TravelHub: Find Your Perfect Getaway

Created by:

Shreyans Tiwari
Email: shreytiw@seas.upenn.edu
Github: https://github.com/shreyans-tiw

Chengyi Zhang
Email: chengyiz@seas.upenn.edu
Github: https://github.com/122095554

Nimisha Salve
Email: nsalve@seas.upenn.edu
Github: https://github.com/nimisha-salve

Kevin Hu
Email: kevinhu@sas.upenn.edu
Github: https://github.com/kevinh2744

Introduction

Travel is a huge part of a person's life, people travel for business and leisure. Planning ones trip beforehand is very important to have a cost saving and seamless experience in the future. As it has been a few years since the pandemic has come to an end and people have started traveling more frequently. It has been very tedious for a user to visit multiple websites to view different aspects related to traveling. They cannot make comparisons among the prices of these aspects and also cannot view the costs of their overall trip. We understand that travelers often face difficulties in comparing prices and making informed decisions due to the lack of an integrated platform. Hence, our website TravelHub provides users with a seamless experience by offering them the convenience of planning their entire trip on one website.

Our platform offers a wide range of options for users to choose from, including flights, hotels, and Airbnb accommodations. By using reliable databases such as Flights, Hotels, and Airbnb, we provide users with accurate and up-to-date information regarding the availability, prices, and ratings of their preferred travel bookings. This enables users to make informed decisions when it comes to selecting their flights and accommodations.

Architecture

The list of technologies that we have used are in Appendix A.

System Architecture

The flow of the website is such that when the user first enters the Homepage where they can view top Airbnbs and Hotels based on ratings along with viewing some of the top cities that have the highly rated Airbnbs and Hotels. Then the user can decide how they want to plan their trips

as we have multiple pages that assist them for this purpose. The flights page helps the user to view and make a booking for a flights across several cities and presents detailed information about each flight. The user can view hotels and airbnbs combined with the flight details on the planner page, which gives them a complete overview of their entire trip. Then finally comes the Deals page where the user can understand what is most cost effective way to schedule a trip based on the location and duration of stay. We generated complicated queries combining information from multiple tables and optimized the database performance by carefully choosing join orders and selections.

Data

We are using 3 overlapping datasets for this project:

Airbnb
The Airbnb dataset includes Airbnb properties across the US up to the year 2020. The size of the csv file is around 36MB, and contains 226030 entries and 17 columns. The columns include name, city, price, room type, reviews score, etc.
The average price of a listing per night is 219.72 with a SD of 570.35 and a maximum of 24999.00. The average number of reviews for a listing is 34.50 with a SD of 63.60. The top 5 cities with the most listings are New York City (45756), Los Angeles (31536), Hawaii (22434) San Diego (12404), and Broward County (10858).
Link: https://www.kaggle.com/datasets/kritikseth/us-airbnb-open-data

Hotel Reviews
This dataset contains a list of hotels and reviews from Datafiniti's Business Database updated between January 2018 and September 2018, and December 2018 and May 2019. The size of the combined csv file is 174 MB, and contains 20000 entries and 25 columns. The columns include dateAdded, city, country, and review.rating.
The average rating of a review is 4.05 with an SD of 1.16. There are 2975 distinct hotels among the 20000 reviews. The cities with the most reviews include San Diego (1252), New Orleans (981), San Francisco (928), Atlanta (916), Chicago (815).
Link: https://www.kaggle.com/datasets/datafiniti/hotel-reviews

Flights
Flight prices is a dataset containing detailed information about flights within the US. It includes columns like starting airports, destination airports, total fares and other 27 columns. The size of the dataset is 31GB and there's more than 5 million entries.
https://www.kaggle.com/datasets/dilwong/flightprices
(Originally scraped here: https://github.com/dilwong/FlightPrices)

Airports
Airline Flight Delays is a dataset containing all flights in the year of 2015 within the US. It has most of the flights across one whole year. This dataset contains multiple tables: airlines, airports, and flights. We have used the Airports table to get the information between cities and IATA codes. Airports have 7 columns including city, state, airport, flights has 31 columns including start airport, destination, date, etc. The size of the files are 600MB and there's over 5,000,000 entries.
Links: https://www.kaggle.com/datasets/gauravmehta13/airline-flight-delays

The Airbnb and Hotel reviews table are used for the user to decide the place of stay. These tables contain columns like rating, prices, latitude and longitude coordinates and cities information. The Flights table has all the information about flights in the USA along with its base fare, total fare and also the flights for each segment to account for one-stop or two-stop flights. To integrate the cities from the Hotels and Airbnbs with the Flights data, we have introduced the Airports data that overlaps the cities to the IATA code required for flights.

