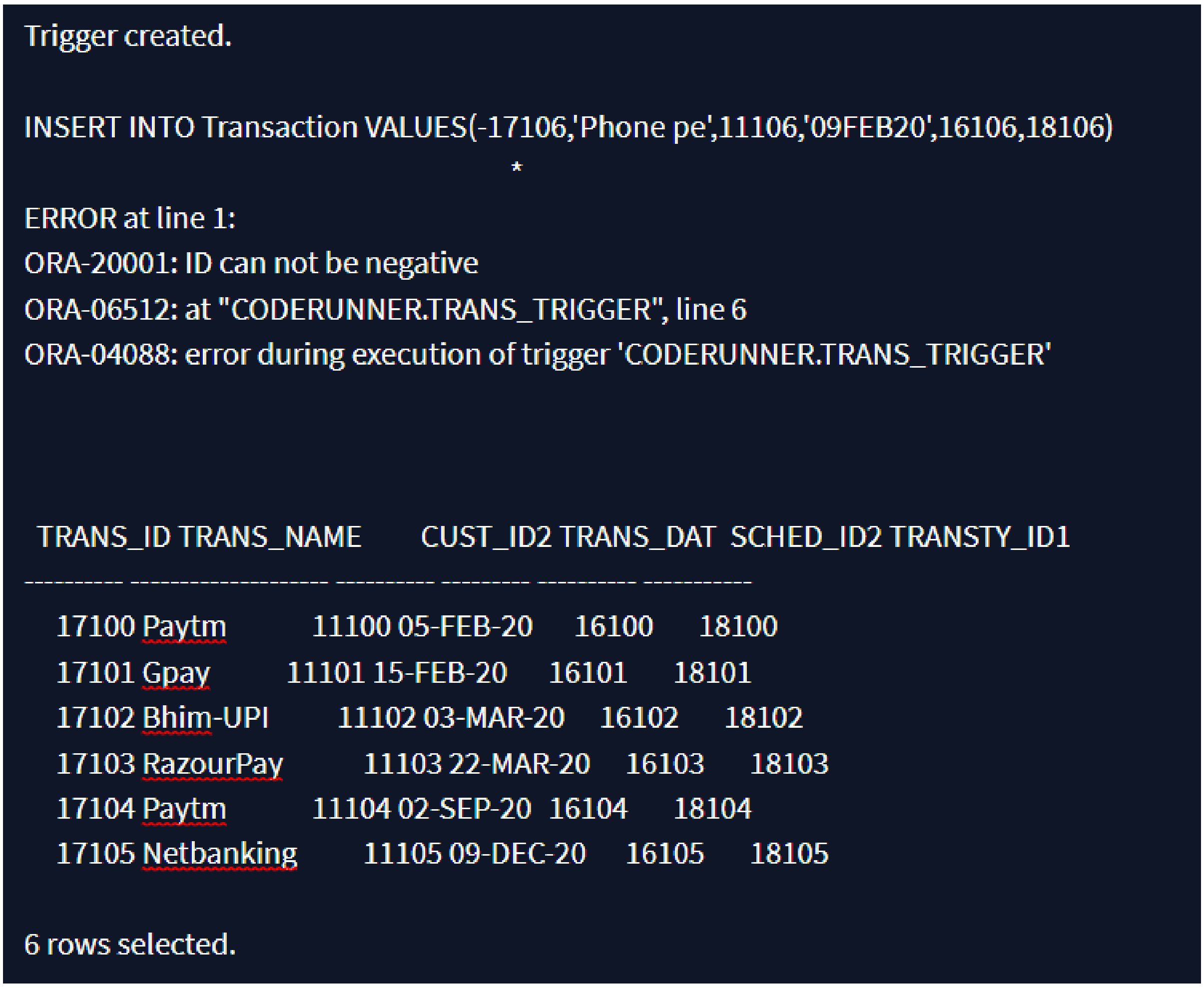
QUERY 4:

To create a trigger for transaction table where trans_ID can’t br negative



```
CREATE OR REPLACE TRIGGER trans_trigger
before INSERT ON Transaction
FOR EACH ROW
DECLARE
trans_excep EXCEPTION;
PRAGMA EXCEPTION_INIT(trans_excep,-20001);
BEGIN
IF :NEW.trans_ID<0 THEN
RAISE_APPLICATION_ERROR(-20001,'ID can not be negative');
ELSE
DBMS_output.put_line('Data Inserted');
END IF;
END;
/

INSERT INTO Transaction VALUES(-17106,'Phone pe',11106,'09FEB20',16106,18106);
select * from Transaction;
```

