# S1 T2 - Flight Tracking


**Prepared by**

Devesh Modi (201901173)

Piyush Suthar (201901404)

Jinesh Kanjiya (201901412)

Harshil Soni (201901416)

**Mentors**

Mr. Mayank sir and Mr. Harsh sir

**DAIICT**



*27-Nov-2021*

# Table of Contents

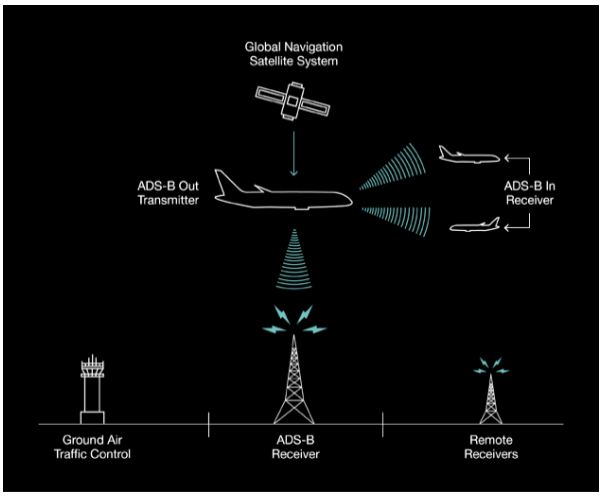
# Section 1: SRS

# A. Introduction

**Problem Statement :** Create a database for a System that tracks flight's sky paths. The system should keep track of flight paths that a plane needs to follow for a given day, and it needs to provide paths & weather information to pilots of those specific flights. It should also keep track of live feeds of plane locations during an ongoing flight. It should also be able to set alternative paths in case of weather or war emergencies. The database should be able to keep track of old trips and store alternate paths for flights operating above war areas during war emergencies.

## 1. Thinking and expanding problem domain

- **Real world working/flow, issues with the existing flow and counter measures to it.**
  Present trackers take data inputs from over sources like from the air traffic control systems, from airlines' systems, and directly from airplane cockpits that are over datalink. Also, most commercial flights are obtained in advance via schedules published by the airlines which remain unchanged until the last minutes.
  Now, the tracking moves to the real time paradigm from static. Pilots or air traffic controllers will push new data. When an aircraft is on the runway the aircraft flags 'out' and when it lifts its landing gears it flags 'off'. Now, when the aircraft is airborne, the tracker gets "realtime" data from the aircraft and also from the ground radar stations. This raw data is processed by the tracker software and presented.
  The landing is detected by the decrease in speed of the aircraft or the 'on' message sent out by the aircraft. On empting the runway the message 'in' is sent.
  The flaw in the above described widely deployed at present is that the real time tracking talked about above depends on the radio waves from fixed ground stations. This can cause a coverage break over many airspaces.
  The other deadly flaw is no nearby aircraft awareness which can result in the Charkhi-Dadri mid air collision in 1996 near Haryana.
  The counter to this limitation is observed as ADS-B technology that stands for Automatic Dependent Surveillance Broadcast. It works somewhat like the GPS in our phone allowing it to transmit its own location with the aid of a satellite. The aircraft can transmit its location and identification number along with other ADS-B data from its ADS-B transponder, which can be picked up by anyone with an ADS-B receiver–on the ground or in the air–within 200 miles.




Source: https://www.fastcompany.com/3044490/how-flight-tracking-apps-work

One more advantage of this novel approach is that the aircraft owners and operators have the option of blocking their tail numbers from public databases, and military flights are typically concealed.
Further, give pilots better awareness of nearby aircraft, and bring safer flows of traffic to the world's increasingly crowding skies.

Some of the limitations still prevailing are delay in the real time input-output, dependable on board system and interference in communication.

- **Purpose**

This database helps analyze, monitor and track Flights. Flight tracking is the process of acquiring real-time flight information, such as longitude, latitude, altitude, and ground speed of a specific aircraft. It is one of the key initiatives undertaken by regulatory agencies towards making aviation safe. Flight tracking is mainly carried out to ensure safe and efficient air traffic control and airline operations, and provide immediate response in case of an incident. Over a period, the flight tracking system has become a vital component in the aviation industry. There have been significant innovations in the flight tracking system front, owing to the technological advancements and improved software systems. The increasing demand for new aircraft, focus on ensuring flight safety, constant monitoring of aircraft, and the growing adoption of ADS-B flight tracking systems are the key factors driving the global market.

The main purpose for our product development should rely on the information fetched by the different target audience and to scale their implementation with reliability.

- **Intended Audience and Reading Suggestions**

The main target audience for this type of flight tracking system are:

- In flight travellers and customers
- Airport Authorities
- Defense companies
- Component Providers
- Investment Agencies
- Government and Regulatory Authorities

This SRS further contains, what is the necessity and future scope of this flight tracking system. How this flight tracking system is implemented is also described below.

- **Product Scope and necessity**

In India, the civil air space is under-utilised, that is only 60 % of the routes are utilised as per government reports. The airlines and authorities are trying to reduce the fuel cost. To reduce fuel cost, airlines often choose unfair means like deducting pilot salaries and over-timing pilots affecting their performance and increasing frequency of accidents. One of the severe accidents took place in Calicut in 2020 where 18 people lost their lives. After several incidents of flight missing in mid of their route ICAO(international civil aviation organization) has also decided to give pace to the world class flight tracking systems. But, on the other hand as per several estimates Rs. 1000 crore are wasted annually due to under-utilization of the air space in India. This is a paradox. Our flight tracking system database helps administer air space to solve these issues.

This database can also motivate private players or industrialists to enter airport authority business so that the country can get high-standard airport infrastructure eg. Adani Group which can bring investment and employment. This can also develop the industry of air ambulance services if the air-traffic management is proper. With acceptance and penetration of drones such databases will become a requirement in future.

- **Description**

The system will keep track of the aircraft path that a plane needs to follow for the flight. This will provide paths and weather information to the crew or the operators. This will keep track of live feeds of plane locations

during an ongoing flight. This will also be able to get alternative or available paths in case of weather, accident, air traffic, airport traffic or war emergencies. The database will have the record of near past trips.

The pilot would get the data of the upcoming trip before take-off. The pilot needs to be given instantaneous inputs if a zone is declared a no-fly zone. We plan to do this using triggers.

We can also consider the maximum number of trips before the maintenance is due and alert the ground crew at the present arrival airport. This can be implemented using triggers.

**The workflow of pilot :**

Firstly the pilot will receive information regarding it's scheduled flight date, time, departure airport. Now on the scheduled date and time the pilot will take off the flight and will aim to reach the destination on scheduled time and date. Now during the flight if the weather conditions are not good for the flight or is a no-fly zone then the pilot will receive a notification regarding the alternate routes. Also if there is an ongoing war at the destination airport location then the pilot needs to change it's destination and hence for this the flight tracking system will provide an alternative route. Also there may be some technical issues in flight due to which the pilot needs to do an emergency landing and for that flight tracking system can be provided with the nearest available airport.

**The workflow for flight :**

Flight includes all the aircrafts deployed by all the registered airlines which need the information about the weather and routing from the Airport authority class.

- Status of flight as explained in workflow
- Source and destination
- Estimated arrival and departure time
- Capacity
- Intermediate airports on the current route

Initially, the flights which are not on a trip will have status as 'available'. When the airport authority schedules a trip for that aircraft, it will update its source, destination, pilot id, scheduled departure and scheduled touch down. At the time of scheduled departure the status of the aircraft should be made 'on the run-way'. When the airport authority signals the aircraft for takeoff the status will be made 'on a trip'. Similarly after touchdown the concerned airport authority will update the status of the flight to 'available' or 'under maintenance' according to the number of trips for that particular flight. During the flight the real time location should be updated at a certain frequency.

**The workflow/ task for airport authority :**

The primary task for airport authority would be to signal the flights for eg. The airport authority will change the status of the flight to 'on runway' on the scheduled arrival time and 'on a trip' if the runway is available at the concerned airport. On touchdown of a flight the concerned airport makes the status of the flight as 'available'.If, the concerned airport authority updates itself as not available for cases like runway not available, bad weather or war has broken out then, the concerned flights has to be given an 'alert'.

**The workflow of users/passengers :**

First, the passenger will check the availability of flights and the arrival and departure time of the specific flight he/she needs, which route the flight will take.

**Salient features**

- **Flight Details** - It includes the originating flight terminal and destination terminal, along with the checkpoints in between. We can maintain flight capacity details and the number of seats booked/available seats on the trip.

- **Airport Details** - It includes all the airport details with airport unique-id, location, weather condition and war emergency situation.
- **Track Details** - It includes the past trip of the flights. Its actual time of arrival and departure and locations it travels.
- **Flight inquiry through query** - arrival date/time, name of airline company, flight number.
- **Airline companies** - It helps us to know which airline company operates the flight of our concern.

- **User Details** - It includes person type passenger/customer details.

- **Pilot Details** - We can maintain person type details for pilots like threshold working hours per week, salary etc.

- **Maintenance requirements** - Along with other aircraft details we can track threshold trips for maintenance requirements of the aircraft and alarm the pilot using triggers.

**Other feasible add-on features**

- This software may be used to even schedule the trips efficiently based on the customers and intermediate stops. for eg. applying optimization algorithms like PSO(Particle Swarm Optimization) to FFPP(Flight frequency programming problem).

# B. Document the Requirements Collection/ Fact Finding Phase

## 1. Background Readings:

- In order to get a proper understanding about flight tracking system i.e. how current location of flight is provided, how the tracks of flights can be tracked, providing weather information to pilots, setting an alternate route for flights, etc. we referred to plenty of websites, blogs, articles, youtube videos, user reviews, airline websites and even talked with one of our groupmate's relative who is currently working in airline services.
- To first get the basic understanding about flight tracking systems we went through the already available flight tracking systems and got the basic information about what all services they are providing, how they are providing and what technology they are using.
- Flightradar24 is a website which tracks every flight globally and represents that data to the user with a graphical interface on the world map with the source and destination of the flight. This is our motivation for this project.
- Previously many airlines used to track the current location of planes using satellite systems. But an ICAO(International Civil Aviation Organization) paper previously noted that for the planes that are having a very long journey, they already have systems on board in order to transmit their position. But sometimes this system is not always turned ON and as a result in some locations, including polar routes, there are gaps in satellite coverage. For eg:- An IFE system that offers broadband satellite communications, can be used to track an aircraft by doing slight modifications in it, which is the same as the datalink airlines which transmits engine performance data to maintenance departments.

**References**


- https://www.flightradar24.com/
- https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database
- https://pdfcoffee.com/airport-management-system-pdf-free.html
- https://timesofindia.indiatimes.com/city/chandigarh/haryanas-charkhi-dadri-plans-a-memorial-for-1996-mid-air-collision-victims/articleshow/81319907.cms
- https://www.fastcompany.com/3044490/how-flight-tracking-apps-work
- https://en.wikipedia.org/wiki/Flight_tracking
- https://www.youtube.com/watch?v=t2ahwbdCO8o
- https://www.youtube.com/watch?v=VacTNjI2-qQ&ab_channel=SouthBridge
- https://www.spidertracks.com/blog/introduction-to-flight-tracking
- https://m.gulf-times.com/story/613733/Tamper-proof-real-time-flight-tracking-is-vital
- https://www.nap.edu/read/10319/chapter/5#62

**Combined Requirements gathered from Background Reading/s.**

- Onboard tamper proof flight tracking system (ADS-B system)
- In case of extensive damage to the database due to catastrophic failure then recovery method should be able to store the previous backed up copy of database and by doing some operations it should be able to reconstruct the database
- The system shall permit only authorized members to do administrator's task and customers to view only their previous orders not other customers order
- The tracking software should use ADS-B transponder rather than using existing satellite communications (Due to delays and budget overrun).
- Manage the information about runway availability and status of the flight eg. on the runway, in air or landing.

## 2. Interviews

## 1. Interview Plan

**System:** Flight tracking Management System
**Project Reference:** JDHP/FTS/2021

**Interviewee:**
**1.** Mr. Satvik Singh (Role Play) **Designation:** CEO of FTA
**Contact Details:** satviks45@gmail.com

**Organization Details:**

**Interviewer:**
**1**. Jinesh Kanjiya
**Designation:** Business Development Executive, JDHP Solutions
**2**. Harshil Soni
**Designation:** Financial Manager and PR, JDHP Solutions

**Date:** 29/09/2021 **Time:** 14:30
**Duration:** 40 min **Place:** Google Meeting

**Purpose of Interview:**
- To get a better insight of the flight tracking system.
- To know about the contact points i.e. from where data can be collected.
- To know about the scale of the project.

**Agenda:**
- What is the role of an administrator.
- Current level of security and abstraction.
- What domains of data should be tracked.
- Required data storage safety

**Documents to be brought to the interview:**
- Rough plan or plan of currently deployed tracking systems.
- Notepad / google-doc (to note down important aspects)

## 1. Interview Summary

**System:** Flight tracking Management System
**Project Reference:** JDHP/FTS/2021

**Interviewee:**
**1.** Mr. Satvik Singh (Role Play) **Designation:** CEO of FTA
**Contact Details:** satviks45@gmail.com

**Organization Details:**

**Interviewer:**
**1**. Jinesh Kanjiya
**Designation:** Business Development Executive, JDHP Solutions
**2**. Harshil Soni
**Designation:** Financial Manager and PR, JDHP Solutions

**Date:** 29/09/2021 **Time:** 14:30
**Duration:** 40 min **Place:** Google Meeting

**Result of Interview:**
- Administrators are mainly concerned about a change in the tracking system.
- Administrators should decide what type of data should be accessible by users, airport authorities, etc.
- When a new flight is deployed or when a new pilot is recruited then it's data should be uploaded to the system.
- Also the data of which flight is travelled from which airport, what is its arrival/departure time and which route the flight has taken should be available.
- Administrators should have access to the entire system.
- Security of administrator level must be as high as possible, so that no confidential data should be leaked.
- Capital limitations/ requirements
- Facility to generate backup of current state of the database when wanted or periodically store the backup.

## 2. Interview Plan

**System:** Flight tracking Management System
**Project Reference:** JDHP/FTS/2021

**Interviewee:**
**1.** Ms. Devanshi Shah(Role Play) **Designation:** Manager, Airport Authority
**Contact Details:** shahdevanshi@gmail.com

**Organization Details:**

**Interviewer:**
**1**. Devesh Modi **Designation:** Developer, JDHP Solutions
**2**. Piyush Suthar **Designation:** Flight expert, JDHP Solutions

**Date:** 1/10/2021 **Time:** 09:00
**Duration:** 45 min **Place:** Google Meeting

**Purpose of Interview:**
- To know the Flight tracking system in more day-to-day functionalities/ requirements from the airport staff perspective.

**Agenda:**
- How frequently data should be tracked.
- How the data should be tracked.

**Documents to be brought to the interview:**
- Notepad/ google-doc (to note down important aspects)

## 2. Interview Summary

**System:** Flight tracking Management System
**Project Reference:** JDHP/FTS/2021

**Interviewee:**
**1.** Ms. Devanshi Shah(Role Play) **Designation:** Manager, Airport Authority
**Contact Details:** shahdevanshi@gmail.com

**Organization Details:**

**Interviewer:**
**1**. Devesh Modi **Designation:** Developer, JDHP Solutions
**2**. Piyush Suthar **Designation:** Flight expert, JDHP Solutions

**Date:** 1/10/2021 **Time:** 09:00
**Duration:** 45 min **Place:** Google Meeting

**Result of Interview:**
- There should be a separate database to store the information about flights and pilots.
- The condition of every airport under the authority should be stored.
- One separate database for tracking the flight, which includes which flight is travelled through which airport, their arrival/departure times. The data should be updated whenever the flight lands-to/take-off-from the airport.
- Data should be fetched with a certain frequency about an ongoing flight.

## 3. Interview Plan

**System:** Flight tracking Management System
**Project Reference:** JDHP/FTS/2021

**Interviewee:**
**1.** Capt. Rishabh Kapur (Role play)
**Designation:** President, Indian Commercial Pilots Association
**Contact Details:** caprishabhkapuricpa@gmail.com

**Organization Details:**

**Interviewer:**
**1**. Devesh Modi **Designation:** Developer, JDHP Solutions
**2**. Piyush Suthar **Designation:** Flight expert, JDHP Solutions

**Date:** 3/10/2021 **Time:** 09:00
**Duration:** 45 min **Place:** Google Meeting

**Purpose of Interview:**
- To know the technical requirements for the Flight tracking system.

**Agenda:**

- What fields are required for flight tracking.
- From where the raw data is to be fetched and how the raw data should be processed.
- To decide the optimal operating environment for the software from a flight perspective.

**Documents to be brought to the interview:**
- General blueprint of the aircraft.
- Notepad/ google-doc (to note down important aspects)

## 3. Interview Summary

**System:** Flight tracking Management System
**Project Reference:** JDHP/FTS/2021

**Interviewee:**
**1.**Capt. Rishabh Kapur (Role play)
**Designation:** President, Indian Commercial Pilots Association
**Contact Details:** caprishabhkapuricpa@gmail.com

**Organization Details:**

**Interviewer:**
**1**. Devesh Modi **Designation:** Developer, JDHP Solutions
**2**. Piyush Suthar **Designation:** Flight expert, JDHP Solutions

**Date:** 3/10/2021 **Time:** 09:00
**Duration:** 45 min **Place:** Google Meeting

**Result of Interview:**
- Got to know about the point of contacts for fetching raw data.
- Pilot details.
- To alert the airlines or pilot on exceeding weekly working hours limit.

## 4. Interview Plan

**System:** Flight tracking Management System
**Project Reference:** JDHP/FTS/2021

**Interviewee:**
**1.** Naimish Soni (Role play) **Designation:** President, Air Passenger Association of India
**Contact Details:** naimish_soni97@gmail.com

**Organization Details:**

**Interviewer:**
**1**. Jinesh Kanjiya
**Designation:** Business Development Executive, JDHP Solutions
**2**. Harshil Soni
**Designation:** Financial Manager and PR, JDHP Solutions

**Date:** 03/10/2021 **Time:** 16:00
**Duration:** 40 min **Place:** Google Meeting

**Purpose of Interview:**
- To know requirements specifically from a user/passenger perspective.

**Agenda:**

- What should be the benefits for the users.
- What domains of data a user will require.

**Documents to be brought to the interview:**

- Some statistical data of present passenger traffic.
- Notepad / google-doc (to note down important aspects)

## 4. Interview Summary

**System:** Flight tracking Management System
**Project Reference:** JDHP/FTS/2021

**Interviewee:**
**1.** Naimish Soni (Role play) **Designation:** President, Air Passenger Association of India
**Contact Details:** naimish_soni97@gmail.com

**Organization Details:**

**Interviewer:**
**1**. Jinesh Kanjiya
**Designation:** Business Development Executive, JDHP Solutions
**2**. Harshil Soni
**Designation:** Financial Manager and PR, JDHP Solutions

**Date:** 03/10/2021 **Time:** 16:00
**Duration:** 40 min **Place:** Google Meeting

**Result of Interview:**

- Users should have access to the data related to flight trips all over the route.
- Accurate arrival and departure time.
- Access to know capacity of flights, occupied and available seat ratios.
- Users should know charges before confirming tickets.
- Updation of the occupied and available seats in flight as per input given by airline companies.

**Combined Requirements gathered from Interview**

- To assign the most superior role to the administrator.
- Capital limitations/requirements.
- Separate database for storing information about flights, pilots, airports and users.
- Updating database when new pilot or flight is added.
- Updating the user database when a new customer tries to find the flight.
- To restrict data that will be viewed by users and airport authorities.
- Also the data of which flight is travelled from which airport, what is its arrival/departure time and which route the flight has taken should be available.
- Airport authorities should be able to communicate with the pilot and update them with the airspace condition.
- The airport authority should be able to reroute the remaining trip if there is alert or assign the nearest landable airport if the current airport is not available.
- Live location of flights at a certain frequency.
- Get required information from the airlines and present them to the users as per the requirements discussed above.
- Keeping track of working hours of the pilots and number of trips of the aircraft.

## 3. Questionnaire

We got the following responses from the google form we had circulated.

- **Question 1**

Your Name *

Your answer

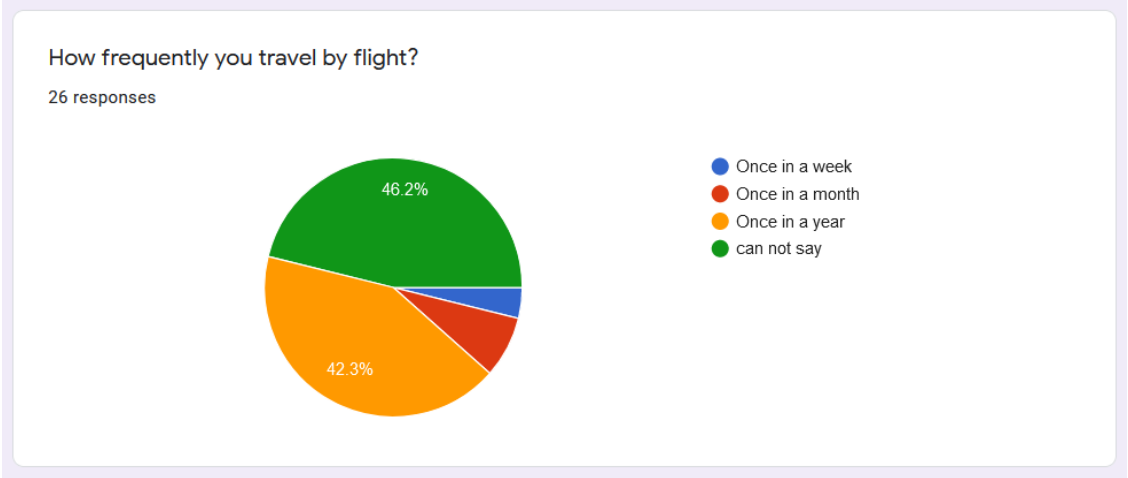
Occupation *

Your answer

- **Question 2**

How frequently you travel by flight? *

- Once in a week
- Once in a month
- Once in a year
- can not say



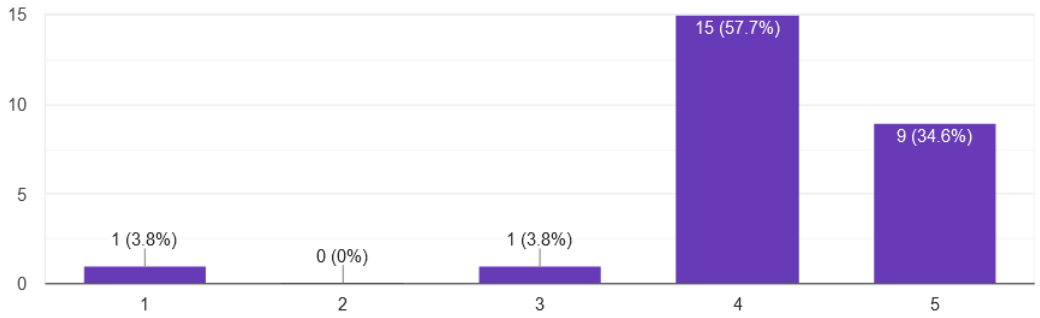
- **Question 3**

Please rate how important it is to create this flight tracking software? *

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Not Important | ○ | ○ | ○ | ○ | ○ | Extremely Important |

Please rate how important it is to create this flight tracking software?

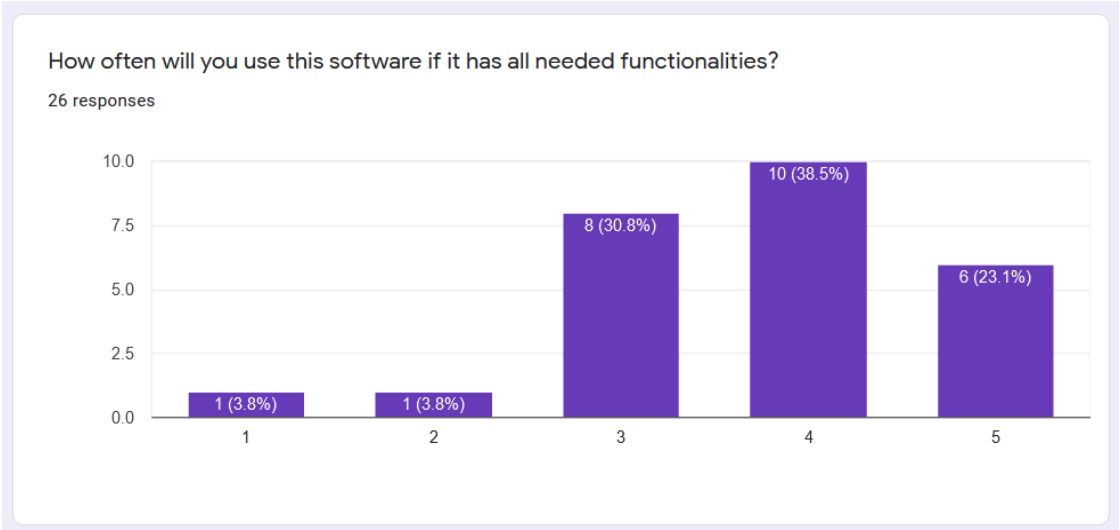
26 responses



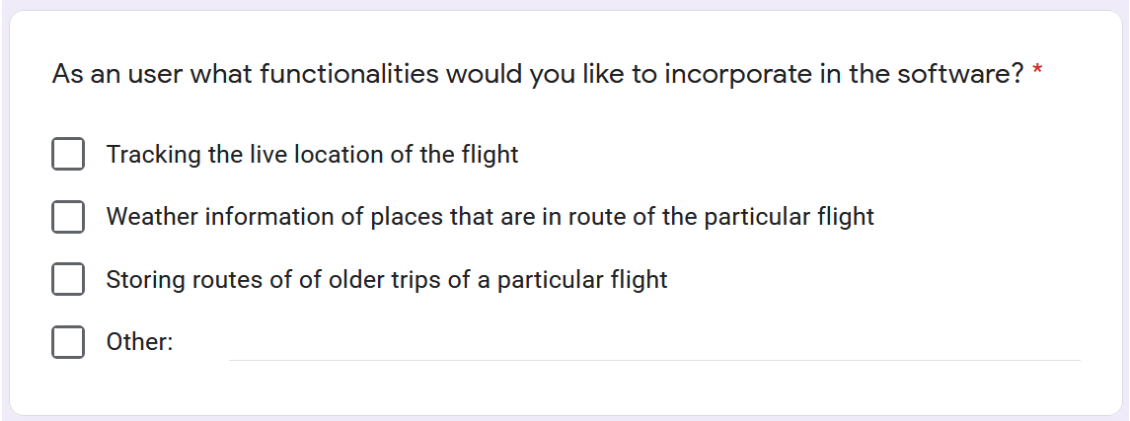

- **Question 4**

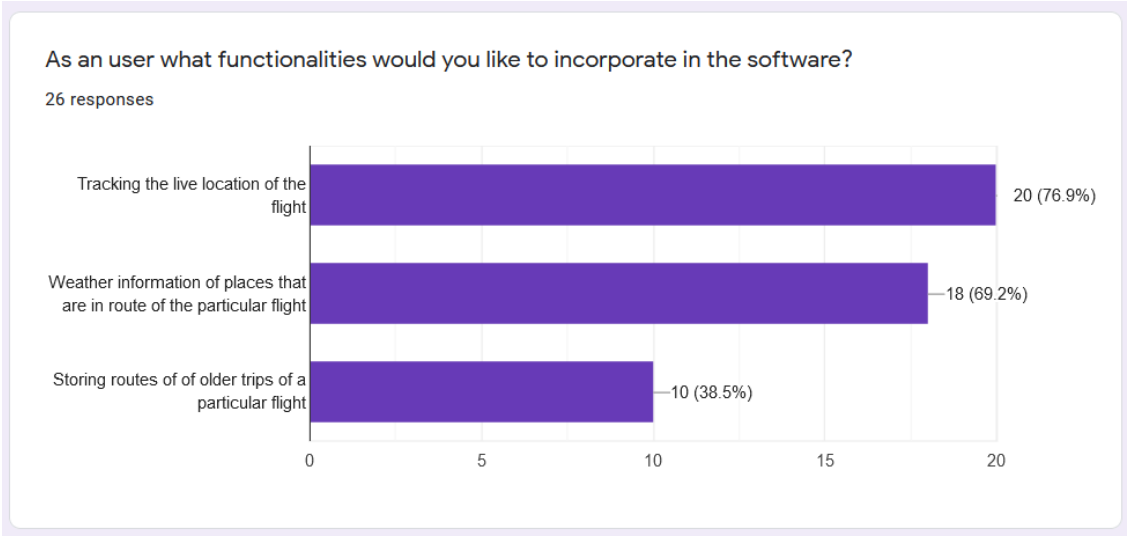
How often will you use this software if it has all needed functionalities? *

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Hardly Sometimes | ○ | ○ | ○ | ○ | ○ | Very Often |


How often will you use this software if it has all needed functionalities?
26 responses
10.0
7.5
5.0
2.5
0.0
1 (3.8%)
1 (3.8%)
8 (30.8%)
10 (38.5%)
6 (23.1%)
1
2
3
4
5

- **Question 5**

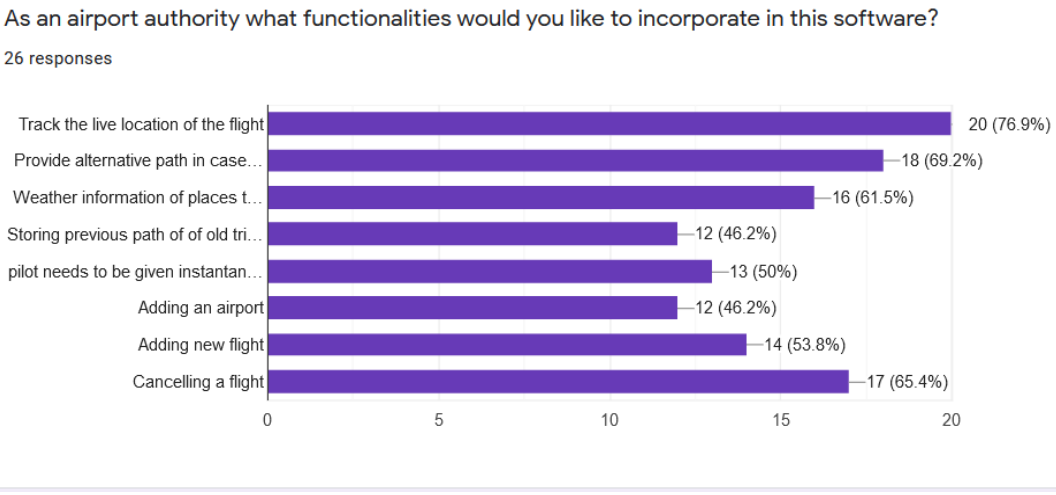

As an user what functionalities would you like to incorporate in the software? *
Tracking the live location of the flight
Weather information of places that are in route of the particular flight
Storing routes of of older trips of a particular flight
Other:


As an user what functionalities would you like to incorporate in the software?
26 responses
Tracking the live location of the flight
20 (76.9%)
Weather information of places that are in route of the particular flight
18 (69.2%)
Storing routes of of older trips of a particular flight
10 (38.5%)
0
5
10
15
20

- **Question 6**

As an airport authority what functionalities would you like to incorporate in this software? *