Database

The datasets Flights and Airbnb did not have duplicate rows as each row in each table had its own unique identifier, i.e., LegID for Flights table and Airbnb ID and Host ID for the Airbnb table. The Hotels table did not have a unique identifier for each instance, so we decided to create an index for that table. We also simulated the price column Hotels to make comparisons between expenses of Hotels and Airbnbs.

Relational Schema:

Airbnb(airbnb_id, airbnb_name, host_id, host_name, city, neighbourood_name, airport_code, lat, lon, room_type, price, minimum_nights, number_of_reivews, last_review, reivew_per_month,calculated_host_listings_count, availibilty_365)

hotels (address, city, latitude, longitude, name, postal_code, province, rating, IATA_code, airport, price, hotel_id)

Flights (legID , flightDate, startingAirport, destinationAirport , isBasicEconomy, isRefundable, baseFare, totalFare, seatsRemaining, totalTravelDistance, segmentsDepartureTimeRaw, segmentsArrivalTimeRaw, segmentsAirlineName, SegmentsAirlineCode, segmentsEquipmentDescription)

airports (IATA_CODE, AIRPORT, CITY, STATE)

The statistics of each table are as follows:

Airbnb - 205858 entries and 17 columns.
Flights - 651615 entries and 15 columns.
Hotels - 751 entries and 12 columns.
Airports - 323entries and 4 columns.

Early in our project planning process we observed in our ER diagram that some data was in 2NF. We manipulated our schema to ensure all our relations are in 3NF and subsequently there is no redundancy in the tables after redistributing our data. Since our ER diagram (shown in Appendix B) has many to one relationships, we included additional columns of IATA_Code in the Airbnb and Hotels table. This satisfies the 3NF conditions. We have not decomposed the relations for performance purposes.

Web App Description

The website TravelHub has 5 pages in total.

1) *Homepage:*

The user can view lists of Airbnb properties and Hotels which have the highest ratings. Only the first 10 listings are visible in the beginning but the user can view more listings. There are city cards below these lists that direct you to the city page to give you more information on the top hotels and airbnbs. This page also have a Navigation Bar at the top which can be used to access other pages on the website.

2) *City Page:*

This page again contains detailed information of Airbnb listings and hotel listings for each city based on their ratings. The user can easily view accomodation listings for the respective cities to plan their travel.

3) *Flights Page:*

The Flights page assists the user to get detailed information of the flights going from one city to another on a particular dates. Herein, the user can apply multiple filters such as choosing between number of stops, number of passengers, cost and whether they want to opt for a one way

trip or a two way trip. The passenger can then book the flight for each leg of the trip and get the booking confirmation.

4) *Planner Page:*

When a user wants to plan a trip using start and end destinations, and the duration of stay along with the starting date, this is the most effective page for such searches. Herein, the complete cost analysis of the flight and stay is presented together and the user could also switch between their choice of accommodation i.e. Hotels and Airbnb to get comparisons.

5) *Deals Page:*

The deals page requires the inputs of start and end destinations and duration of the trip after which it outputs the most expensive trip and pocket friendly trip. The user can make the accommodation comparisons for each location and understand how the prices would vary.

API Specification

The API specification for 8 queries including both simple and complex in nature is mentioned in Appendix C.

Queries

The 5 complex queries that were used in our dataset to view and make comparisons amongst prices and rating are shown in Appendix D.

Performance Evaluation

Using a step-by-step join approach and filtering the table at each step can be slow because our temporary table lacks an index. To optimize our queries, we have made use of pushing down the projections and selections as much as possible and selected the appropriate join orders and placed the smaller relation on the outside which has led to significant improvement in the performance of the queries. Below is the table of some complex queries and their runtimes pre and post optimization. Please refer to Appendix E to find the comparisons of these queries.

Query Description	Pre-optimization Runtime	Post-optimization Runtime
Search non-stop, 1-stop and 2-stop one-way flights, given the starting and destination airports, flight date, number of travellers and total fare range.	42s	~ 4 s
Detailed Information of round trip flights using Airbnb	Explodes	629 ms
Most expensive trip with Airbnb	47s	~ 1 s

For the flights table, two indices were added:

1. A non-unique composite index on (startingAirport, flightDate, destinationAirport) called idx_airportsanddate. This allowed for a *Unique Index Scan* instead of a Full Table Scan when searching for non-stop and the first leg of 1-stop and 2-stop flights.

