Flight Price Analysis & API-Integration

James Garcia
Lovecy Thomas
Seyhr Waqas

Overview

- Data-Driven Travel Decisions: Flight prices fluctuate frequently. Analyzing this data can provide insights into price trends and the best times to book.
- Efficient Flight Data Management: The ETL process will clean, structure, and store data in a normalized format for better accessibility.
- Real-World Applications: Airlines, travel agencies, and consumers can use this system to retrieve flight details efficiently via a user-friendly API.

Agenda

- We utilized GoFlightLabs APIs to fetch:
  - Airport Data API – Retrieves airport details
  - Airline Data API – Provides airline-related information
  - Flight Price Data API – Fetches real-time flight pricing data
- ETL (Extract, Transform, Load)
- ERD
- Postgres
- Flask API Development & User Interaction
- AWS RDS Cloud

API Workflow

- Used three APIs provided by GoFlightLabs to perform flight price analysis.

Focused on the flight data from March 2025 to February 2026 for the major airports in Canada and the US. The airports included in our analysis are:

1. Toronto Pearson International (YYZ)
2. Ottawa International (YOW)
3. Toronto Island (YTZ)
4. Montreal Pierre Elliott Trudeau (YUL)
5. Dallas Fort Worth International (DFW)
6. Denver International (DEN)
7. Atlanta Hartsfield-Jackson (ATL)
8. Chicago O'Hare International (ORD)
9. Los Angeles International (LAX)

Airport Data API

Step 1: Retrieved airport data for Canada and the U.S. and saved the results as a CSV file.

Retrieve Airports API

This API endpoint allows you to retrieve a list of airports located at a specified location by using a query. The information provided by this endpoint will be useful as an input parameter for the Flight Prices API.

Example API Request:

```
https://www.goflightlabs.com/retrieveAirport?access_key=YOUR_ACCESS_KEY&query=New
```

Run endpoint | Run In Postman

HTTP GET Request Parameters:

Object	Description	
access_key	required	Your API access key, which can be found in your account dashboard.
query	required	Location name of the situated Airport.

```python
# GoFlightLabs API endpoint for fetching airport data
base_url = f"https://www.goflightlabs.com/airports-by-filter"
country = ['US', 'CA']

# Make an API request to the GoFlightLabs API
url = f"{base_url}?access_key={API_KEY}&country_code={country}"

# Make API request
response = requests.get(url)

# Check if the API request was successful (HTTP status code 200 means success)
if response.status_code == 200:
    try:

        # Parse the API response as JSON
        data = response.json()

        # Initialize an empty list to store airport details
        airport_list = []

        # Check if the response contains data and has at least one airport
        if "data" in data and len(data["data"]) > 0:

            # Iterate through each country in the response data
            for each_country in country:

            # Iterate through each airport in the response data
                for airport in data["data"]:

                    # Extract relevant airport details and add them to the list
                    airport_list.append({
                    "Name": airport.get("name", "N/A"),
                        "IATA Code": airport.get("iata_code", "N/A"),
                        "ICAO Code": airport.get("icao_code", "N/A"),
                        "Country": airport.get("country_code", "N/A"),
                        "City": airport.get("city", "N/A"),
                        "City Code": airport.get("city_code", "N/A"),
                        "Latitude": airport.get("lat", "N/A"),
                        "Longitude": airport.get("lng", "N/A"),
                        "Timezone": airport.get("timezone", "N/A"),
                        "Departures": airport.get("departures", "N/A"),
                    })
```

Airline Data API

Step 2: Fetched airline data globally and stored it as a CSV file for further analysis.

Retrieve Airlines API

To obtain information about airlines, you can use this endpoint, filtering by IataAirline and Iso2Country.

