Database Management Project

ONLINE FLIGHT BOOKING

INDEX

Requirements of system

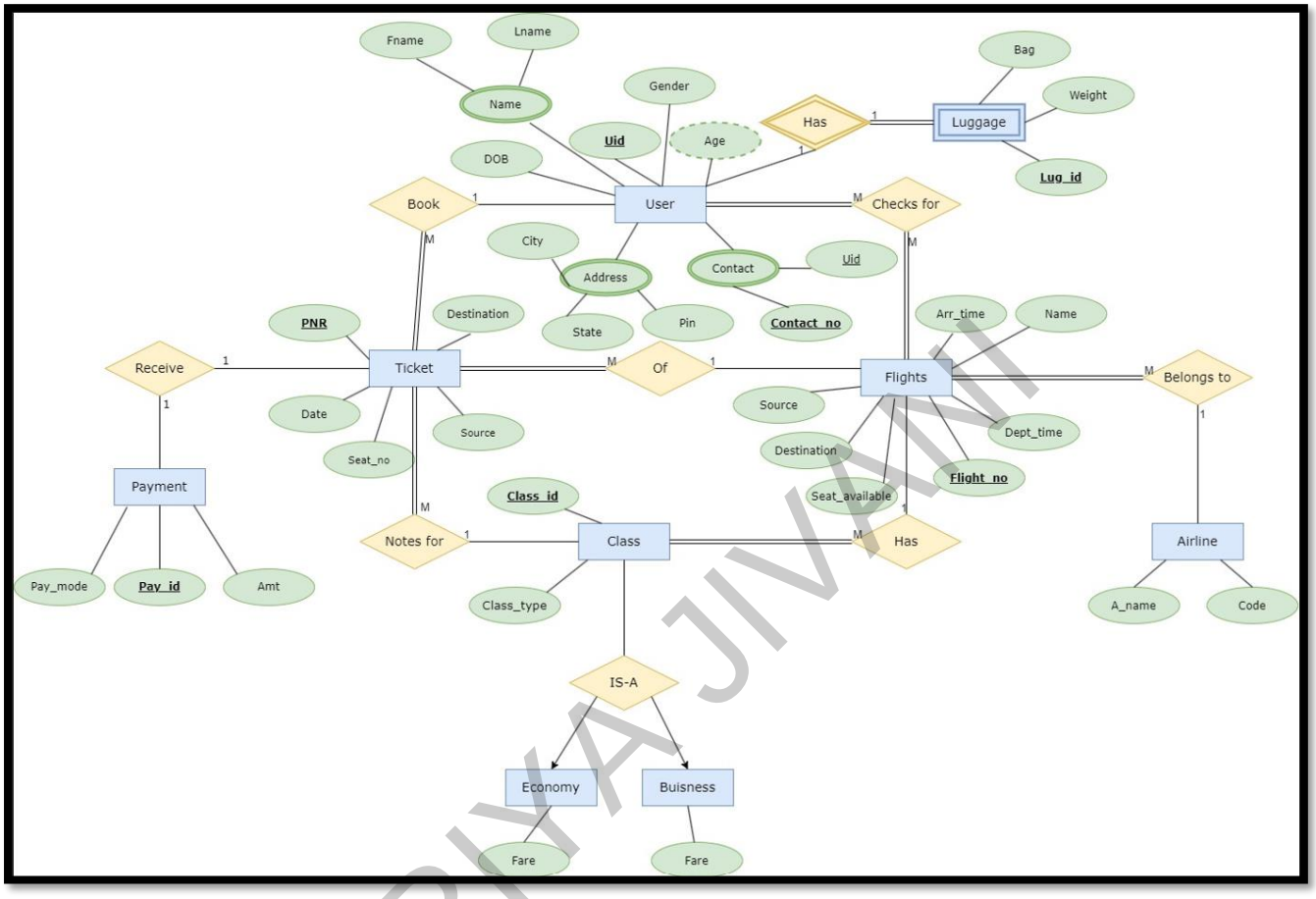
We want to make a website for flight booking as per the demands of users.

- Users should be able to search for flights for a given date and source/destination airport.
- Users should be able to reserve a ticket for any scheduled flight.
- Users of the system can check flight schedules, their departure time, available seats, arrival time, and other flight details.
- Users can make reservations for multiple passengers under one itinerary.
- Only the admin of the system can add new aircrafts, flights, and flight schedules. Admin can cancel any pre-scheduled flight.
- Users can cancel their reservation and itinerary.
- The system should be able to handle payments for reservations.

Description of Flight Booking Database

- The details of Ticket is store into Ticket tables respective with all tables.
- Each entity (User, Ticket, Luggage, Flight, Payment, Airline) contains Primary key.
- The entity (Ticket, Flight, Payment, Class) contains foreign key.
- There is one-to-one relationship between Ticket and Payment and one-to-many relationships available between luggage and user, user and ticket, flight and ticket, flight and class, airline and flight, user and flight. (Ticket and class)

Entity-Relationship Model



Relational Schema

User	User_id, fname, lname, gender, city, pin, state, DOB
Contact	User_id, contact_id, contact_no
Luggage	Lug_id,User_id, bag,weight
Airline	Air_name, code
Flight	Flight_no, name, seat_available, Air_name, destination, source, Arr_time, dep_time
Paymnet	Pay_id, PNR, Amt, Pay_mode
Ticket	PNR, User_id, Flight_no, seat_no, date,destination, source
Class Economy Buisness	Class_id, class_type, Flight_no, PNR Class_id, fare Class_id, fare
Checks For	Chk_id, Flight_no, User_id