2. A non-unique index on destinationAirport called idx_destairport. This allowed for a *Unique Index Scan* instead of a Full Table Scan when searching for the middle leg pf 2-stop flights, and the last leg of 1-stop and 2-stop flights.

This, along with query optimization by pushing projections and selections down, the cost was decreased from 71079 to 542.

Technical Challenges

Initially, we encountered a challenge with the complexity of our data sources. While integrating Kaggle into our production database (MySQL) was a straightforward process, we had to extract flights data from unstructured web sources, which required scraping. Furthermore, we experienced difficulty combining various datasets, as some instances in Hotels did not possess proper identification or location data, apart from coordinates. Similarly, the Airbnb data solely provided postal codes instead of city information, prompting us to use an API that converted postal codes into neighborhood names, ultimately providing us with the cities. The Flights dataset comprised numerous records, with over 20,000 flights distributed over different dates, necessitating optimization of aggregation queries, which can be intricate. Selecting the appropriate join orders was critical to ensure our web application remained responsive.

Appendices

Appendix A:

1) Data cleaning and preprocessing

a. Python
b. Spark

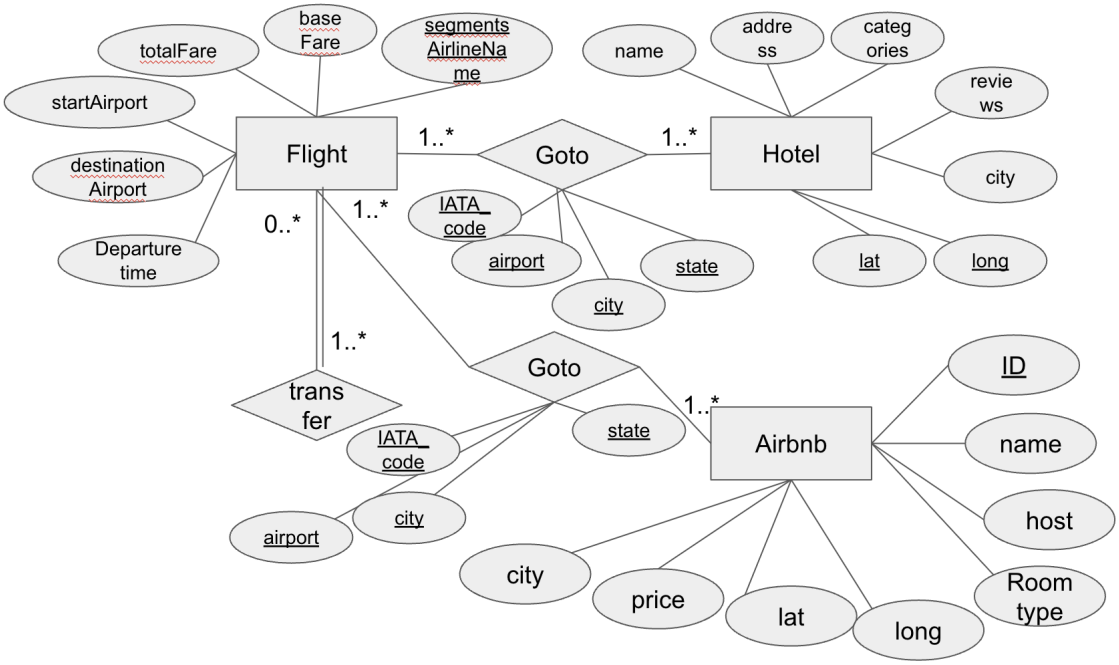
2) Database setup for server

c. SQL - MySQL
d. AWS

3) Front end development

e. React.js
f. Node.js
g. Javascript

Appendix B:



Appendix C:

1. **One way trip and hotel**

   Path: `'/oneway_hotel/'`
   Param:

```
const start_airport = ('startap' in req.query) ? req.query.startap : '';
  const dest_airport = ('destap' in req.query) ? req.query.destap:'';
  const flight_date = ('date' in req.query) ? req.query.date:today;
  const stay_duration = ('duration' in req.query) ? req.query.duration:1
```

   Response Param:

```
hotel_name,hotel_address, hotel_price, entire flight information and
total_price
```