API request example for information about a specific airline, you can search based on IATA airline code

```
https://www.goflightlabs.com/airlines?access_key=YOUR_ACCESS_KEY&codeIataAirline=AA
```

Run endpoint | Run in Postman

HTTP GET Request Parameters:

Object	Description	
access_key	required	Your API access key, which can be found in your account dashboard.
codeIataAirline	optional	Use this parameter to get information about a specific airline, you can search based on IATA airline code.
codeIso2Country	optional	Use this parameter to get information the airlines based on the country codes.

```python
# GoFlightLabs API endpoint for fetching airline data
base_url = f"https://www.goflightlabs.com/flights-by-airline"

# Make an API request to the GoFlightLabs API
url = f"{base_url}?access_key={API_KEY}"

# Make API request
response = requests.get(url)

# Parse the API response as JSON
data = response.json()

# Create an empty list to extract airlines data
airlines_list = []

# Check if the response contains data and has at least one airline
if "data" in data and len(data["data"]) > 0:
        # Iterate through each airline in the response data
        for airline in data["data"]:
            # Extract relevant airline details and add them to the list
            airlines_list.append({
                "Name": airline.get("name", "N/A"),
                "country_code": airline.get("country_code", "N/A"),
                "iata_code": airline.get("iata_code", "N/A"),
                "iata_prefix": airline.get("iata_prefix", "N/A"),
                "iata_accounting": airline.get("iata_accounting", "N/A"),
                "callsign": airline.get("callsign", "N/A"),
                "is international": airline.get("is international", "N/A"),
                "iosa_registered": airline.get("iosa_registered", "N/A"),
                "iosa_expiry": airline.get("iosa_expiry", "N/A"),
                "is_passenger": airline.get("is_passenger", "N/A"),
                "is_cargo":airline.get("is_cargo", "N/A"),
                "is_scheduled": airline.get("is_scheduled", "N/A"),
                "total_aircrafts": airline.get("total_aircrafts", "N/A"),
                "average_fleet_age": airline.get("average_fleet_age", "N/A"),
                "accidents_last_5y": airline.get("accidents_last_5y", "N/A"),
                "crashes_last_5y": airline.get("crashes_last_5y", "N/A"),
                "website": airport.get("website", "N/A"),
                "twitter": airport.get("twitter", "N/A"),
                "facebook": airline.get("facebook", "N/A"),
                "instagram": airline.get("instagram", "N/A"),
                "linkedin": airline.get("linkedin", "N/A"),
                })
```

Reactivating terminals

Flight Price Data API

Step 3 : Collected flight price data for flights from Canadian airports to Texas airports.

The following Endpoint will provide you with the price of a flight based on the input parameters entered. In addition to the price, it will provide you with detailed information about the origin, destinat

NOTE: Some of the input parameters need to be obtained beforehand from the Retrieve Airports API and/or Retrieve Countries API.

If you receive incomplete results, please note that sometimes heavy queries may take longer to process. You can use the Retrieve Flights Incomplete endpoint below to fetch the complete information. Please be awa

Example API Request:

```
https://www.goflightlabs.com/retrieveFlights?access_key=YOUR_ACCESS_KEY&originSkyId=LOND&destinationSkyId=NYCA&originEntityId=27544008&destinationEntityId=27537542&date=2025-03-04
```

I'm not a robot reCAPTCHA Privacy - Terms

Test endpoint | Test In Postman

HTTP GET Request Parameters:

Parameter		Description
access_key	required	Your API access key, which can be found in your account dashboard.
originSkyId	required	Extract the originSkyId code from the retrieveAirport endpoint.
destinationSkyId	required	Retrieve the destinationSkyId code from the retrieveAirport endpoint.
originEntityId	required	The originEntityId code is attainable from the retrieveAirport endpoint.
destinationEntityId	required	The destinationEntityId code is retrievable from the retrieveAirport endpoint.
date	required	Date of departure or travel. Format: YYYY-MM-DD.

```python
# Define the GoFlightLabs API endpoint for retrieving flight data
BASE_URL = "https://www.goflightlabs.com/retrieveFlights"

# Define different parameters for flight searches

# List of available cabin classes
cabin_classes = ["economy", "premium_economy", "business", "first"]

# Sorting options for flights (e.g., best, highest price, fastest flight)
sort_options = ["best", "price_high", "fastest"]

# List of dates for which flight data should be retrieved
date_options =  ["2025-11-01","2025-11-02","2025-11-03","2025-11-04","2025-11-05","2025-11-06","2025-11-07","2025-11-08","2
# Initialize an empty list to store structured flight records
flight_records = []
# Loop through each date in the date options list
for date in date_options:
    # Loop through each cabin class option
    for cabin in cabin_classes:
        # Loop through each sorting option
        for sort in sort_options:
            params = {
                "access_key": API_KEY,
                "originSkyId": "YTOA",
                "destinationSkyId": "DFWA",
                "originEntityId": "27536640",
                "destinationEntityId": "27536457",
                "date": date,
                "currency": "CAD",
                "cabinClass": cabin,
                "sortBy": sort
            }

            # Make an API request with the specified parameters
            response = requests.get(BASE_URL, params=params)

            # Check if the request was successful
            if response.status_code == 200:

                # Parse the response JSON data
                data = response.json()
```

Data Import and Cleaning Strategies
ETL for Airport dataset

Filter the merged airport data to get the required Canadian and US airportS

```
# Filter the merged Canada-US airport dataset to include only the busiest airports based on their IATA codes.
can_us_busiest_airport = merged_canada_us_airport_data.loc[merged_canada_us_airport_data['IATA Code'].isin(["YUL", "YYZ", "YTZ", "YOW", "YKF", "ATL", "DFW", "DEN", "ORD","LAX"])]

#Reset the index of the filtered DataFrame to ensure it starts from 0 and remove the old index
airports_df = can_us_busiest_airport.reset_index(drop=True)

# Display the filtered DataFrame
airports_df.head()
```

Python

	Name	IATA Code	ICAO Code	Country	City	City Code	# Latitude	# Longitude
0	Billy Bishop Toronto City Airport	YTZ	CYTZ	CA	Toronto	YTO	43.62974	-79.39
1	Kitchener/Waterloo Airport	YKF	CYKF	CA	Breslau	YKF	43.45747	-80.38
2	Montreal-Pierre Elliott Trudeau Inter	YUL	CYUL	CA	Montreal	YMQ	45.46106	-73.75
3	Ottawa Macdonald-Cartier Internatic	YOW	CYOW	CA	Ottawa	YOW	45.3225	-75.66
4	Toronto Pearson International Airpor	YYZ	CYYZ	CA	Toronto	YTO	43.68066	-79.61

5 rows x 10 cols 10 per page Page 1 of 1

```
# Remove Leading & Trailing Spaces
airport_df = airport_df.astype(str).apply(lambda x: x.str.strip() if x.dtype == 'object' else x)
# Convert 'departures' to numeric, forcing errors to NaN (if there are any non-numeric values)
airport_df['departures'] = pd.to_numeric(airport_df['departures'], errors='coerce')
# Convert to int64 after numeric conversion
airport_df['departures'] = airport_df['departures'].astype('int64')
airport_df['latitude'] = airport_df['departures'].astype(float)
airport_df['longitude'] = airport_df['longitude'].astype(float)
```