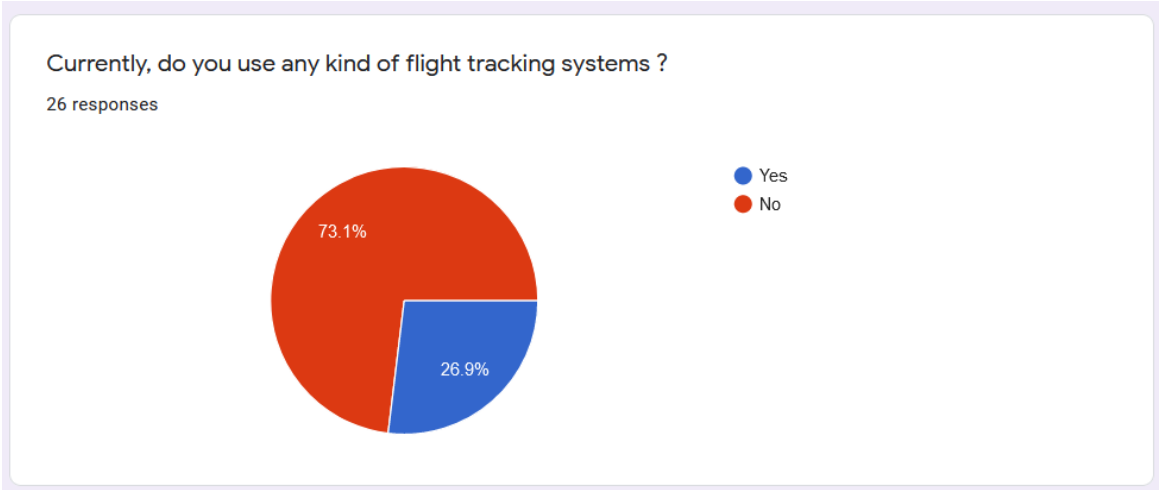
- [ ] Track the live location of the flight
- [ ] Provide alternative path in case of any emergency(i.e. Bad weather, War)
- [ ] Weather information of places that are in route of the particular flight
- [ ] Storing previous path of of old trips of a particular flight
- [ ] pilot needs to be given instantaneous input if any of the places in the flight route declared as a no-fly zone
- [ ] Adding an airport
- [ ] Adding new flight
- [ ] Cancelling a flight
- [ ] Other: _____



- **Question 7**

Currently, do you use any kind of flight tracking systems ? *

- Yes
- No



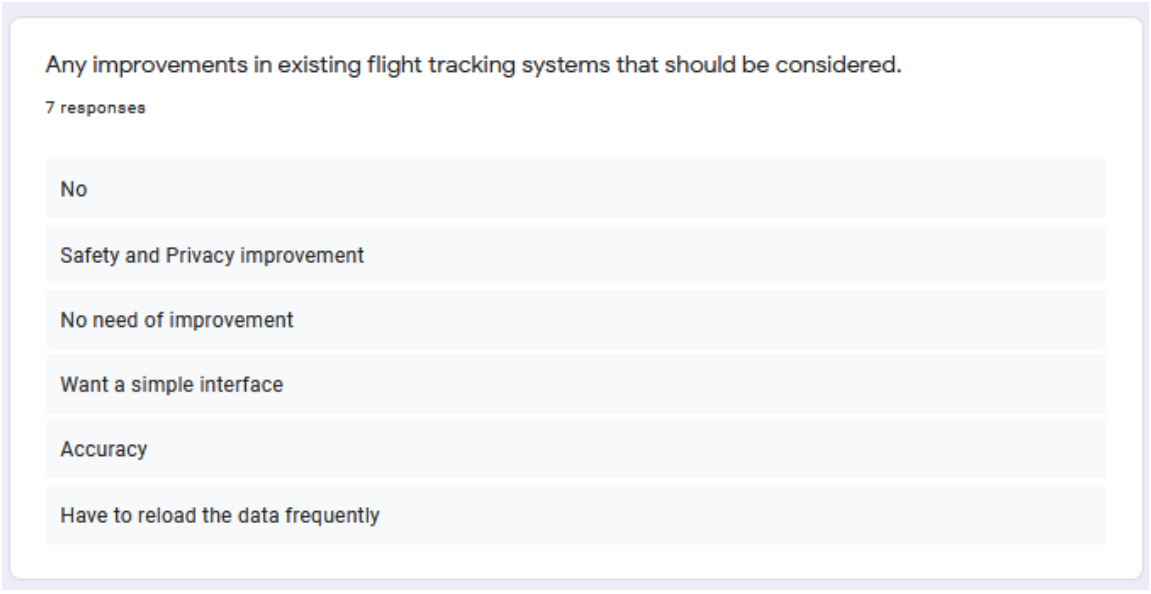

- **Question 8**

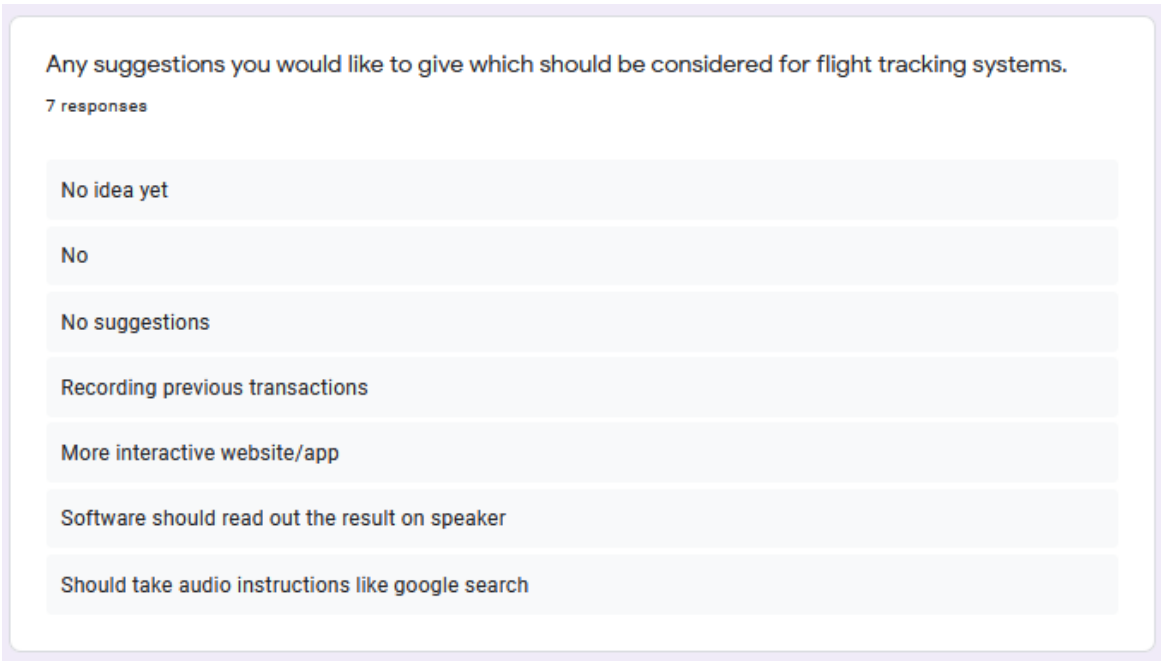
Any improvements in existing flight tracking systems that should be considered.

Your answer

Any suggestions you would like to give which should be considered for flight tracking systems.

Your answer




**Combined Requirements gathered from Questionnaire.**

- How frequent the passenger will travel from flight is 46.2% - can not say, 42.3 - once in a year. From this we can say that the use of the flight tracking system by the passengers would be very less. We need to design the system in such a way that the user who is not much familiar with it should also be able to use the system easily.
- From the third question we can say there is a high requirement of the flight tracking system.
- From the fourth question, the use of the flight tracking system would be high if we manage to achieve all the possible functionalities in our system.
- From the fifth question, the most admirable functionality for the normal users is to track the live location of the flight. Then the weather information and at last keeping track of old trips would be the least requirement for normal/users.
- From the sixth question, the most required functionalities for the airport authority would be tracking the live location of the flights, weather information in the route of particular flights, providing alternative paths in case

of emergency(bad weather), a continuous interaction with the pilot (in case of no-fly zone declared in any route), adding/cancelling a flight.

- From the seventh question, approximately 75% of people are not using the flight tracking system. So, we should also focus on the users other than passengers like pilots and airport authorities along with the ease of use for the passengers who are using it for the first time.

## 4. Observation

**JDHP Solutions: Observations**
**System:** Flight Tracking Management System
**Project Reference:** JDHP/FTS/2021
**Observations by:** JDHP Team
**Date:** 05/10/2021 **Time:** 10:00
**Duration:** 2 hours
**Place:** Google meet

**Observations:**

- Onboard tamper proof flight tracking system (ADS-B system)
- In case of extensive damage to the database due to catastrophic failure then recovery method should be able to store the previous backed up copy of database and by doing some operations it should be able to reconstruct the database
- The system shall permit only authorized members to do administrator's task and customers to view only their previous orders not other customers order
- The tracking software should use ADS-B transponder rather than using existing satellite communications (Due to delays and budget overrun).
- Manage the information about runway availability and status of the flight eg. on the runway, in air or landing.
- Assign the most superior role to the administrator.
- Separate database for storing information about flights, pilots, airports and users.
- Updating database when new pilot or flight is added.
- Updating the user database when a new customer tries to find the flight.
- To restrict data that will be viewed by users and airport authorities.
- Also the data of which flight is travelled from which airport, what is its arrival/departure time and which route the flight has taken should be available.
- Airport authorities should be able to communicate with the pilot and update them with the airspace condition.
- The airport authority should be able to reroute the remaining trip if there is alert or assign the nearest landable airport if the current airport is not available.
- Live location of flights at a certain frequency.
- Keeping track of working hours of the pilots and number of trips of the aircraft.
- How frequent the passenger will travel from flight is 46.2% - can not say, 42.3 - once in a year. From this we can say that the use of the flight tracking system by the passengers would be very less. We need to design the system in such a way that the user who is not much familiar with it should also be able to use the system easily.
- The use of the flight tracking system would be high if we manage to achieve all the possible functionalities in our system.
- The most admirable functionality for the normal users is to track the live location of the flight. Then the weather information and at last keeping track of old trips would be the least requirement for normal/users.
- The most required functionalities for the airport authority would be tracking the live location of the flights, weather information in the route of particular flights, providing alternative paths in case of emergency(bad weather), a continuous interaction with the pilot (in case of no-fly zone declared in any route), adding/cancelling a flight.
- Approximately 75% of people are not using the flight tracking system. We need to take care of the above points to increase the use of the Flight Tracking Management System.

## C. Fact Finding Chart

| Objective | Technique | Subject | Time Commitment |
|---|---|---|---|
| To get a proper understanding about flight tracking system | Background Reading | Blogs, Research papers, youtube | 1 day |
| Opinions/suggestion from people | Questionnaire | Batchmates and relatives | 1 day |
| To get a better insight of the flight tracking system. | Interview | CEO of FTA | 40 min |
| To know the Flight tracking system in a more day to day functional way. | Interview | Manager, Airport Authority | 45 min |
| To know the technical requirements for the Flight tracking system. | Interview | President, Indian Commercial Pilots Association | 45 min |
| To know requirements specifically from user perspective | Interview | President, Air Passenger Association of India | 40 min |
| Points taken out from the background reading, questionnaire and interviews | Observation | Business development executive, Developer, Financial manager and PR, Flight expert | 2 hours |

## D. List Requirements

- Diverse level of access rights. Full and partial level access to the database should be given accordingly.
  - ❖ Here for example users should be given access to view only some part of the database and users should not be able to modify that part. Similarly airport authority should be allowed to access some part of the database and also modify that part as per the need. Hence the system should provide different levels of access right according to the type of person using that database.

- Registered airlines and ability to add new airlines
  - ❖ All the registered airlines should be uploaded in the database and the database should have the ability to add new airlines.

- Data of pilots
  - ❖ Database should be able to modify the data of the pilots that are currently working with the airlines and also add new data of pilots as per the need.

- Location of airports and its availability.
  - ❖ There should be a real time update about the availability status of airports because if there is a war going at the airport location then that airport should not be available and hence it's status should be updated accordingly.

- Pilots, users, airport authorities should be granted specific rights to the system and administrators should have the ability to revoke or grant rights at any time.
    - ❖ System should be able to grant or revoke rights to the users, pilots, airport authorities separately so that we can have shared use of the system at the same time.

- Realtime/ high frequency update of airspace conditions.
    - ❖ In order to provide alternate routes to pilots in case of bad weather conditions there should be a real time update of weather in the system or if real time is not possible then the update of weather conditions should occur at high frequency.

- Aircraft details like remaining trips left before maintenance.
    - ❖ After the aircraft has successfully completed it's flight then the number of trips of that flight should be increased and after that it should be checked whether the flight should be sent under maintenance or not.

- Data backup needs
    - ❖ Generate backup of the current state of the database when wanted and also periodically store the backup.

- Number of runways available at the given instance at the given airport.
    - ❖ System should contain the data about the currently available runways at the specific airport

# E. User Class And Characteristics

## 1. Customer/Passenger Class:

This class includes normal person type users/passengers. This class will be interested in knowing the flight details like flight arrival/departure times, route, current location.

## 2. Airport Authority Class:

Airport authorities are the on ground staff which may be dedicated either to the airport crew or the runway security or maintenance staff. Airport authority which refers to the operators of the airports like Airport Authority of India(AAI), Adani Airport Holdings Ltd. (AAHL) etc.

- Runway availability information
- Flight arrival and departure time
- Routes of every flight
- Add or remove flight in service

## 3. Pilot/Crew member Class:

Pilot class includes all the pilots/crew employed by all the registered airlines which need the information about the weather and routing from the Airport authority class.

- Current flight location
- Working hours per week
- Salary
- Assigned flight
- Employer airline
- Flight details

## 4. Administrator Class:

Administrators class can be the dedicated staff for maintaining the database.

- JDHP Solutions

- Can limit or grant access to each of the above operator classes.

# F. Operating environment

## 1. Hardware, Software or Connectivity Requirements

- Stable internet connection.
- High speed internet connection- 10 Mbps or more for user
- Browser requirements
    - Chrome - 63 or above
    - Firefox - 52 or above
- PC/mobile with at least 2 GB of RAM for user
- Relevant setting/ permissions like storage permissions.
- At least 40 GB storage for database
- At least 1.2 GHz processor

## 2. External Interface Requirements (Hardware/Software/Third-party APIs and other things taken from other external sources.)

- A google account.
- C++ language
- Aviation Edge API (https://geekflare.com/flight-data-api/)
- SQL Server
- IDE (eg. VS code)

# G. Product Functions

1. Log in / Sign up
    - As per the type of user whether it's customer, airport authority, pilot or administrator the he/she should be able use the software.

2. Get flight details
    - With the help of this function the customer, pilot and airport authorities should be able to get/view details of the flights as per his/her need.

3. Get weather details
    - Airport authorities and pilots can view the weather details of the route of a specific flight by the help of this function.

4. Get route details
    - This function will provide the route of the specific flight.

5. Provide with an alternate route
    - If there is severe weather conditions or there is a war broken out for the upcoming route then this functionality will provide an alternate route to the pilot.

6. Emergency rerouting
    - If some conditions arise due to which the pilot needs to do an emergency rerouting then this function will provide the pilot with the nearest airport.

7. Working hours for pilots
    - This function will tell us about the number of hours the pilot has been working in this week.

8. Get current location of a flight
    - This will help us fetch the real-time location of an aircraft in terms of longitude and latitude.

9. Get the number of the trips
    - To know the number of many trips a particular flight has completed and from that we can decide whether to send that flight for maintenance or not.

10. Get status of a flight
    - By the help of this function we can know whether a particular flight is under maintenance, available, on runway, or on a trip as explained in workflow above.

11. Get the number of available runways
    - This function will provide us with the number of runways that are available at a specific airport.

12. Generate backup
    - By the help of this function the administrator will be able to create a master backup of the database.

## H. Privileges

The modules which each class can access :

- User Class
    - Log In / Sign Up
    - Get flight details
    - Get status of flight

- Pilot Class
    - Log In / Sign Up
    - Get flight details
    - Get weather details
    - Get route details
    - Get status of flight
    - Emergency landing

- Airport authority Class
    - Log in / Sign Up
    - Get flight details
    - Get status of flight
    - Provide with an alternate route
    - Get working hours of pilots
    - Get number of the trips for an aircraft
    - Get weather details
    - Provide an alternate route
    - Get the number of available runways
    - Grant pilot role to someone
    - Remove someone from pilot role

- Administrator Class

- All available functionalities
- Create/Delete a role
- Grant any privileges to any role
- Revoke any privileges from any role
- Generate backup

## I. Assumptions

- The weather conditions are available with the airport authorities.
- The airport authorities are always able to communicate with the flight.
- Technical information like Air Navigation Service Providers (ANSPs) for surveillance information are available from ADS-B transponders.
- The inputs are assumed for eg. When the flight takes off the input from the pilot has to be assumed that the runway is freed.

## J. Business Constraint

- The open source data like weather conditions would be available easily but the data like the broken out war would not be open source.
- There may be complexities like data sharing and competition when the different companies are the airport authorities of different airports.
- Also, the airline companies may be reluctant to share the data of their aircraft.
- The updating and maintaining the airspace conditions at all points and at all time instances is computationally, storage wise and economically not feasible which has to be compromised by frequent updates only.
- Storage services are required from the service providers to back up data.

# Section 2: Noun Analysis

# A. Final Problem Description

The system will primarily keep track of the current aircraft path that a plane is following or needs to follow for the upcoming trip. This will also store and update the paths and weather information.

This will also enable the pilot to get alternative available paths in case of bad weather, accident, air traffic, airport traffic or broken out war emergencies. The database will have the record of near past trips.

The pilot would get the data of the upcoming trip before take-off in advance along with the status of the flight like 'on a runway', 'on a trip', 'landing', 'waiting/ rerouting'. We can also consider the maximum number of trips before the maintenance is due and alert the ground crew at the present arrival airport authority.

The pilot needs to be given instantaneous inputs if a zone is declared a no-fly zone.

## Problem from pilot perspective :

In the first place, the pilot will receive information regarding it's scheduled flight date, time, departure airport. Now on the scheduled date and time the pilot will take off the flight and pilot will be able to see the map of route that the flight will follow and pilot will aim to reach the destination on scheduled time and date.

During an ongoing flight if there are war conditions at the checkpoints which come in the route of a flight or if the destination airport is not safe for the flight to land or is a no-fly zone then the pilot will receive a notification regarding the alternate route.

Also in this case the pilot needs to change it's destination and hence for this the flight tracking system will provide an alternative route/destination along with longitude and latitude of checkpoints in the alternate route.

If there is some technical issue with the flight, then the pilot immediately contacts the airport authority to provide the nearest landing location.

During the flight if the weather between the current source(checkpoint) and next checkpoint(destination) is not suitable for the flight then the pilot needs to reroute and for that, he should be able to get the information about new coordinates for alternate routes based on the coordinates of next checkpoints(destination) based on ranks of preference for new checkpoint.

## Problem from flight/ aircraft perspective :

Initially, the flights which are not on a trip or under maintenance should have status as 'available'.

When the airport authority schedules a trip for that aircraft, it will update its source, destination, pilot id, scheduled departure and scheduled touch down.

At the time of scheduled departure the status of the aircraft should be made 'on the run-way' if available.

When the airport authority signals the aircraft for takeoff the status should be made 'on a trip'.

Similarly after touchdown the concerned airport authority will update the status of the flight to 'available' or 'under maintenance' according to the number of trips for that particular flight.

During the flight the real time location should be updated in the real time domain.

## Problem from airport perspective :

The airport authority will have the weather data and its safety conditions.

The primary problem of the airport is to route the flights with a sense of above two data in consideration.

The primary task for airport authority would be to signal the flights for eg. The airport authority will change the status of the flight to ‘on runway’ on the scheduled arrival time and ‘on a trip’ if the runway is available at the concerned airport.

On touchdown of a flight the concerned airport makes the status of the flight as ‘available’. If the concerned airport authority updates the airport status as not available because of cases like runway not available, bad weather or war has broken out then, the concerned flights have to be given an ‘alert’.

## Problem from passenger perspective :

First, the passenger will check the availability of flights and the arrival and departure time of the specific flight he/she needs, which route the flight will take.

The passenger should only see the data like scheduled/ actual arrival or departure time, whichever is available, status of the aircraft and the current or the latest location of the flight.

## Features:

- **Flight Details** - It includes the originating flight terminal and destination terminal along with the checkpoints in between the status of the trip. We can maintain flight capacity details.
- **Airport Details** - It includes all the airport details with airport unique-id, location, weather condition and war emergency situation.
- **Old trip Details** - It includes the past trip of the flights. Its actual time of arrival and departure and intermediate locations it travels.
- **Rerouting** - In case of emergency from flight side like bird-hit or from airport side like war or bad weather, an alternate route is to be provided to the pilot.
- **Flight inquiry through query** - arrival date/time, name of airline company, flight number, current location.
- **Airline companies** - It helps us to know which airline company operates the flight of our concern.
- **User Details** - It includes person type passenger/ customer details.
- **Pilot Details** - We can maintain person type details for pilots like threshold working hours per week, salary etc.
- **Maintenance requirements** - Along with other aircraft details we can track threshold trips for maintenance requirements of the aircraft and alarm the pilot using triggers.

The problem demands that the initial trip details should be available from the real world, for example, say an airline's schedule. Also, at times, the human real world triggers or inputs should be obtained and based on the behaviour defined on particular inputs the database should be recorded.

To further understand the problem, we are required to conduct interviews and circulate questionnaires.

The tracking software should use ADS-B transponder based approach rather than using existing satellite communications.

The onboard hardware interface should be tamper proof that means minimal effect of the human error by crew on the data sent.

In case of extensive damage to the record due to any failure then the recovery method should be able to restore the latest available version of the backed up copy of the database.

The problem description demands to store information about flights, pilots, airports and users separately.

As per the objective of the problem, the users **only need to** view the scheduled flights with its details like source, destination, expected arrival and departure times and live location. On the sidelines of the main requirement we can store the user id and corresponding password. Even for airport authorities and pilots to see their relevant details we can store id and password.

The past or old trips are the records in the trips which have status completed.

If there arises a need to assign the alternative path, the concerned intermediate node is to be changed. Thus, we will have the actual path followed by the flight at last.

The database should be updated when a new pilot or flight is added.

In an aircraft there will be many sensors details like sensors which give us the value of the current altitude, sensors which will notify us about how much fuel is left, also some sensors will help us to know about the speed of the aircraft and we need to have the readings of these sensors. Flight will have fuel capacity details already stored.

Airport authorities should be able to communicate with the pilot and update them with the airspace condition.

If there is an alert like bad weather condition for an ongoing flight or war ongoing at an airport then airport authority should be able to provide an alternate route as per the mechanism discussed above.

The primary problem requirement is that the flight tracking system should provide the live location of a flight and update it at a certain frequency.

Flight tracking system should be able to keep track of the number of trips an aircraft has completed and on the basis of that it will decide whether the aircraft should be sent for maintenance or not. Also the number of hours a pilot had worked should also be tracked.

The flight tracking system should be designed in such a way that it is user friendly and also the user who is using it for the first time should be able to use all the functionalities available in the system.

Some of the functionalities like tracking the live location of the flights, weather information in the route of flights, a continuous interaction with pilot when there is no fly zone, adding or cancelling a flight, providing alternative paths in case of an emergency, etc. are most used by the airport authorities and so these functionalities should be available to airport authorities.

In order to increase the use of the flight tracking management system we need to take care of the above points and include it in the flight tracking management system.

## B. Noun Analysis

### 1. Noun and Verbs table

| **NOUN** | **VERB** |
|---|---|
| software | will |
| contents | following |
| list | enable |
| weather | broken |
| phase | take - off |
| classes | needs |
| maintenance | can |
| airlines | receive |
| data | store |
| details | issue |
| flight | repair |
| user | having |
| questionnaire | travel |
| arrival | frequently |
| phase | assign |
| checkpoints | available |
| authority | inquiry |
| ability | provided |
| depo | keep |
| domain | trip |
| specification | get |
| domain | out |
| background | trip |
| alternative | keep |

| | |
|---|---|
| requirements | assign |
| location | regarding |
| class | time |
| locations | landing |
| war | check |
| sensor | issue |
| frequency | booking |
| capacity | view |
| source | condition |
| departure | communicate |
| routes | visit |
| condition | delete |
| administrator | track |
| query | store |
| problem | fly |
| zone | war |
| fact | estimate |
| interview | declared |
| coordinates | working |
| user | consider |
| old_trips | was |
| information | scheduled |
| characteristics | provide |
| airport | take - off |
| classes | get |
| longitude | find |
| latitude | alert |
| plane | is |

| | |
|---|---|
| trips | restrict |
| live | recovery |
| passenger | increase |
| location | need |
| requirements | add |
| problem | is |
| airspace | update |
| system | traffic |
| fund | advance |
| bird hit | reroute |
| runway | alert |
| speed | inquiry |
| real | view |
| version | arrival |
| map | restrict |
| characteristics | takes |
| observation | reading |
| intermediate | visit |
| product | be |
| Airport | fly |
| problem | register |
| aircraft | collects |
| paths | keep |
| rank | estimate |
| pilot | will |
| old_trips | refers |
| system | update |
| emergencies | store |

| | |
|---|---|
| function | available |
| status | find |
| trip | gathers |
| speed | notify |
| airport | |
| track | |
| hours | |
| weather | |
| altitude | |
| reason | |
| data | |
| proof | |
| password | |

## 2. Truncating initial noun list

- **Accepted Noun and Verb List**

| **Candidate entity set** | **Candidate Attribute set** | **Candidate relationship set** |
|---|---|---|
| flight | flight_ID, capacity, airline, status, n_trips, location, speed, fuel_capacity | on |
| user | user_ID, username, email, password | --- |
| trips | trip_ID, arrival_time, departure time, source,destination,trips_checkpoints | on, readings, takes, refer, collects, flies |
| sensor_details | rank, time,fuel_value, altitude, longitude, latitude, speed | readings |
| coordinates | c_ID, current_location, next_location, condition, alternate | refer, map |
| all checkpoints | longitude, latitude, name | takes, map |

| | | |
|---|---|---|
| airport | airport_ID, name, condition, location, runways, alternate_airports | collects, notify |
| pilot | pilot_ID, password, working_hours | flies, notify |

- **Rejected Noun and Verb List**

| **Noun/verb** | **Reject reason** |
|---|---|
| software | General |
| contents | General |
| list | Vague |
| weather | Attributes |
| phase | Irrelevant |
| classes | General |
| maintenance | Vague |
| airlines | Attributes |
| data | general |
| details | General |
| questionnaire | Irrelevant |
| arrival | Attributes |
| phase | Duplicates |
| authority | Associations |
| ability | General |
| depo | General |
| domain | General |
| fact | General |
| interview | Irrelevant |
| information | Irrelevant |
| characteristics | General |
| classes | Vague |
| plane | Irrelevant |

| | |
|---|---|
| live | General |
| passenger | Duplicates |
| location | Attributes |
| requirements | General |
| problem | General |
| airspace | Attributes |
| system | Vague |
| fund | Vague |
| bird hit | Vague |
| runway | Attributes |
| real | General |
| specification | General |
| background | General |
| alternative | Attributes |
| location | Duplicates |
| class | Duplicates |
| locations | Duplicates |
| war | Attributes |
| frequency | General |
| capacity | Attributes |
| source | Attributes |
| departure | Attributes |
| condition | Vague |
| administrator | General |
| query | Vague |
| problem | Irrelevant |
| zone | Vague |
| version | Irrelevant |

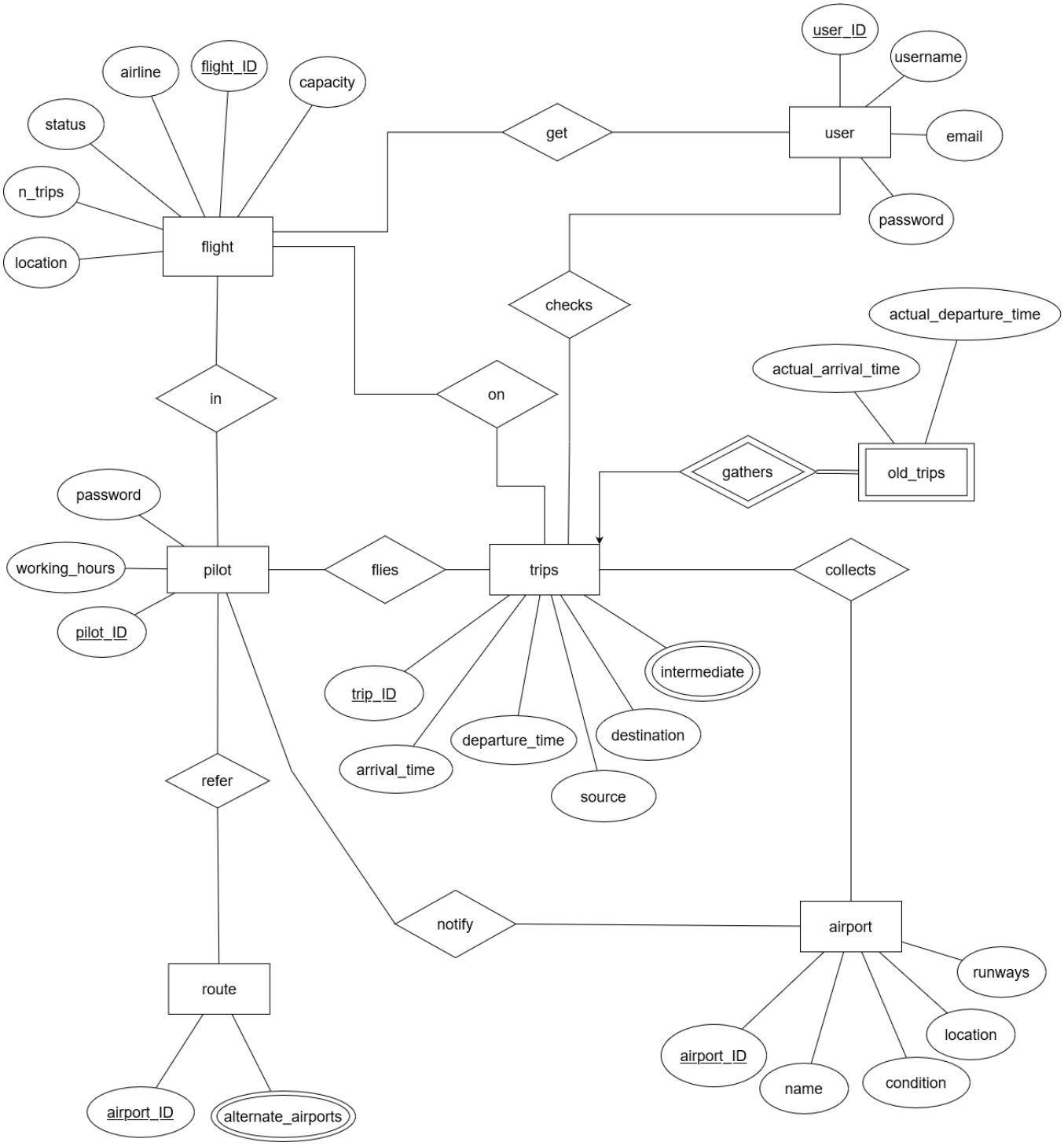
| | |
|---|---|
| observation | Irrelevant |
| product | Vague |
| problem | Duplicates |
| paths | Irrelevant |
| system | General |
| emergencies | Vague |
| function | General |
| status | Attributes |
| weather | Irrelevant |
| proof | Vague |
| aircraft | General |
| will | Irrelevant |
| following | Irrelevant |
| enable | General |
| broken | Vague |
| take - off | General |
| needs | Irrelevant |
| can | Irrelevant |
| receive | General |
| store | General |
| issue | Irrelevant |
| repair | General |
| having | General |
| travel | General |
| frequently | Vague |
| assign | General |
| inquiry | General |
| keep | Irrelevant |

| | |
|---|---|
| get | General |
| out | General |
| trip | Duplicates |
| keep | Duplicates |
| assign | Duplicates |
| regarding | General |
| time | General |
| landing | General |
| check | Associations |
| booking | Vague |
| view | Vague |
| condition | Attributes |
| communicate | General |
| working | General |
| delete | Vague |
| track | General |
| store | General |
| fly | General |
| war | Irrelevant |
| estimate | Vague |
| declared | General |
| consider | Irrelevant |
| scheduled | General |
| provide | General |
| take - off | irrelevant |
| get | General |
| is | General |
| restrict | Vague |