2. **Find a random trip with a random destination given the start airport**

   Path: `'/randomdest/:startAirport'`
   Params:

   startAirport
   Date (today's date)

   Response Param:

   Outputs entire flight information

3. **Ranking of hotels based on rating**

   Path: `/hotelrank/`
   Params: /
   Response Param:

   Outputs entire hotels information

4. **One way trip with Airbnb**

   Path: `/oneway_airbnb/`
   Params:

```
req.query.page_size >=0 ? req.query.page_size : 10;
  const start_airport = ('startap' in req.query) ? req.query.startap : '';
  const dest_airport = ('destap' in req.query) ? req.query.destap:'';
  const flight_date = ('date' in req.query) ? req.query.date:today;
  const stay_duration = ('duration' in req.query) ? req.query.duration:1;
```

   Response Param:

```
airbnb_name,airbnb_id, airbnb_price, entire flight information and
total_price
```

5. **Round trip with Airbnb**

Route: `/round_airbnb/`

Params:

```
 const start_airport = ('startap' in req.query) ? req.query.startap : '';
  const dest_airport = ('destap' in req.query) ? req.query.destap:'';
  const departure_date = ('dep_date' in req.query) ? req.query.dep_date:today;
  const return_date = ('ret_date' in req.query) ? req.query.ret_date:today;
  const stay_duration = ('duration' in req.query) ? req.query.duration:1;
```

Response Param:

```
airbnb_name, host_name, airbnb_price, departure_date, dep_startingAirport,
  dep_airline,dep_departureTime, return_date, ret_startingAirport,
  Ret_airline, ret_departureTime, total_price
```

6. **Detailed flight and cost information using airbnb**

Path: `/detailed_airbnb_trip/`

Param:

```
 req.query.page_size >=0 ? req.query.page_size : 10;
  const start_airport = req.query.start_airport ? req.query.start_airport :
'';
  const dest_airport = req.query.dest_airport ? req.query.start_airport:'';
  const start_city = req.query.start_city ? req.query.start_city : '';
  const dest_city = req.query.dest_city ? req.query.dest_city : '';
  const cost = req.query.cost ? req.query.cost: 1000000;
  const departure_date = req.query.departure_date ? req.query.departure_date
:today;
  const return_date = req.query.return_date ? req.query.return_date :today;
  const stay_duration = req.query.duration ? req.query.duration:1;
```

Response Param:

```
 outboundStartingAirport,outboundStartingCity, outboundDestinationAirport,
outboundDestinationCity, outboundFlightPrice, returnStartingAirport,
returnStartingCity, returnDestinationAirport, returnDestinationCity,
returnFlightPrice, flightPrice,  airbnbPrice, a.airbnb_name,
airbnbRating, totalAirbnbPrice, totalTripPrice
```

7. **Cheapest 'X' day trip with hotel information**

Path: `/cheap_X_trip_hotel/`
Param:

```
  const departure_airport = req.query.departure_airport ?
req.query.departure_airport : '';
  const destination_airport = req.query.destination_airport ?
req.query.destination_airport : '';
  const num_days = req.query.num_days ? req.query.num_days : 1;
```

Response Param:
`hotel_name, hotel_address, hotel_price, departure_flight_price, return_flight_price, total_price`

8. **Most expensive trip with Airbnb**

Path: `/most_expensive_trip_airbnb/`
Param:

```
  const departure_date = req.query.departure_date ? req.query.departure_date :today;
  const return_date = req.query.return_date ? req.query.return_date :today;
  const stay_duration = req.query.duration ? req.query.duration:1;
```

Response Param:
`airbnb_name, startingAirport,destinationAirport, flight_fare, airbnb_fare, total_cost`

Appendix D:

1) **Round trip with Airbnb with total cost trip based on departure date and return date as well as start and end destinations**

```
WITH start_ap AS(
    SELECT IATA_CODE as START_IATA_CODE
    FROM airports
    WHERE AIRPORT = 'CLT'
    ),
    end_ap AS(
        SELECT IATA_CODE as END_IATA_CODE
        FROM airports
        WHERE AIRPORT = 'JFK'
    ),
    Departure_flights AS(
```

```
      SELECT *
      FROM Flights JOIN start_ap S ON S.START_IATA_CODE =
Flights.startingAirport
      JOIN end_ap E ON E.END_IATA_CODE = Flights.destinationAirport
      AND flightDate = '2022-07-28 00:00:00'
    ),
    Return_flights AS(
      SELECT *
      FROM Flights JOIN end_ap E ON E.END_IATA_CODE = Flights.startingAirport
      JOIN start_ap S ON S.START_IATA_CODE = Flights.destinationAirport
      AND flightDate = '2022-11-12 00:00:00'
    ),
    airbnb_options AS(
      SELECT airbnb_name , host_name, price AS airbnb_price, airport_code,
city
      FROM Airbnb
    )
    SELECT
      a.hotel_name, a.host_name, a.airbnb_price,
      df.flightDate AS departure_date, df.startingAirport AS
dep_startingAirport, df.segmentsAirlineName AS dep_airline, df.departureTime
AS segmentsdepartureTimeRaw, rf.flightDate AS return_date, rf.startingAirport
AS ret_startingAirport, rf.segmentsAirlineName AS ret_airline,
rf.segmentsdepartureTimeRaw AS ret_departureTime,
      (a.airbnb_price * 5 + df.totalFare + rf.totalFare) AS total_price
    FROM airbnb_options a
    JOIN Departure_flights df ON a.IATA_CODE = df.destinationAirport AND
a.city = 'JFK'
    JOIN Return_flights rf ON a.IATA_CODE = rf.startingAirport AND a.city =
'CLT'
    ORDER BY total_price ASC;
```

2) Round trip with Hotel with total cost trip based on departure date and return date as well as start and end destinations

```
WITH start_ap AS(
    SELECT IATA_CODE as START_IATA_CODE
    FROM airports
    WHERE AIRPORT = 'CLT'
    ),
    end_ap AS(
        SELECT IATA_CODE as END_IATA_CODE
        FROM airports
        WHERE AIRPORT = 'JFK'
    ),
    Departure_flights AS(
```

```
      SELECT *
      FROM Flights JOIN start_ap S ON S.START_IATA_CODE =
Flights.startingAirport
      JOIN end_ap E ON E.END_IATA_CODE = Flights.destinationAirport
      AND flightDate = '2022-11-12 00:00:00'
    ),
    Return_flights AS(
      SELECT *
      FROM Flights JOIN end_ap E ON E.END_IATA_CODE = Flights.startingAirport
      JOIN start_ap S ON S.START_IATA_CODE = Flights.destinationAirport
      AND flightDate = '2022-07-28 00:00:00'
    ),
    hotel_options AS(
      SELECT name AS hotel_name, address AS hotel_address, price AS
hotel_price, IATA_CODE, city
      FROM hotels
    )
    SELECT
      ho.hotel_name, ho.hotel_address, ho.hotel_price,
      df.flightDate AS departure_date, df.startingAirport AS
dep_startingAirport,
      df.segmentsAirlineName AS dep_airline, df.segmentsdepartureTimeRaw AS
dep_departureTime,
      rf.flightDate AS return_date, rf.startingAirport AS ret_startingAirport,
      rf.segmentsAirlineName AS ret_airline, rf.segmentsdepartureTimeRaw AS
ret_departureTime,
      (ho.hotel_price * 5 + df.totalFare + rf.totalFare) AS total_price
    FROM hotel_options ho
    JOIN Departure_flights df ON ho.IATA_CODE = df.destinationAirport AND
ho.city = 'CLT'
    JOIN Return_flights rf ON ho.IATA_CODE = rf.startingAirport AND ho.city =
'JFK'
    ORDER BY total_price ASC;
```

3) Detailed flight and cost information of most expensive trip using airbnb based on departure date, duration and start and end destinations

```
  WITH one_way_city_info AS (
    SELECT *
    FROM airports
    WHERE CITY = 'Charlotte'),

    two_way_city_info AS (
    SELECT *
    FROM airports
```

```
    WHERE CITY = 'New York'),

    Departure_flight AS(
    SELECT *, DATE_ADD(flightDate, INTERVAL 5 DAY) AS return_date
        FROM Flights JOIN one_way_city_info ow ON Flights.startingAirport =
ow.IATA_CODE
         ),
    Return_flight AS(
        SELECT *
        FROM Flights JOIN two_way_city_info tw ON Flights.startingAirport =
tw.IATA_CODE
        )
    SELECT H.name AS hotel_name, H.city , H.price AS hotel_price,
      DF.totalFare AS departure_flight_price, RF.totalFare AS
return_flight_price,  ROUND(H.price* 5 + DF.totalFare + RF.totalFare,1) AS
total_price
      FROM hotels H JOIN Departure_flight DF ON H.IATA_CODE=DF.startingAirport
      JOIN Return_flight RF ON DF.destinationAirport = RF.startingAirport
      AND DATE_ADD(DF.flightDate, INTERVAL 5 DAY)= RF.flightDate
      ORDER BY total_price ASC
      LIMIT 1
```