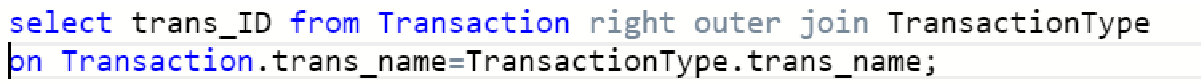
Output:



```
Trigger created.

INSERT INTO Transaction VALUES(-17106,'Phone pe',11106,'09FEB20',16106,18106)
                                                  *

ERROR at line 1:
ORA-20001: ID can not be negative
ORA-06512: at "CODERUNNER.TRANS_TRIGGER", line 6
ORA-04088: error during execution of trigger 'CODERUNNER.TRANS_TRIGGER'



 TRANS_ID TRANS_NAME       CUST_ID2 TRANS_DAT  SCHED_ID2 TRANSTY_ID1
---------- -------------------- ---------- --------- ---------- -----------
    17100 Paytm            11100 05-FEB-20     16100      18100
    17101 Gpay             11101 15-FEB-20     16101      18101
    17102 Bhim-UPI         11102 03-MAR-20     16102      18102
    17103 RazourPay        11103 22-MAR-20     16103      18103
    17104 Paytm            11104 02-SEP-20     16104      18104
    17105 Netbanking       11105 09-DEC-20     16105      18105

6 rows selected.
```

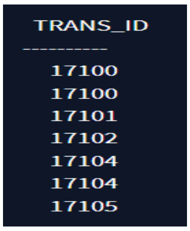
QUERY 5:

To get trans_ID from where transaction name are equal in both transaction table and transaction type table



```
select trans_ID from Transaction right outer join TransactionType
on Transaction.trans_name=TransactionType.trans_name;
```

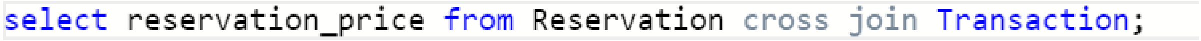
Output:



```
 TRANS_ID
----------
    17100
    17100
    17101
    17102
    17104
    17104
    17105
```

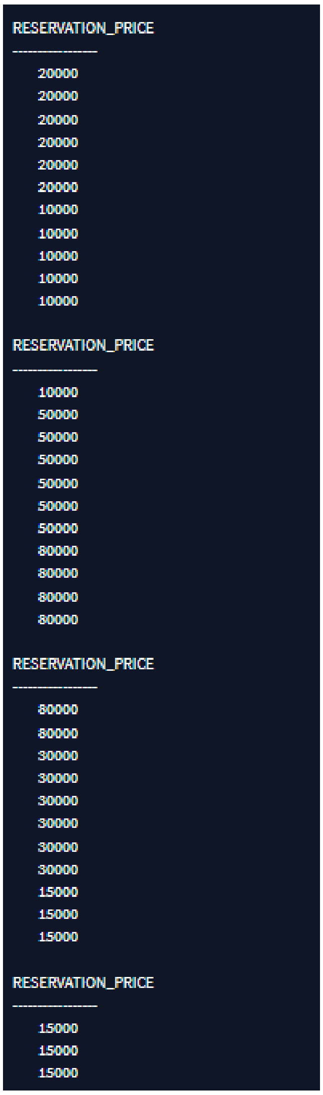
Query 6:

To get reservation price from transactions made



```
select reservation_price from Reservation cross join Transaction;
```

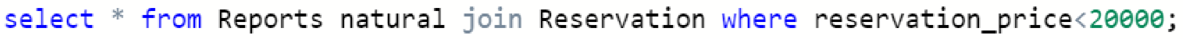
Output:



```
RESERVATION_PRICE
-----------------
    20000
    20000
    20000
    20000
    20000
    20000
    10000
    10000
    10000
    10000
    10000

RESERVATION_PRICE
-----------------
    10000
    50000
    50000
    50000
    50000
    50000
    50000
    80000
    80000
    80000
    80000

RESERVATION_PRICE
-----------------
    80000
    80000
    30000
    30000
    30000
    30000
    30000
    30000
    15000
    15000
    15000

RESERVATION_PRICE
-----------------
    15000
    15000
    15000
```