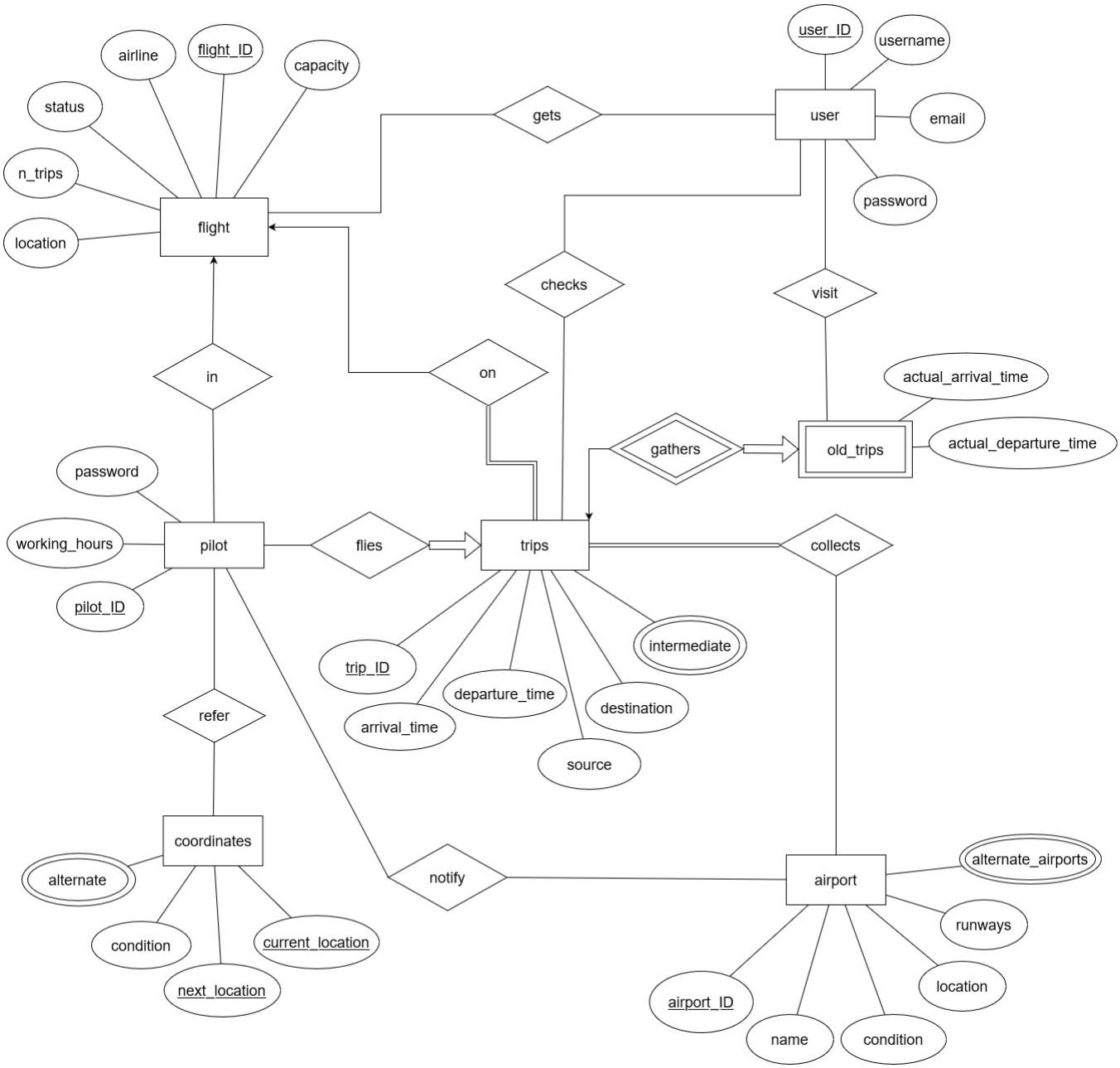
| | |
|---|---|
| recovery | General |
| increase | General |
| need | General |
| add | General |
| is | Duplicates |
| update | Irrelevant |
| traffic | Irrelevant |
| advance | Irrelevant |
| alert | Irrelevant |
| view | Duplicates |
| arrival | Vague |
| reading | Vague |
| be | General |
| fly | Irrelevant |
| register | Irrelevant |
| keep | General |
| will | Duplicates |
| update | General |
| reroute | Irrelevant |
| store | Duplicates |
| available | Irrelevant |
| find | General |

# Section 3: ER-Diagrams all versions.

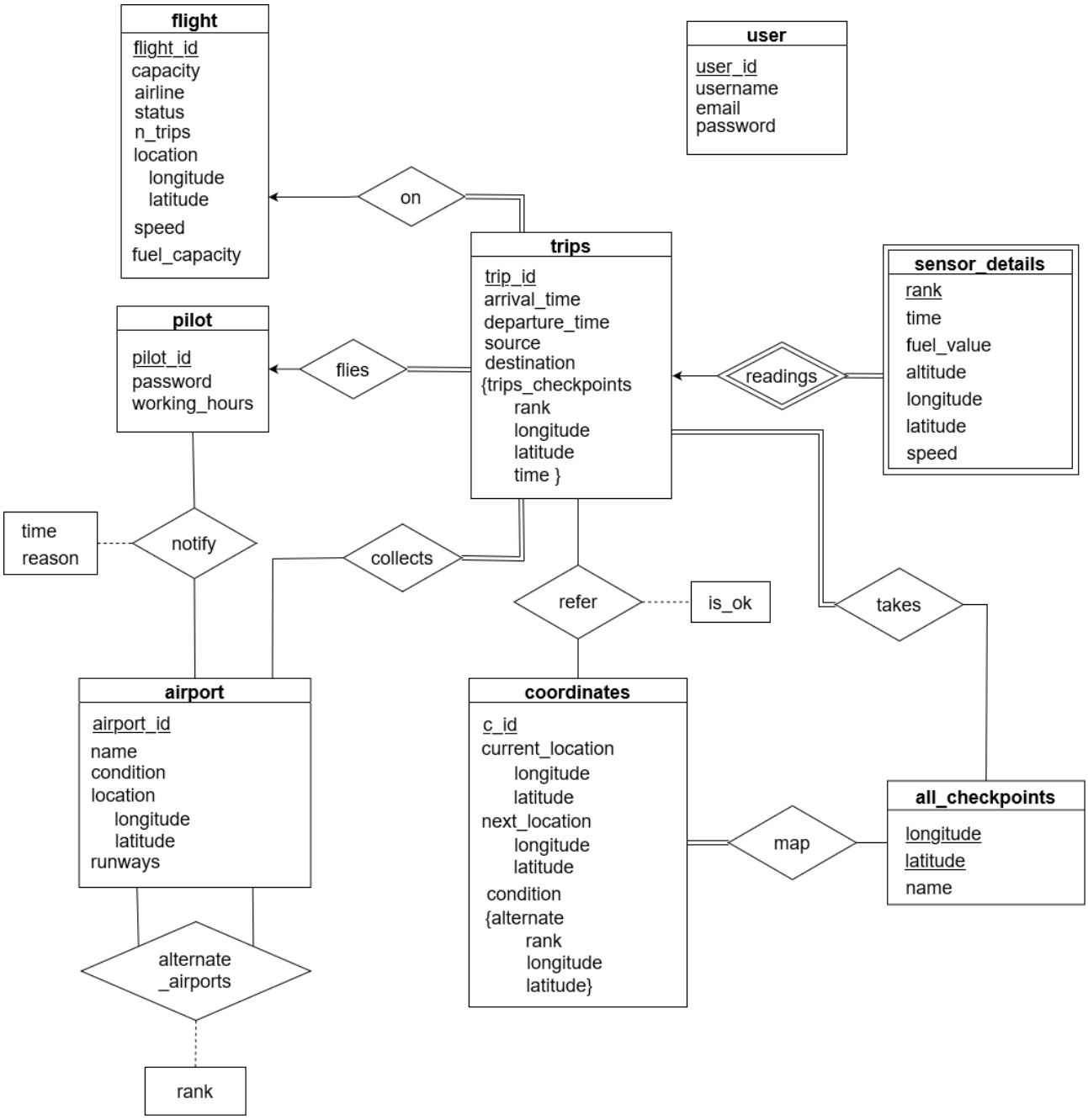
# I. ER Diagram Version-1



# II. ER Diagram Version-2



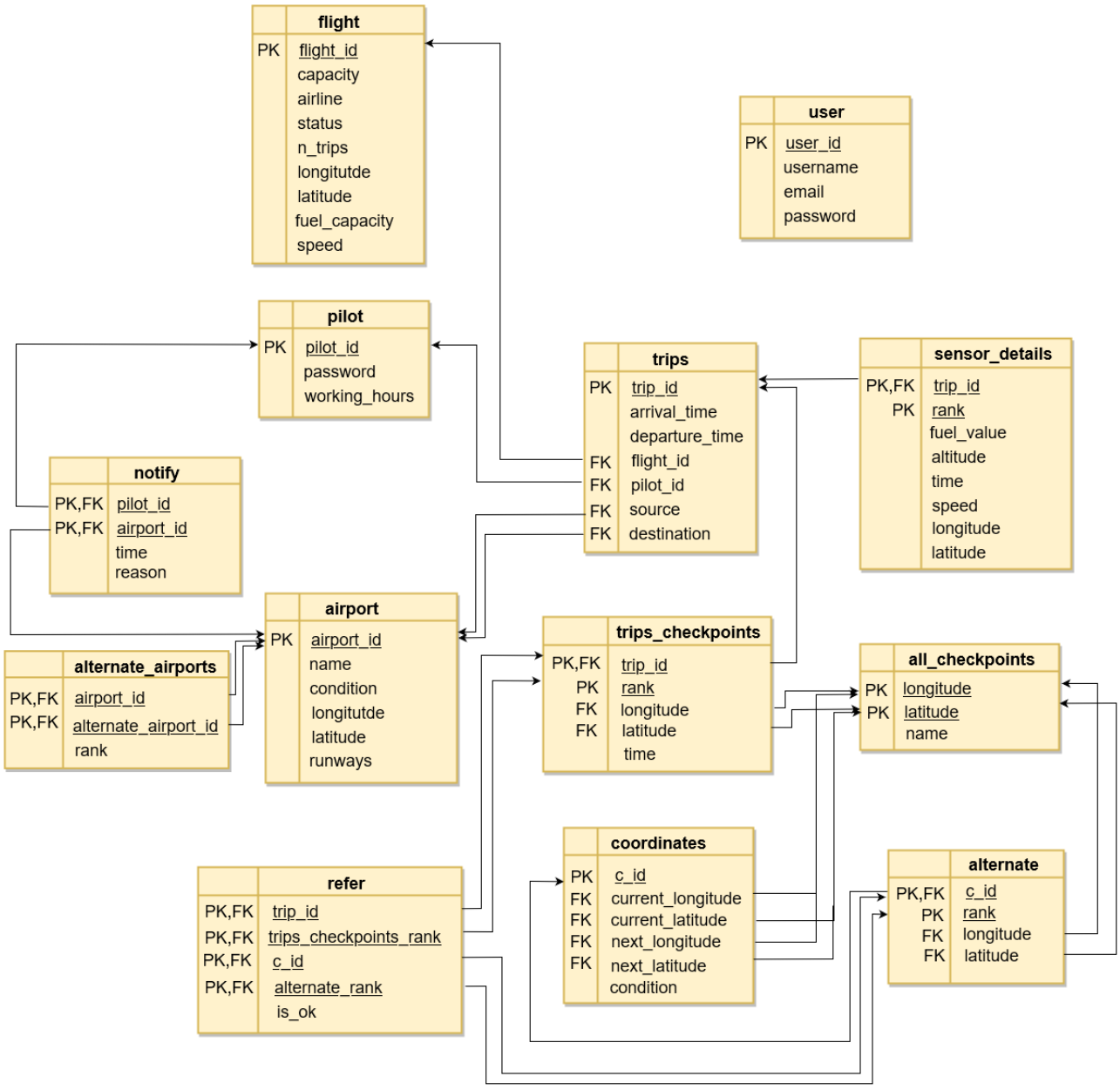
## III. ER Diagram Version-3 (Final)



# Section 4: Conversion of FinalER-Diagram to Relational Model.

# I. Mapping E-R Model to Relational Model

## 1. Relational model

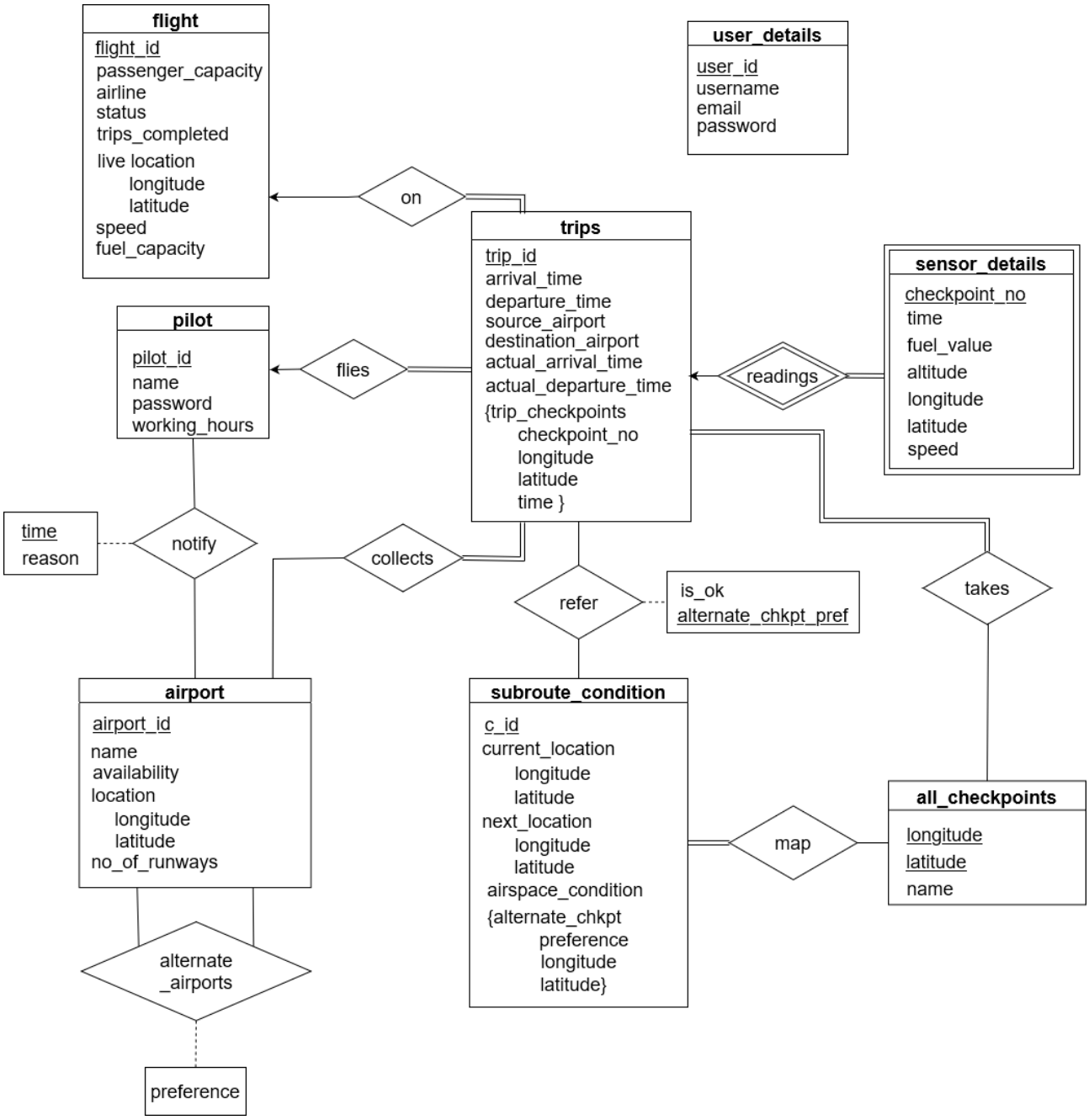


## 2. Relational Schema

1. user(**user_id**, username, email, password)
2. flight(**flight_id**, capacity, airline, status, n_trips, longitude, latitude, fuel_capacity, speed)
3. pilot(**pilot_id**, password, working_hours)
4. trips(**trip_id**, source, destination, arrival_time, departure_time, flight_id, pilot_id)
5. sensor_details(**trip_id, rank**, fuel_value, altitude, time, speed, longitude, latitude)
6. airport(**airport_id**, name, condition, longitude, latitude, runways)
7. alternate_airports(**airport_id**, **rank**, alternate_airport_id)
8. trips_checkpoints(**trip_id**, **rank**, longitude, latitude, time)
9. all_checkpoints(**longitude**, **latitude**, name)
10. coordinates(**c_id**, current_longitude, current_latitude, next_longitude, next_latitude, condition)
11. alternate(**c_id**, **rank**, longitude, latitude)
12. notify(**pilot_id**, **airport_id**, time, reason)
13. refer(**trip_id**, **checkpoint.rank**, **c_id**, **alternate.rank**, is_ok)

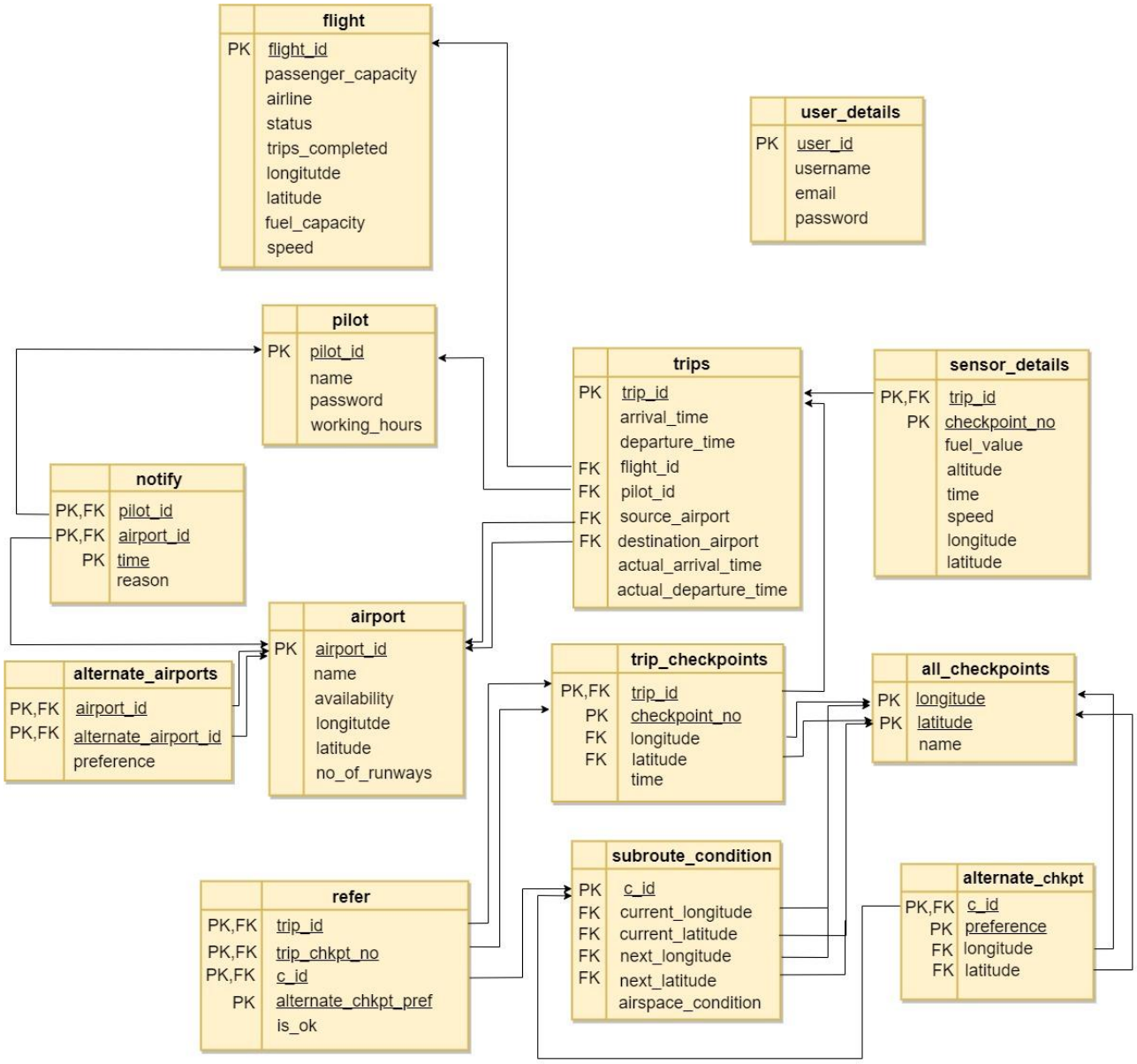
# Section 5: Normalization and Schema Refinement.

# I. Final ER Diagram



# II. Normalization & Schema Refinement

## 1) Relational Model(Original Design of Database)



## 2) Identify and list all types of dependencies ( PK, FK, Functional Dependencies) for each relation

1. user_details
    - CK({user_id},{email})
    - PK(user_id)
    - NO FK
    - Functional dependencies (user_id->username , user_id->email , user_id->password , email->username , email->email , email->password)

2. flight
    - CK({flight_id})
    - PK(flight_id)
    - NO FK
    - Functional dependencies (flight_id->passenger_capacity , flight_id->airline , flight_id->status , flight_id->trips_completed , flight_id->longitude , flight_id->latitude , flight_id->fuel_capacity , flight_id->speed)

3. pilot
    - CK({pilot_id})
    - PK(pilot_id)
    - NO FK
    - Functional dependencies (pilot_id->name , pilot_id->password , pilot_id->working_hours)

4. trips
    - CK({trip_id}, {flight_id, departure_time}, {pilot_id, departure_time})
    - PK(trip_id)
    - FK flight_id references flight(flight_id)
    - FK pilot_id references pilot(pilot_id)
    - FK source_airport references airport(airport_id)
    - FK destination_airport references airport(airport_id)
    - Functional dependencies(trip_id->arrival_time , trip_id->departure_time , trip_id->flight_id , trip_id->pilot_id , trip_id->source_airport , trip_id->destination_airport , trip_id->actual_arrival_time , trip_id->actual_departure_time , {flight_id, departure_time} -> {arrival_time , departure_time , flight_id , pilot_id , source_airport , destination_airport , actual_arrival_time , actual_departure_time } , {pilot_id, departure_time} -> {arrival_time , departure_time , flight_id , pilot_id , source_airport , destination_airport , actual_arrival_time , actual_departure_time} )

5. sensor_details
    - CK({trip_id, checkpoint_no}, {trip_id, time})
    - PK({trip_id, checkpoint_no})
    - FK trip_id references trips(trip_id)
    - Functional dependencies ( {trip_id, checkpoint_no} -> fuel_value, {trip_id, checkpoint_no} ->altitude , {trip_id, checkpoint_no} -> time, {trip_id, checkpoint_no} ->speed, {trip_id, checkpoint_no} ->longitude, {trip_id, checkpoint_no} ->latitude , {trip_id, time} -> {fuel_value, altitude, time, speed, longitude, latitude} )

6. airport
    - CK({airport_id}, {longitude, latitude})

- PK(airport_id)
- NO FK
- Functional dependencies(airport_id->name , airport_id->availability , airport_id->longitude , airport_id->latitude , airport_id->no_of_runways , {longitude,latitude} -> {airport_id,name,availability,no_of_runways} )

7. alternate_airports
   - CK({airport_id, alternate_airport_id}, {airport_id, preference})
   - PK(airport_id, alternate_airport_id)
   - FK airport_id references airport(aiport_id)
   - FK alternate_airport_id references airport(aiport_id)
   - Functional dependencies({airport_id,alternate_airport_id}->preference , {airport_id, preference} -> {airport_id, alternate_airport_id, preference})

8. trip_checkpoints
   - CK({trip_id, checkpoint_no})
   - PK({trip_id, checkpoint_no})
   - FK trip_id references trips(trip_id)
   - FK (longitude, latitude) references all_checkpoints(longitude, latitude)
   - Functional dependencies({trip_id, checkpoint_no}->longitude , {trip_id, checkpoint_no}->latitude , {trip_id, checkpoint_no}->time)

9. all_checkpoints
   - CK({longitude, latitude})
   - PK({longitude, latitude})
   - NO FK
   - Functional dependencies({longitude, latitude}->name)

10. subroute_condition
    - CK(c_id)
    - PK(c_id)
    - FK (current_longitude, current_latitude) references all_checkpoints(longitude, latitude)
    - FK (next_longitude, next_latitude) references all_checkpoints(longitude, latitude)
    - Functional dependencies(c_id->current_longitude , c_id->current_latitude , c_id->next_longitude , c_id->next_latitude , c_id->airspace_condition)

11. alternate_chkpt
    - CK({cid, preference})
    - PK({c_id, preference})
    - FK c_id references subroute_condition(c_id)
    - FK (longitude, latitude) references all_checkpoints(longitude, latitude)
    - Functional dependencies({c_id, preference}->longitude , {c_id, preference}->latitude)

12. notify
    - CK({pilot_id, airport_id, time})
    - PK({pilot_id, airport_id, time})
    - FK pilot_id references pilot(pilot_id)
    - FK airport_id references airport(airport_id)
    - Functional dependencies({pilot_id, airport_id, time}->reason)

13. refer
    - CK({trip_id, trip_chkpt_no, c_id, alternate_chkpt_pref})
    - PK({trip_id, trip_chkpt_no, c_id, alternate_chkpt_pref})
    - FK trip_id references trip_checkpoints(trip_id)
    - FK trip_chkpt_no references trip_checkpoints(checkpoint_no)
    - FK c_id references subroute_condition(c_id)
    - Functional dependencies({trip_id, trip_chkpt_no, c_id, alternate_chkpt_pref}->is_ok)

## 3) Investigate every schema for normalization and schema refinement:

1. user_details
    - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
    - Here, we only have primary dependencies that determinant is candidate key only and there is no overlapping candidate key. So, our model is in 2NF and BCNF as we have more than one candidate keys.
    - It is obvious that there are no insert, delete, update anomalies .

2. flight
    - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
    - Here, we only have primary dependencies that determinant is candidate key and there is only one candidate key. So, our model is in 2NF and 3NF.
    - It is obvious that there are no anomalies.

3. pilot
    - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
    - Here, we only have primary dependencies that is determinant is candidate key only and there is only one candidate key. So, our model is in 2NF and 3NF.
    - It is obvious that there are no anomalies.

4. trips
    - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
    - Here, we do not have partial dependencies. So, our model is in 2NF
    - There are more than one candidate key. We only have candidate keys as determinant in dependencies.
    - We also have overlapping candidate key but we do not have pilot_id <-> flight_id dependency. So, our model is in BCNF.
    - It is obvious that there are no anomalies.
    - We can update any tuple easily as we can change any attribute based on the candidate or primary key as there are no redundancies because every determinant in dependency is a candidate key.
    - Similar will be the case with insert and delete though we do not want to delete any trip as we need to store every detail of every trip so far.

5. sensor_details
    - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
    - Here, we do not have partial dependencies. So, our model is in 2NF
    - There are more than one candidate key. We only have candidate keys as determinant in dependencies.
    - We also have overlapping candidate key but we do not have pilot_id <-> dependency. So, our model is in BCNF .

- It is obvious that there are no anomalies.
- We can update any tuple easily as we can change any attribute based on the candidate or primary key as there are no redundancies because every determinant in dependency is a candidate key though we do not need to update because they are the real world values.
- Similar will be the case with insert and delete though we do not want to delete as we need to store every detail of every trip so far.

6. airport
   - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
   - Here, we only have primary dependencies. So, our 2NF.
   - We have multiple candidate keys but all determinants are candidate key and there is no overlapping candidate key. So, our model is in BCNF.
   - It is obvious that there are no anomalies.

7. alternate_airports
   - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
   - Here, we do not have partial dependencies. So, our model is in 2NF
   - There are more than one candidate key. We only have candidate keys as determinant in dependencies.
   - We also have overlapping candidate key but we do not have alternate_airport_id<-> preference. So, our model is in BCNF .
   - On insertion of new airport_id based on its value and preference we can insert its alternate airport and if (airport_id, alternate_airport_id) is present then it will be rejected as desired because it is a primary key. Same will be the case with update.
   - When delete a preference or an alternate_airport_id for an airport we do not loose other preference.. Also even if we delete all alternate airports for an aiport_id we do not loose airport data as it is stored in airport schema.
   - Also when we only delete a preference then also alternate airport data for the given airport will not be deleted which is desirable.

8. trip_checkpoints
   - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
   - Here, we only have primary dependencies. So, our model is in 2NF.
   - Also determinants are candidate key only and there is only one candidate key. So, our model is in 3NF.
   - It is obvious that there are no anomalies.

9. all_checkpoints
   - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
   - Here, we only have primary dependencies. So, our model is in 2NF.
   - Also determinants are candidate key only and there is only one candidate key. So, our model is in 3NF.
   - It is obvious that there are no anomalies.

10. subroute_condition
    - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
    - Here, we only have primary dependencies. So, our model is in 2NF.
    - Also determinants are candidate key only and there is only one candidate key. So, our model is in 3NF.
    - It is obvious that there are no anomalies.

11. alternate_chkpt
    - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
    - Here, we only have primary dependencies. So, our model is in 2NF.
    - Also determinants are candidate key only and there is only one candidate key. So, our model is in 3NF.
    - It is obvious that there are no anomalies.

12. notify
    - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
    - Here, we only have primary dependencies. So, our model is in 2NF.
    - Also determinants are candidate key only and there is only one candidate key. So, our model is in 3NF.
    - It is obvious that there are no anomalies.

13. refer
    - Here, all attributes are non composite, single valued and atomic. So, our model is in 1NF.
    - Here, we only have primary dependencies. So, our model is in 2NF.
    - Also determinants are candidate key only and there is only one candidate key. So, our model is in 3NF.
    - It is obvious that there are no anomalies.