4) **Round trip one-stop flights within particular price range for a particular date and start and end destination, and number of passengers**

```
  WITH NonStopFlights AS (
          SELECT legId AS TripId,
          segmentsAirlineName AS Airline,
          startingAirport AS DepartFrom,
          destinationAirport AS ArriveAt,
          NULL AS FirstVia,
          NULL AS SecondVia,
          segmentsDepartureTimeRaw AS DepartureTime,
          NULL AS FirstConnectingArrivalTime,
          NULL AS FirstLayover,
          NULL AS FirstConnectingDepartureTime,
          NULL AS SecondConnectingArrivalTime,
          NULL AS SecondLayover,
          NULL AS SecondConnectingDepartureTime,
          segmentsArrivalTimeRaw AS ArrivalTime,
          TIMEDIFF(segmentsArrivalTimeRaw, segmentsDepartureTimeRaw) AS
TotalTime,
          totalFare AS Fare
          FROM Flights
          WHERE startingAirport = 'CLT'
          AND destinationAirport = 'JFK'
```

```
          AND DATE(flightDate) = '2022-11-12 00:00:00'
          AND seatsRemaining >= 5
          AND totalFare BETWEEN 10 AND 1000
        ),
        OneStopFlights AS (
          SELECT CONCAT_WS(',', a.legId, b.legId) AS TripId,
          a.segmentsAirlineName AS Airline,
          a.startingAirport AS DepartFrom,
          b.destinationAirport AS ArriveAt,
          a.destinationAirport AS FirstVia,
          NULL AS SecondVia,
          a.segmentsDepartureTimeRaw AS DepartureTime,
          a.segmentsArrivalTimeRaw AS FirstConnectingArrivalTime,
          TIMEDIFF(b.segmentsDepartureTimeRaw, a.segmentsArrivalTimeRaw) AS
FirstLayover,
          b.segmentsDepartureTimeRaw AS FirstConnectingDepartureTime,
          NULL AS SecondConnectingArrivalTime,
          NULL AS SecondLayover,
          NULL AS SecondConnectingDepartureTime,
          b.segmentsArrivalTimeRaw AS ArrivalTime,
          TIMEDIFF(b.segmentsArrivalTimeRaw, a.segmentsDepartureTimeRaw) AS
TotalTime,
          ROUND((a.totalFare + b.totalFare), 2) AS Fare
          FROM Flights a JOIN Flights b
          ON a.destinationAirport = b.startingAirport
          AND a.segmentsAirlineName = b.segmentsAirlineName
          AND TIMESTAMPDIFF(MINUTE, a.segmentsArrivalTimeRaw,
b.segmentsDepartureTimeRaw) BETWEEN 45 AND 360
          WHERE a.startingAirport = 'CLT'
          AND b.destinationAirport = 'JFK'
          AND DATE(a.flightDate) = '2022-11-12 00:00:00'
          AND a.seatsRemaining >= 1
          AND b.seatsRemaining >= 1
          AND (a.totalFare + b.totalFare) BETWEEN 10 AND 1000
        )
        (SELECT * FROM NonStopFlights)
        UNION
        (SELECT * FROM OneStopFlights)
        ORDER BY Fare, TotalTime;
```