Data Dictionary

4.1 Admin

```
postgres=# \d admin;
                    Table "public.admin"
 Column |          Type          | Collation | Nullable | Default
--------+------------------------+-----------+----------+---------
 a_id   | character varying(10) |           | not null |
 a_name | character varying(20) |           |          |
Indexes:
    "admin_pkey" PRIMARY KEY, btree (a_id)
```

4.2 Users

```
postgres=# \d Users;
                   Table "public.users"
 Column  |         Type          | Collation | Nullable | Default
---------+-----------------------+-----------+----------+---------
 user_id | character varying(5)  |           | not null |
 fname   | character varying(10) |           | not null |
 lname   | character varying(10) |           | not null |
 gender  | character varying(6)  |           |          |
 city    | character varying(15) |           |          |
 pin     | integer               |           |          |
 state   | character varying(15) |           |          |
 dob     | date                  |           |          |
Indexes:
    "users_pkey" PRIMARY KEY, btree (user_id)
Referenced by:
    TABLE "checksfor" CONSTRAINT "cheaksfor_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
    TABLE "contact" CONSTRAINT "contact_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
    TABLE "luggage" CONSTRAINT "luggage_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
    TABLE "ticket" CONSTRAINT "ticket_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
```

4.3 Contact

```
postgres=# \d Contact;
                   Table "public.contact"
 Column  |         Type          | Collation | Nullable | Default
---------+-----------------------+-----------+----------+---------
 cnt_id  | character varying(5) |           | not null |
 user_id | character varying(5) |           |          |
 cnt_no  | numeric(10,0)         |           | not null |
Indexes:
    "contact_pkey" PRIMARY KEY, btree (cnt_id)
Foreign-key constraints:
    "contact_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
```

4.4 Luggage

```
postgres=# \d Luggage;
                    Table "public.luggage"
 Column  |         Type          | Collation | Nullable | Default
---------+-----------------------+-----------+----------+---------
 lug_id  | character varying(5) |           | not null |
 user_id | character varying(5) |           | not null |
 bag     | integer               |           |          |
 weight  | integer               |           |          |
Indexes:
    "luggage_pkey" PRIMARY KEY, btree (lug_id, user_id)
Foreign-key constraints:
    "luggage_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
```

4.5 Airline

```
postgres=# \d Airline;
                    Table "public.airline"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 air_name | character varying(20) |           | not null |
 code     | integer               |           |          |
Indexes:
    "airline_pkey" PRIMARY KEY, btree (air_name)
Referenced by:
    TABLE "flight" CONSTRAINT "flight_air_name_fkey" FOREIGN KEY (air_name) REFERENCES airline(air_name)
```

4.6 Flight

```
postgres=# \d Flight;
                     Table "public.flight"
     Column     |          Type          | Collation | Nullable | Default
----------------+------------------------+-----------+----------+---------
 flight_no      | integer                |           | not null |
 air_name       | character varying(20)  |           |          |
 flight_name    | character varying(20)  |           |          |
 seat_available | integer                |           |          |
 destination    | character varying(20)  |           |          |
 source         | character varying(20)  |           |          |
 arr_time       | time without time zone |           |          |
 dep_time       | time without time zone |           |          |
Indexes:
    "flight_pkey" PRIMARY KEY, btree (flight_no)
Foreign-key constraints:
    "flight_air_name_fkey" FOREIGN KEY (air_name) REFERENCES airline(air_name)
Referenced by:
    TABLE "checksfor" CONSTRAINT "cheaksfor_flight_no_fkey" FOREIGN KEY (flight_no) REFERENCES flight(flight_no)
    TABLE "class" CONSTRAINT "class_flight_no_fkey" FOREIGN KEY (flight_no) REFERENCES flight(flight_no)
    TABLE "ticket" CONSTRAINT "ticket_flight_no_fkey" FOREIGN KEY (flight_no) REFERENCES flight(flight_no)
```

4.7 Payment

```
postgres=# \d Payment;
                      Table "public.payment"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 pay_id   | integer               |           | not null |
 pnr      | character varying(5)  |           |          |
 amount   | numeric(10,2)         |           |          |
 pay_mode | character varying(8)  |           |          |
Indexes:
    "payment_pkey" PRIMARY KEY, btree (pay_id)
Foreign-key constraints:
    "payment_pnr_fkey" FOREIGN KEY (pnr) REFERENCES ticket(pnr)
```

4.8 Ticket

```
postgres=# \d Ticket;
                       Table "public.ticket"
    Column    |         Type          | Collation | Nullable | Default
--------------+-----------------------+-----------+----------+---------
 pnr          | character varying(5)  |           | not null |
 user_id      | character varying(5)  |           |          |
 flight_no    | integer               |           |          |
 seat_no      | character varying(5)  |           |          |
 journey_date | date                  |           |          |
 destination  | character varying(20) |           |          |
 source       | character varying(20) |           |          |
Indexes:
    "ticket_pkey" PRIMARY KEY, btree (pnr)
Foreign-key constraints:
    "ticket_flight_no_fkey" FOREIGN KEY (flight_no) REFERENCES flight(flight_no)
    "ticket_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
Referenced by:
    TABLE "class" CONSTRAINT "class_pnr_fkey" FOREIGN KEY (pnr) REFERENCES ticket(pnr)
    TABLE "payment" CONSTRAINT "payment_pnr_fkey" FOREIGN KEY (pnr) REFERENCES ticket(pnr)
```