Airport Dataset

Before Cleaning

Name	IATA Code	ICAO Code	Country	City	City Code	Latitude	Longitude	Timezone	Departures
Abbotsfor	YXX	CYXX	CA	Abbotsfor	YVR	49.02529	-122.377	America/V	941
Aklavik/Fr	LAK	CYKD	CA	Aklavik	LAK	68.22333	-135.006	America/Y	348
Akulivik Ai	AKV	CYKO	CA	Akulivik	AKV	60.81861	-78.1486	America/T	270
Alberni Va	YPB		CA	Port Alber	YPB	49.31933	-124.93	America/Vancouver	
Alert Bay A	YAL	CYAL	CA	Alert Bay		50.5822	-126.916	America/Vancouver	
Alert Airpc	YLT	CYLT	CA	Alert		82.51778	-62.2806	America/Iqaluit	
Alice Arm/	ZAA		CA			55.47185	-129.495	America/Vancouver	
Allan J. M	YPS	CYPD	CA	Port Hawkesbury		45.65667	-61.3681	America/Halifax	
Alma Airp	YTF	CYTF	CA	Alma	YTF	48.5089	71.6419	America/Toronto	

```
airport_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   name        10 non-null     object
 1   iata code   10 non-null     object
 2   country     10 non-null     object
 3   city        10 non-null     object
 4   latitude    10 non-null     float64
 5   longitude   10 non-null     float64
 6   timezone    10 non-null     object
 7   departures  10 non-null     float64
dtypes: float64(3), object(5)
memory usage: 768.0+ bytes
```

After Cleaning

name	iata code	country	city	latitude	longitude	timezone	departures
Billy Bisho	YTZ	CA	Toronto	752	-79.3983	EST	752
Kitchener/	YKF	CA	Breslau	454	-80.3859	EST	454
Montreal-F	YUL	CA	Montreal	35190	-73.7502	EST	35190
Ottawa Ma	YOW	CA	Ottawa	10293	-75.6692	EST	10293
Toronto Pe	YYZ	CA	Toronto	72355	-79.6129	EST	72355
Chicago O'	ORD	US	Chicago	177167	-87.9045	CST	177167
Dallas/For	DFW	US	Dallas-Ft V	172447	-97.0372	CST	172447
Denver Inte	DEN	US	Denver	100565	-104.673	MST	100565
Hartsfield-	ATL	US	Atlanta	168164	-84.4227	EST	168164
Los Angele	LAX	US	Los Angele	112571	-118.409	PST	112571

```
airport_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   name        10 non-null     object
 1   iata code   10 non-null     object
 2   country     10 non-null     object
 3   city        10 non-null     object
 4   latitude    10 non-null     float64
 5   longitude   10 non-null     float64
 6   timezone    10 non-null     object
 7   departures  10 non-null     int64
dtypes: float64(2), int64(1), object(5)
memory usage: 768.0+ bytes
```

ETL for Flight Price dataset

```
# Confirming that only those records with required airports were selected
airports = pd.Series(list(set(selected_flight_price_data['origin_airport']) | set(selected_flight_price_data['destination_airport'])))  # Uni
airports
```

```
0      Toronto Pearson International
1                    Toronto Island
2    Montreal Pierre Elliott Trudeau
3               Denver International
4          Los Angeles International
5       Chicago O'Hare International
6         Atlanta Hartsfield-Jackson
7               Ottawa International
8    Dallas Fort Worth International
dtype: object
```

Click to add a breakpoint

```
irlines from marketing airline and operating airline fields of the required data
unique_airlines = pd.Series(list(set(selected_flight_price_data['operating_airline']) | set(selected_flight_price_data['marketing_airline'])))
unique_airlines
```

```
0                      Frontier Airlines
1                Air Canada Express - Jazz
2                      Porter Airlines Inc
3                SkyWest DBA United Express
4                                   Delta
5                          Alaska Airlines
6                         American Airlines
7              Envoy Air As American Eagle
8                          Spirit Airlines
9                                  WestJet
10      Republic Airways AS American Eagle
11     Republic Airways DBA United Express
12                              GOL Linhas
13                              Air Canada
14                               Lufthansa
15            Porter Airlines (Canada) Ltd
16                                  United
dtype: object
```