## 4) Final Relational Schema

1. user_details(**user_id**, username, email, password)
2. flight(**flight_id**, passenger_capacity, airline, status, trips_completed, longitude,latitude, fuel_capacity, speed)
3. pilot(**pilot_id**, name, password, working_hours)
4. trips(**trip_id**, source_airport, destination_airport, arrival_time, departure_time, flight_id, pilot_id, actual_arrival_time, actual_departure_time )
5. sensor_details(**trip_id, checkpoint_no**, fuel_value, altitude, time, speed, longitude, latitude)
6. airport(**airport_id**, name, availability, longitude, latitude, no_of_runways)
7. alternate_airports(**airport_id**, **alternate_airport_id**, preference)
8. trip_checkpoints(**trip_id**, **checkpoint_no**, longitude, latitude, time)
9. all_checkpoints(**longitude**, **latitude**, name)
10. subroute_condition(**c_id**, current_longitude, current_latitude,next_longitude, next_latitude, airspace_condition)
11. alternate_chkpt(**c_id**, **preference**, longitude, latitude)
12. notify(**pilot_id**, **airport_id**, **time**, reason)
13. refer(**trip_id**, **trip_chkpt_no**, **c_id**, **alternate_chkpt_pref**, is_ok)

# Section 6: SQL: Final DDL Scripts, Insert statements, 40 SQL Queries with Snapshots of output of each query.

# I. DDL Script

## 1. user

```
CREATE TABLE IF NOT EXISTS user_details
(
        user_id bigint NOT NULL,
        username character varying(50) NOT NULL,
        email character varying(50),
        password character varying(50) NOT NULL,
        PRIMARY KEY (user_id)
);

COPY ft.user_details(user_id,username,email,password) FROM 'D:\College\Sem 5\DBMS\user_details.csv'
DELIMITER ',' CSV HEADER;
```

## 2. flight

```
CREATE TABLE IF NOT EXISTS flight
(
        flight_id bigint NOT NULL,
        passenger_capacity integer,
        airline character varying(50),
        status character varying(50),
        trips_completed integer DEFAULT(0),
        longitude real,
        latitude real,
        fuel_capacity integer,
        speed real,
        PRIMARY KEY (flight_id)
);

COPY ft.flight(flight_id,passenger_capacity,airline,status,trips_completed,longitude,latitude,fuel_capacity,speed)
FROM 'D:\College\Sem 5\DBMS\flight.csv' DELIMITER ',' CSV HEADER;
```

## 3. pilot

```
CREATE TABLE IF NOT EXISTS pilot
(
        pilot_id bigint NOT NULL,
   name character varying(40),
        password character varying(20) NOT NULL,
        working_hours integer,
        PRIMARY KEY (pilot_id)
);

COPY ft.pilot(pilot_id,name,password,working_hours) FROM 'D:\College\Sem 5\DBMS\pilot.csv' DELIMITER ','
CSV HEADER;
```

## 4. airport

```
CREATE TABLE IF NOT EXISTS airport
```

```
(
        airport_id bigint NOT NULL,
        name character varying(50),
        availability character varying(10) NOT NULL,
        longitude real NOT NULL,
        latitude real NOT NULL,
        no_of_runways integer,
        PRIMARY KEY (airport_id),
        UNIQUE(longitude,latitude)
);

COPY ft.airport(airport_id,name,availability,longitude,latitude,no_of_runways) FROM 'D:\College\Sem
5\DBMS\airport.csv' DELIMITER ',' CSV HEADER;
```

### 5. trips

```
CREATE TABLE IF NOT EXISTS trips
(
        trip_id bigint NOT NULL,
        arrival_time character varying(50) NOT NULL,
        departure_time character varying(50) NOT NULL,
        flight_id bigint,
   pilot_id bigint,
        source_airport bigint,
        destination_airport bigint,
   actual_arrival_time character varying(50) default NULL,
        actual_departure_time character varying(50) default NULL,
        PRIMARY KEY (trip_id),
   FOREIGN KEY (pilot_id)
        REFERENCES pilot(pilot_id) ON DELETE SET NULL ON UPDATE CASCADE,
   FOREIGN KEY (flight_id)
   REFERENCES flight(flight_id) ON DELETE SET NULL ON UPDATE CASCADE,
   FOREIGN KEY (source_airport)
   REFERENCES airport(airport_id) ON DELETE SET NULL ON UPDATE CASCADE,
   FOREIGN KEY (destination_airport)
   REFERENCES airport(airport_id) ON DELETE SET NULL ON UPDATE CASCADE,
   UNIQUE(flight_id,departure_time),
   UNIQUE(pilot_id,departure_time)
);

COPY
ft.trips(trip_id,arrival_time,departure_time,flight_id,pilot_id,source_airport,destination_airport,actual_arrival_time,actu
al_departure_time) FROM 'D:\College\Sem 5\DBMS\trips.csv' DELIMITER ',' CSV HEADER;
```

### 6. sensor_details

```
CREATE TABLE IF NOT EXISTS sensor_details
(
        trip_id bigint NOT NULL,
        checkpoint_no integer NOT NULL,
        fuel_value integer,
        altitude bigint,
        time character varying(50) NOT NULL,
```

```
        speed real,
        longitude real,
        latitude real,
        PRIMARY KEY (trip_id, checkpoint_no),
  FOREIGN KEY (trip_id)
        REFERENCES trips(trip_id) ON DELETE RESTRICT ON UPDATE CASCADE,
        UNIQUE(trip_id,time)
);

COPY ft.sensor_details(trip_id,checkpoint_no,fuel_value,altitude,time,speed,longitude,latitude) FROM
'D:\College\Sem 5\DBMS\sensor_details.csv' DELIMITER ',' CSV HEADER;
```

### 7. notify

```
CREATE TABLE IF NOT EXISTS notify
(
        pilot_id bigint NOT NULL,
        airport_id bigint NOT NULL,
        time character varying(50) NOT NULL,
        reason character varying(20),
        PRIMARY KEY (pilot_id, airport_id, time),
  FOREIGN KEY (pilot_id)
        REFERENCES pilot(pilot_id) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (airport_id)
        REFERENCES airport(airport_id) ON DELETE CASCADE ON UPDATE CASCADE
);

COPY ft.notify(pilot_id,airport_id,time,reason) FROM 'D:\College\Sem 5\DBMS\notify.csv' DELIMITER ',' CSV
HEADER;
```

### 8. alternate_airports

```
CREATE TABLE IF NOT EXISTS alternate_airports
(
        airport_id bigint NOT NULL,
        alternate_airport_id bigint NOT NULL,
        preference integer NOT NULL,
        PRIMARY KEY (airport_id, alternate_airport_id),
  FOREIGN KEY (airport_id)
        REFERENCES airport(airport_id) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (alternate_airport_id)
        REFERENCES airport(airport_id) ON DELETE CASCADE ON UPDATE CASCADE,
        UNIQUE(airport_id,preference)
);

COPY ft.alternate_airports(airport_id,alternate_airport_id,preference) FROM 'D:\College\Sem
5\DBMS\alternate_airports.csv' DELIMITER ',' CSV HEADER;
```

### 9. all_checkpoints

```
CREATE TABLE IF NOT EXISTS all_checkpoints
(
```

```
        longitude real NOT NULL,
        latitude real NOT NULL,
        name character varying(20),
        PRIMARY KEY (longitude, latitude)
);

COPY ft.all_checkpoints(longitude,latitude,name) FROM 'D:\College\Sem 5\DBMS\all_checkpoints.csv' DELIMITER
',' CSV HEADER;
```

### 10. trip_checkpoints

```
CREATE TABLE IF NOT EXISTS trip_checkpoints
(
        trip_id bigint NOT NULL,
        checkpoint_no integer NOT NULL,
        longitude real,
        latitude real,
        time character varying(50),
        PRIMARY KEY (trip_id, checkpoint_no),
        FOREIGN KEY (trip_id)
        REFERENCES trips(trip_id) ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (longitude,latitude)
        REFERENCES all_checkpoints(longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE
);

COPY ft.trip_checkpoints(trip_id,checkpoint_no,longitude,latitude,time) FROM 'D:\College\Sem
5\DBMS\trip_checkpoints.csv' DELIMITER ',' CSV HEADER;
```

### 11. subroute_condition

```
CREATE TABLE IF NOT EXISTS subroute_condition
(
        c_id bigint NOT NULL,
        current_longitude real NOT NULL,
        current_latitude real NOT NULL,
        next_longitude real NOT NULL,
        next_latitude real NOT NULL,
        airspace_condition character varying(10),
        PRIMARY KEY (c_id),
        FOREIGN KEY (current_longitude,current_latitude)
        REFERENCES all_checkpoints(longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (next_longitude,next_latitude)
        REFERENCES all_checkpoints(longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE
);

COPY ft.subroute_condition(c_id,current_longitude,current_latitude,next_longitude,next_latitude,airspace_condition)
FROM 'D:\College\Sem 5\DBMS\subroute_condition.csv' DELIMITER ',' CSV HEADER;
```

### 12. alternate_chkpt

```
CREATE TABLE IF NOT EXISTS alternate_chkpt
(
```

```
        c_id bigint NOT NULL,
        preference integer NOT NULL,
        longitude real,
        latitude real,
        PRIMARY KEY (c_id, preference),
  FOREIGN KEY (c_id)
        REFERENCES subroute_condition(c_id) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (longitude,latitude)
        REFERENCES all_checkpoints(longitude,latitude) ON DELETE CASCADE ON UPDATE CASCADE
);

COPY ft.alternate_chkpt(c_id,preference,longitude,latitude) FROM 'D:\College\Sem 5\DBMS\alternate_chkpt.csv'
DELIMITER ',' CSV HEADER;
```

### 13. refer

```
CREATE TABLE IF NOT EXISTS refer
(
        trip_id bigint NOT NULL,
        trip_chkpt_no integer NOT NULL,
        c_id bigint NOT NULL,
        alternate_chkpt_pref integer NOT NULL DEFAULT(0),
        is_ok character varying(10),
        PRIMARY KEY (trip_id, trip_chkpt_no, c_id, alternate_chkpt_pref),
        FOREIGN KEY (trip_id,trip_chkpt_no)
        REFERENCES trip_checkpoints(trip_id,checkpoint_no) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (c_id)
        REFERENCES subroute_condition(c_id) ON DELETE CASCADE ON UPDATE CASCADE
);

COPY ft.refer(trip_id,trip_chkpt_no,c_id,alternate_chkpt_pref,is_ok) FROM 'D:\College\Sem 5\DBMS\refer.csv'
DELIMITER ',' CSV HEADER
```

## II. SQL QUERIES

---

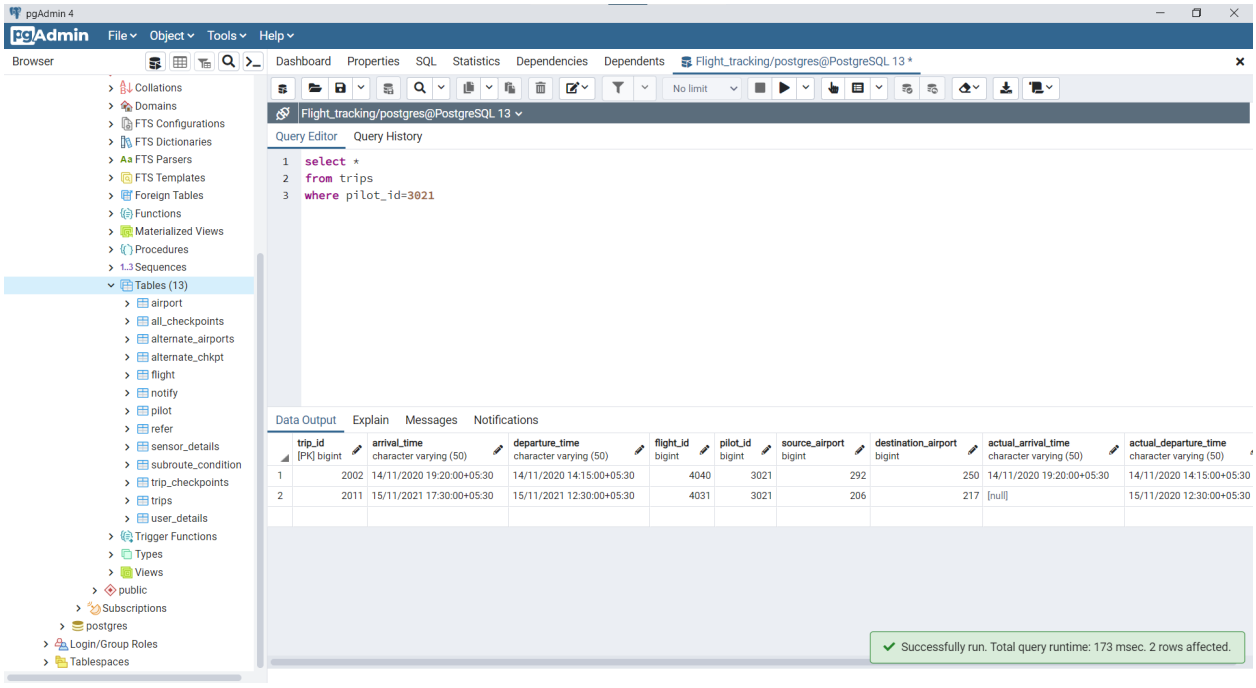
**SIMPLE QUERIES**

---

**1.)** Give details of all the trip where pilot_id=3021

**SQL Query -**

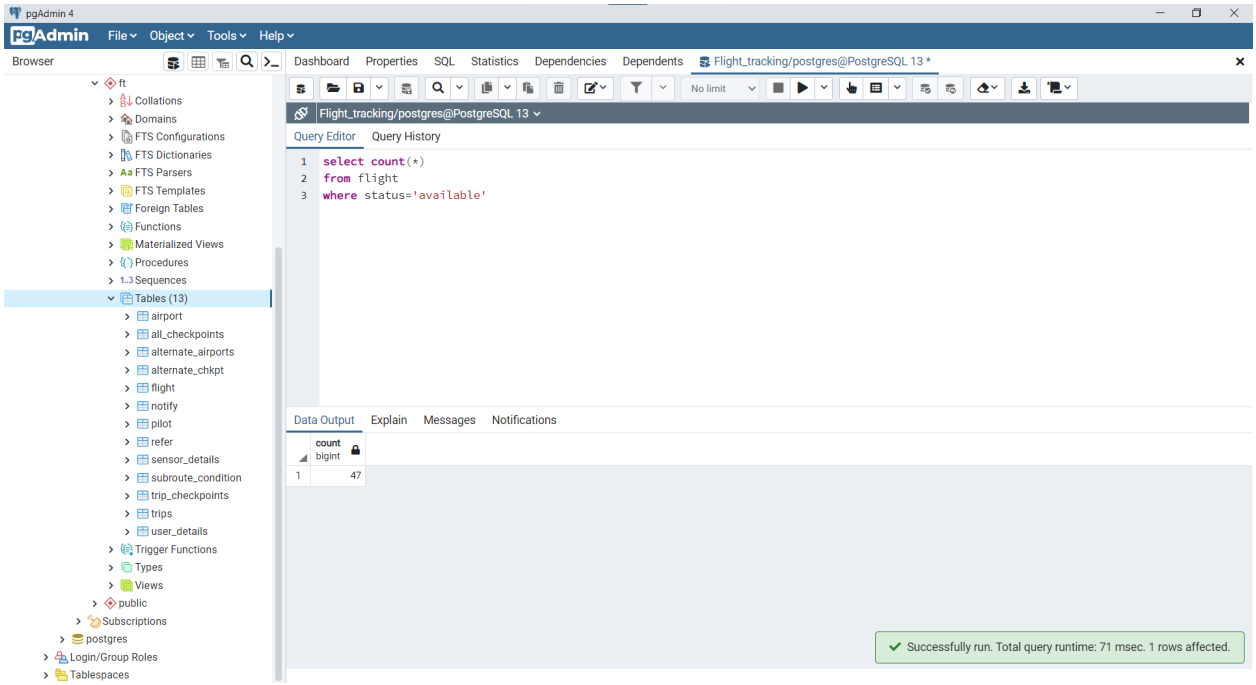
```
select *
from trips
where pilot_id=3021
```




**2.)** To count the number of aircrafts available.

**SQL Query -**

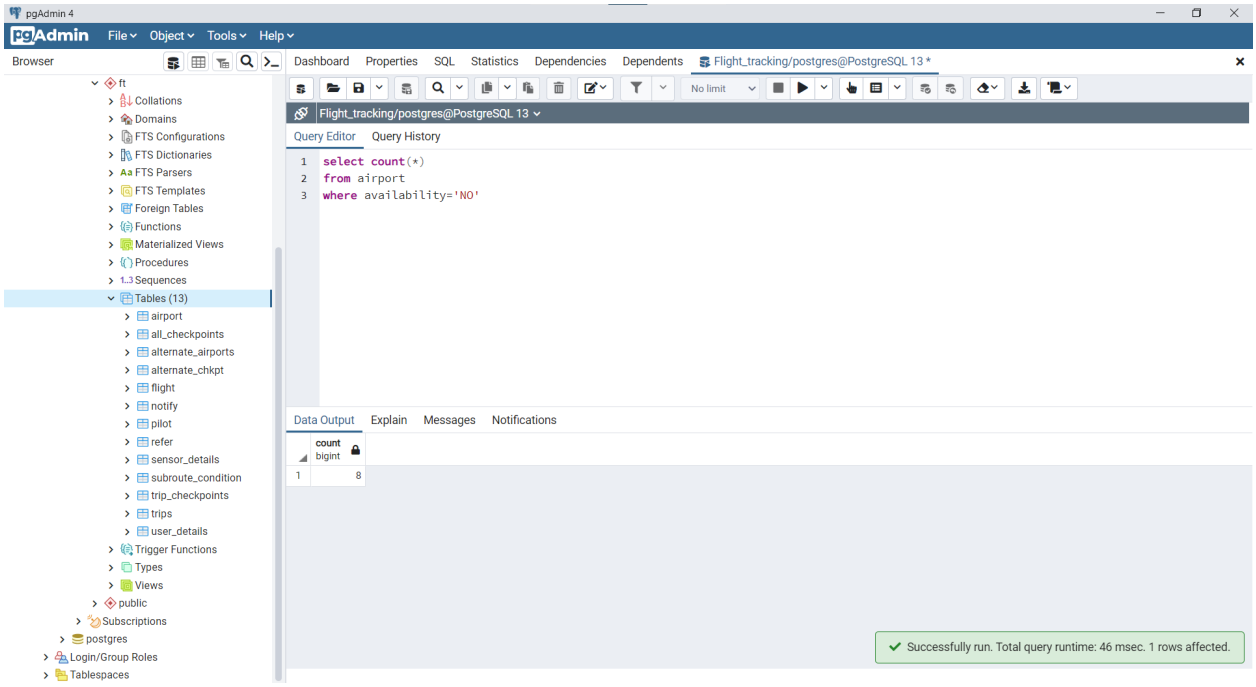
```
select count(*)
from flight
where status='available'
```



**3.)** To count the non available airports.

**SQL Query -**

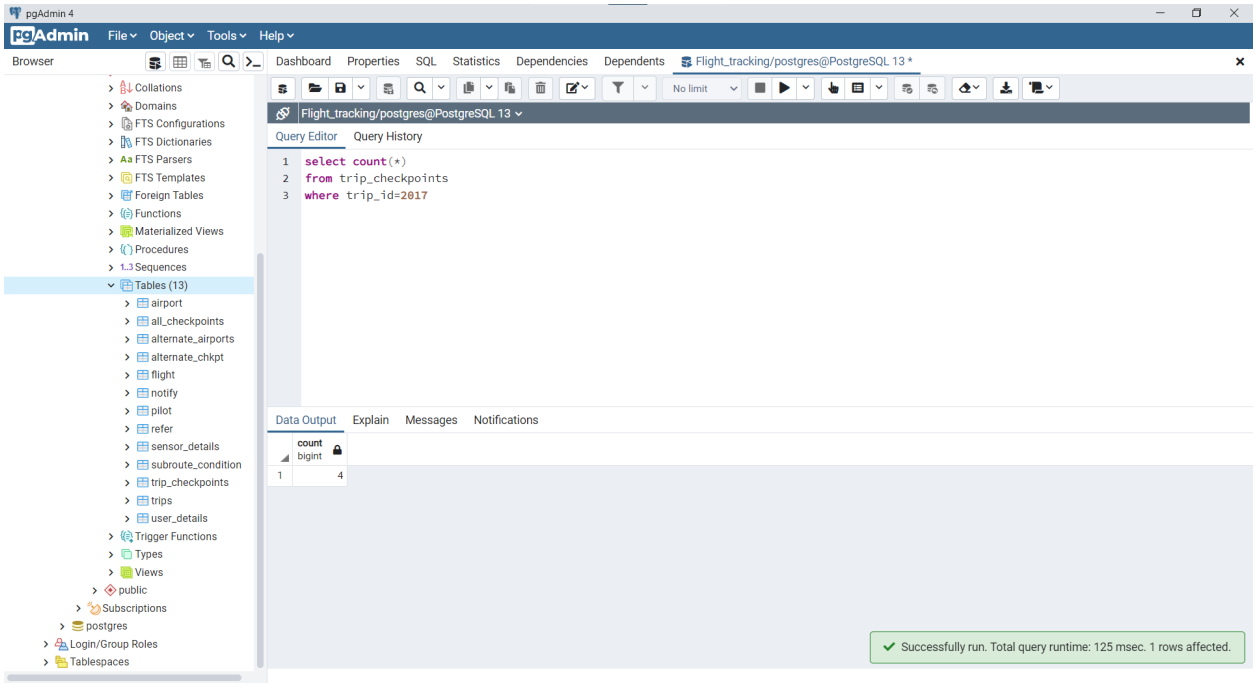
```
select count(*)
from airport
where availability='NO'
```




**4.)** How many checkpoints had the trip_id 2017 have

**SQL Query -**

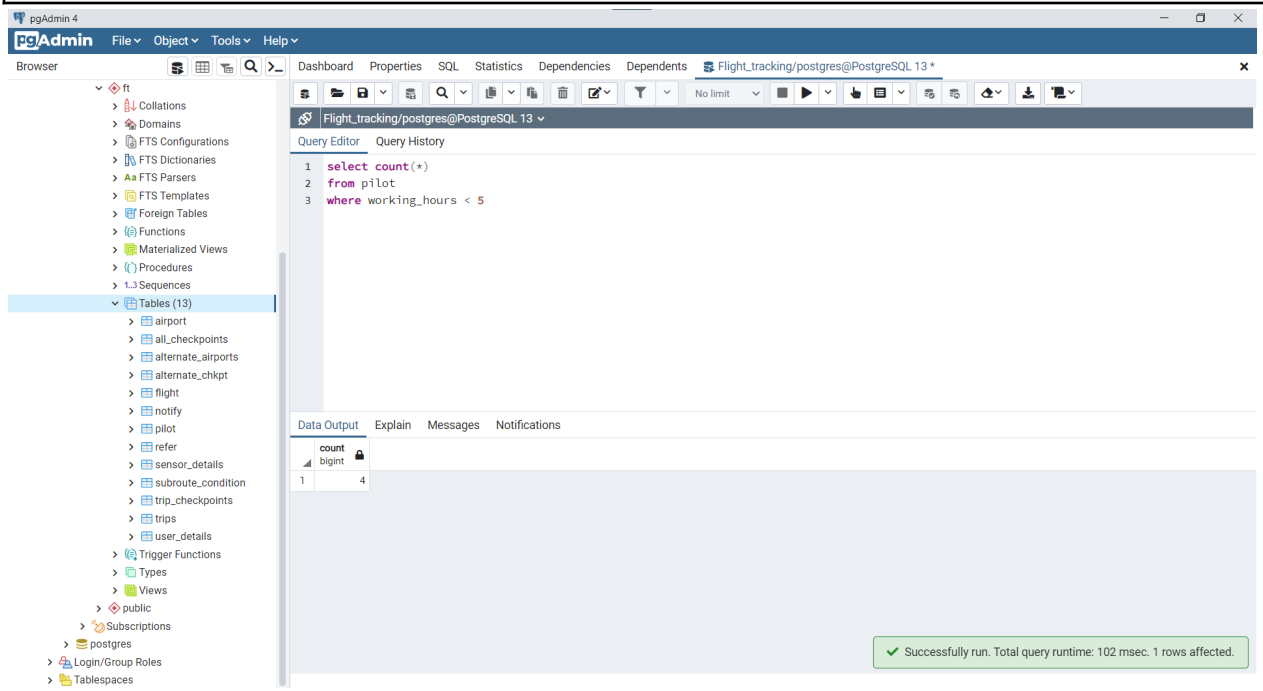
```
select count(*)
from trip_checkpoints
where trip_id=2017
```




**5.)** Find the number of pilots who have worked less than 5 hours in the week.

**SQL Query -**

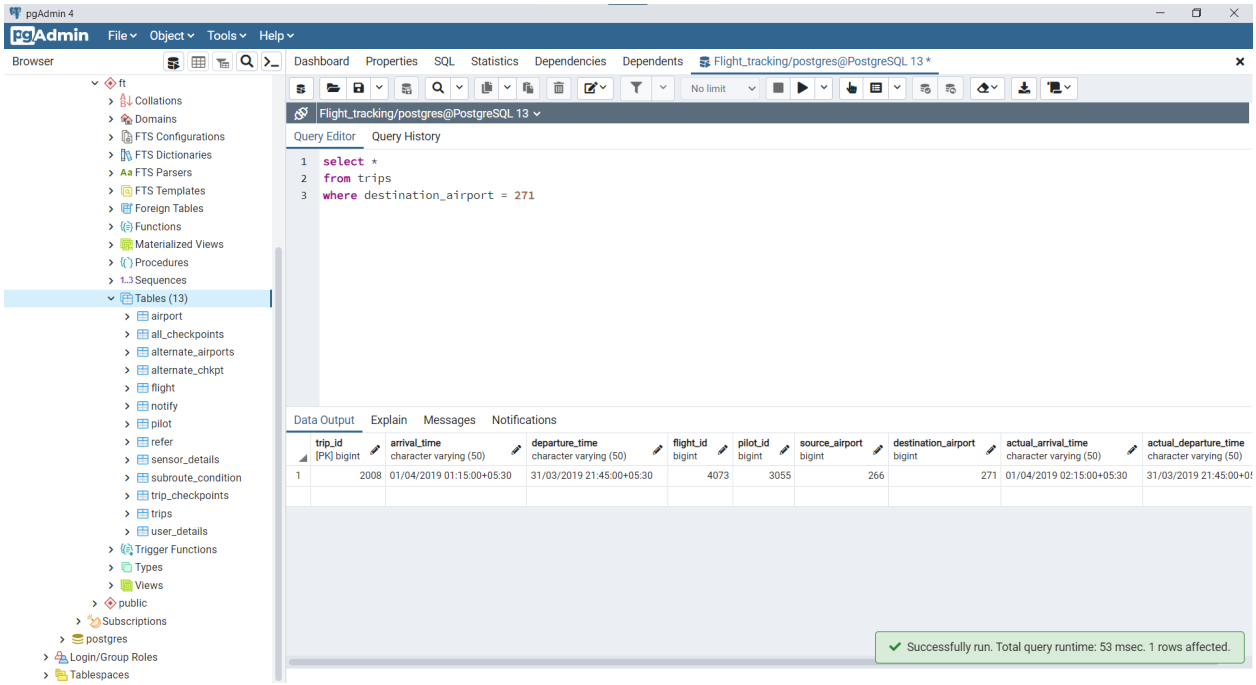
```
select count(*)
from pilot
where working_hours < 5
```



**6.)** Get the trip details for the trips having destination ‘271’.

**SQL Query -**

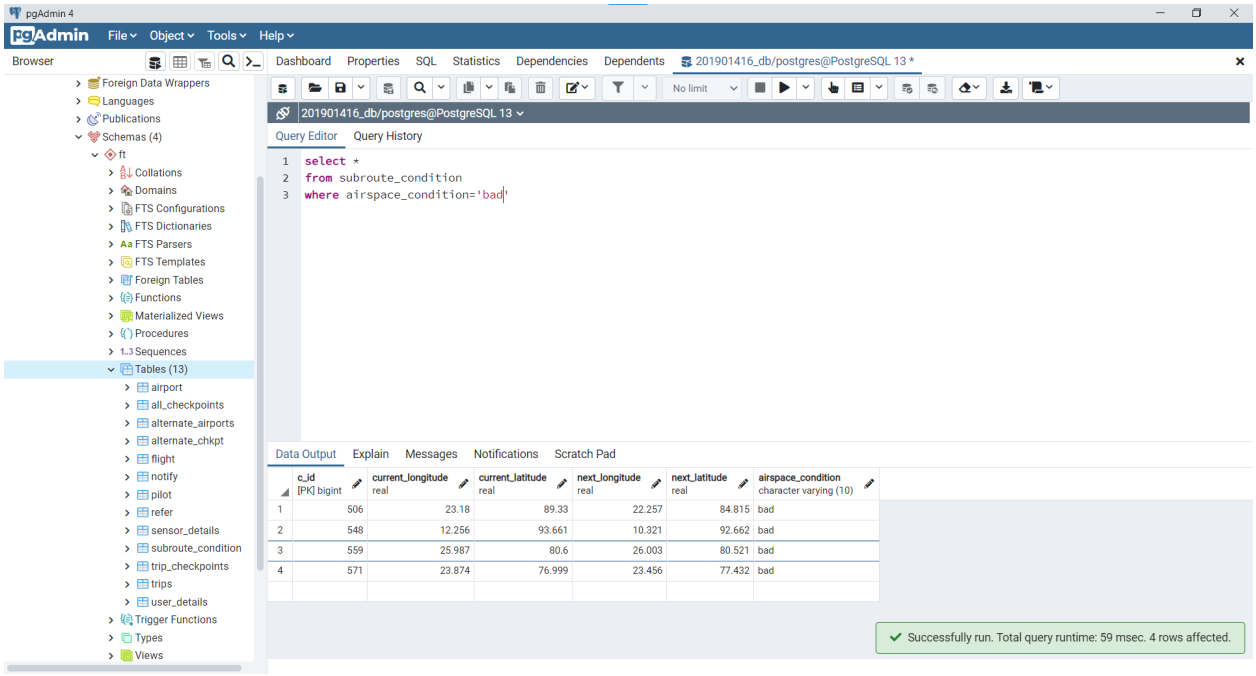
```
select *
from trips
where destination_airport = 271
```




**7.)** List the subroutes segments in bad condition.

**SQL Query -**

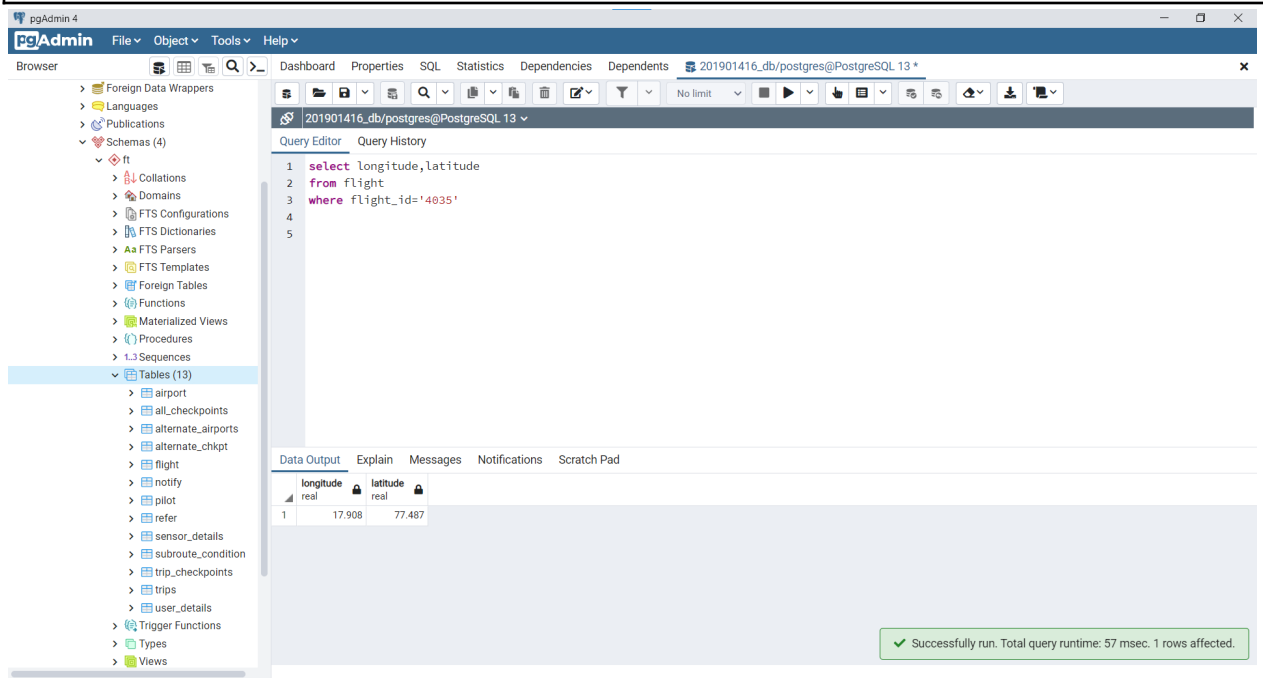
```
select *
from subroute_condition
where airspace_condition = 'bad'
```




**8.)** Get the live location of flight '4035'

**SQL Query -**

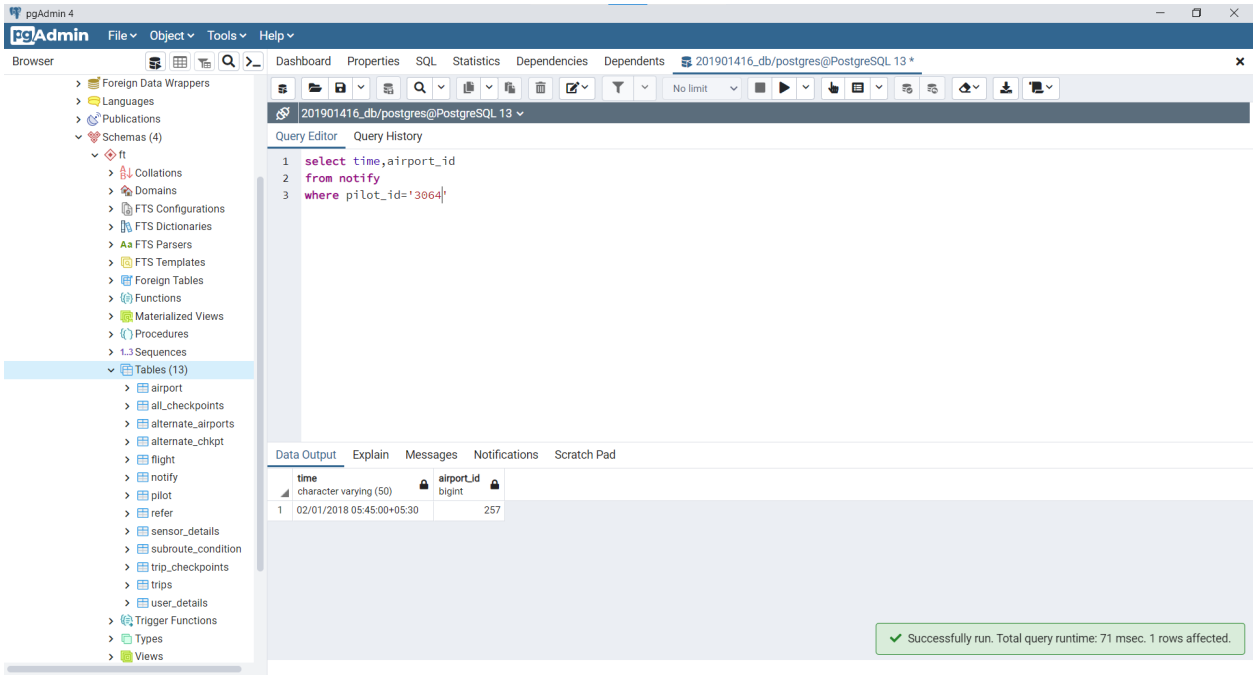
```
select longitude,latitude
from flight
where flight_id='4035'
```



**9.)** When was pilot '3064' notified and which airport notified it.

**SQL Query -**

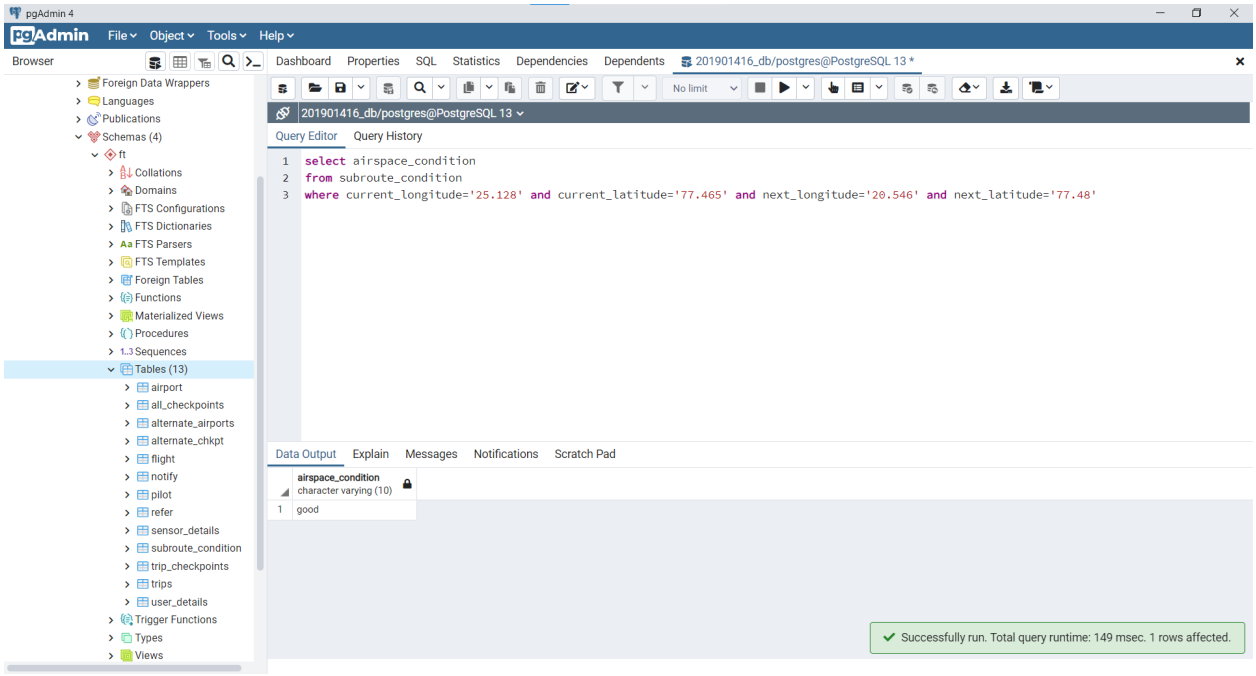
```
select time,airport_id
from notify
where pilot_id='3064'
```




**10.)** What is the airspace condition between the points having longitude and latitude as (25.128,77.465) to (20.546,77.48) .