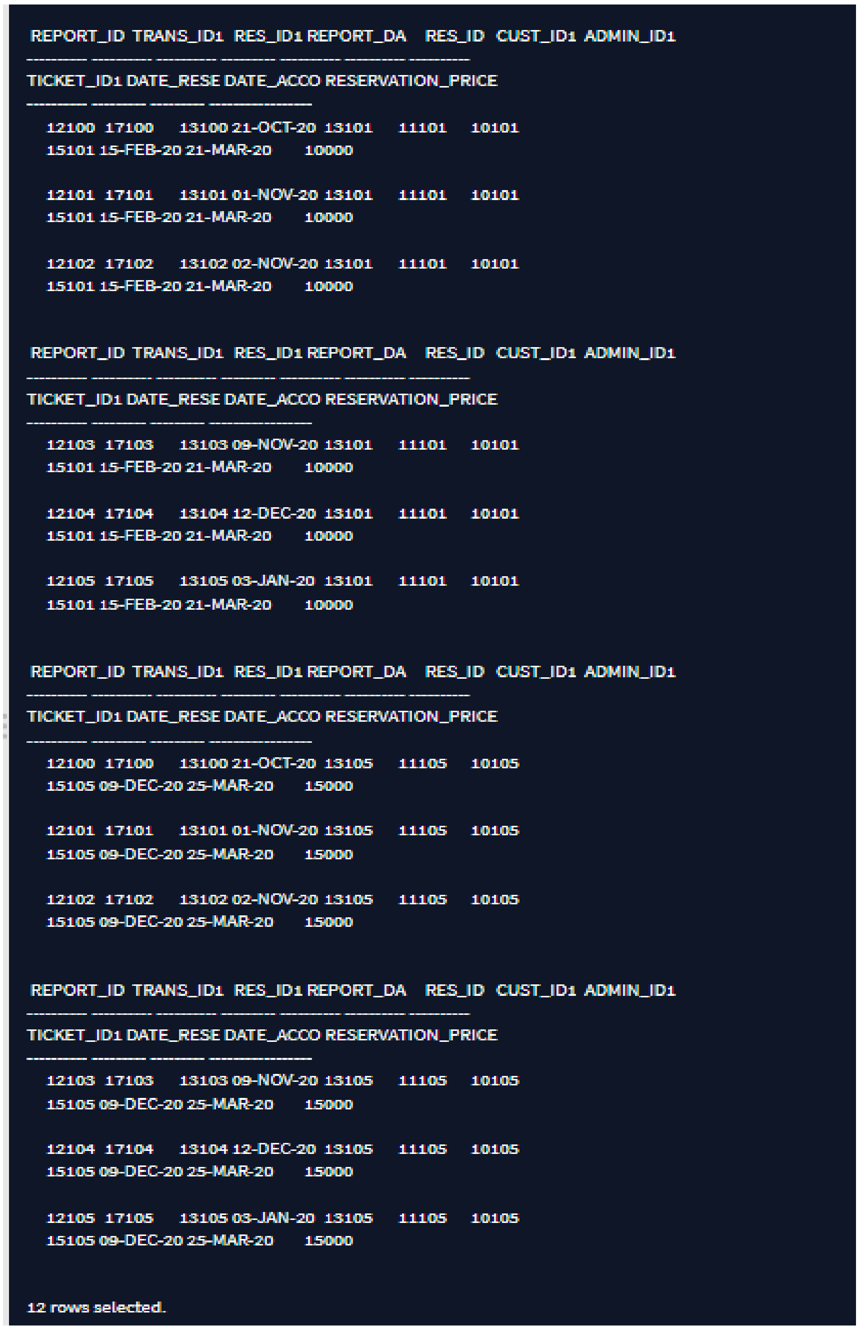
QUERY 7:

To fetch reports where reservation price is less than 20000



```
select * from Reports natural join Reservation where reservation_price<20000;
```