4.9 Class

```
postgres=# \d Class;
                    Table "public.class"
   Column   |         Type         | Collation | Nullable | Default
------------+----------------------+-----------+----------+---------
 class_id   | integer              |           | not null |
 flight_no  | integer              |           |          |
 pnr        | character varying(5) |           |          |
 class_type | character varying(1) |           |          |
Indexes:
    "class_pkey" PRIMARY KEY, btree (class_id)
Foreign-key constraints:
    "class_flight_no_fkey" FOREIGN KEY (flight_no) REFERENCES flight(flight_no)
    "class_pnr_fkey" FOREIGN KEY (pnr) REFERENCES ticket(pnr)
Referenced by:
    TABLE "business" CONSTRAINT "business_class_id_fkey" FOREIGN KEY (class_id) REFERENCES class(class_id)
    TABLE "economy" CONSTRAINT "economy_class_id_fkey" FOREIGN KEY (class_id) REFERENCES class(class_id)
```

4.10 Economy

```
postgres=# \d Economy;
                  Table "public.economy"
  Column  |     Type      | Collation | Nullable | Default
----------+---------------+-----------+----------+---------
 class_id | integer       |           |          |
 fare     | numeric(10,2) |           |          |
Foreign-key constraints:
    "economy_class_id_fkey" FOREIGN KEY (class_id) REFERENCES class(class_id)
```

4.11 Business

```
postgres=# \d Business;
                  Table "public.business"
  Column  |     Type      | Collation | Nullable | Default
----------+---------------+-----------+----------+---------
 class_id | integer       |           |          |
 fare     | numeric(10,2) |           |          |
Foreign-key constraints:
    "business_class_id_fkey" FOREIGN KEY (class_id) REFERENCES class(class_id)
```

4.12 Checksfor

```
postgres=# \d Checksfor;
                    Table "public.checksfor"
  Column   |         Type          | Collation | Nullable | Default
-----------+-----------------------+-----------+----------+---------
 chk_id    | integer               |           | not null |
 user_id   | character varying(5) |           |          |
 flight_no | integer               |           |          |
Indexes:
    "cheaksfor_pkey" PRIMARY KEY, btree (chk_id)
Foreign-key constraints:
    "cheaksfor_flight_no_fkey" FOREIGN KEY (flight_no) REFERENCES flight(flight_no)
    "cheaksfor_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
```

Data Implementation

A) Schema

Admin

```
create table Admin(a_id varchar(10) primary key,a_name varchar(20));
```

Users

```
create table Users(user_id varchar(5) primary key,fname varchar(10) not null,lname varchar(10) not null,gender varchar(6),city varchar(15),pin int,state varchar(15),dob date);
```

Contact

```
create table Contact(cnt_id varchar(5) primary key,user_id varchar(5),cnt_no numeric(10) not null,foreign key (user_id) references Users(user_id));
```

Luggage

```
create table Luggage(lug_id varchar(5),user_id varchar(5) not null,bag int,weight int,primary key(lug_id,user_id), foreign key(user_id) references Users(user_id) on delete cascade);
```

Airline

```
create table Airline(air_name varchar(20) primary key,code int);
```

Flight

```
create table Flight(flight_no int primary key,air_name varchar(20),flight_name varchar(20),seat_available int,destination varchar(20),source varchar(20),arr_time time,dep_time time,foreign key(air_name) references Airline(air_name));
```

Payment

```
create table Payment(pay_id int primary key,PNR varchar(5),amount
numeric(10,2),pay_mode varchar(8),foreign key(PNR) references
Ticket(PNR));
```

Ticket

```
create table Ticket(PNR varchar(5) primary key,user_id
varchar(5),flight_no int,seat_no varchar(5),journey_date date,destination
varchar(20),source varchar(20),foreign key(user_id) references
Users(user_id),foreign key(flight_no) references Flight(flight_no));
```

Class

```
create table Class(class_id int primary key,flight_no int,PNR
varchar(5),class_type varchar(1),foreign key(flight_no) references
Flight(flight_no),foreign key(PNR) references Ticket(PNR));
```

Economy

```
create table Economy(class_id int,fare numeric(10,2),foreign key(class_id)
references Class(class_id));
```

Business

```
create table Business(class_id int,fare numeric(10,2),foreign key(class_id)
references Class(class_id));
```

Checksfor

```
create table cheaksfor(chk_id int primary key,user_id varchar(5),flight_no
int,foreign key(flight_no) references Flight(flight_no),foreign key(user_id)
references Users(user_id));
```

B) Data Insertion

Admin

```
insert into Admin(a_id,a_name) values('21ITUOF071','RIYA JIVANI'),
('22ITUOD006','MILONI MEHTA');
```