Transformations in Flight Price Dataset

```
Click to add a breakpoint  remove commas and convert to integer
price_df['one_way_price_candollar'] = price_df['one_way_price_candollar'].replace({'C\$': '', ',': ''}, regex=True).astype('int64')
```
✓ 0.2s

```
# Convert flight_number to integer
price_df[['flight_number','stop_count']] = price_df[['flight_number','stop_count']].astype('int64')
```
✓ 0.0s

```
# Convert departure_date_time column to datetime format
price_df['departure_date_time']=pd.to_datetime(price_df['departure_date_time'], errors='coerce')
```
✓ 0.0s

```
# Convert arrival_date_time column to datetime format
price_df['arrival_date_time']=pd.to_datetime(price_df['arrival_date_time'], errors='coerce')
```
✓ 0.0s

```
# Convert duration_in_hours to float type
price_df['duration_in_hours'] = price_df['duration_in_hours'].astype(float)
```
✓ 0.0s

```
# Convert is_self_transfer to boolean type
price_df['is_self_transfer'] = price_df['is_self_transfer'].astype(bool)
```
✓ 0.0s

Flight Price

Before cleaning

date	itinerary_i	cabin_cla	sort_by	price_raw	price_forn	currency	flight_nun	origin_air	origin_city	origin_cou	destinatio	destinatio	destinatio	departure	arrival_tin	duration_i	stop_cour	marketing	operating	change_al	cancellati	is_self_tra	has_flexib	score
4/1/2025	18467-250	economy	fastest	476.6	C$477		5545	Toronto Pearson Int	Canada		Detroit Wayne Count	United Sta		2025-04-0	2025-04-0	79	1	WestJet	SkyWest D	FALSE	FALSE	FALSE	FALSE	0.999
4/1/2025	18467-250	economy	fastest	476.6	C$477		8400	Detroit Wayne Count	United Sta		Dallas Fort Worth Int	United Sta		2025-04-0	2025-04-0	176	1	WestJet	Delta	FALSE	FALSE	FALSE	FALSE	0.999
4/1/2025	18467-250	economy	fastest	476.6	C$477		6342	Toronto Pearson Int	Canada		Detroit Wayne Count	United Sta		2025-04-0	2025-04-0	92	1	WestJet	Endeavor	FALSE	FALSE	FALSE	FALSE	0.611641
4/1/2025	18467-250	economy	fastest	476.6	C$477		7021	Detroit Wayne Count	United Sta		Dallas Fort Worth Int	United Sta		2025-04-0	2025-04-0	186	1	WestJet	Delta	FALSE	FALSE	FALSE	FALSE	0.611641
4/1/2025	18467-250	economy	fastest	476.6	C$477		6343	Toronto Pearson Int	Canada		Detroit Wayne Count	United Sta		2025-04-0	2025-04-0	76	1	WestJet	Endeavor	FALSE	FALSE	FALSE	FALSE	0.627755
4/1/2025	18467-250	economy	fastest	476.6	C$477		7019	Detroit Wayne Count	United Sta		Dallas Fort Worth Int	United Sta		2025-04-0	2025-04-0	179	1	WestJet	Delta	FALSE	FALSE	FALSE	FALSE	0.627755
4/1/2025	18467-250	economy	fastest	449.3	C$450		6476	Toronto Pearson Int	Canada		Detroit Wayne Count	United Sta		2025-04-0	2025-04-0	79	1	WestJet	SkyWest D	FALSE	FALSE	FALSE	FALSE	0.827052
4/1/2025	18467-250	economy	fastest	449.3	C$450		8330	Detroit Wayne Count	United Sta		Dallas Fort Worth Int	United Sta		2025-04-0	2025-04-0	175	1	WestJet	Delta	FALSE	FALSE	FALSE	FALSE	0.827052
4/1/2025	18467-250	economy	fastest	476.6	C$477		6340	Toronto Pearson Int	Canada		Atlanta Hartsfield-Ja	United Sta		2025-04-0	2025-04-0	152	1	WestJet	Delta	FALSE	FALSE	FALSE	FALSE	0.544887