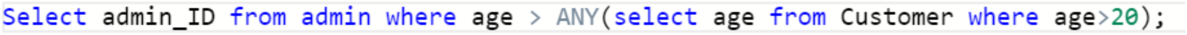
Output:



```
 REPORT_ID TRANS_ID1  RES_ID1 REPORT_DA    RES_ID  CUST_ID1 ADMIN_ID1
---------- ---------- ---------- --------- ---------- ---------- ----------
TICKET_ID1 DATE_RESE DATE_ACCO RESERVATION_PRICE
---------- --------- --------- -----------------
    12100  17100      13100 21-OCT-20  13101      11101      10101
    15101 15-FEB-20 21-MAR-20        10000

    12101  17101      13101 01-NOV-20  13101      11101      10101
    15101 15-FEB-20 21-MAR-20        10000

    12102  17102      13102 02-NOV-20  13101      11101      10101
    15101 15-FEB-20 21-MAR-20        10000


 REPORT_ID TRANS_ID1  RES_ID1 REPORT_DA    RES_ID  CUST_ID1 ADMIN_ID1
---------- ---------- ---------- --------- ---------- ---------- ----------
TICKET_ID1 DATE_RESE DATE_ACCO RESERVATION_PRICE
---------- --------- --------- -----------------
    12103  17103      13103 09-NOV-20  13101      11101      10101
    15101 15-FEB-20 21-MAR-20        10000

    12104  17104      13104 12-DEC-20  13101      11101      10101
    15101 15-FEB-20 21-MAR-20        10000

    12105  17105      13105 03-JAN-20  13101      11101      10101
    15101 15-FEB-20 21-MAR-20        10000


 REPORT_ID TRANS_ID1  RES_ID1 REPORT_DA    RES_ID  CUST_ID1 ADMIN_ID1
---------- ---------- ---------- --------- ---------- ---------- ----------
TICKET_ID1 DATE_RESE DATE_ACCO RESERVATION_PRICE
---------- --------- --------- -----------------
    12100  17100      13100 21-OCT-20  13105      11105      10105
    15105 09-DEC-20 25-MAR-20        15000

    12101  17101      13101 01-NOV-20  13105      11105      10105
    15105 09-DEC-20 25-MAR-20        15000

    12102  17102      13102 02-NOV-20  13105      11105      10105
    15105 09-DEC-20 25-MAR-20        15000


 REPORT_ID TRANS_ID1  RES_ID1 REPORT_DA    RES_ID  CUST_ID1 ADMIN_ID1
---------- ---------- ---------- --------- ---------- ---------- ----------
TICKET_ID1 DATE_RESE DATE_ACCO RESERVATION_PRICE
---------- --------- --------- -----------------
    12103  17103      13103 09-NOV-20  13105      11105      10105
    15105 09-DEC-20 25-MAR-20        15000

    12104  17104      13104 12-DEC-20  13105      11105      10105
    15105 09-DEC-20 25-MAR-20        15000

    12105  17105      13105 03-JAN-20  13105      11105      10105
    15105 09-DEC-20 25-MAR-20        15000


12 rows selected.
```

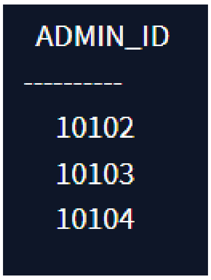
QUERY 8:

To select admin_ID of the admin that manages the customer with age above 20



```
Select admin_ID from admin where age > ANY(select age from Customer where age>20);
```

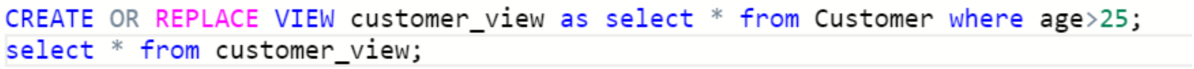
Output:



```
 ADMIN_ID
----------
    10102
    10103
    10104
```

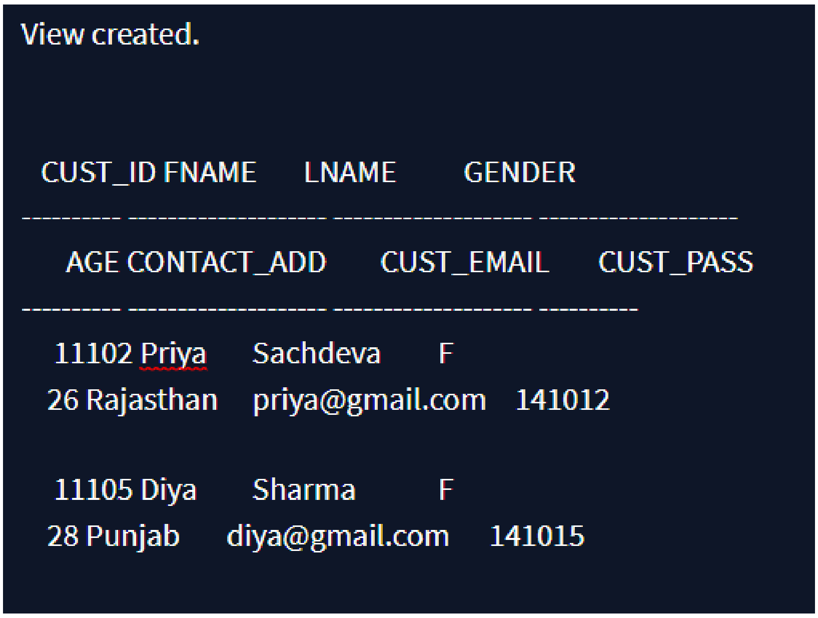
Query 9:

To get a view from Customer table where their age would be more than 25



```
CREATE OR REPLACE VIEW customer_view as select * from Customer where age>25;
select * from customer_view;
```

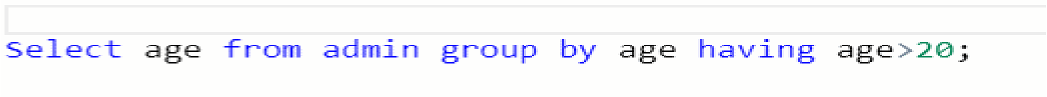
Output:



```
View created.


  CUST_ID FNAME     LNAME      GENDER
---------- -------------------- -------------------- --------------------
    AGE CONTACT_ADD      CUST_EMAIL      CUST_PASS
---------- -------------------- -------------------- ----------
    11102 Priya      Sachdeva      F
   26 Rajasthan     priya@gmail.com    141012

    11105 Diya       Sharma        F
   28 Punjab      diya@gmail.com     141015
```

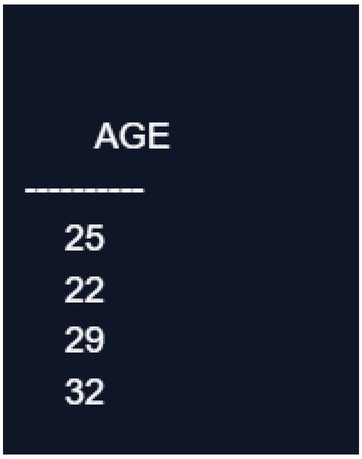
Query 10:

To get admin age where age of admin is greater than 20



```
Select age from admin group by age having age>20;
```

Output:



```
      AGE
----------
    25
    22
    29
    32
```

NORMALISATION

Admin Table :-

Here,

Admin_ID is Primary Key

admin_ID is Candidate Key as It is traversing all other attributes.

1NF -> This table is already normalised to 1st Normal Form as it has no multivalued attributes/Atomicity.

2NF -> This table is already normalised to 2nd Normal Form as it is in 1NF and it has no partial Dependency.

3NF -> This table is not Normalised to 3rd Normal Form as condition/Functional Dependency= Candidate Key->Non-Prime is there.

So, we will Decompose the table into two tables.

DECOMPOSED TABLE:

Admin_ID
Fname
Lname
Gender
Age
Contact_email

Table1:- Table2:-

Admin_ID
Fname
Lname
Gender
Age

Contact_email
Admin_pass

Customer Table:-

Here,

cust_ID is Primary Key

cust_ID is Candidate Key as It is traversing all other attributes.

1NF -> This table is already Normalised to 1st Normal Form as it has no multivalued attributes/Atomicity.

2NF -> This table is already normalised to 2nd Normal Form as it is in 1NF and it has no partial Dependency.

3NF -> This table is not Normalised to 3rd Normal Form as condition/Functional Dependency= Candidate Key->NonPrime is there .

So we will Decompose the table into two tables.

Decomposed Table:

Table1:- Table2:-

Cust_ID
Fname
Lname
Gender

Age
Contact_add
Cust_email
Cust_pass
Contact_add
Cust_email
Cust_ID

Reports:-

Here,

report_ID is Primary Key

report_ID is candidate key

1NF -> This table is already Normalised to 1st Normal Form as it has no multivalued attributes/Atomicity.

2NF -> This table is already normalised to 2nd Normal Form as it is in 1NF and it has no partial Dependency.

3NF -> This table is not Normalised to 3rd Normal Form as condition/Functional Dependency= Non-Prime->Non-Prime is there .