Users

```
INSERT INTO Users (user_id, fname, lname, gender, city, pin, state, dob)
VALUES('U0001', 'Aarav', 'Sharma', 'Male', 'Mumbai', 400001,
'Maharashtra', '1990-01-01'),('U0002', 'Sneha', 'Patel', 'Female',
'Ahmedabad', 380001, 'Gujarat', '1995-03-15'),('U0003', 'Rohan', 'Chopra',
'Male', 'New Delhi', 110001, 'Delhi', '1985-07-22'),('U0004', 'Aanya',
'Gupta', 'Female', 'Kolkata', 700001, 'West Bengal', '1992-11-30'),('U0005',
'Aditya', 'Mehra', 'Male', 'Bengaluru', 560001, 'Karnataka', '1987-04-
10'),('U0006', 'Aashi', 'Singh', 'Female', 'Jaipur', 302001, 'Rajasthan', '1998-
09-18'),('U0007', 'Arjun', 'Rao', 'Male', 'Chennai', 600001, 'Tamil Nadu',
'1991-02-28'),('U0008', 'Isha', 'Nair', 'Female', 'Hyderabad', 500001,
'Telangana', '1994-06-12'),('U0009', 'Anish', 'Kumar', 'Male', 'Lucknow',
226001, 'Uttar Pradesh', '1989-12-05'),('U0010', 'Kavya', 'Menon',
'Female', 'Pune', 411001, 'Maharashtra', '1997-08-25');
```


Contact

```
INSERT INTO Contact (cnt_id, user_id, cnt_no) VALUES ('C0001',
'U0001', 9876543210),('C0002', 'U0001', 8765432109),('C0003', 'U0002',
7654321098),('C0004', 'U0003', 6543210987),('C0005', 'U0004',
5432109876),('C0006', 'U0004', 4321098765),('C0007', 'U0005',
3210987654),('C0008', 'U0006', 2109876543),('C0009', 'U0007',
1098765432),('C0010', 'U0007', 9876543210),('C0011', 'U0008',
8765432109),('C0012', 'U0009', 7654321098),('C0013', 'U0010',
6543210987);
```

Luggage

```
INSERT INTO Luggage (lug_id, user_id, bag, weight) VALUES ('L0001',
'U0001', 2, 15),('L0002', 'U0003', 1, 10),('L0003', 'U0005', 3, 20),('L0004',
```

'U0004', 1, 8),('L0005', 'U0002', 2, 12),('L0006', 'U0008', 1, 6),('L0007', 'U0009', 4, 25),('L0008', 'U0006', 2, 14),('L0009', 'U0010', 3, 18),('L0010', 'U0007', 1, 9);

Airline

INSERT INTO Airline (air_name, code) VALUES ('Air India', 100),('IndiGo', 200),('GoAir', 500),('AirAsia India', 600),('Vistara', 400),('Star Air', 800),('Alliance Air', 1200);

Flight

INSERT INTO Flight (flight_no, air_name, flight_name, seat_available, destination, source, arr_time, dep_time) VALUES (1001, 'Air India', 'AI 101', 120, 'Mumbai', 'Delhi', '13:30:00', '11:00:00'),(1002, 'Air India', 'AI 102', 95, 'Delhi', 'Mumbai', '18:00:00', '15:30:00'),(2001, 'IndiGo', '6E 201', 150, 'Bangalore', 'Delhi', '16:30:00', '14:00:00'),(2002, 'IndiGo', '6E 202', 80, 'Delhi', 'Bangalore', '21:00:00', '18:30:00'),(3001, 'SpiceJet', 'SG 301', 100, 'Chennai', 'Mumbai', '17:00:00', '14:30:00'),(3002, 'SpiceJet', 'SG 302', 75, 'Mumbai', 'Chennai', '22:00:00', '19:30:00'),(4001, 'Vistara', 'UK 401', 125, 'Delhi', 'Kolkata', '11:30:00', '09:00:00'),(4002, 'Vistara', 'UK 402', 85, 'Kolkata', 'Delhi', '15:30:00', '13:00:00'),(5001, 'GoAir', 'G8 501', 80, 'Mumbai', 'Bangalore', '14:30:00', '12:00:00'),(5002, 'GoAir', 'G8 502', 70, 'Bangalore', 'Mumbai', '19:00:00', '16:30:00'),(6001, 'AirAsia India', 'I5 601', 120, 'Chennai', 'Bangalore', '11:00:00', '09:30:00'),(6002, 'AirAsia India', 'I5 602', 90, 'Bangalore', 'Chennai', '14:30:00', '12:00:00'),(7001, 'Alliance Air', '9I 701', 60, 'Delhi', 'Jaipur', '10:00:00', '08:30:00'),(7002, 'Alliance Air', '9I 702', 50, 'Jaipur', 'Delhi', '12:30:00', '11:00:00'),(8002, 'Star Air', 'OG 801', 40, 'Belgaum', 'Bengaluru', '16:00:00', '14:30:00');

Payment

INSERT INTO Payment (pay_id, PNR, amount, pay_mode) VALUES(1,'ABC11',11375.25, 'Debit'),(2,'ABC05',23450.00, 'Cash'),(3,'ABC09',12500.75, 'Credit'),(4,'ABC06',80000.00, 'Debit'),(5,'ABC02',31500.00, 'Cash'),(6,'ABC03',90000.25, 'Debit'),(7,'ABC10',20550.00, 'Cash'),(8,'ABC07',17175.75, 'Credit'),(9,'ABC04',55000.00, 'Debit'),(10,'ABC12',110000.00, 'Cash'),(11,'ABC01',10000.50, 'Credit');