**SQL Query -**

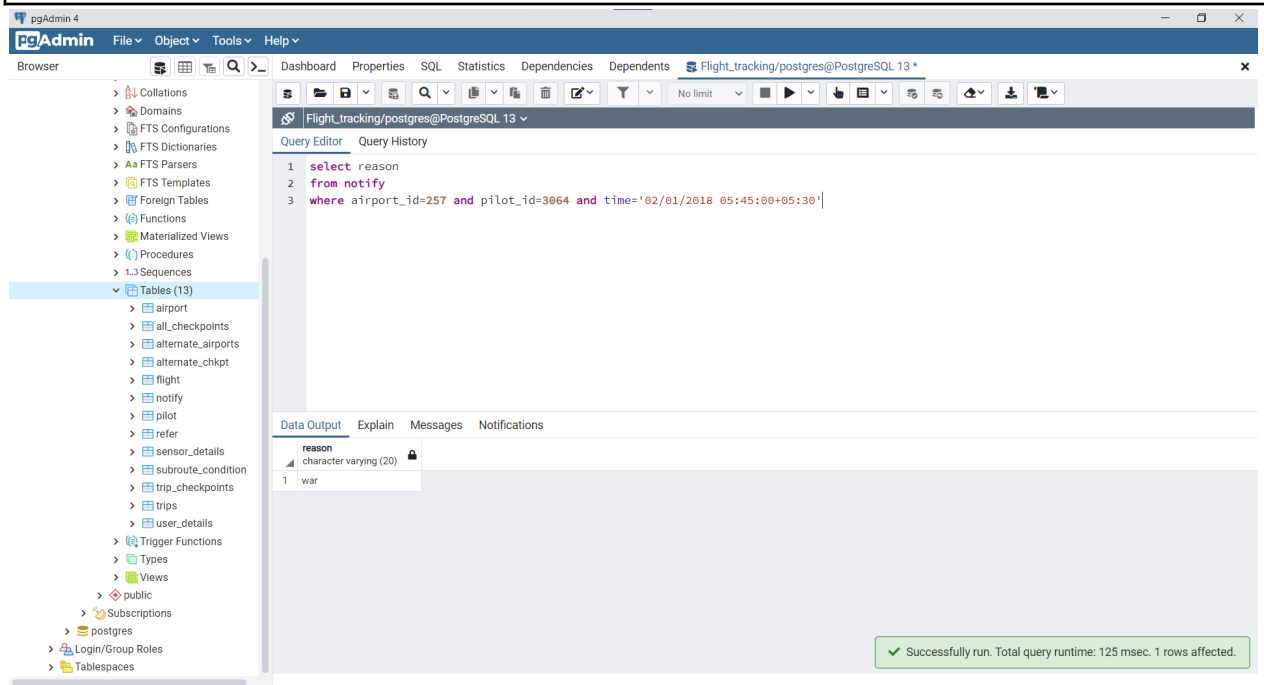
```
select airspace_condition
from subroute_condition
where
current_longitude='25.128' and current_latitude='77.465' and
next_longitude='20.546' and next_latitude='77.48'
```




**11.)** Find the reason when airport with airport_id=257 notified pilot with pilot pilot_id=3064 at time = '02/01/2018 05:45:00+05:30'

**SQL Query -**

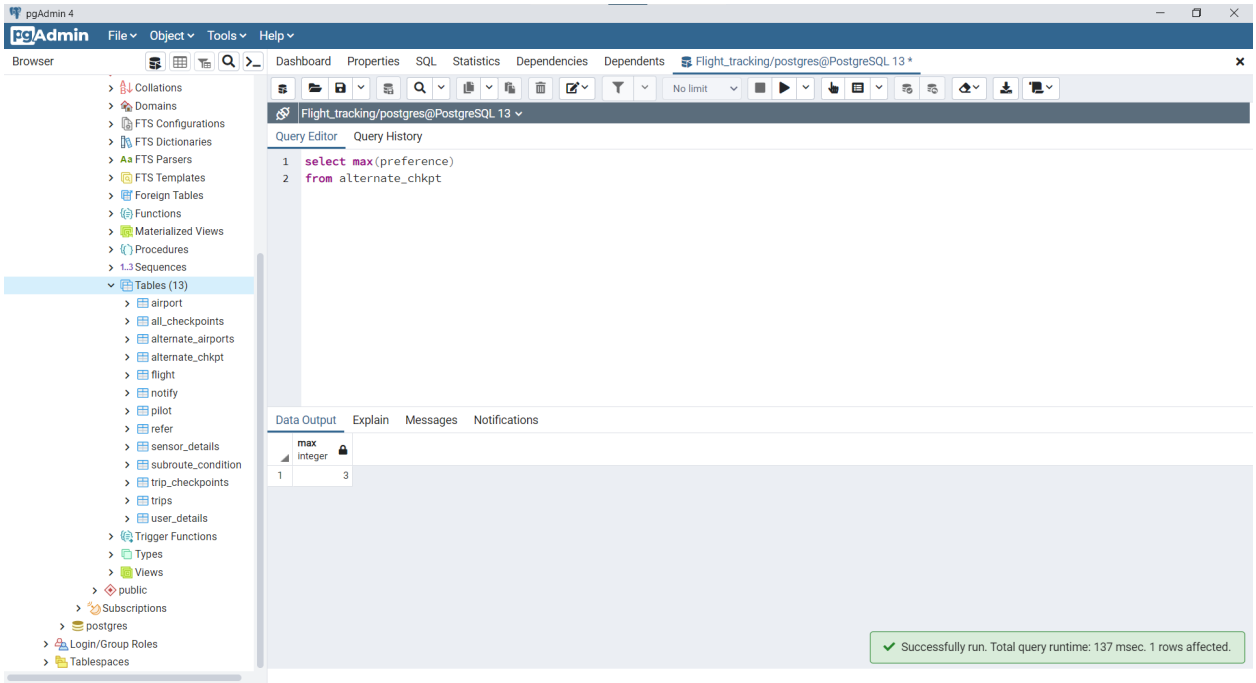
```
select reason
from notify
where airport_id=257 and pilot_id=3064 and time='02/01/2018 05:45:00+05:30'
```



**12.)** What are the maximum alternate checkpoints any checkpoint has?

**SQL Query -**

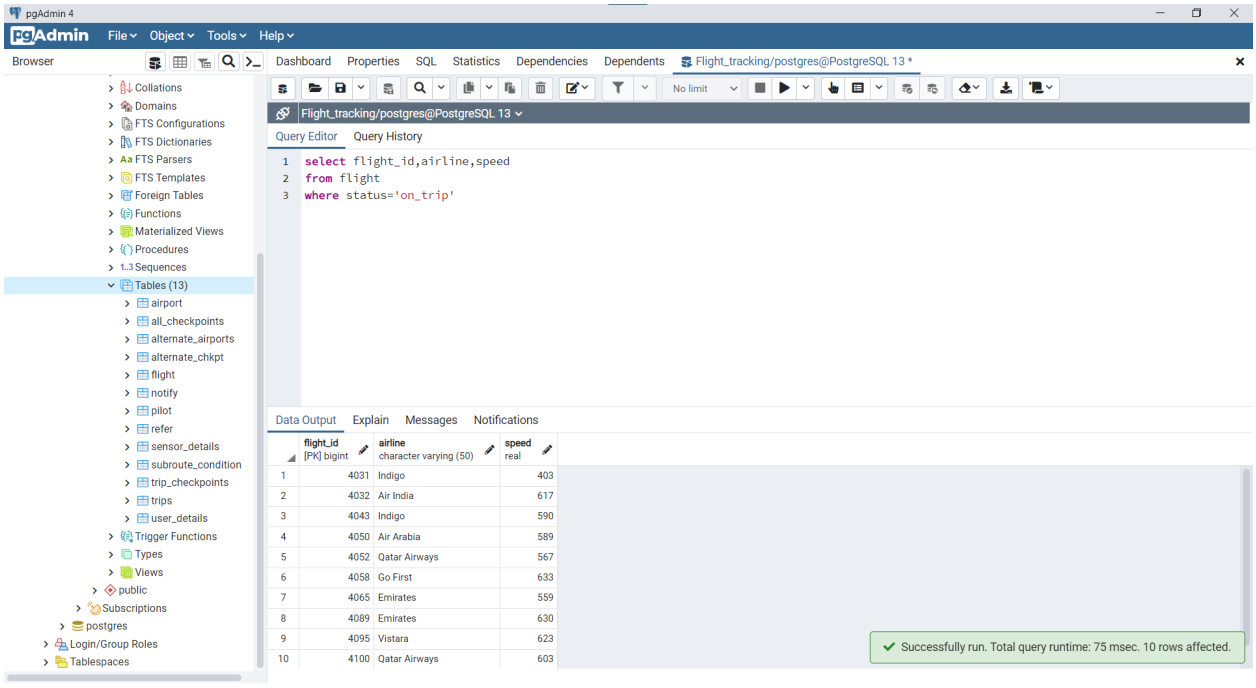
```
select max(preference)
from alternate_chkpt
```




**13.)** Give details of all the flights and their current speed which are on trip.

**SQL Query -**

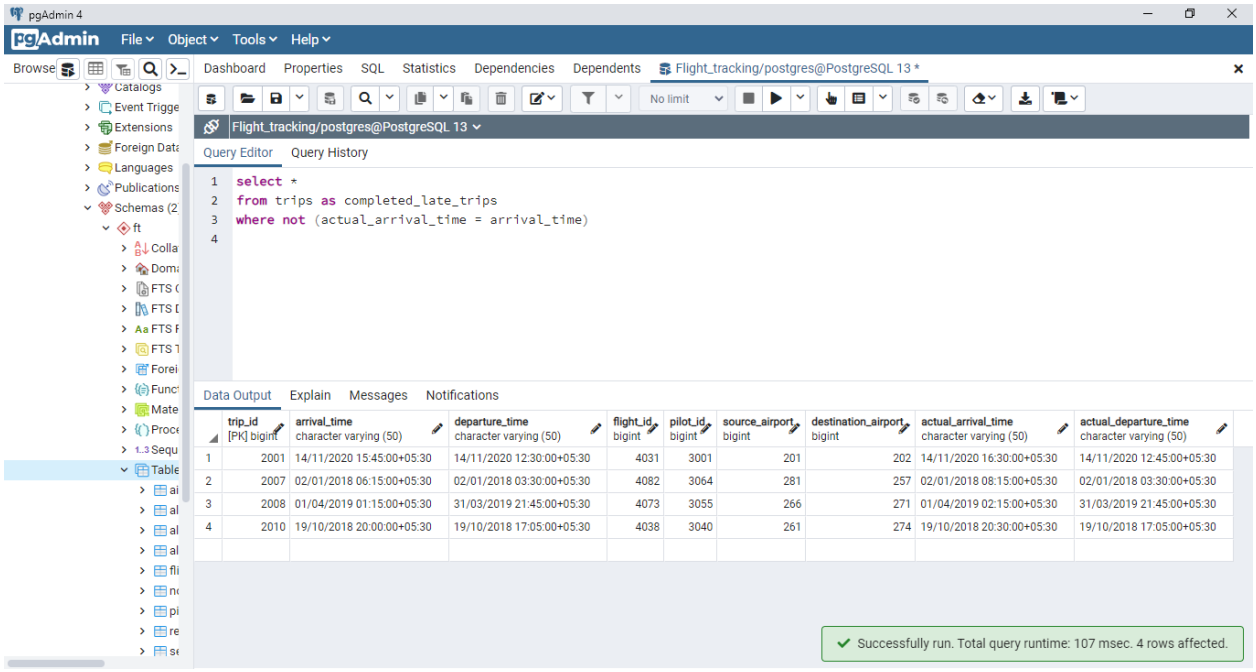
```
select flight_id,airline,speed
from flight
where status='on_trip'
```




**14.)** Find the trip details which were completed late.

**SQL Query -**

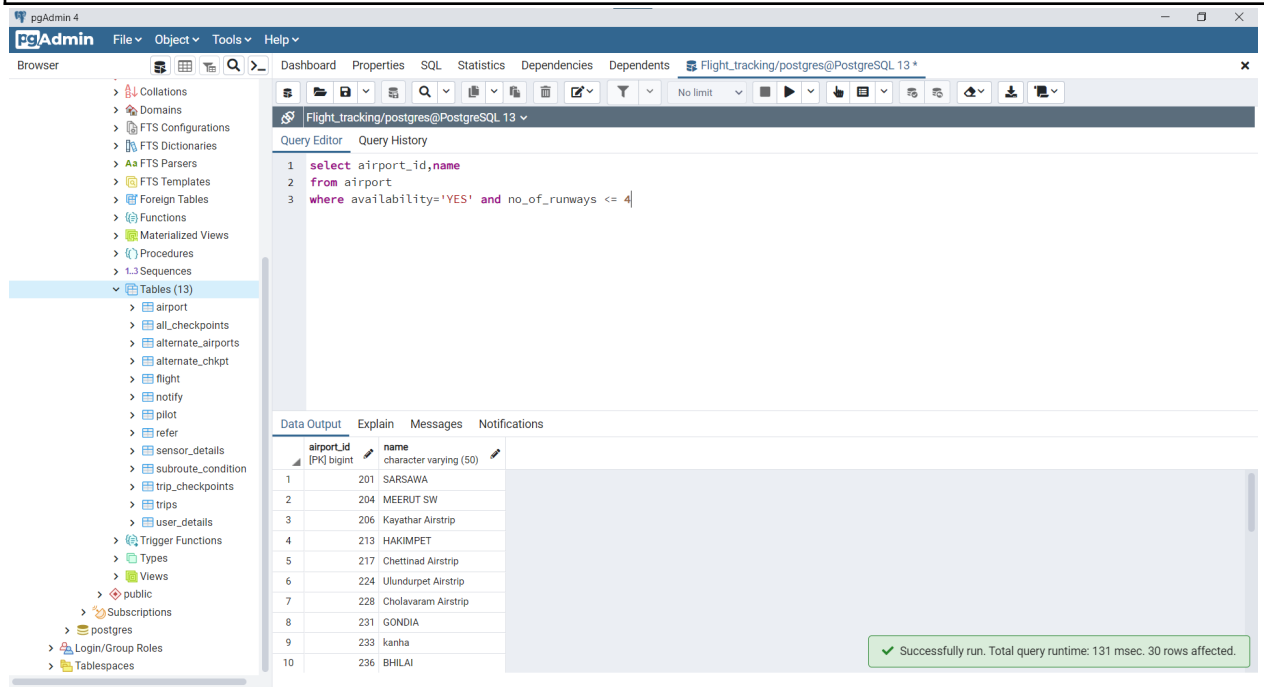
```
select *
from trips as completed_late_trips
where not (actual_arrival_time = arrival_time)
```



**15.)** List the airports which are available and have less than or equal to 4 runways

**SQL Query -**

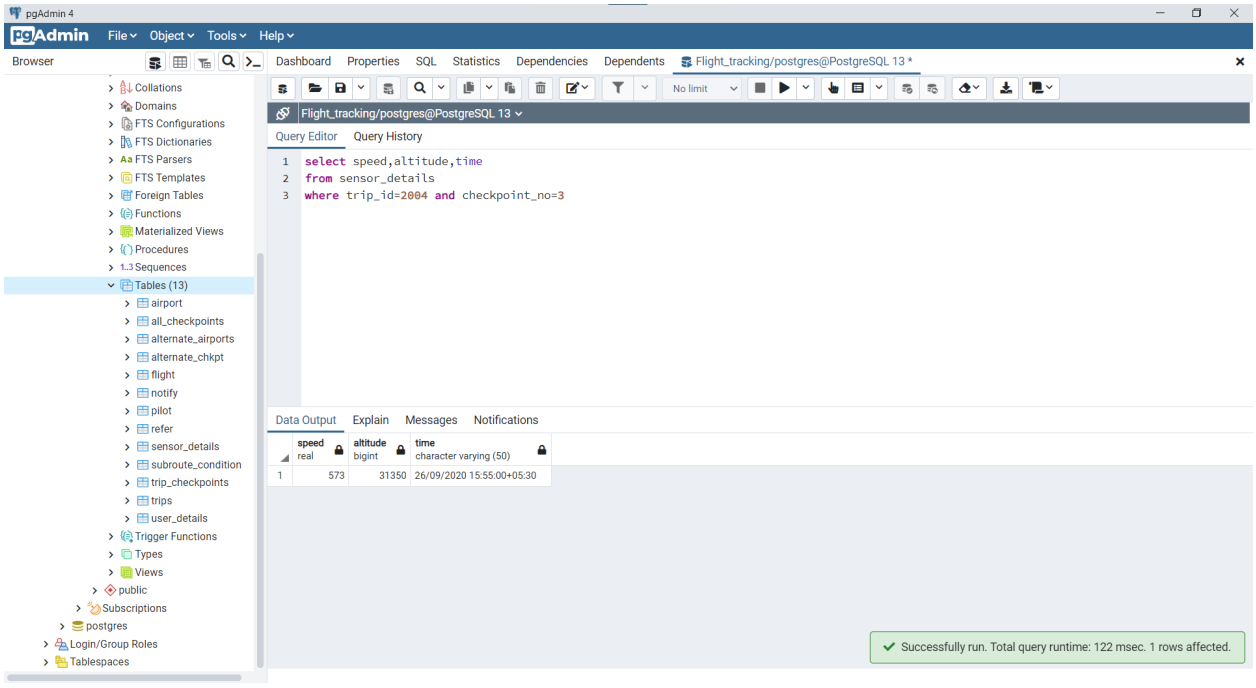
```
select airport_id,name
from airport
where availability='YES' and no_of_runways <= 4
```




**16.)** What was the speed,altitude and time of the flight when it is at checkpoint 3 in trip with trip_id=2004

**SQL Query -**

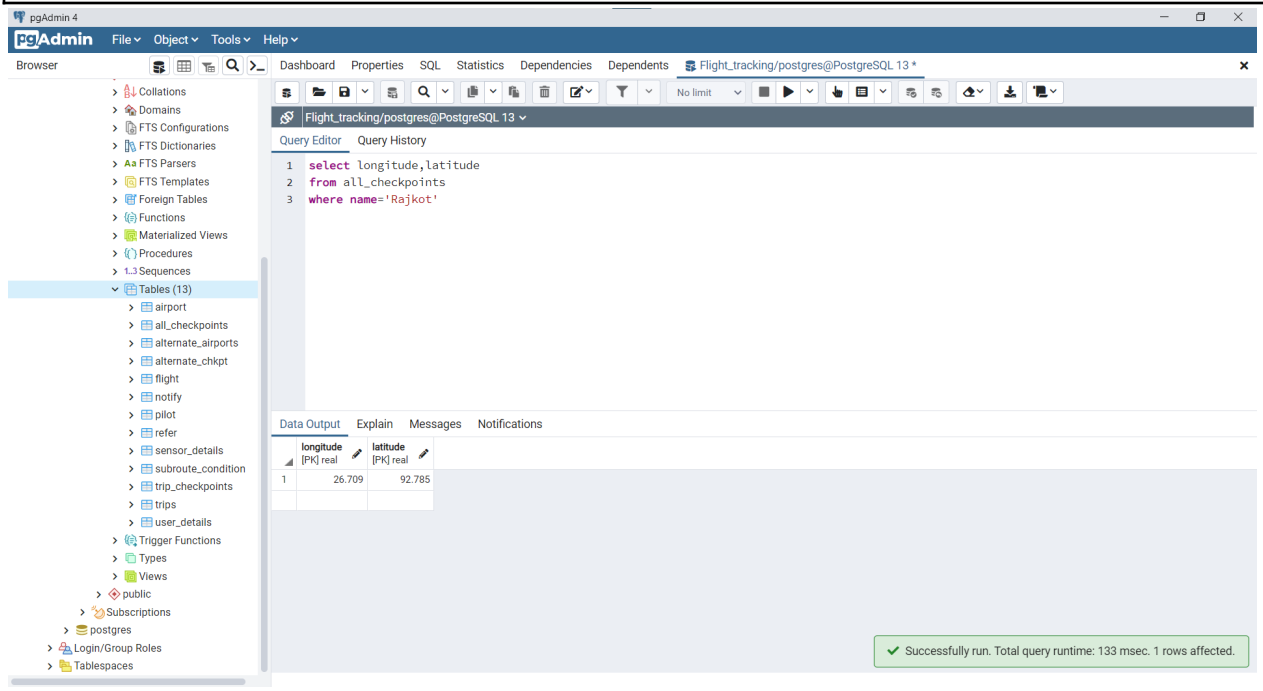
```
select speed,altitude,time
from sensor_details
where trip_id=2004 and checkpoint_no=3
```




**17.)** Which checkpoints are there over Rajkot.

**SQL Query -**

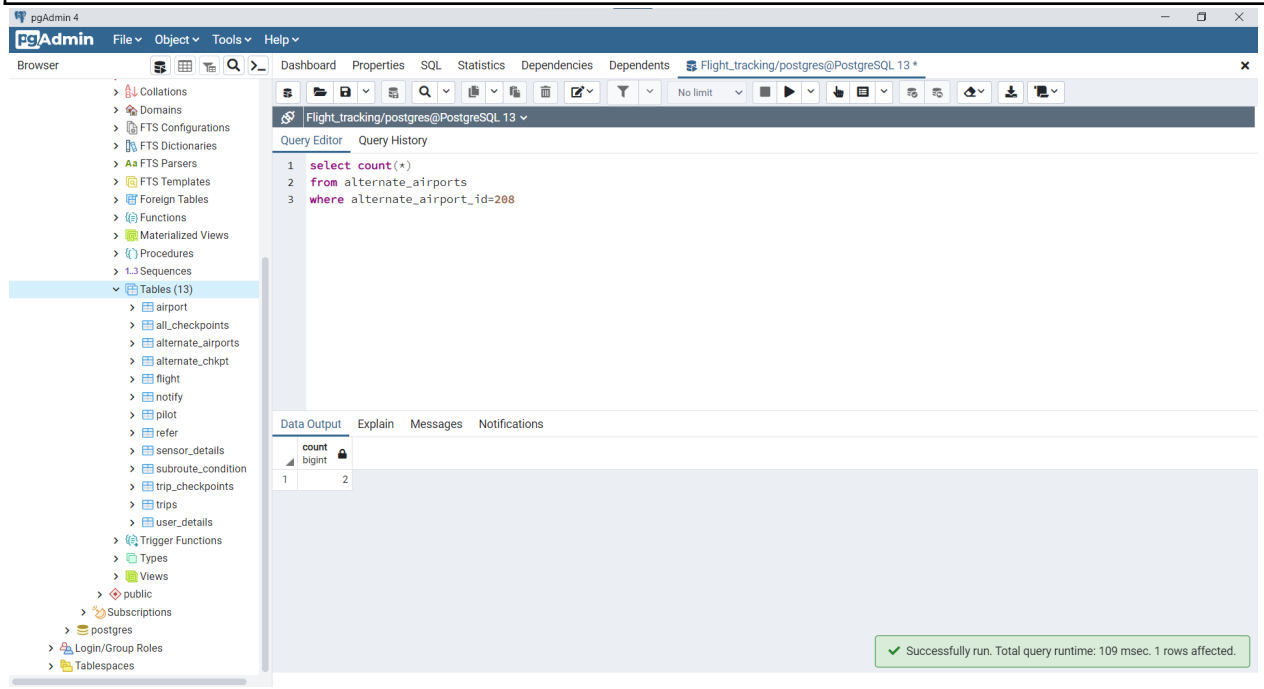
```
select longitude,latitude
from all_checkpoints
where name='Rajkot'
```



**18.)** How many airport has airport 208 as its alternate_airport_id

**SQL Query -**

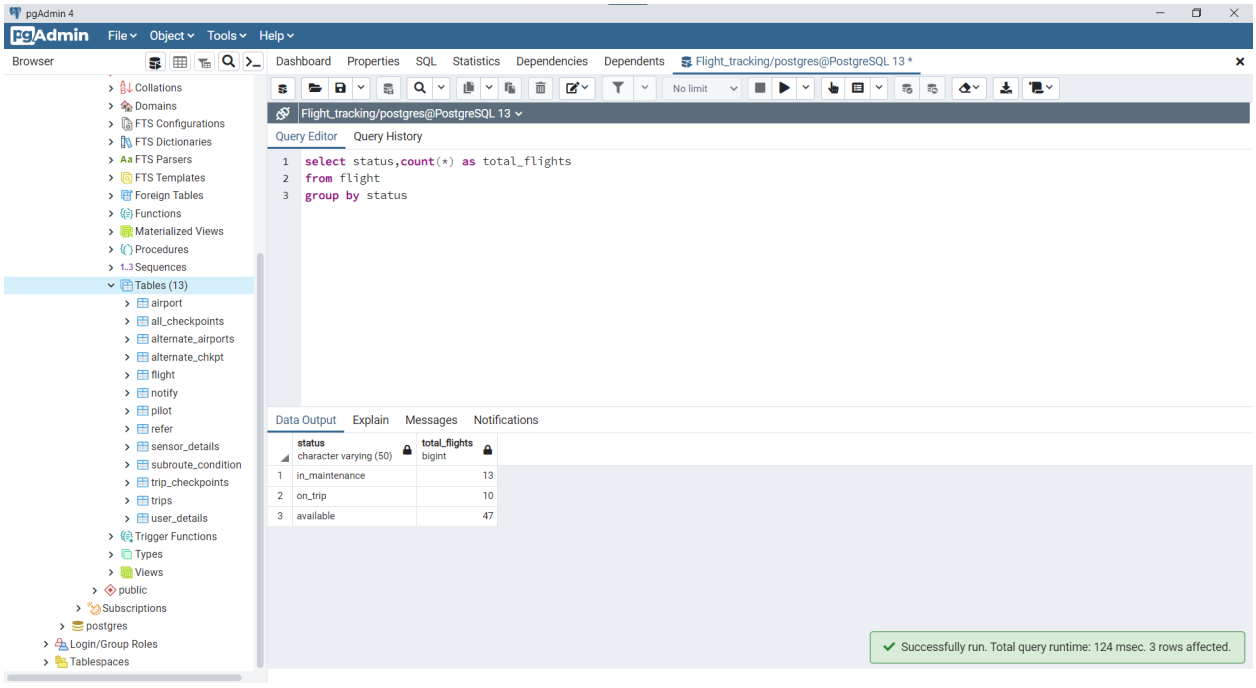
```
select count(*)
from alternate_airports
where alternate_airport_id=208
```




**19.)** How many flights are ‘on_trip’, ‘available’ and ‘in_maintenance’

**SQL Query -**

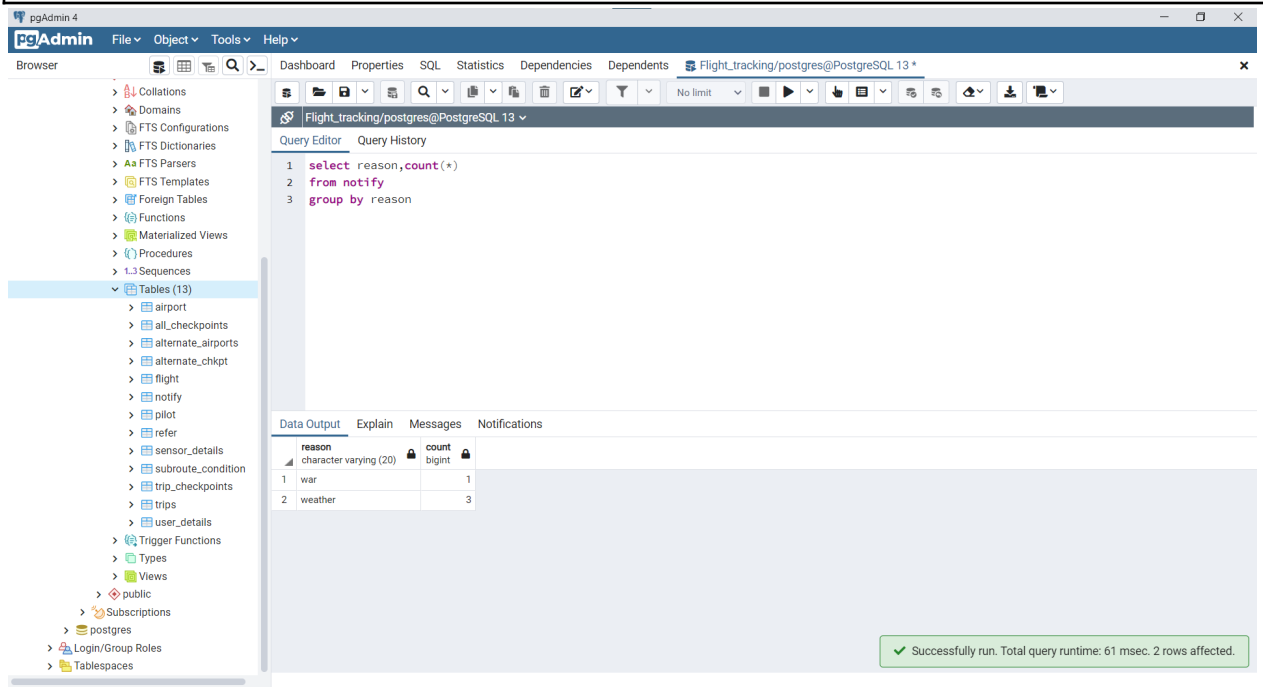
```
select status,count(*) as total_flights
from flight
group by status
```