So we will Decompose the table into two tables.

Decomposed Table:

Table1:-

Report_ID
Trans_ID

Table2:-

Res_ID
Report_date
res_ID
report_date

Reservation Table:-

Here,

res_ID is Primary Key

admin_ID is candidate key

admin_ID,cust_ID,ticket_ID is foreign key

1NF -> This table is already Normalised to 1st Normal Form as it has no multivalued attributes/Atomicity.

2NF -> This table is already Normalised to 2nd Normal Form as it is in 1NF and it has no partial Dependency.

3NF -> This table is not Normalised to 3rd Normal Form as condition/Functional Dependency= Non Prime->NonPrime is there .

So we will Decompose the table into three tables.

Decomposed Table:

Table1:- Table2:-

Res_ID
Cust_ID
Admin_ID
Ticket_ID
Res_ID
Cust_ID
Ticket_ID
Date_reserve
Date_accom

Table 3:-

Cust_ID
Ticket_D

Schedule Table:-

Here,

sched_ID is Primary Key

sched_ID is candidate key

admin_ID,cust_ID,ticket_ID is foreign key

1NF -> This table is already Normalised to 1st Normal Form as it has no multivalued attributes/Atomicity.

2NF -> This table is already Normalised to 2nd Normal Form as it is in 1NF and it has no partial Dependency.

3NF -> This table is already Normalised to 3rd Normal Form as condition/Functional Dependency= Prime->NonPrime is there also No Transitive Dependency.

Ticket Table:-

Here,

ticket_ID is Primary Key

ticket_ID is candidate key

1NF -> This table is already Normalised to 1st Normal Form as it has no multivalued attributes/Atomicity.

2NF -> This table is already Normalised to 2nd Normal Form as it is in 1NF and it has no partial Dependency.

3NF -> This table is not Normalised to 3rd Normal Form as condition/Functional Dependency= Candidate Kay->NonPrime is there .

So we will Decompose the table into two tables.

Decomposed Table:

Table 1:-   Table2:-

Ticket_ID
Ticket_num
Ticket_num
Date_avail
Date_flight
Time_depart
Time_land
Destination

Transaction Table:-

Here,

cust_ID is Primary Key

trans_ID is Primary key

sched_ID is foreign key

transty_ID is foreign key

transty_ID is Candidate Key

1NF -> This table is already Normalised to 1st Normal Form as it has no multivalued attributes/Atomicity.

2NF -> This table is already Normalised to 2nd Normal Form as it is in 1NF and it has no partial Dependency.

3NF -> This table is not Normalised to 3rd Normal Form as condition/Functional Dependency= Non Prime->Prime is there.

So we will Decompose the table into three tables.

Decomposed Table:

Transty_Id
Trans_ID
Trans_name
Trans_date
Trans_ID
Cust_Id
Sched_ID
Trans_ID

Table1:-   Table2:-   Table3:-

Transaction Type:-

Here,

transty_ID is Primary Key

sched_ID is foreign key

ticket_ID is foreign key

1NF -> This table is already Normalised to 1st Normal Form as it has no multivalued attributes/Atomicity.

2NF -> This table is already Normalised to 2nd Normal Form as it is in 1NF and it has no partial Dependency.

3NF -> This table is not Normalised to 3rd Normal Form as Transitive Dependency is there.

So we will Decompose the table into three tables.

Decomposed Table:

Sched_ID
Ticket_Id

Table1:- Table 2:- Table3:-

Transty_Id
Ticket_ID
Transty_ID
Trans_name

Conclusion and Future work:

We are trying to give a live reporting which is updated by Airline Companies so that customer gets a live Flights checking, Available seats, Pricing and also planning to provide seats as per theirs choice so that they can travel very comfortably their journey. We will be trying to provide food facility and choice to customers so that they can feel like their home and more effective amenities. We are also trying to make more attention on Business class people and their requirements. Our future planning is to take this project towards an Android App and QR Code Scanning. So that a Customer can easily contact to the Airlines and they are getting quick Services from Airlines. We also want in future to place in market so that customer can take more advantages and saves their important time. We are also finding and approaching to companies which are using this type of software.

BIBLIOGRAPHY:

- **https://www.lucidchart.com/**
- **https://codequotient.com/code**
- **https://www.flightslogic.com/airline-ticket-reservation-system.php**