Ticket

INSERT INTO Ticket (PNR, user_id, flight_no, seat_no, journey_date, destination, source) VALUES ('ABC01', 'U0001', 3001, 'A1', '2023-03-15', 'Chennai', 'Mumbai'),('ABC02', 'U0002', 6002, 'B2', '2023-03-16','Bangalore', 'Chennai'),('ABC03', 'U0004', 5001, 'C3', '2023-03-17', 'Mumbai', 'Bangalore'),('ABC04', 'U0005', 2002, 'D4', '2023-03-18', 'Delhi', 'Bangalore'),('ABC05', 'U0005',2002, 'E5', '2023-03-19', 'Delhi', 'Bangalore'),('ABC06', 'U0003', 1002, 'F6', '2023-03-20', 'Delhi', 'Mumbai'),('ABC07', 'U0007', 7001, 'G7', '2023-03-21', 'Delhi', 'Jaipur'),('ABC09', 'U0006', 6001, 'I9', '2023-03-23', 'Chennai', 'Bangalore'),('ABC10', 'U0010', 4001, 'J10', '2023-03-24', 'Delhi', 'Kolkata'),('ABC11', 'U0009', 8002, 'K11', '2023-03-25','Belgaum', 'Bengaluru'),('ABC12', 'U0009', 8002, 'L12', '2023-03-26', 'Belgaum', 'Bengaluru');

Class

INSERT INTO Class (class_id, flight_no, PNR, class_type) VALUES (1,3001,'ABC01','E'),(2,6002,'ABC02','B'),(3,5001,'ABC03','E'),(4,2002,'ABC04','B'),(5,2002,'ABC05','E'),(6,1002,'ABC06','B'),(7,7001,'ABC07','E'),(8,6001,'ABC09','B'),(9,4001,'ABC10','E'),(10,8002,'ABC11','B'),(11,8002,'ABC12','E');

Economy

INSERT INTO Economy(class_id ,fare) VALUES (1,10000.50),(3,90000.25),(5,23450.00),(7,17175.75),(9,20550.00),(11,110000.00);

Business

INSERT INTO Business(class_id ,fare) VALUES (2,31500.00),(4,55000.00),(6,80000.00),(8,12500.75),(10,11375.25);

Checksfor

INSERT INTO Cheaksfor (chk_id, user_id, flight_no) VALUES (1, 'U0001',3001),(2, 'U0002',6002),(3, 'U0003',1002),(4, 'U0003',4001),(5, 'U0004',5001),(6, 'U0005',2002),(7, 'U0007',7001),(8, 'U0008',5001),(9, 'U0009',8002),(10, 'U0009',1002),(11, 'U0010',4001),(12, 'U0001',8002),(13, 'U0004',1001),(14, 'U0005',2002),(15, 'U0001',5002);

Insertion Output

Admin

```
postgres=# select * from Admin;
     a_id     |     a_name
--------------+----------------
 21ITUOF071 | RIYA JIVANI
 22ITUOD006 | MILONI MEHTA
(2 rows)
```

Users

```
postgres=# select * from Users;
 user_id | fname  | lname  | gender |   city    |  pin   |     state     |    dob
---------+--------+--------+--------+-----------+--------+---------------+------------
 U0001   | Aarav  | Sharma | Male   | Mumbai    | 400001 | Maharashtra   | 1990-01-01
 U0002   | Sneha  | Patel  | Female | Ahmedabad | 380001 | Gujarat       | 1995-03-15
 U0003   | Rohan  | Chopra | Male   | New Delhi | 110001 | Delhi         | 1985-07-22
 U0004   | Aanya  | Gupta  | Female | Kolkata   | 700001 | West Bengal   | 1992-11-30
 U0005   | Aditya | Mehra  | Male   | Bengaluru | 560001 | Karnataka     | 1987-04-10
 U0006   | Aashi  | Singh  | Female | Jaipur    | 302001 | Rajasthan     | 1998-09-18
 U0007   | Arjun  | Rao    | Male   | Chennai   | 600001 | Tamil Nadu    | 1991-02-28
 U0008   | Isha   | Nair   | Female | Hyderabad | 500001 | Telangana     | 1994-06-12
 U0009   | Anish  | Kumar  | Male   | Lucknow   | 226001 | Uttar Pradesh | 1989-12-05
 U0010   | Kavya  | Menon  | Female | Pune      | 411001 | Maharashtra   | 1997-08-25
(10 rows)
```

Contact

```
postgres=# select * from Contact;
 cnt_id | user_id |   cnt_no
--------+---------+------------
 C0001  | U0001   | 9876543210
 C0002  | U0001   | 8765432109
 C0003  | U0002   | 7654321098
 C0004  | U0003   | 6543210987
 C0005  | U0004   | 5432109876
 C0006  | U0004   | 4321098765
 C0007  | U0005   | 3210987654
 C0008  | U0006   | 2109876543
 C0009  | U0007   | 1098765432
 C0010  | U0007   | 9876543210
 C0011  | U0008   | 8765432109
 C0012  | U0009   | 7654321098
 C0013  | U0010   | 6543210987
(13 rows)
```

Luggage

```
postgres=# select * from Luggage;
 lug_id | user_id | bag | weight
--------+---------+-----+--------
 L0001  | U0001   |   2 |     15
 L0002  | U0003   |   1 |     10
 L0003  | U0005   |   3 |     20
 L0004  | U0004   |   1 |      8
 L0005  | U0002   |   2 |     12
 L0006  | U0008   |   1 |      6
 L0007  | U0009   |   4 |     25
 L0008  | U0006   |   2 |     14
 L0009  | U0010   |   3 |     18
 L0010  | U0007   |   1 |      9
(10 rows)
```