**20.)** How many notifications are for weather and war.

**SQL Query -**

```
select reason,count(*)
from notify
group by reason
```



---

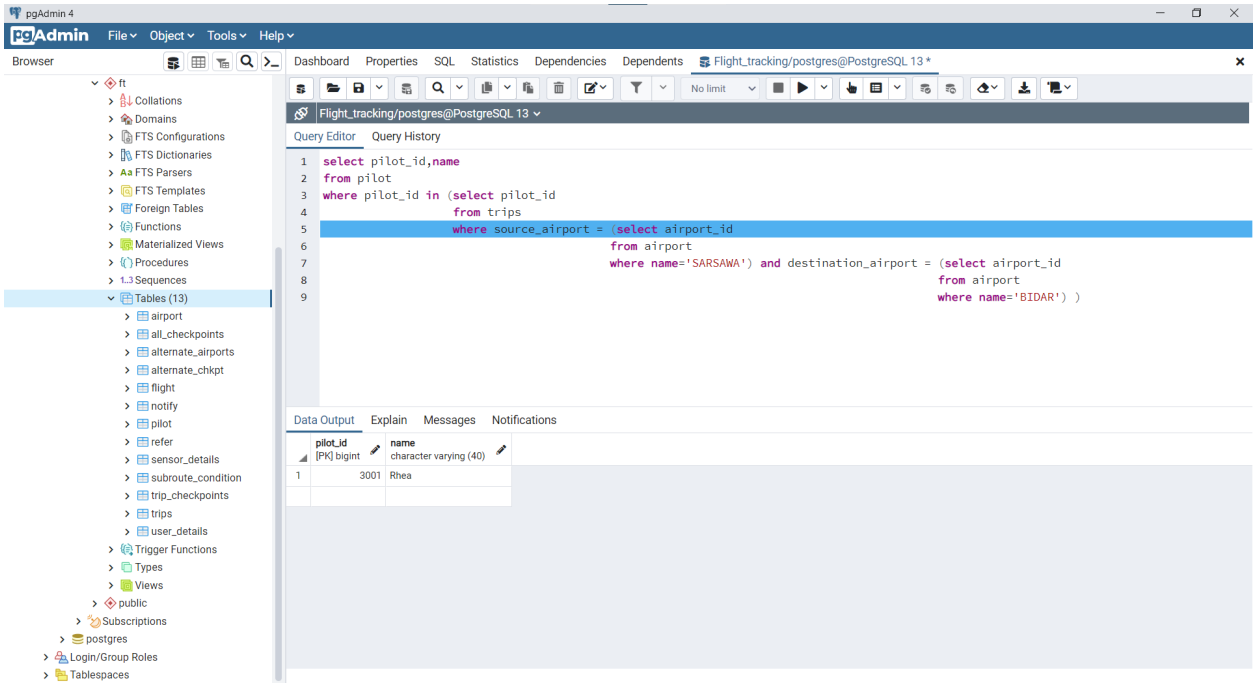
## COMPLEX QUERIES

---

**21.)** To know the pilot of the trip from SARSAWA to BIDAR

**SQL Query -**

```
select pilot_id,name
from pilot
where pilot_id in (select pilot_id
                   from trips
                   where
                source_airport = (select airport_id
                                  from airport
                                  where
                                name='SARSAWA') and
                                destination_airport = (select airport_id
                                                       from airport
                                                            where name='BIDAR') )
```

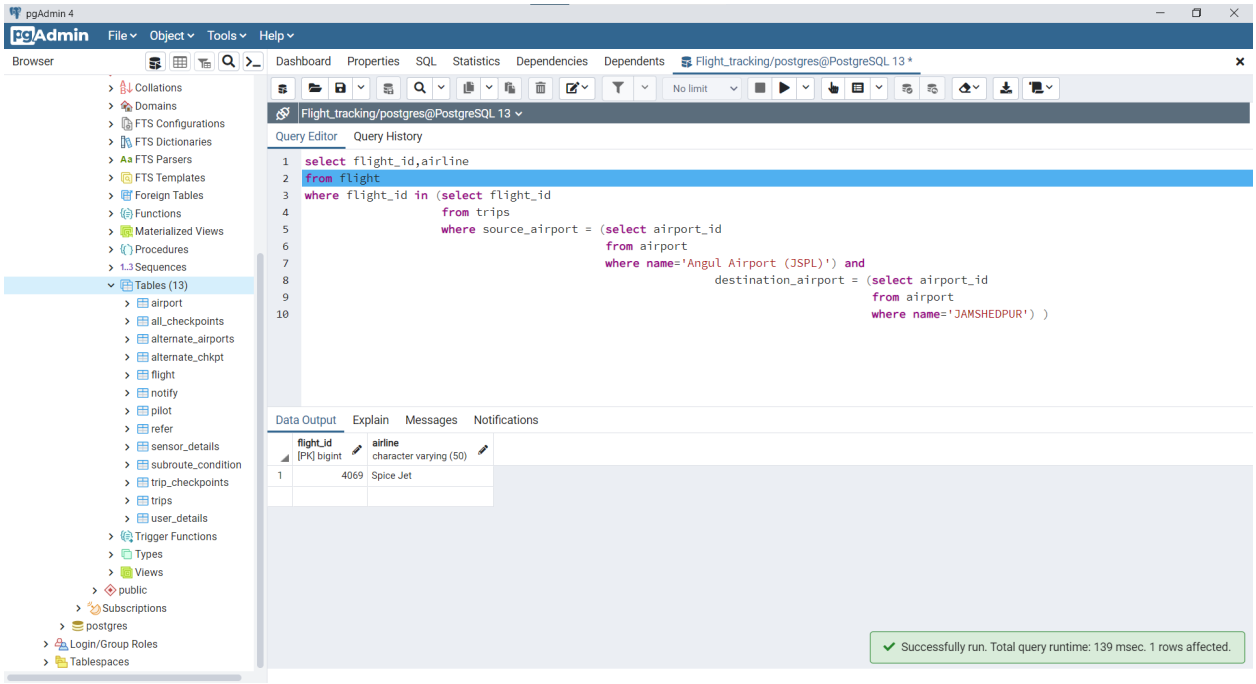



**22.)** To know the flight details of the trip from Angul Airport (JSPL) to JAMSHEDPUR

**SQL Query -**

```
select flight_id,airline
from flight
where flight_id in (select flight_id
          from trips
```

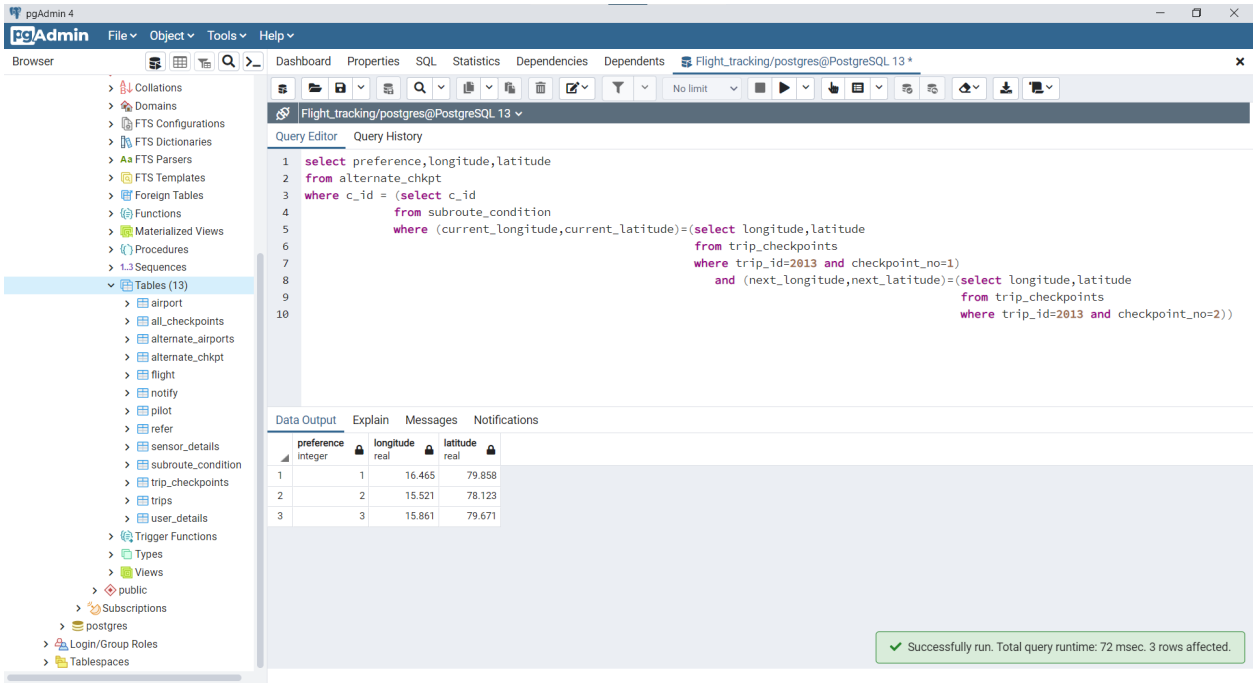
```
Where source_airport = (select airport_id
                        from airport
                        where
                           name='Angul Airport (JSPL)') and
                                    destination_airport = (select airport_id
                                    from airport
                                    where name='JAMSHEDPUR') )
```




**23.)** Suppose flight of trip_id=2013 is at 1st checkpoint and found the weather in the route from 1st checkpoint to 2nd checkpoint is bad, so give all the alternate checkpoints of 2nd chkpt.

**SQL Query -**

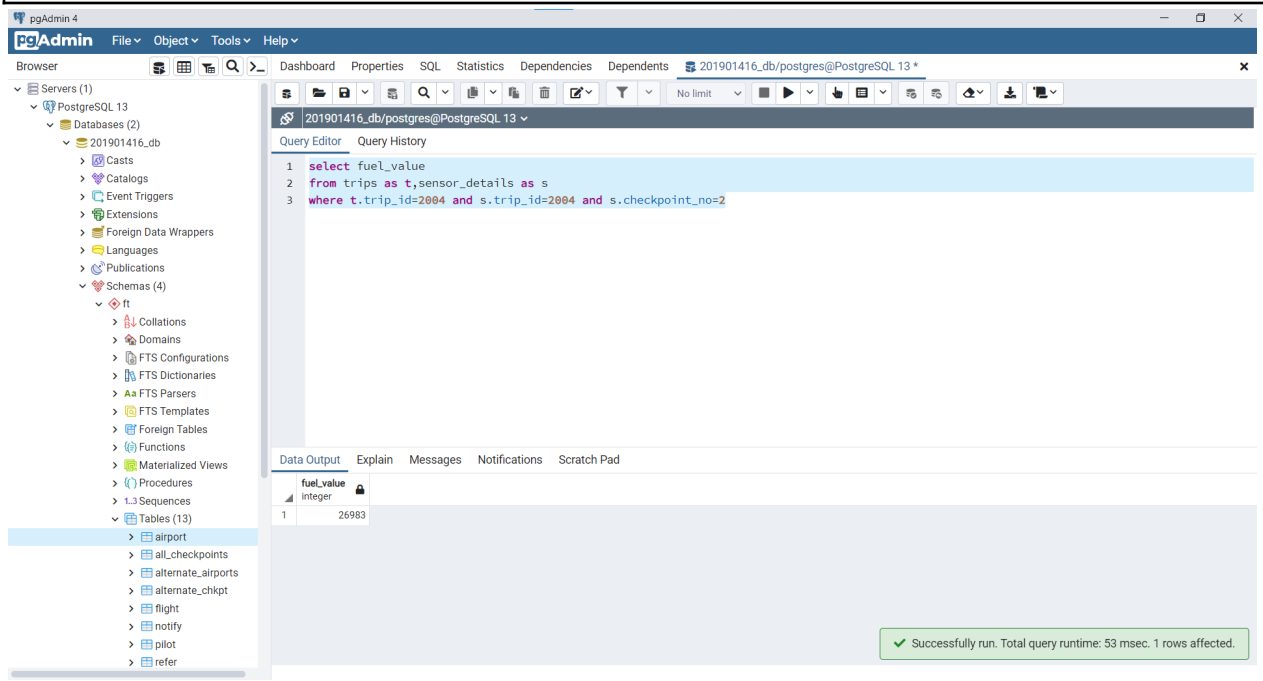
```
select preference,longitude,latitude
from alternate_chkpt
where c_id = (select c_id
                    from subroute_condition
                    where (current_longitude,current_latitude)=(select longitude,latitude
                                                from trip_checkpoints
                                                    where trip_id=2013 and checkpoint_no=1)
                                            and (next_longitude,next_latitude)=(select longitude,latitude
                                                from trip_checkpoints
                                                where trip_id=2013 and checkpoint_no=2))
```




**24.)** Find the fuel value of flight at checkpoint number 2 of flight going from FUKCHE to KANHA having trip id 2004.

**SQL Query -**

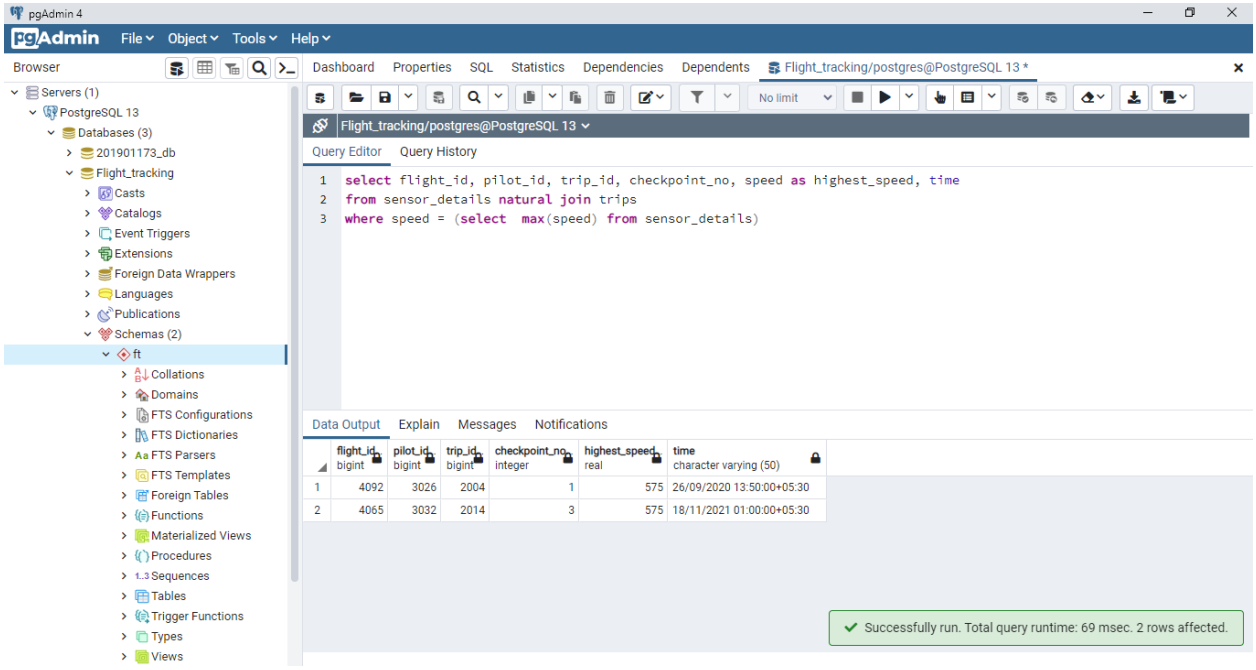
```
select fuel_value
from trips as t,sensor_details as s
where t.trip_id=2004 and s.trip_id=2004 and s.checkpoint_no=2
```



**25.)** Find the all time highest speed of any aircraft and who was flying it

**SQL Query -**

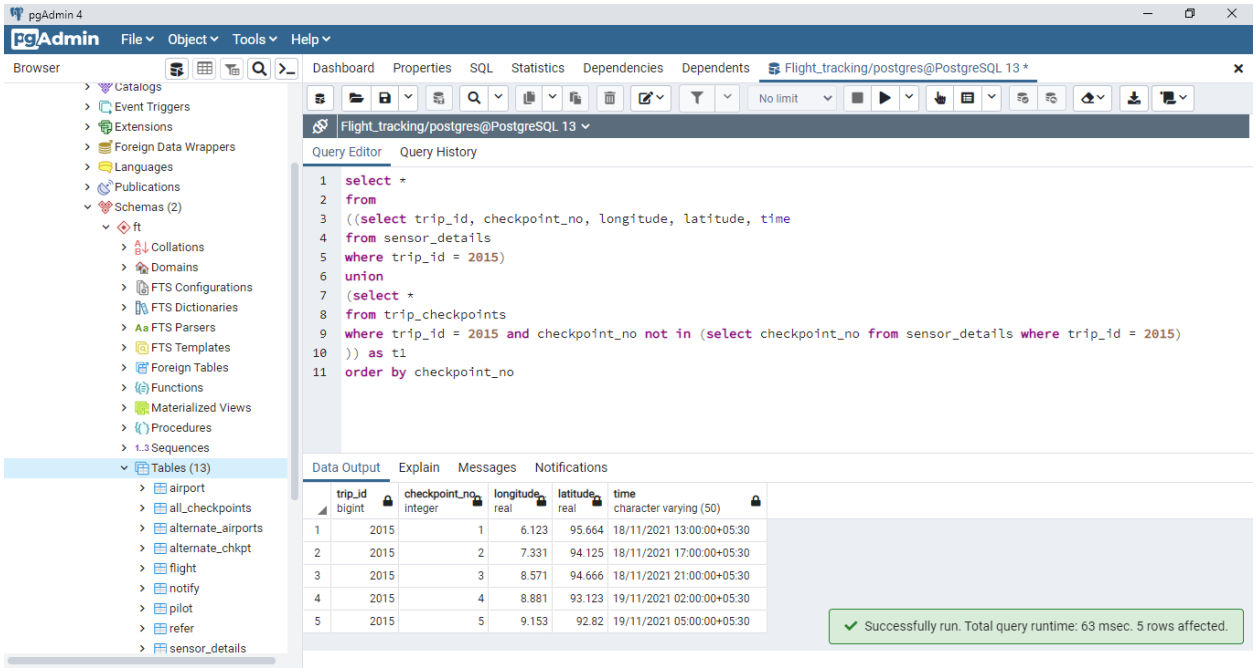
```
select flight_id, pilot_id, trip_id, checkpoint_no, speed as highest_speed, time
from sensor_details natural join trips
where speed = (select  max(speed)  from sensor_details)
```




**26.)** Print the path of the trip '2015' till now along with its estimated time if checkpoint not travelled or actual time if travelled.

**SQL Query -**

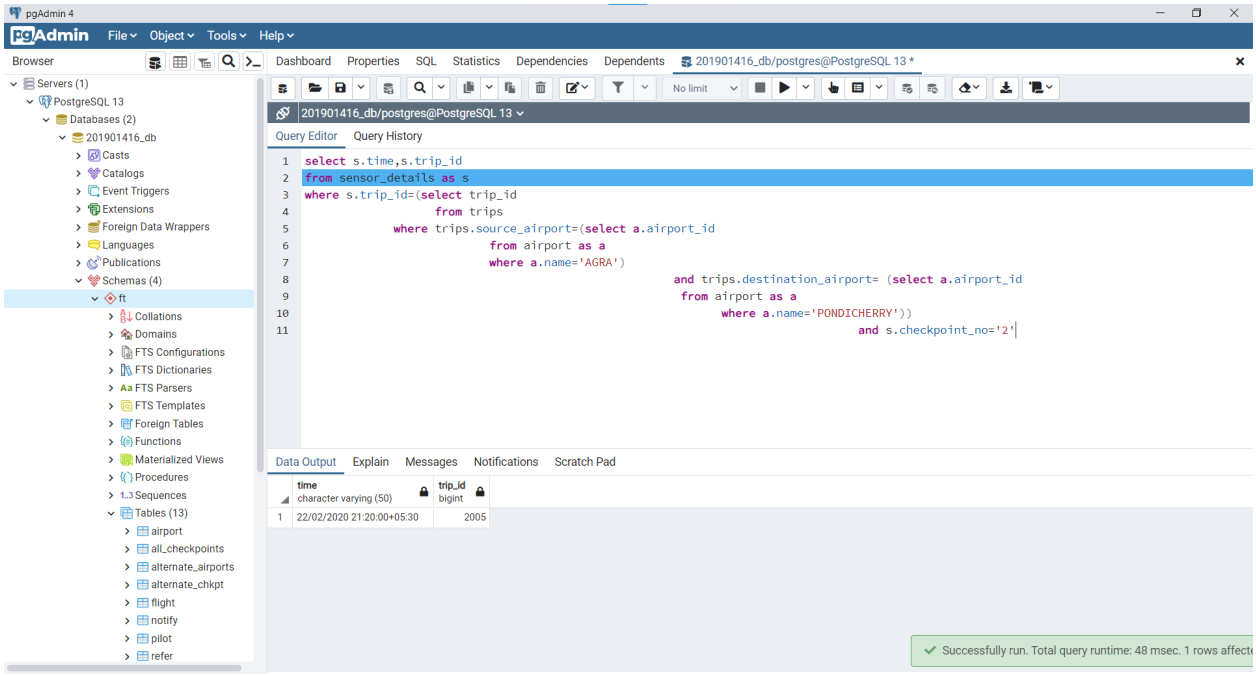
```
select *
from
((select trip_id, checkpoint_no, longitude, latitude, time
from sensor_details
where trip_id = 2015)
union
(select *
from trip_checkpoints
where trip_id = 2015 and checkpoint_no not in (select checkpoint_no from sensor_details where trip_id = 2015)
)) as t1
order by checkpoint_no
```




**27.)** What was the arrival time of flight on a trip from AGRA to PONDICHERRY when it reached checkpoint number 2?

**SQL Query -**

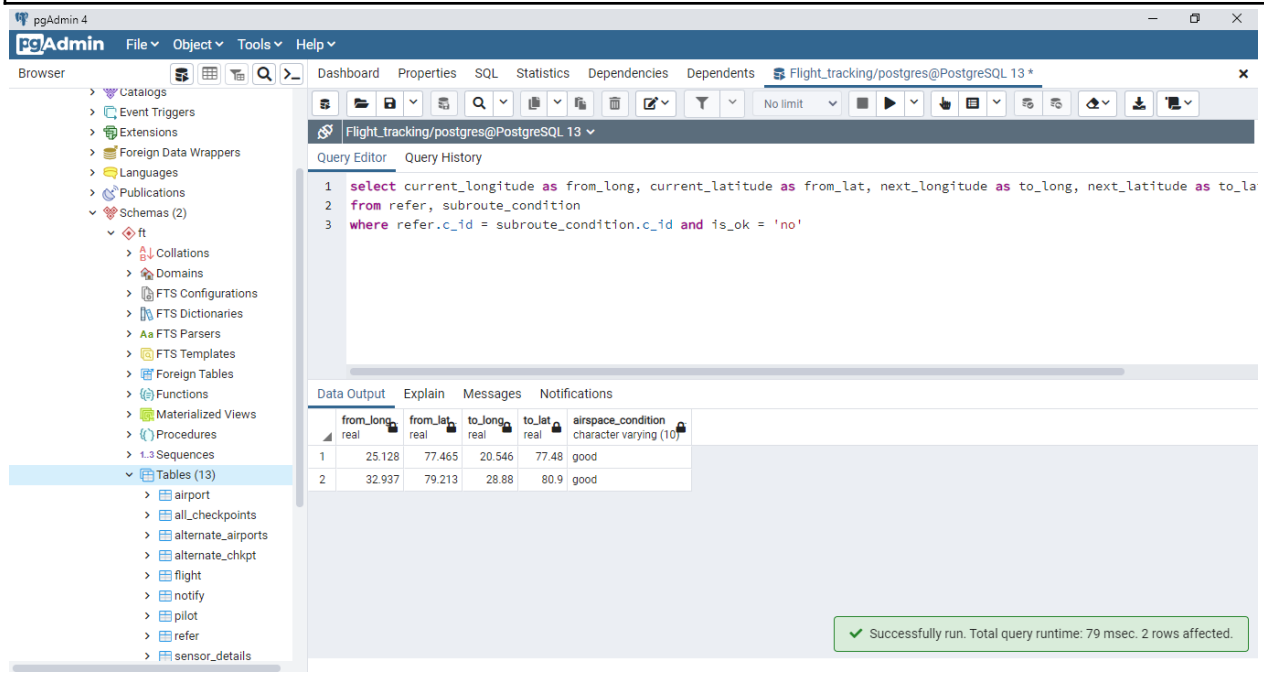
```
select s.time,s.trip_id
from sensor_details as s
where s.trip_id=(select trip_id
                 from trips
                    where trips.source_airport=(select a.airport_id
                                                from airport as a
                                                where a.name='AGRA')
                                            and trips.destination_airport= (select a.airport_id
                                                                            from airport as a
                                                                           where a.name='PONDICHERRY'))
                                                                        and s.checkpoint_no='2''
```




**28.)** What is the condition of subroute now which was bad for some flights in the past.

**SQL Query -**

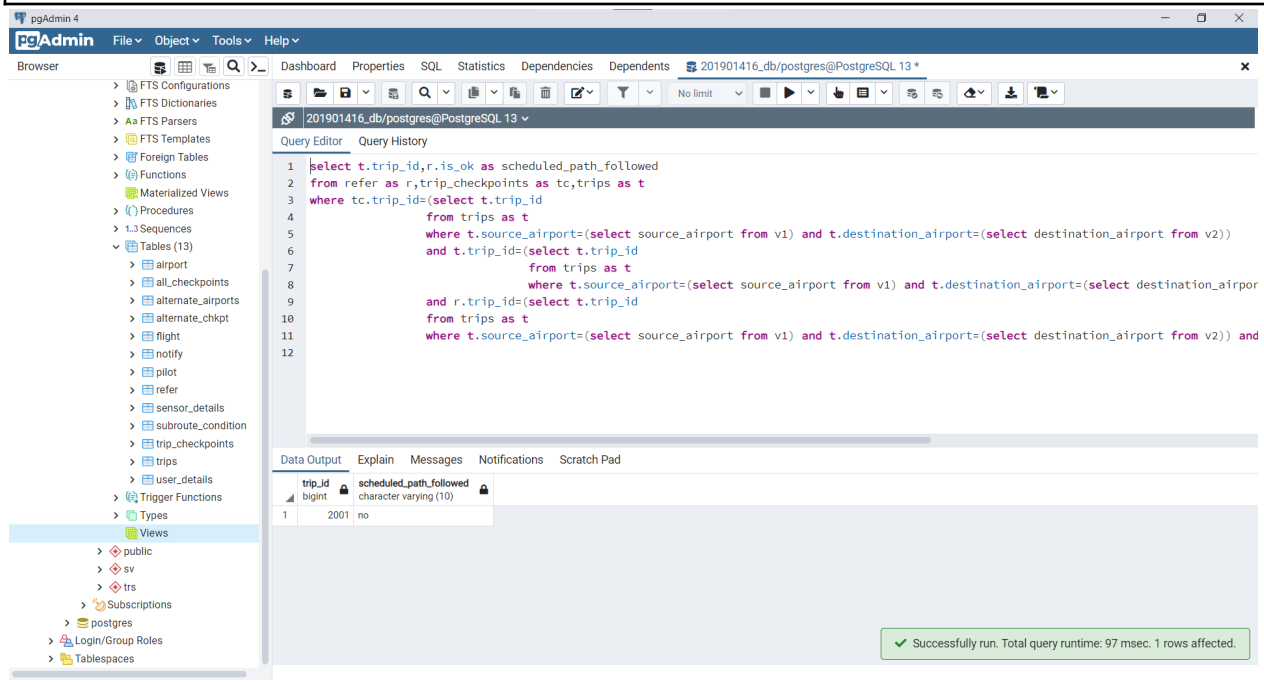
```
select current_longitude as from_long, current_latitude as from_lat, next_longitude as to_long, next_latitude as
to_lat, airspace_condition
from refer, subroute_condition
where
refer.c_id = subroute_condition.c_id and is_ok = 'no'
```



**29.)** Did all the flights on the trip from SARSAWA to BIDAR follow the scheduled route from checkpoint 1 to checkpoint 2?