5) **Round trip non-stop, one-stop and two-stop flights within particular price range for a particular date and start and end destination and number of passengers**

```
      WITH NonStopFlights AS (
   SELECT legId AS TripId,
   segmentsAirlineName AS Airline,
   startingAirport AS DepartFrom,
```

```
          destinationAirport AS ArriveAt,
          NULL AS FirstVia,
          NULL AS SecondVia,
          segmentsDepartureTimeRaw AS DepartureTime,
          NULL AS FirstConnectingArrivalTime,
          NULL AS FirstLayover,
          NULL AS FirstConnectingDepartureTime,
          NULL AS SecondConnectingArrivalTime,
          NULL AS SecondLayover,
          NULL AS SecondConnectingDepartureTime,
          segmentsArrivalTimeRaw AS ArrivalTime,
          TIMEDIFF(segmentsArrivalTimeRaw, segmentsDepartureTimeRaw) AS TotalTime,
          totalFare AS Fare
          FROM Flights
          WHERE startingAirport = 'CLT'
          AND destinationAirport = 'JFK'
          AND DATE(flightDate) = '2022-11-12 00:00:00'
          AND seatsRemaining >= 1
          AND totalFare BETWEEN 10 AND 1000
        ),
        OneStopFlights AS (
          SELECT CONCAT_WS(',', a.legId, b.legId) AS TripId,
          a.segmentsAirlineName AS Airline,
          a.startingAirport AS DepartFrom,
          b.destinationAirport AS ArriveAt,
          a.destinationAirport AS FirstVia,
          NULL AS SecondVia,
          a.segmentsDepartureTimeRaw AS DepartureTime,
          a.segmentsArrivalTimeRaw AS FirstConnectingArrivalTime,
          TIMEDIFF(b.segmentsDepartureTimeRaw, a.segmentsArrivalTimeRaw) AS
FirstLayover,
          b.segmentsDepartureTimeRaw AS FirstConnectingDepartureTime,
          NULL AS SecondConnectingArrivalTime,
          NULL AS SecondLayover,
          NULL AS SecondConnectingDepartureTime,
          b.segmentsArrivalTimeRaw AS ArrivalTime,
          TIMEDIFF(b.segmentsArrivalTimeRaw, a.segmentsDepartureTimeRaw) AS
TotalTime,
          ROUND((a.totalFare + b.totalFare), 2) AS Fare
          FROM Flights a JOIN Flights b
          ON a.destinationAirport = b.startingAirport
          AND a.segmentsAirlineName = b.segmentsAirlineName
          AND TIMESTAMPDIFF(MINUTE, a.segmentsArrivalTimeRaw,
b.segmentsDepartureTimeRaw) BETWEEN 45 AND 360
          WHERE a.startingAirport = 'CLT'
          AND b.destinationAirport = 'JFK'
          AND DATE(a.flightDate) = '2022-11-12 00:00:00'
          AND a.seatsRemaining >= 1
```

```
          AND b.seatsRemaining >= 1
          AND (a.totalFare + b.totalFare) BETWEEN 10 AND 1000
        ),
        TwoStopFlights AS (
          SELECT CONCAT_WS(',', a.legId, b.legId, c.legId) AS TripId,
          a.segmentsAirlineName AS Airline,
          a.startingAirport AS DepartFrom,
          c.destinationAirport AS ArriveAt,
          a.destinationAirport AS FirstVia,
          b.destinationAirport AS SecondVia,
          a.segmentsDepartureTimeRaw AS DepartureTime,
          a.segmentsArrivalTimeRaw AS FirstConnectingArrivalTime,
          TIMEDIFF(b.segmentsDepartureTimeRaw, a.segmentsArrivalTimeRaw) AS
FirstLayover,
          b.segmentsDepartureTimeRaw AS FirstConnectingDepartureTime,
          b.segmentsArrivalTimeRaw AS SecondConnectingArrivalTime,
          TIMEDIFF(c.segmentsDepartureTimeRaw, b.segmentsArrivalTimeRaw) AS
SecondLayover,
          c.segmentsDepartureTimeRaw AS SecondConnectingDepartureTime,
          b.segmentsArrivalTimeRaw AS ArrivalTime,
          TIMEDIFF(b.segmentsArrivalTimeRaw, a.segmentsDepartureTimeRaw) AS
TotalTime,
          ROUND((a.totalFare + b.totalFare), 2) AS Fare
          FROM Flights a JOIN Flights b
          ON a.destinationAirport = b.startingAirport
          AND a.segmentsAirlineName = b.segmentsAirlineName
          AND TIMESTAMPDIFF(MINUTE, a.segmentsArrivalTimeRaw,
b.segmentsDepartureTimeRaw) BETWEEN 45 AND 180
          JOIN Flights c
          ON b.destinationAirport = c.startingAirport
          AND b.segmentsAirlineName = c.segmentsAirlineName
          AND TIMESTAMPDIFF(MINUTE, b.segmentsArrivalTimeRaw,
c.segmentsDepartureTimeRaw) BETWEEN 45 AND 180
          WHERE a.startingAirport = 'CLT'
          AND a.destinationAirport <> 'JFK'
          AND b.destinationAirport <> 'CLT'
          AND c.destinationAirport = 'JFK'
          AND DATE(a.flightDate) = '2022-11-12 00:00:00'
          AND a.seatsRemaining >= 1
          AND b.seatsRemaining >=2
          AND c.seatsRemaining >= 1
          AND (a.totalFare + b.totalFare + c.totalFare) BETWEEN 10 AND 5000
        )
        (SELECT * FROM NonStopFlights)
        UNION
        (SELECT * FROM OneStopFlights)
        UNION
        (SELECT * FROM TwoStopFlights)
```

```
        ORDER BY Fare, TotalTime;
```