Airline

```
postgres=# select * from Airline;
   air_name    | code
---------------+------
 Air India     |  100
 IndiGo        |  200
 GoAir         |  500
 AirAsia India |  600
 SpiceJet      | 1000
 Vistara       |  400
 Star Air      |  800
 Alliance Air  | 1200
(8 rows)
```

Flight

```
postgres=# select * from Flight;
 flight_no |   air_name    | flight_name | seat_available | destination |  source   | arr_time | dep_time
-----------+---------------+-------------+----------------+-------------+-----------+----------+----------
      1001 | Air India     | AI 101      |            120 | Mumbai      | Delhi     | 13:30:00 | 11:00:00
      1002 | Air India     | AI 102      |             95 | Delhi       | Mumbai    | 18:00:00 | 15:30:00
      2001 | IndiGo        | 6E 201      |            150 | Bangalore   | Delhi     | 16:30:00 | 14:00:00
      2002 | IndiGo        | 6E 202      |             80 | Delhi       | Bangalore | 21:00:00 | 18:30:00
      3001 | SpiceJet      | SG 301      |            100 | Chennai     | Mumbai    | 17:00:00 | 14:30:00
      3002 | SpiceJet      | SG 302      |             75 | Mumbai      | Chennai   | 22:00:00 | 19:30:00
      4001 | Vistara       | UK 401      |            125 | Delhi       | Kolkata   | 11:30:00 | 09:00:00
      4002 | Vistara       | UK 402      |             85 | Kolkata     | Delhi     | 15:30:00 | 13:00:00
      5001 | GoAir         | G8 501      |             80 | Mumbai      | Bangalore | 14:30:00 | 12:00:00
      5002 | GoAir         | G8 502      |             70 | Bangalore   | Mumbai    | 19:00:00 | 16:30:00
      6001 | AirAsia India | I5 601      |            120 | Chennai     | Bangalore | 11:00:00 | 09:30:00
      6002 | AirAsia India | I5 602      |             90 | Bangalore   | Chennai   | 14:30:00 | 12:00:00
      7001 | Alliance Air  | 9I 701      |             60 | Delhi       | Jaipur    | 10:00:00 | 08:30:00
      7002 | Alliance Air  | 9I 702      |             50 | Jaipur      | Delhi     | 12:30:00 | 11:00:00
      8002 | Star Air      | OG 801      |             40 | Belgaum     | Bengaluru | 16:00:00 | 14:30:00
(15 rows)
```

Payment

```
postgres=# select * from Payment;
 pay_id |  pnr   |  amount   | pay_mode
--------+--------+-----------+----------
      1 | ABC11  |  11375.25 | Debit
      2 | ABC05  |  23450.00 | Cash
      3 | ABC09  |  12500.75 | Credit
      4 | ABC06  |  80000.00 | Debit
      5 | ABC02  |  31500.00 | Cash
      6 | ABC03  |  90000.25 | Debit
      7 | ABC10  |  20550.00 | Cash
      8 | ABC07  |  17175.75 | Credit
      9 | ABC04  |  55000.00 | Debit
     10 | ABC12  | 110000.00 | Cash
     11 | ABC01  |  10000.50 | Credit
(11 rows)
```

Ticket

```
postgres=# select * from Ticket;
  pnr   | user_id | flight_no | seat_no | journey_date | destination |  source
--------+---------+-----------+---------+--------------+-------------+-----------
 ABC01  | U0001   |      3001 | A1      | 2023-03-15   | Chennai     | Mumbai
 ABC02  | U0002   |      6002 | B2      | 2023-03-16   | Bangalore   | Chennai
 ABC03  | U0004   |      5001 | C3      | 2023-03-17   | Mumbai      | Bangalore
 ABC04  | U0005   |      2002 | D4      | 2023-03-18   | Delhi       | Bangalore
 ABC05  | U0005   |      2002 | E5      | 2023-03-19   | Delhi       | Bangalore
 ABC06  | U0003   |      1002 | F6      | 2023-03-20   | Delhi       | Mumbai
 ABC07  | U0007   |      7001 | G7      | 2023-03-21   | Delhi       | Jaipur
 ABC09  | U0006   |      6001 | I9      | 2023-03-23   | Chennai     | Bangalore
 ABC10  | U0010   |      4001 | J10     | 2023-03-24   | Delhi       | Kolkata
 ABC11  | U0009   |      8002 | K11     | 2023-03-25   | Belgaum     | Bengaluru
 ABC12  | U0009   |      8002 | L12     | 2023-03-26   | Belgaum     | Bengaluru
(11 rows)
```

Class

```
postgres=# select * from Class;
 class_id | flight_no |  pnr   | class_type
----------+-----------+--------+------------
        1 |      3001 | ABC01  | E
        2 |      6002 | ABC02  | B
        3 |      5001 | ABC03  | E
        4 |      2002 | ABC04  | B
        5 |      2002 | ABC05  | E
        6 |      1002 | ABC06  | B
        7 |      7001 | ABC07  | E
        8 |      6001 | ABC09  | B
        9 |      4001 | ABC10  | E
       10 |      8002 | ABC11  | B
       11 |      8002 | ABC12  | E
(11 rows)
```

Economy

```
postgres=# select * from Economy;
 class_id |    fare
----------+-----------
        1 |  10000.50
        3 |  90000.25
        5 |  23450.00
        7 |  17175.75
        9 |  20550.00
       11 | 110000.00
(6 rows)
```

Business

```
postgres=# select * from Business;
 class_id |    fare
----------+----------
        2 | 31500.00
        4 | 55000.00
        6 | 80000.00
        8 | 12500.75
       10 | 11375.25
(5 rows)
```