**SQL Query -**

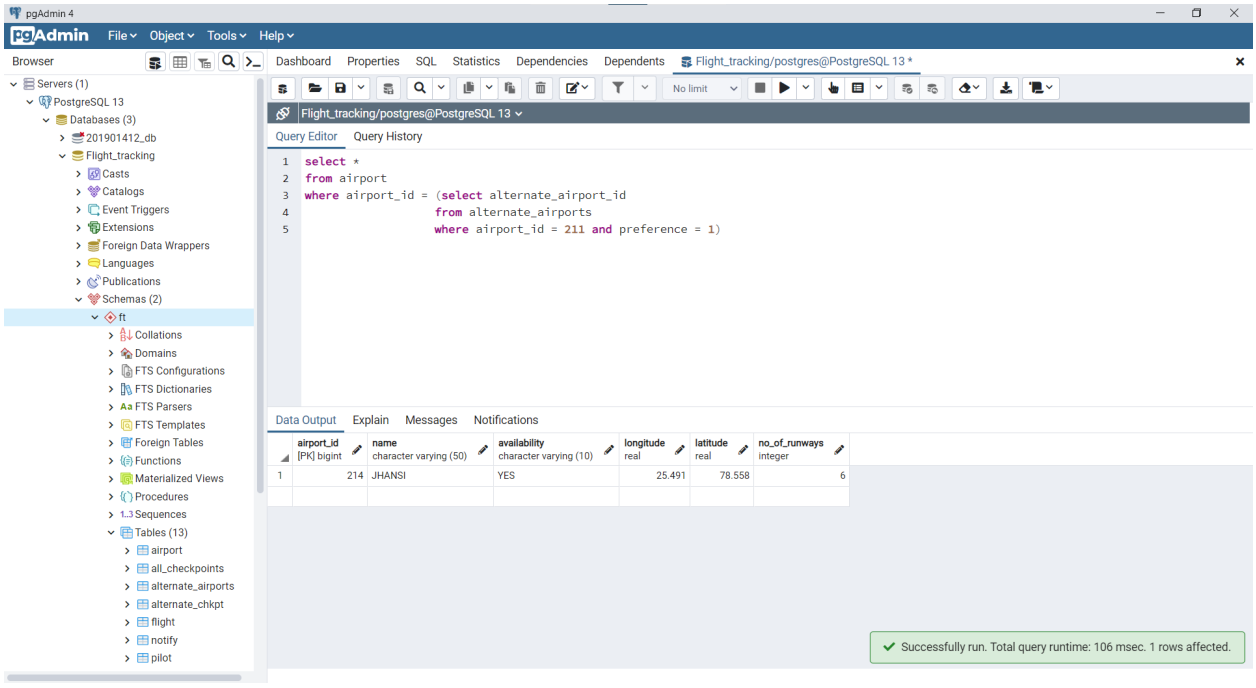
```
select t.trip_id,r.is_ok as scheduled_path_followed
from refer as r,trip_checkpoints as tc,trips as t
where tc.trip_id=(select t.trip_id
                  from trips as t
                  where t.source_airport=(select source_airport from v1) and t.destination_airport=(select
destination_airport from v2))
                                    and t.trip_id=(select t.trip_id
                                    from trips as t
                                    where t.source_airport=(select source_airport from v1) and
t.destination_airport=(select destination_airport from v2))
                                    and r.trip_id=(select t.trip_id
                                    from trips as t
                                    where t.source_airport=(select source_airport from v1) and
t.destination_airport=(select destination_airport from v2)) and r.trip_chkpt_no='2' and tc.checkpoint_no='2' and
r.alternate_chkpt_pref='0'
```




**30.)** Give all details of the first preference of alternate airport for airport id '211'.

**SQL Query -**

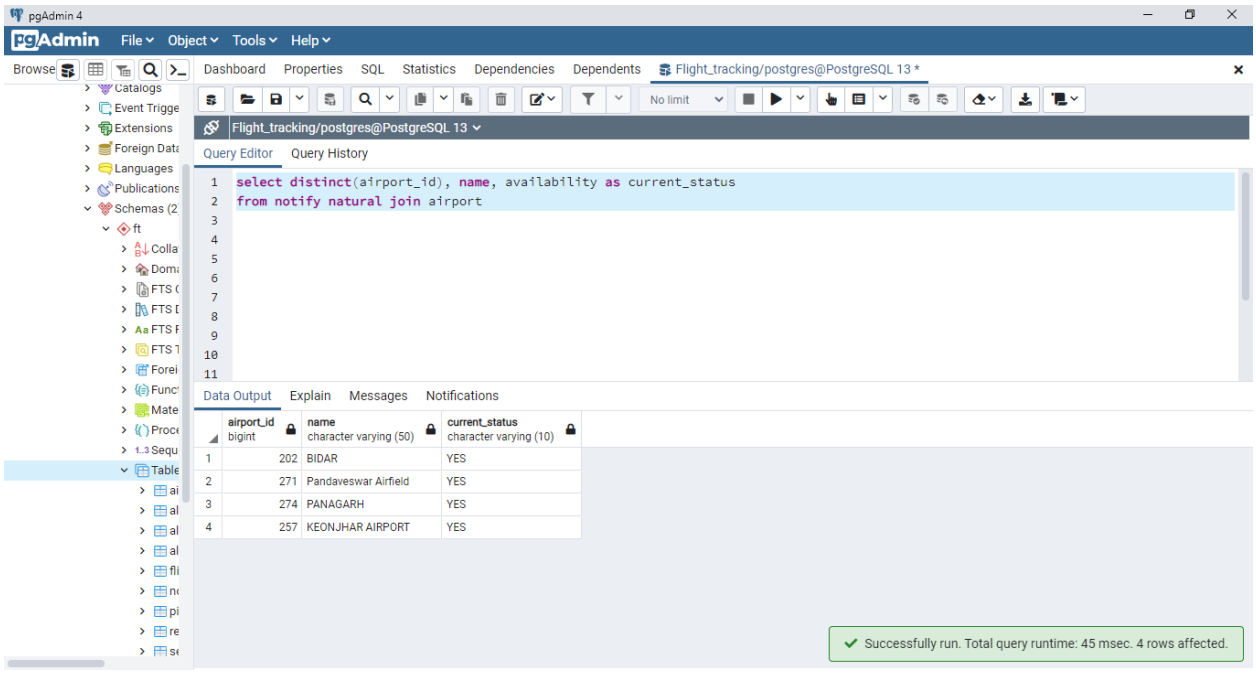
```
select *
from airport
where airport_id = (select alternate_airport_id
                                from alternate_airports
                                where airport_id = 211 and preference = 1)
```




**31.)** What is the condition of the airport now which notified the pilot once.

**SQL Query -**

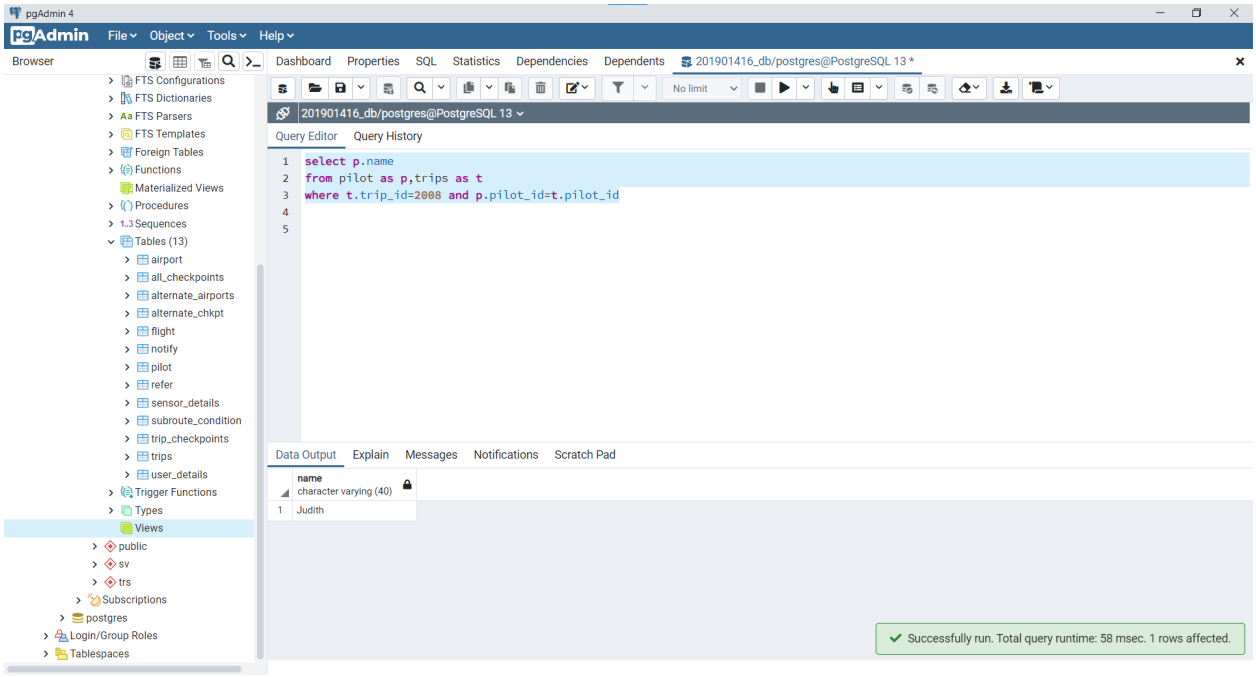
```
select airport_id, name, availability as current_status
from notify natural join airport
```



**32.)** What is the name of the pilot who had completed the trip having trip id 2008?

**SQL Query -**

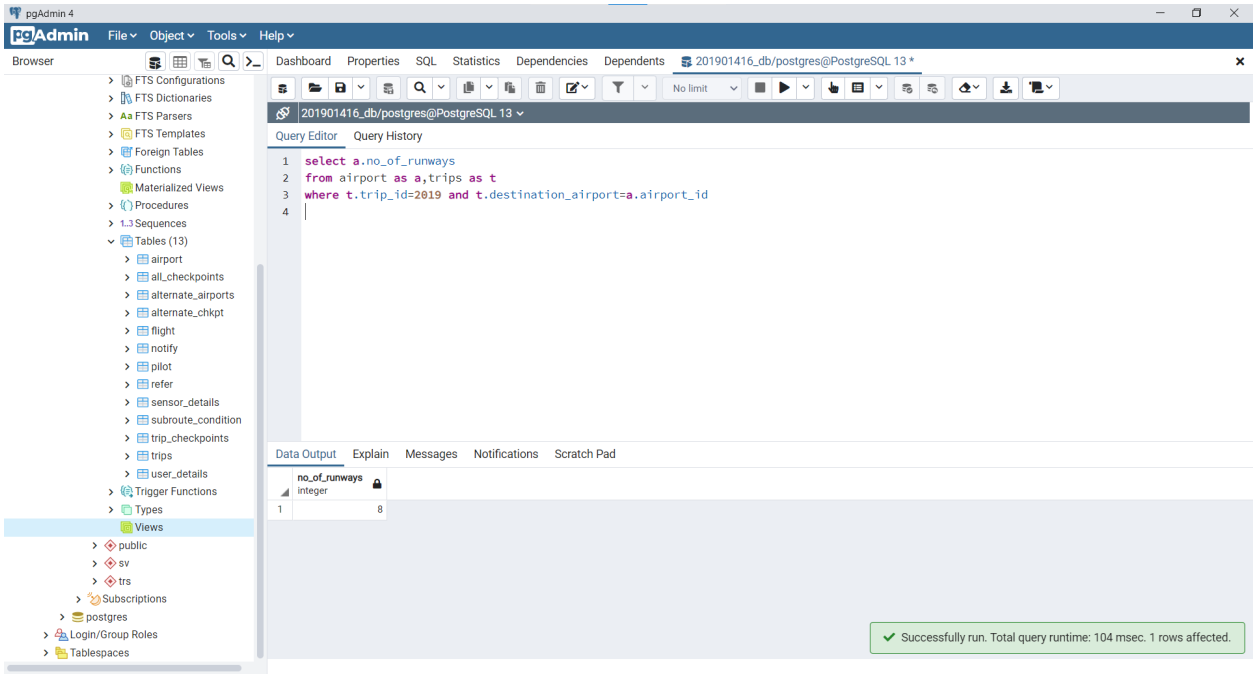
```
select p.name
from pilot as p,trips as t
where t.trip_id=2008 and p.pilot_id=t.pilot_id
```




**33.)** How many runways are present at the destination airport for the trip_id 2019?

**SQL Query -**

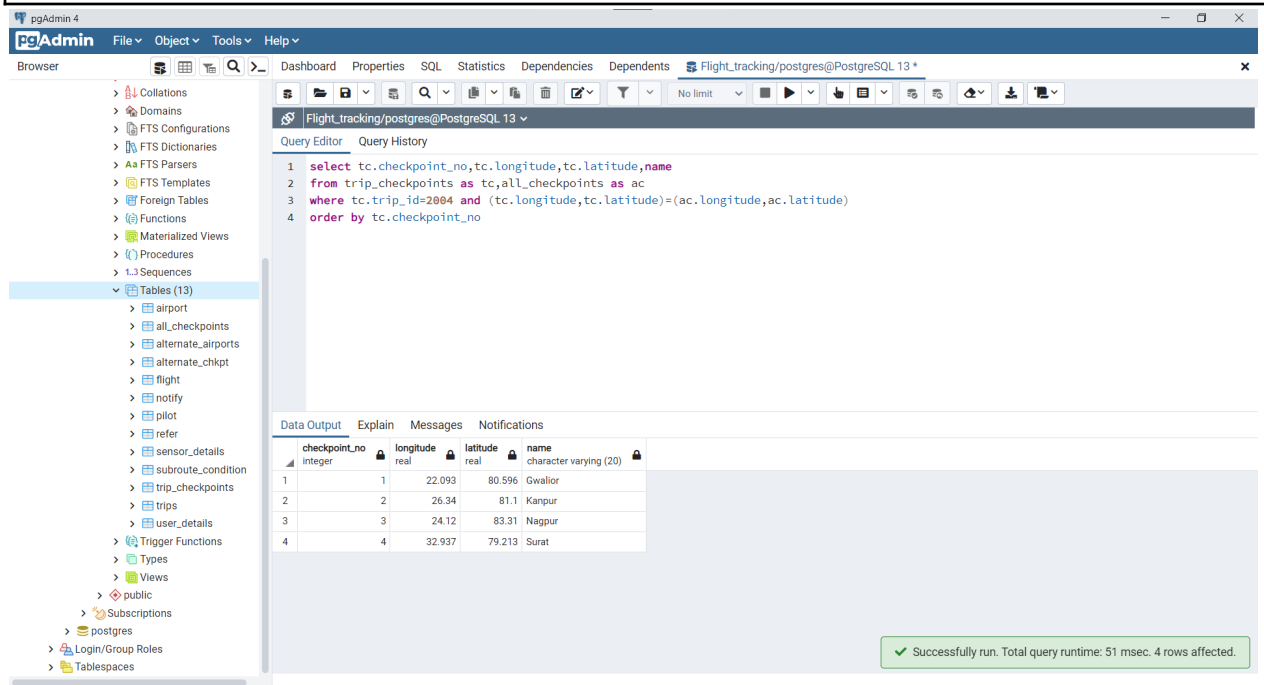
```
select a.no_of_runways
from airport as a,trips as t
where t.trip_id=2019 and t.destination_airport=a.airport_id
```




**34.)** Give longitude,latitude and name of location of all the checkpoints of trip_id=2004

**SQL Query -**

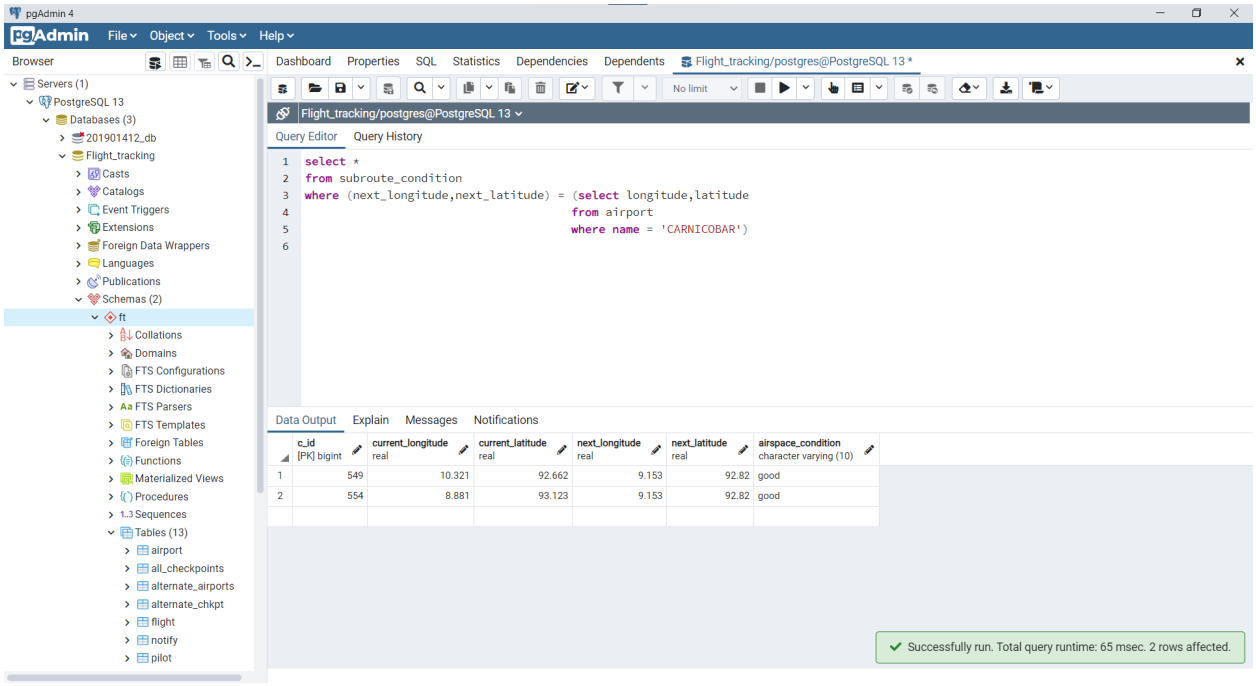
```
select tc.checkpoint_no,tc.longitude,tc.latitude,name
from trip_checkpoints as tc,all_checkpoints as ac
where tc.trip_id=2004 and (tc.longitude,tc.latitude)=(ac.longitude,ac.latitude)
order by tc.checkpoint_no
```



**35.)** Find the condition of the subroute approaching CARNICOBAR airport.

**SQL Query -**

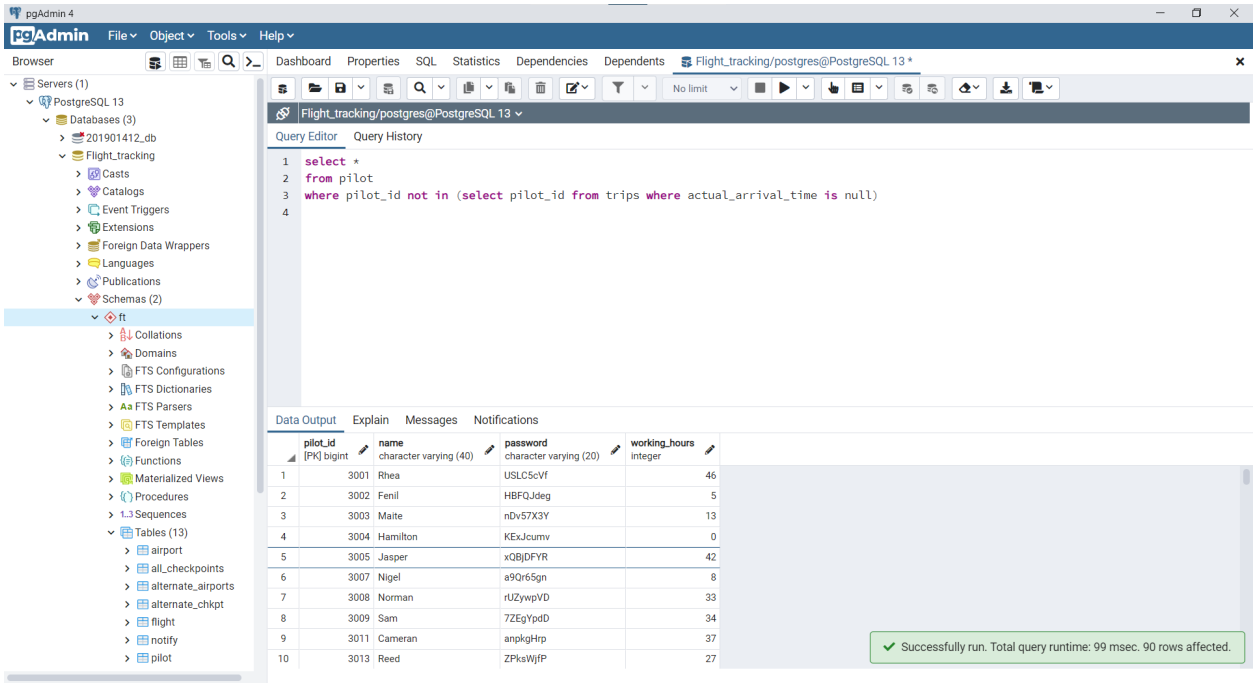
```
select *
from subroute_condition
where (next_longitude,next_latitude) = (select longitude,latitude from airport
                                                    where name = 'CARNICOBAR')
```




**36.)** List the pilots which can be alloted a trip now.

**SQL Query -**

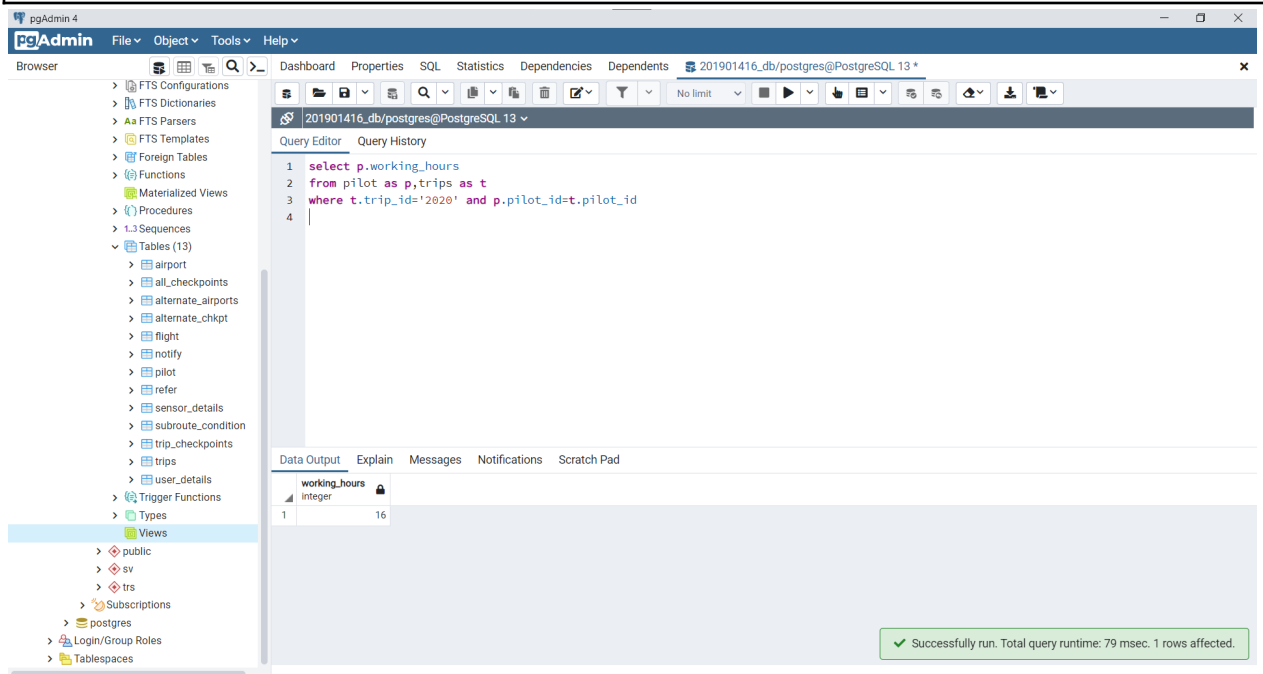
```
select *
from pilot
where pilot_id not in (select pilot_id from trips where actual_arrival_time is null)
```




**37.)**What are the working hours of the pilot who is going on a trip having trip id 2020?

**SQL Query -**

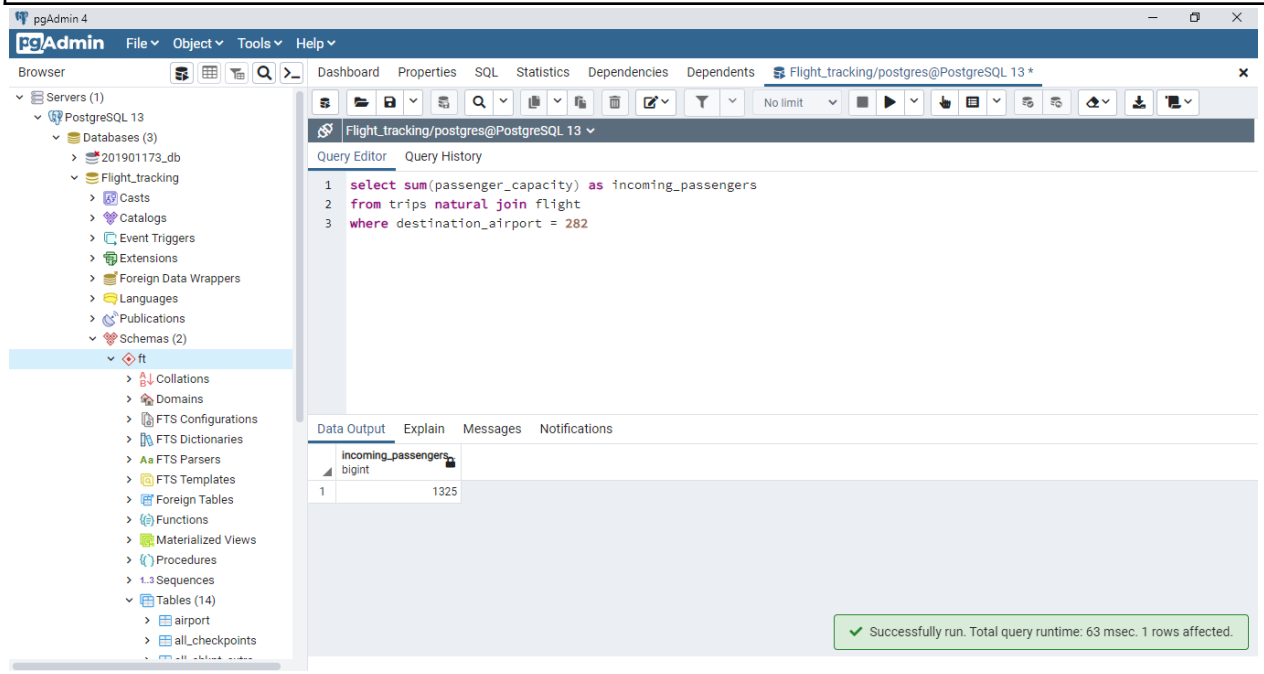
```
select p.working_hours
from pilot as p,trips as t
where t.trip_id='2020' and p.pilot_id=t.pilot_id
```



**38.)** Find the expected incoming flow of passengers at airport 282 assuming the related flights are running full.

**SQL Query -**

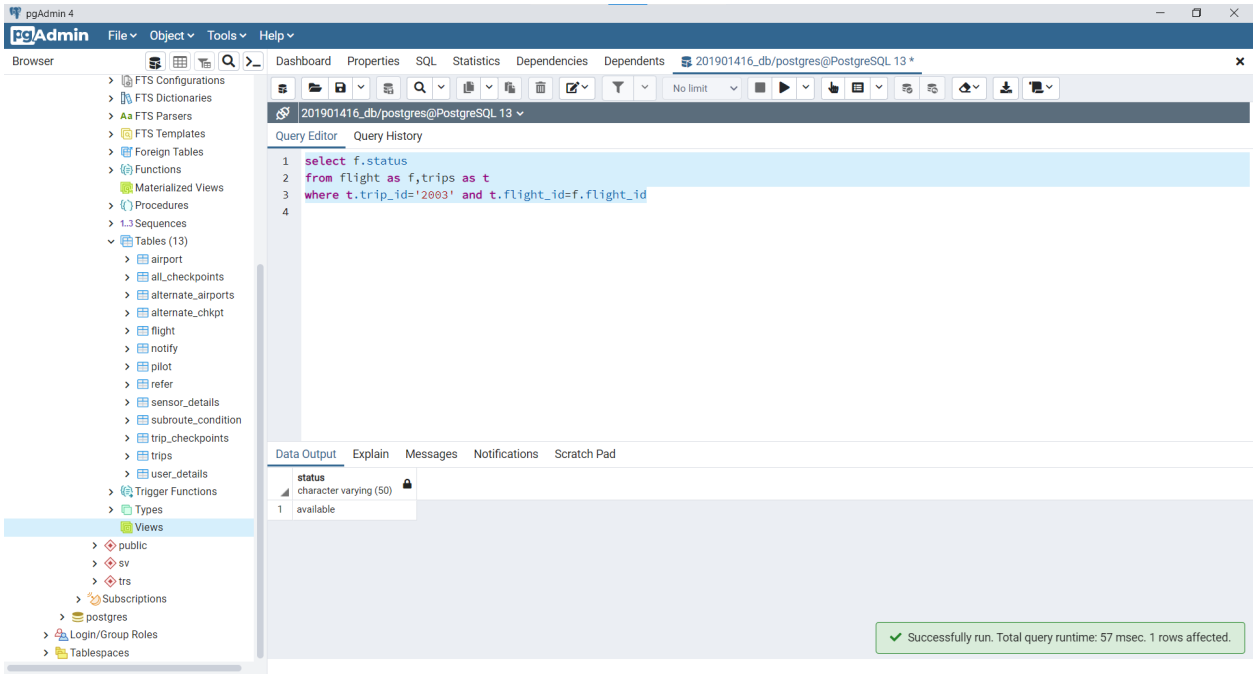
```
select sum(passenger_capacity) as incoming_passengers
from trips natural join flight
where destination_airport = 282
```




**39.)** What is the current status of the flight that was on trip having trip id 2003?

**SQL Query -**

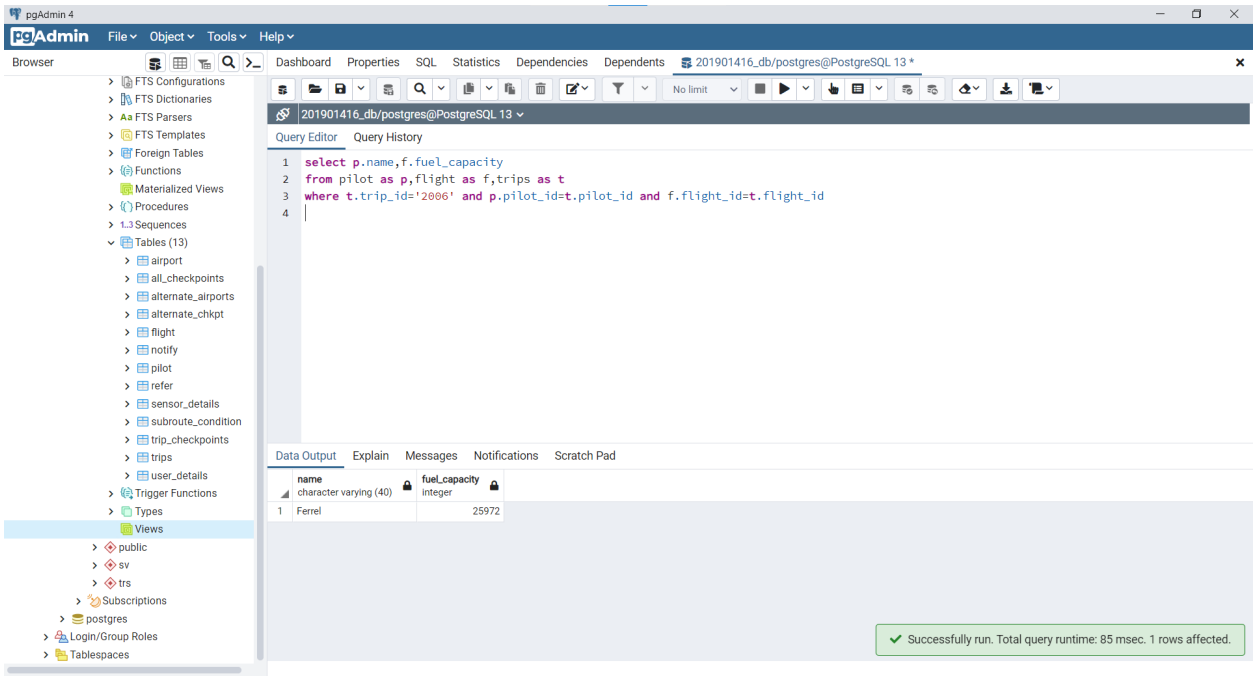
```
select f.status
from flight as f,trips as t
where t.trip_id='2003' and t.flight_id=f.flight_id
```




**40.)** What was the fuel capacity of flight and pilot name who was on a trip having trip id 2006?

**SQL Query -**

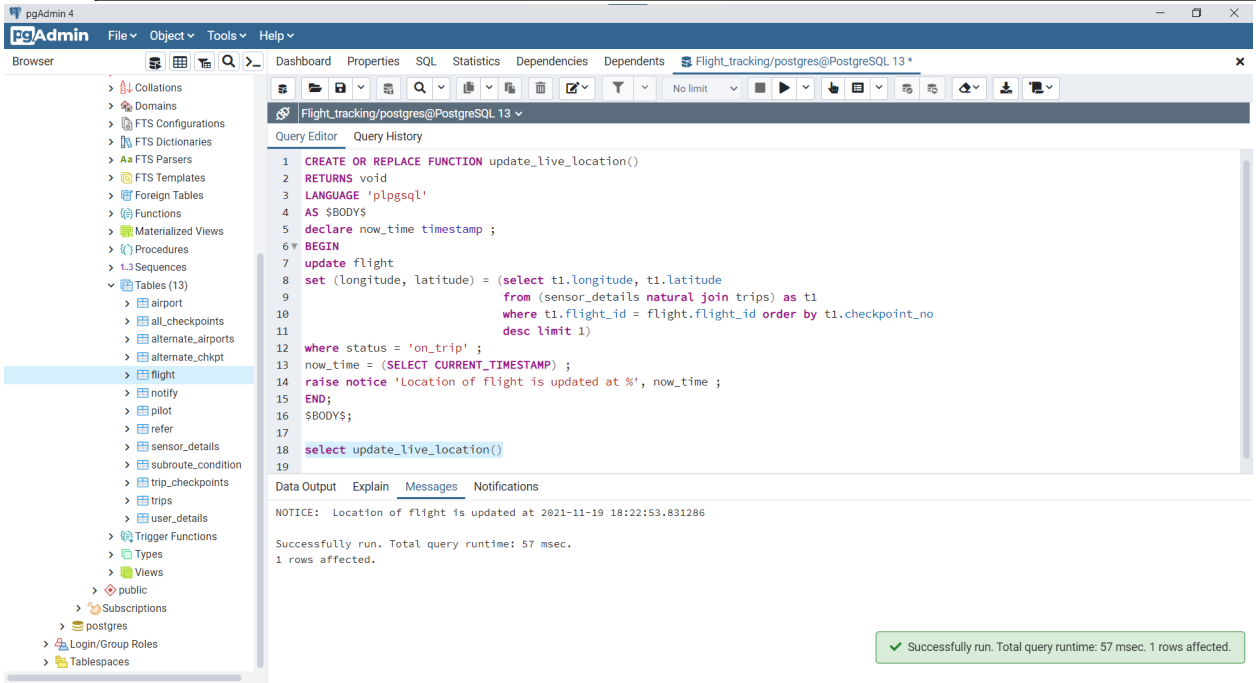
```
select p.name,f.fuel_capacity
from pilot as p,flight as f,trips as t
where t.trip_id='2006' and p.pilot_id=t.pilot_id and f.flight_id=t.flight_id
```



# III. Functions

**1.)** On calling this function, update the longitude and latitude in the flight table with the longitude and latitude of the last travelled checkpoint longitude and latitude.