Appendix E:

Before optimization:

```
SELECT
      F1.startingAirport AS outboundStartingAirport,
      A1.CITY AS outboundStartingCity,
      F1.destinationAirport AS outboundDestinationAirport,
      A2.CITY AS outboundDestinationCity,
      F1.totalFare AS outboundFlightPrice,
      F2.startingAirport AS returnStartingAirport,
      A2.CITY AS returnStartingCity,
      F2.destinationAirport AS returnDestinationAirport,
      A1.CITY AS returnDestinationCity,
      F2.totalFare AS returnFlightPrice,
      F1.totalFare AS flightPrice,
      a.price AS airbnbPrice,
      a.airbnb_name,
      a.review_per_month AS airbnbRating,
      (a.price * (DATEDIFF('2022-11-12 00:00:00', '2022-07-28 00:00:00') + 1)) AS
totalAirbnbPrice,
      (F1.totalFare + F2.totalFare + (a.price * DATEDIFF('2022-11-12 00:00:00',
'2022-07-28 00:00:00'))) AS totalTripPrice
    FROM Flights F1
    JOIN airports A1 ON F1.startingAirport = A1.IATA_CODE
    JOIN airports A2 ON F1.destinationAirport = A2.IATA_CODE
    JOIN Flights F2 ON F2.startingAirport = A2.IATA_CODE AND F2.destinationAirport =
A1.IATA_CODE
    JOIN Airbnb a ON a.CITY = A2.CITY
    WHERE
      A1.IATA_CODE = 'CLT' AND
      A2.IATA_CODE= 'JFK' AND
      F1.flightDate = '2022-11-12 00:00:00' AND F2.flightDate = '2022-07-28 00:00:00'
AND (F1.totalFare + F2.totalFare + (a.price * DATEDIFF('2022-11-12 00:00:00',
'2022-07-28 00:00:00'))) <= 1000
    ORDER BY totalTripPrice ASC;
```

After optimization:

```
       SELECT
         F1.startingAirport AS outboundStartingAirport,
         A1.CITY AS outboundStartingCity,
         F1.destinationAirport AS outboundDestinationAirport,
         A2.CITY AS outboundDestinationCity,
```

```
  F1.totalFare AS outboundFlightPrice,
  F2.startingAirport AS returnStartingAirport,
  A2.CITY AS returnStartingCity,
  F2.destinationAirport AS returnDestinationAirport,
  A1.CITY AS returnDestinationCity,
  F2.totalFare AS returnFlightPrice,
  F1.totalFare AS flightPrice,
  a.price AS airbnbPrice,
  a.airbnb_name,
  a.review_per_month AS airbnbRating,
  (a.price * (DATEDIFF('2022-11-12 00:00:00', '2022-07-28 00:00:00') + 1)) AS
totalAirbnbPrice,
  (F1.totalFare + F2.totalFare + (a.price * DATEDIFF('2022-11-12 00:00:00',
'2022-07-28 00:00:00'))) AS totalTripPrice
FROM (
  SELECT *
  FROM Flights
  WHERE startingAirport = 'CLT' AND flightDate = '2022-11-12 00:00:00'
) F1
JOIN airports A1 ON F1.startingAirport = A1.IATA_CODE
JOIN airports A2 ON F1.destinationAirport = A2.IATA_CODE
JOIN (
  SELECT *
  FROM Flights
  WHERE startingAirport = 'JFK' AND destinationAirport = 'CLT' AND flightDate
= '2022-07-28 00:00:00'
) F2 ON F2.startingAirport = A2.IATA_CODE
JOIN Airbnb a ON a.CITY = A2.CITY
WHERE
 (F1.totalFare + F2.totalFare + (a.price * DATEDIFF('2022-11-12 00:00:00',
'2022-07-28 00:00:00'))) <= 1000
ORDER BY totalTripPrice ASC;
```