Checksfor

```
postgres=# select * from Checksfor;
 chk_id | user_id | flight_no
--------+---------+-----------
      1 | U0001   |      3001
      2 | U0002   |      6002
      3 | U0003   |      1002
      4 | U0003   |      4001
      5 | U0004   |      5001
      6 | U0005   |      2002
      7 | U0007   |      7001
      8 | U0008   |      5001
      9 | U0009   |      8002
     10 | U0009   |      1002
     11 | U0010   |      4001
     12 | U0001   |      8002
     13 | U0004   |      1001
     14 | U0005   |      2002
     15 | U0001   |      5002
(15 rows)
```

Queries using basic dbms constructs join & subqueries:

Select empyoee ids.

```
postgres=# SELECT E_ID FROM EMPLOYEE;
 e_id
------
  501
  502
  503
  504
  505
  506
  507
  508
  509
  510
(10 rows)
```

Display employee name who lives in nadiad.

```
postgres=# SELECT E_name FROM employee WHERE E_address='NADIAD';
 e_name
--------
 AJAY
 KIYARA
 MIHIR
(3 rows)
```

Display blood stock before date 2022-09-20.

```
postgres=# SELECT * FROM blood_stock WHERE BS_date<'2022-09-20';
 bs_id |  bs_date   | i_id
-------+------------+------
  1901 | 2022-09-01 |
  1902 | 2022-09-02 |
  1903 | 2022-09-03 |
  1904 | 2022-09-04 |
  1905 | 2022-09-05 |
  1906 | 2022-09-06 |
  1907 | 2022-09-07 |
  1908 | 2022-09-08 |
  1909 | 2022-09-09 |
  1910 | 2022-09-10 |
(10 rows)
```

Display employee details with salary greater than 50000.

```
postgres=# SELECT * FROM employee e inner join salary s on e.S_id=s.S_id where s.Salary>50000;
 e_id | a_id | e_name | e_address | e_phoneno  |   e_dob    | s_id | s_id | salary
------+------+--------+-----------+------------+------------+------+------+--------
  501 |  101 | AJAY   | NADIAD    | 9898160331 | 2012-09-04 | 2102 | 2102 |  60000
  503 |  102 | SHIVA  | SURAT     | 7898763523 | 2013-07-09 | 2110 | 2110 | 100000
  504 |  103 | STACY  | BARODA    | 7987653432 | 2013-06-12 | 2108 | 2108 |  88000
  506 |  102 | DHYEY  | AHEMDABAD | 6733737373 | 2015-06-02 | 2105 | 2105 |  90000
  507 |  103 | MIHIR  | NADIAD    | 6574323342 | 2015-06-09 | 2104 | 2104 |  66000
  508 |  101 | HIL    | SURAT     | 7865653432 | 2013-09-03 | 2107 | 2107 | 100000
  510 |  102 | OM     | SURAT     | 8769765346 | 2015-09-08 | 2109 | 2109 |  85000
(7 rows)
```

Display blood details where blood group is AB+ve.

```
postgres=# SELECT * FROM blood where B_Group='AB+';
 b_id | b_group
------+---------
  203 | AB+
  212 | AB+
  216 | AB+
(3 rows)
```

Display count of blood having blood group AB+ve.

```
postgres=# SELECT count(B_Group) FROM blood where B_Group='AB+';
 count
-------
     3
(1 row)
```

Display employee name which employee number is highest.

```
postgres=# select *from employee WHERE E_id=(select max(E_id) FROM employee);
 e_id | a_id | e_name | e_address | e_phoneno  |   e_dob    | s_id
------+------+--------+-----------+------------+------------+------
  510 |  102 | OM     | SURAT     | 8769765346 | 2015-09-08 | 2109
(1 row)
```

Display Average Salary of emplyoee

```
postgres=# SELECT avg(salary) from salary;
        avg
--------------------
 70700.000000000000
(1 row)
```

Display the donor count grouped by blood group.

```
postgres=# select count(b.B_Group),b.B_Group as doner_group from blood b inner join doner d on b.B_id=d.B_id group by b.B_Group;
 count | doner_group
-------+-------------
     1 | AB-
     1 | AB+
     2 | A+
     1 | B-
     2 | B+
     1 | A-
     2 | O+
(7 rows)
```

Display name of employee and the query associated with him/her.

```
postgres=# select e.E_name,h.HD_Query from hd_emp he inner join help_desk h on he.HD_id= h.HD_id inner join employee e on he.E_id=e.E_id;
 e_name |       hd_query
--------+---------------------
 HIL    | BLOOD BOTTLE LOST
 MIHIR  | PAYMENT FAILED
 MIHIR  | IMPROPER SERVICE
 SHIVA  | WRONG BLOOD GROUP
 SUMAN  | DONOR NOT FOUND
 STACY  | BLOOD GROUP MISMATCH
 HIL    | PAYMENT FAILED
(7 rows)
```


Display id and name of employee in asc ordee whose salary is greater 50000 and id lies bertween 501 to 507

```
postgres=# SELECT E_id,E_name
postgres-# from
postgres-# employee
postgres-# where (E_id between 501 and 507
postgres(# AND S_id in (
postgres(#     select
postgres(#     S_id
postgres(#     from
postgres(#     salary
postgres(#     where
postgres(#     salary>50000
postgres(# ))
postgres-# order by
postgres-# E_name ASC;
 e_id | e_name
------+--------
  501 | AJAY
  506 | DHYEY
  507 | MIHIR
  503 | SHIVA
  504 | STACY
(5 rows)
```


Pl/Sql (Views):

Create view with employye name and address and display all details from it.

```
postgres=# create view E_name_address as select E_id,E_address from employee;
CREATE VIEW
postgres=# select * from E_name_address;
 e_id | e_address
------+-----------
  501 | NADIAD
  502 | NADIAD
  503 | SURAT
  504 | BARODA
  505 | ANAND
  506 | AHEMDABAD
  507 | NADIAD
  508 | SURAT
  509 | BARODA
  510 | SURAT
(10 rows)
```


5.6 FUNCTION & TRIGGERS:

Create a trigger to get new salary input which should not be less than 50000.