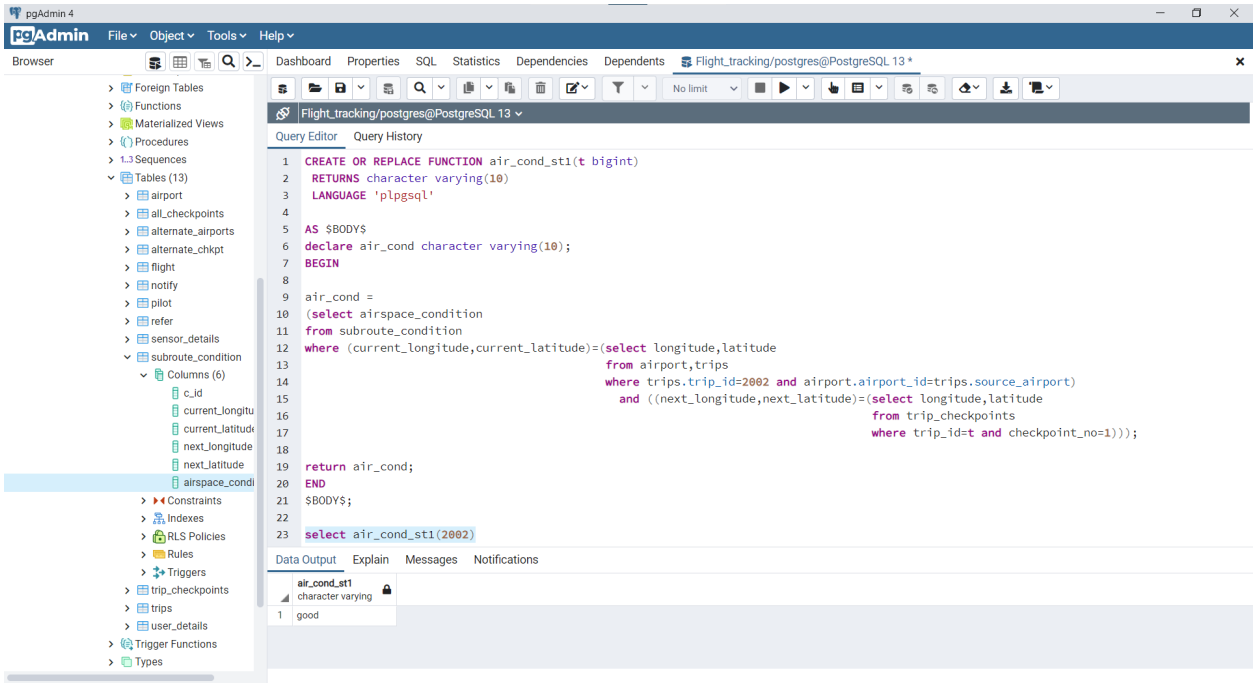
```
CREATE OR REPLACE FUNCTION update_live_location()
RETURNS void
LANGUAGE 'plpgsql'
AS $BODY$
declare now_time timestamp ;
BEGIN
update flight
set (longitude, latitude) = (select t1.longitude, t1.latitude
                                    from (sensor_details natural join trips) as t1
                                    where t1.flight_id = flight.flight_id
                             order by t1.checkpoint_no desc limit 1)
                            where status = 'on_trip' ;
now_time = (SELECT CURRENT_TIMESTAMP) ;
raise notice 'Location of flight is updated at %', now_time ;
END;
$BODY$;

-- calling function
select update_live_location()
```




**2.)** What is the airspace condition from source airport to 1st checkpoint in a given trip.

```
CREATE OR REPLACE FUNCTION air_cond_st1(t bigint)
RETURNS character varying(10)
LANGUAGE 'plpgsql'
```

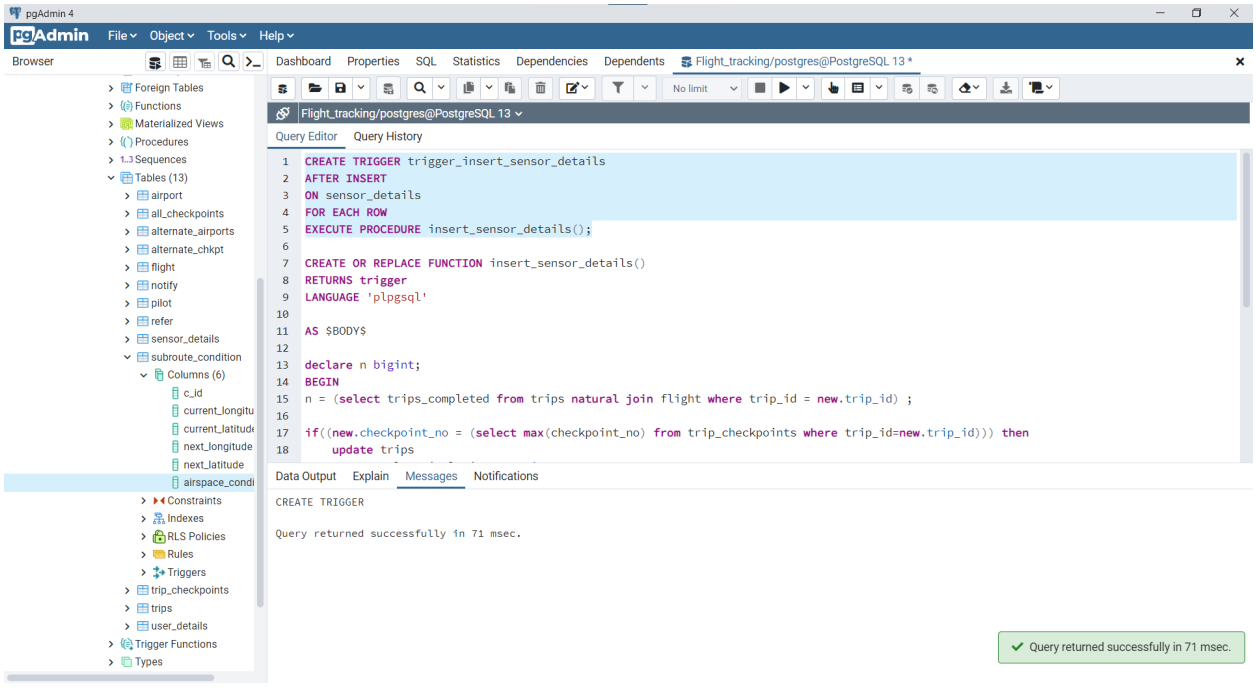
```
AS $BODY$
declare air_cond character varying(10);
BEGIN

air_cond =
(select airspace_condition
from subroute_condition
where (current_longitude,current_latitude)=(select longitude,latitude
                    from airport,trips
                       where trips.trip_id=t and airport.airport_id=trips.source_airport)
                       and ((next_longitude,next_latitude)=(select longitude,latitude
                                                  from trip_checkpoints
                                                  where trip_id=t and checkpoint_no=1)));

        return air_cond;
          END
$BODY$;
```

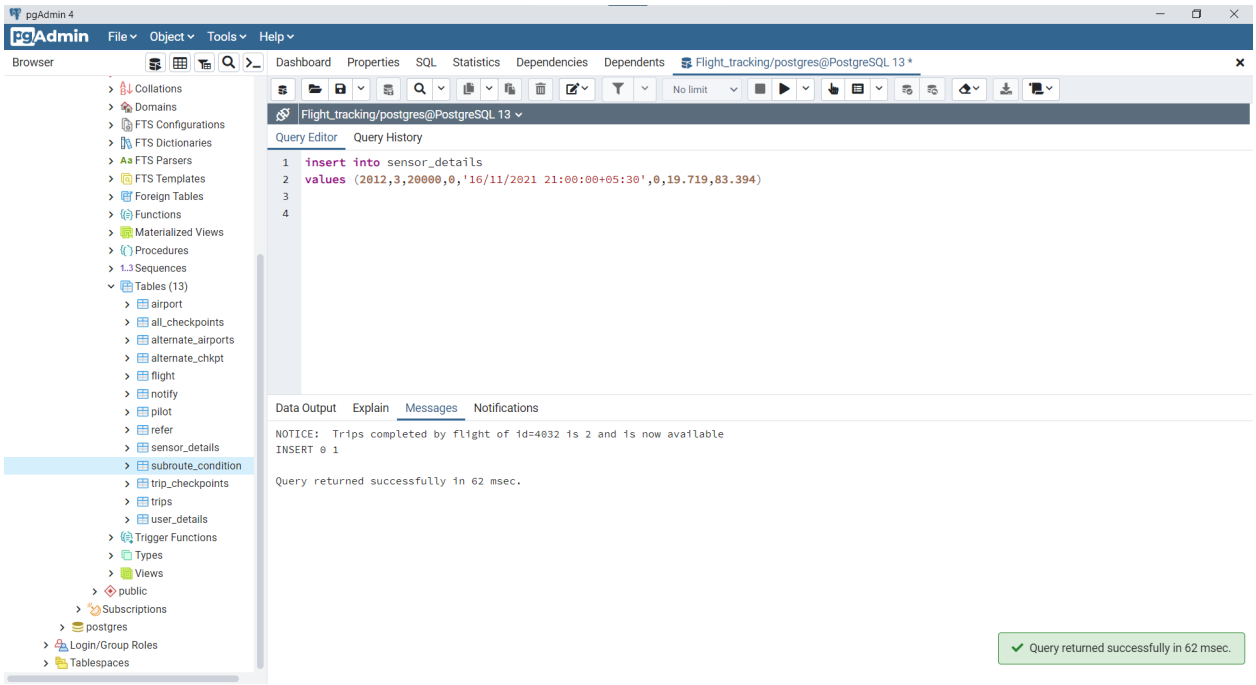



# IV. Triggers and trigger functions

**1.)** When the detail for the last checkpoint for a trip is inserted in sensor details then update the actual arrival time of that trip in trips and increment the number of trips completed by that flight. If it passes the threshold then make its status ‘in_maintainence’ else make it available.

```
CREATE TRIGGER trigger_insert_sensor_details
AFTER INSERT
ON sensor_details
FOR EACH ROW
EXECUTE PROCEDURE insert_sensor_details();

CREATE OR REPLACE FUNCTION insert_sensor_details()
RETURNS trigger
LANGUAGE 'plpgsql'

AS $BODY$

declare n integer;
declare f_id bigint;
BEGIN

f_id = (select flight_id from trips natural join flight where trip_id = new.trip_id) ;

if((new.checkpoint_no = (select max(checkpoint_no) from trip_checkpoints where trip_id=new.trip_id)))
then
        update trips
        set actual_arrival_time=new.time
        where trip_id=new.trip_id;
        update flight
        set trips_completed = trips_completed + 1, status = 'available'
        where flight_id = f_id;

        n = (select trips_completed from trips natural join flight where trip_id = new.trip_id) ;

        if(n>16) then
        update flight
        set status = 'in_maintenance', trips_completed=0
        where flight_id = f_id;
        raise notice 'Flight of id=% is now in maintenance',f_id;
        else
        raise notice 'Trips completed by flight of id=% is % and is now available',f_id,n;
        end if;
end if;

RETURN NEW;
END
$BODY$;

insert into sensor_details
values (2012,3,20000,0,'16/11/2021 21:00:00+05:30',0,19.719,83.394)
```


pgAdmin 4
File Object Tools Help
Browser
Dashboard Properties SQL Statistics Dependencies Dependents Flight_tracking/postgres@PostgreSQL 13 *
Flight_tracking/postgres@PostgreSQL 13
Query Editor Query History
CREATE TRIGGER trigger_insert_sensor_details
AFTER INSERT
ON sensor_details
FOR EACH ROW
EXECUTE PROCEDURE insert_sensor_details();
CREATE OR REPLACE FUNCTION insert_sensor_details()
RETURNS trigger
LANGUAGE 'plpgsql'
AS $BODY$
declare n bigint;
BEGIN
n = (select trips_completed from trips natural join flight where trip_id = new.trip_id) ;
if((new.checkpoint_no = (select max(checkpoint_no) from trip_checkpoints where trip_id=new.trip_id))) then
update trips
Data Output Explain Messages Notifications
CREATE TRIGGER
Query returned successfully in 71 msec.
Query returned successfully in 71 msec.


pgAdmin 4
File Object Tools Help
Browser
Dashboard Properties SQL Statistics Dependencies Dependents Flight_tracking/postgres@PostgreSQL 13 *
Flight_tracking/postgres@PostgreSQL 13
Query Editor Query History
insert into sensor_details
values (2012,3,20000,0,'16/11/2021 21:00:00+05:30',0,19.719,83.394)
Data Output Explain Messages Notifications
NOTICE: Trips completed by flight of id=4032 is 2 and is now available
INSERT 0 1
Query returned successfully in 62 msec.
Query returned successfully in 62 msec.

# Section 7: Project Code with output screenshots.

## 1. Python and html code :

This code connects to our database. Inserts a new user and lists all the users.

- query_runner.py

```
from flask import Flask, render_template, request
import cgi
import time
import pymonetdb
import psycopg2

query_runner = Flask(__name__)

@query_runner.route('/')
def enter_query() :
   return render_template('enter_query.html')

@query_runner.route('/runmyquery', methods=['POST'])
def runmyquery() :
   connection = psycopg2.connect(host="localhost",database="Flight_tracking",user="postgres",password="admin")
   cursor=connection.cursor()
   print_line = request.form['query']
   cursor.execute(print_line)
   connection.commit()
   if print_line[0] == 's' :
      cursor.execute("Select * FROM (" + print_line + ") as xyz LIMIT 0")
      colnames = [desc[0] for desc in cursor.description]
      mssg = cursor.fetchall()
      return render_template('show_result.html', cols=colnames, mssg=mssg)
   return render_template('pass.html')

@query_runner.route('/userinsert', methods=['GET', 'POST'])
def userinsert() :
   newuser_id = request.form['userid']
   newuser_name = request.form['name']
   newuser_emailid = request.form['email']
   newuser_psswd = request.form['password']
   connection = psycopg2.connect(host="localhost",database="Flight_tracking",user="postgres",password="admin")
   cursor=connection.cursor()
   print_line="insert into ft.user_details values(" + str(newuser_id) + ",\'"+ str(newuser_name) + "\',\'" +
str(newuser_emailid) + "\',\'" + str(newuser_psswd) + "\')"
   cursor.execute(print_line)
   connection.commit()
   return render_template('pass.html')

@query_runner.route('/pilotinsert', methods=['GET', 'POST'])
def pilotinsert() :
   newuser_id = request.form['pilotid']
   newuser_name = request.form['name']
   newuser_psswd = request.form['password']
   newuser_hours = request.form['hours']
   connection = psycopg2.connect(host="localhost",database="Flight_tracking",user="postgres",password="admin")
   cursor=connection.cursor()
```

```
    print_line="insert into ft.pilot values(" + str(newuser_id) + ",\'"+ str(newuser_name) + "\',\'" + str(newuser_psswd) +
"\',\'" + str(newuser_hours) + "\')"
    cursor.execute(print_line)
    connection.commit()
    return render_template('pass.html')

@query_runner.route('/getroute', methods=['GET', 'POST'])
def getroute() :
    fortripid = request.form['tripid']
    connection = psycopg2.connect(host="localhost",database="Flight_tracking",user="postgres",password="admin")
    cursor=connection.cursor()
    # print_line="set search_path to ft"
    # cursor.execute(print_line)
    print_line="select * from ((select trip_id, checkpoint_no, longitude, latitude, time from ft.sensor_details where trip_id
= 2015) union (select * from ft.trip_checkpoints where trip_id = 2015 and checkpoint_no not in (select checkpoint_no
from ft.sensor_details where trip_id = " + str(fortripid) + ") )) as t1 order by checkpoint_no"
    cursor.execute(print_line)
    mssg = cursor.fetchall()
    cursor.execute("Select * FROM (select * from ((select trip_id, checkpoint_no, longitude, latitude, time from
ft.sensor_details where trip_id = 2015) union (select * from ft.trip_checkpoints where trip_id = 2015 and checkpoint_no
not in (select checkpoint_no from ft.sensor_details where trip_id = " + str(fortripid) + ") )) as t1 order by
checkpoint_no) as xyz LIMIT 0")
    colnames = [desc[0] for desc in cursor.description]
    connection.commit()
    return render_template('show_result.html', cols=colnames, mssg=mssg)

if __name__ == '__main__' :
    query_runner.run(debug=True)
```

- enter_query.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>S1 T2</title>
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">
  </head>
  <body>
    <div class="container">
      <div class="row col-md-6 col-md-offset-3">
        <div class="panel panel-primary">
          <div class="panel-heading text-center">
            <h1>Enter you query</h1>
          </div>
          <div class="panel-body">
            <form name="passdata" action="/runmyquery" method="post">
              <div class="form-group">
                <label for="Enter SQL query">Your query</label>
                <input
                  type="text"
                  class="form-control"
                  id="query"
```

```
        name="query"
      />
    </div>
    <input type="submit" class="btn btn-primary" value="Run"/>
  </form>

  <br><br><br>
  <div class="panel-heading text-center">
  <h1>Insert new user</h1>
  </div>
  <form name="passdata" action="/userinsert" method="post">
    <div class="form-group">
      <label for="userid">User ID</label>
      <input
        type="text"
        class="form-control"
        id="userid"
        name="userid"
      />
    </div>
    <div class="form-group">
      <label for="name">Name</label>
      <input
        type="text"
        class="form-control"
        id="name"
        name="name"
      />
    </div>
    <div class="form-group">
      <label for="email">Email</label>
      <input
        type="text"
        class="form-control"
        id="email"
        name="email"
      />
    </div>
    <div class="form-group">
      <label for="password">Password</label>
      <input
        type="password"
        class="form-control"
        id="password"
        name="password"
      />
    </div>
    <input type="submit" class="btn btn-primary" value="Insert user"/>
  </form>


  <br><br><br>
  <div class="panel-heading text-center">
  <h1>Insert new pilot</h1>
  </div>
  <form name="passdata" action="/pilotinsert" method="post">
```

```
        <div class="form-group">
          <label for="pilotid">Pilot ID</label>
          <input
            type="text"
            class="form-control"
            id="pilotid"
            name="pilotid"
          />
        </div>
        <div class="form-group">
          <label for="name">Name</label>
          <input
            type="text"
            class="form-control"
            id="name"
            name="name"
          />
        </div>
        <div class="form-group">
          <label for="password">Password</label>
          <input
            type="password"
            class="form-control"
            id="password"
            name="password"
          />
        </div>
        <div class="form-group">
          <label for="hours">Working hours(weekly)</label>
          <input
            type="text"
            class="form-control"
            id="hours"
            name="hours"
          />
        </div>
        <input type="submit" class="btn btn-primary" value="Insert pilot"/>
      </form>

      <br><br><br>
      <div class="panel-heading text-center">
      <h1>Get path details of a trip</h1>
      </div>
      <form name="passdata" action="/getroute" method="post">
        <div class="form-group">
          <label for="tripid">Trip ID</label>
          <input
            type="text"
            class="form-control"
            id="tripid"
            name="tripid"
          />
        </div>
        <input type="submit" class="btn btn-primary" value="Get path"/>
      </form>
    </div>
```

```
      </div>
    </div>
  </div>
</div>
 </body>
</html>
```

- show_result.html

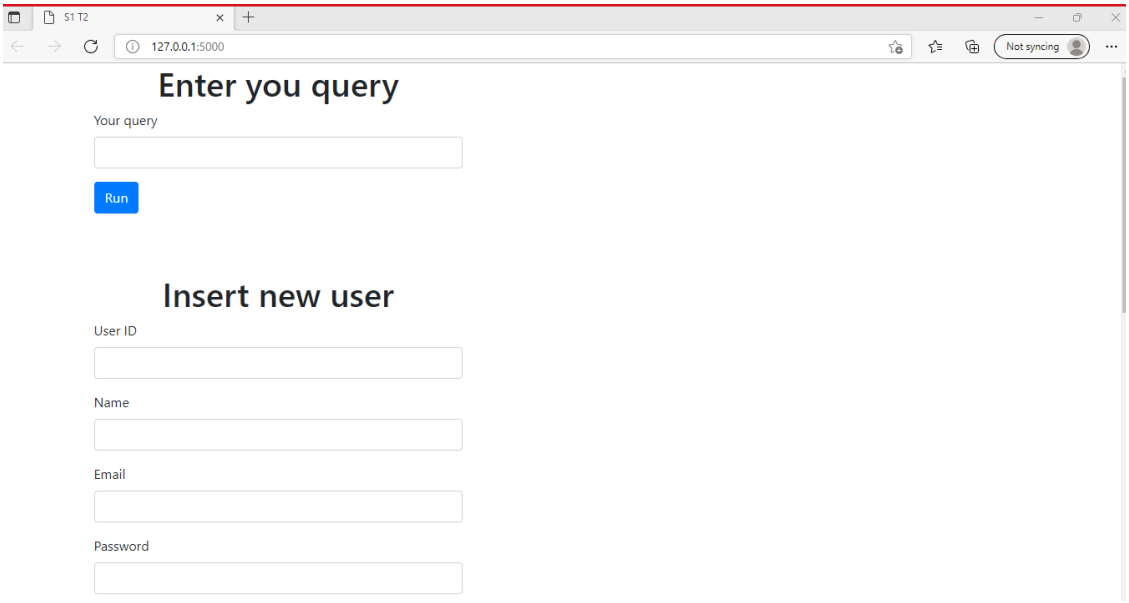
```
<!DOCTYPE html>
<html lang="en">
 <head>
    <meta charset="UTF-8">
  <title>show_result</title>
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet"
id="bootstrap-css">
 </head>
 <body>

 <div class="row">
  <div class="col-md-4">
  <div class="col-md-8">
    <table id="example" class="table table-striped table-bordered" style="width:100%">
      <thead>
        <tr>
          <td>{{cols[0]}}</td>
          <td>{{cols[1]}}</td>
          <td>{{cols[2]}}</td>
          <td>{{cols[3]}}</td>
          {% if cols[4] %}
          <td>{{cols[4]}}</td>
          {% endif %}
        </tr>
      </thead>
      <tbody>
        {% for row in mssg %}
        <tr>
          <td>{{row[0]}}</td>
          <td>{{row[1]}}</td>
          <td>{{row[2]}}</td>
          <td>{{row[3]}}</td>
          {% if row[4] %}
          <td>{{row[4]}}</td>
          {% endif %}
        </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>
 </div>
</div>
</div>
 </body>
</html>
```

- pass.html

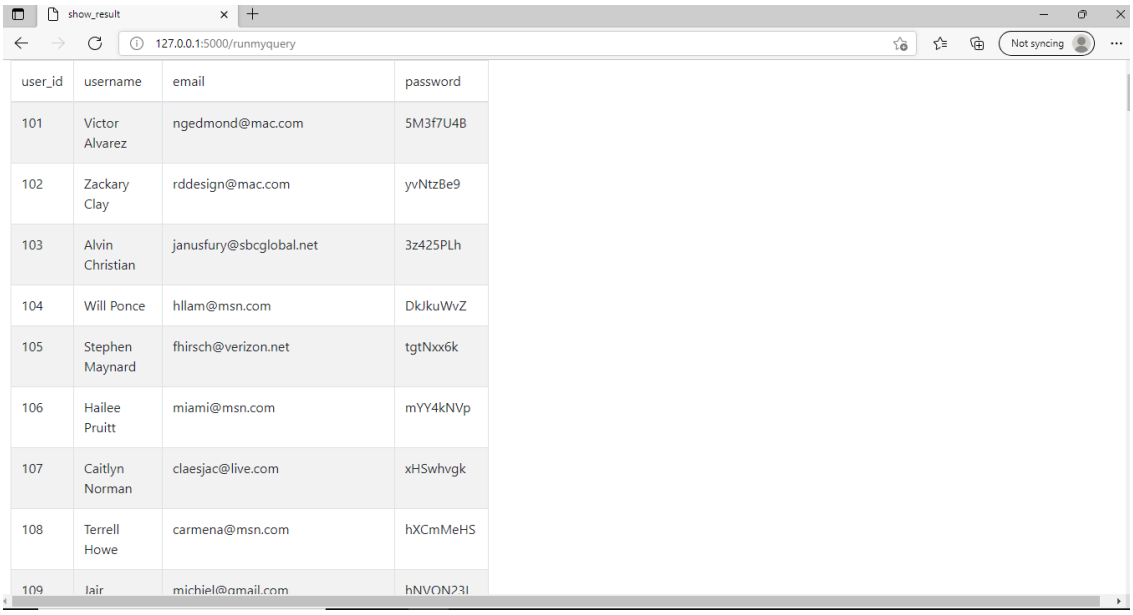
```
<!DOCTYPE html>
<html lang="en">
 <head>
    <meta charset="UTF-8">
  <title>pass</title>
 </head>
 <body>
  <h1>Success</h1>
 </body>
</html>
```

## 2. Screenshots :

1. Web-page

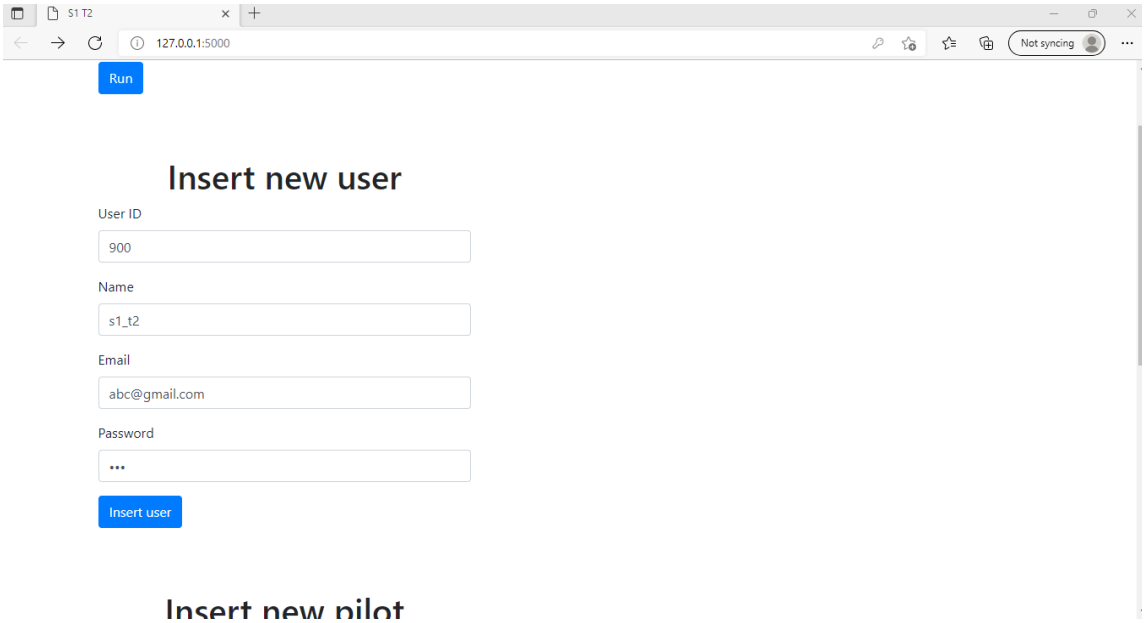


**2.** Showing ‘select * from user_details’ result on web-page

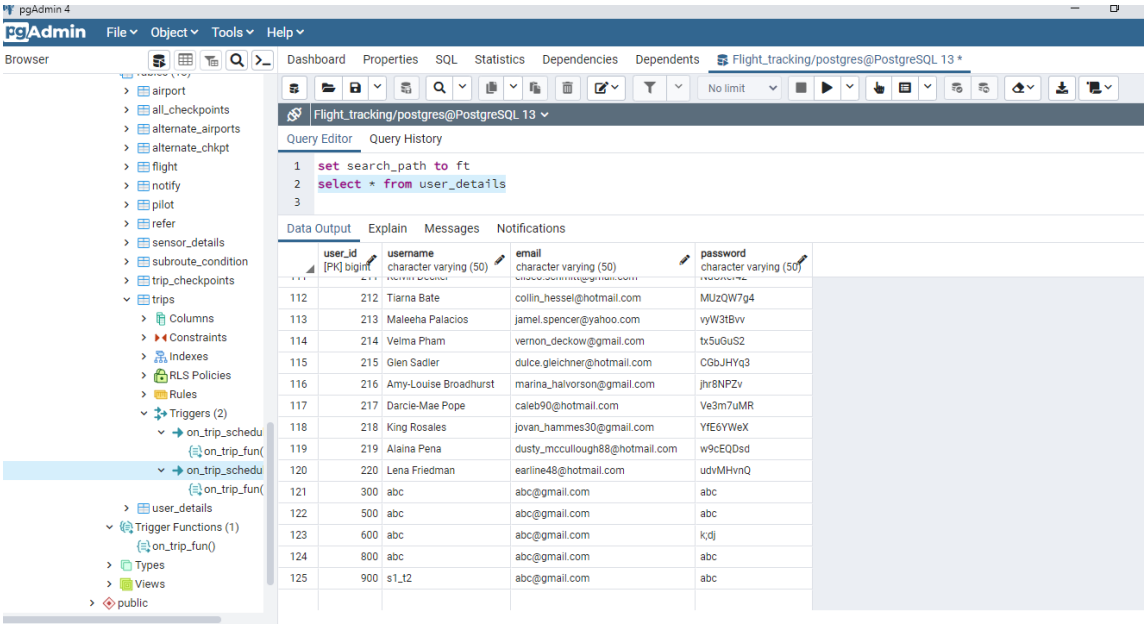



| user_id | username | email | password |
|---|---|---|---|
| 101 | Victor Alvarez | ngedmond@mac.com | 5M3f7U4B |
| 102 | Zackary Clay | rddesign@mac.com | yvNtzBe9 |
| 103 | Alvin Christian | janusfury@sbcglobal.net | 3z425PLh |
| 104 | Will Ponce | hllam@msn.com | DkJkuWvZ |
| 105 | Stephen Maynard | fhirsch@verizon.net | tgtNxx6k |
| 106 | Hailee Pruitt | miami@msn.com | mYY4kNVp |
| 107 | Caitlyn Norman | claesjac@live.com | xHSwhvgk |
| 108 | Terrell Howe | carmena@msn.com | hXCmMeHS |
| 109 | Jair | michiel@gmail.com | hNVON23I |

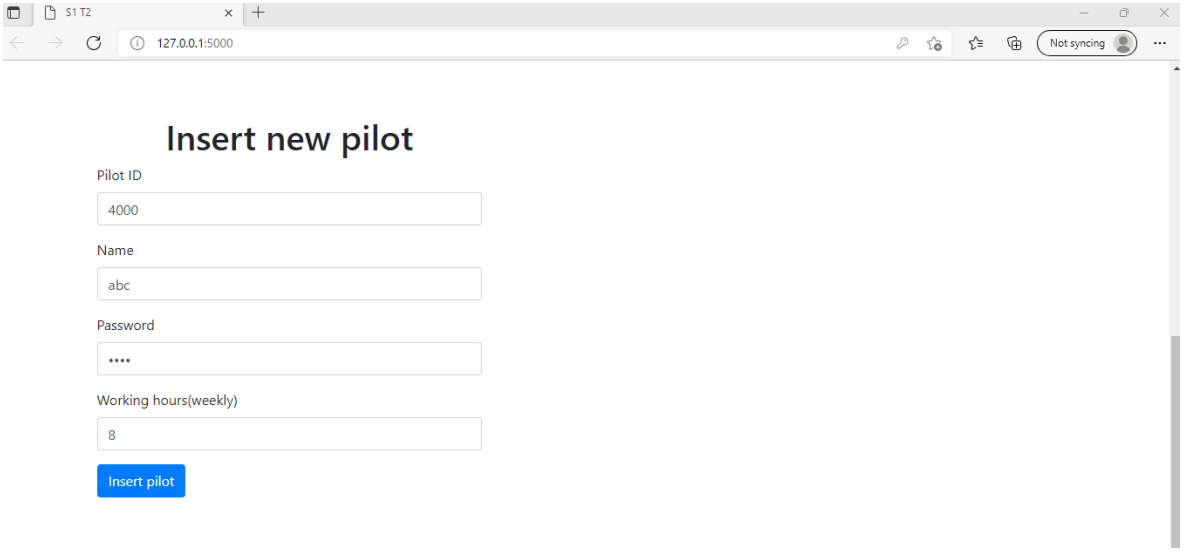
**3.** Inserting new user tuple in database



**4.** Successfully inserted new user tuple in ‘user_details’ database




**5.** Inserting new pilot tuple in database



**6.** Successfully inserted new pilot tuple in ‘pilot’ database