Function:

```
create function check_sal() returns trigger as $$
BEGIN
 if NEW.salary<50000 then
 raise exception 'Salary Must be greater than 50000';  end
if;
return NEW;
 END;
$$
 LANGUAGE plpgsql;
```

Trigger:

```
CREATE trigger sal_check
 BEFORE INSERT OR UPDATE
 ON salary
 FOR EACH ROW
 EXECUTE PROCEDURE check_sal();
```



```
postgres=# create function check_sal() returns trigger as $$
postgres$#  BEGIN
postgres$#  if NEW.salary<50000 then
postgres$#  raise exception 'Salary Must be greater than 50000';
postgres$#  end if;
postgres$# return NEW;
postgres$#  END;
postgres$# $$
postgres-#  LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# create trigger sal_check
postgres-#  BEFORE INSERT OR UPDATE
postgres-#  ON salary
postgres-#  FOR EACH ROW
postgres-#  EXECUTE PROCEDURE check_sal()
postgres-# ;
CREATE TRIGGER
```

Check: insert into salary values (2111,19000);

```
postgres=# insert into salary values (2111,19000);
ERROR:  Salary Must be greater than 50000
CONTEXT:  PL/pgSQL function check_sal() line 4 at RAISE
```

Create a function to count numbers of employees with salary between a user defined range.

Function:

```
CREATE function employeewithsalarybetween(salaryfrom int , salaryto int)
returns int
language plpgsql
as  $$
declare  numberofemp
integer;
begin select
count(*) into
numberofemp
from salary
where salary between salaryfrom and salaryto;
return numberofemp;
end; $$;
```

Check: select employee with salary between (50000,90000);

```
postgres=# CREATE function employeewithsalarybetween(salaryfrom int , salaryto int)
postgres-# returns int
postgres-# language plpgsql
postgres-# as
postgres-# $$
postgres$# declare
postgres$# numberofemp integer;
postgres$# begin
postgres$# select count(*)
postgres$# into numberofemp
postgres$# from salary
postgres$# where salary between salaryfrom and salaryto;
postgres$# return numberofemp;
postgres$# end;
postgres$# $$;
CREATE FUNCTION
postgres=# select employeewithsalarybetween(50000,90000);
 employeewithsalarybetween
---------------------------
                         6
(1 row)
```

Create a trigger for insertion of new employee details which should not belong to nadiad.

Function :

```
CREATE function check_emp_city4() returns trigger as $$
BEGIN
 if(NEW.E_address='Nadiad') then
 raise exception 'Employee must not belongs to nadiad!';  end
if;
return NEW;
 END;
$$
 LANGUAGE plpgsql;
```

Trigger:

```
create trigger city_check4
 BEFORE INSERT OR UPDATE
 ON employeed
 FOR EACH ROW
 EXECUTE PROCEDURE check_emp_city4();
```

Check: insert into employeed(E-id, A_id, E_ address, E_phoneno, E_dob, s_id) values(522,101,'AJAY','nadiad',989816031,'2012-09-04',2102);

```
postgres=# create function check_emp_city4() returns trigger as $$
postgres$#  BEGIN
postgres$#  if(NEW.E_address='Nadiad') then
postgres$#  raise exception 'Employee must not belongs to nadiad!';
postgres$#  end if;
postgres$# return NEW;
postgres$#  END;
postgres$# $$
postgres-#  LANGUAGE plpgsql;
CREATE FUNCTION
postgres=#
postgres=# create trigger city_check4
postgres-#  BEFORE INSERT OR UPDATE
postgres-#  ON employeed
postgres-#  FOR EACH ROW
postgres-#  EXECUTE PROCEDURE check_emp_city4();
CREATE TRIGGER
postgres=# INSERT INTO employeed (E_id, A_id, E_name, E_address, E_phoneno, E_dob, S_id) VALUES
postgres-# (522, 101, 'AJAY', 'Nadiad', 989816031, '2012-09-04', 2102);
ERROR:  Employee must not belongs to nadiad!
CONTEXT:  PL/pgSQL function check_emp_city4() line 4 at RAISE
```

Cursor

Create a cursor to select employee ID with greater than 501 and fetch the next employee ID

CURSOR SYNTAX:
Begin;
declare my_e_id cursor for
select e_id from employee where e_id > 501;
fetch next from my_e_id; close my_e_id;
cursor ;

```
postgres=# begin;
BEGIN
postgres=*# declare my_e_id cursor for
postgres-*# select e_id from employeed where e_id > 501;
DECLARE CURSOR
postgres=*# fetch next from my_e_id;
 e_id
------
  502
(1 row)


postgres=*# close my_e_id;
CLOSE CURSOR
```