After cleaning

itinerary_i	cabin_cla	one_way_j	flight_nun	origin_air	destinatio	departure_date_tim	arrival_date_tir	stop_cour	marketing	operating	is_self_tra	duration_in_hours
18467-250	economy	477	6340	YYZ	ATL	4/1/2025 7:45	4/1/2025 10:17	1	Westjet	Delta Air L	TRUE	2.53
18467-250	economy	477	6482	ATL	DFW	4/1/2025 11:20	4/1/2025 12:49	1	Westjet	Delta Air L	TRUE	2.48
18390-250	economy	253	2205	YTZ	YOW	4/1/2025 7:00	4/1/2025 7:59	2	Porter Airl	Porter Airl	TRUE	0.98
18467-250	premium_	581	2662	YYZ	ATL	4/1/2025 6:15	4/1/2025 8:49	1	Delta Air L	Delta Air L	TRUE	2.57
18467-250	premium_	581	2988	YYZ	ATL	4/1/2025 7:45	4/1/2025 10:17	1	Delta Air L	Delta Air L	TRUE	2.53
18467-250	premium_	581	414	ATL	DFW	4/1/2025 11:20	4/1/2025 12:49	1	Delta Air L	Delta Air L	TRUE	2.48
18467-250	premium_	581	2835	YYZ	ATL	4/1/2025 12:20	4/1/2025 14:40	1	Delta Air L	Delta Air L	TRUE	2.33

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22129 entries, 0 to 22128
Data columns (total 25 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   date                  22129 non-null  object
 1   itinerary_id          22129 non-null  object
 2   cabin_class           22129 non-null  object
 3   sort_by               22129 non-null  object
 4   price_raw             22129 non-null  float64
 5   price_formatted       22129 non-null  object
 6   currency              0 non-null      float64
 7   flight_number         22129 non-null  int64
 8   origin_airport        22129 non-null  object
 9   origin_city           0 non-null      float64
 10  origin_country        22129 non-null  object
 11  destination_airport   22129 non-null  object
 12  destination_city      0 non-null      float64
 13  destination_country   22129 non-null  object
 14  departure_time        22129 non-null  object
 15  arrival_time          22129 non-null  object
 16  duration_minutes      22129 non-null  int64
 17  stop_count            22129 non-null  int64
 18  marketing_airline     22129 non-null  object
 19  operating_airline     22129 non-null  object
...
 23  has_flexible_options  22129 non-null  bool
 24  score                 22129 non-null  float64
dtypes: bool(4), float64(5), int64(3), object(13)
```

```
price_df.info()
✓ 0.0s
<class 'pandas.core.frame.DataFrame'>
Index: 11127 entries, 8 to 22128
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   itinerary_id            11127 non-null  object
 1   cabin_class             11127 non-null  object
 2   one_way_price_candollar 11127 non-null  int64
 3   flight_number           11127 non-null  int64
 4   origin_airport          11127 non-null  object
 5   destination_airport     11127 non-null  object
 6   departure_date_time     11127 non-null  datetime64[ns]
 7   arrival_date_time       11127 non-null  datetime64[ns]
 8   stop_count              11127 non-null  int64
 9   marketing_airline       11127 non-null  object
 10  operating_airline       11127 non-null  object
 11  is_self_transfer        11127 non-null  bool
 12  duration_in_hours       11127 non-null  float64
dtypes: bool(1), datetime64[ns](2), float64(1), int64(3), object(6)
memory usage: 1.1+ MB
```

ETL for Airlines Dataset

```
# Get only the required airline names from airline dataset
available_airlines = airlines_data.loc[airlines_data['iata_code'].isin(["OO", "MQ", "F9", "DL", "YX", "LH", "PD", "WS", "AA","UA","QK","AS","AC","NK","G3","P3"])]
airlines_info = available_airlines.reset_index(drop=True)
airlines_info['Name']
```

✓ 0.2s

```
0                    Alaska Airlines
1                   American Airlines
2                         Air Canada
3                   Delta Air Lines
4                          Envoy Air
5                   Frontier Airlines
6     GOL Linhas Aereas Inteligentes
7                      Jazz Aviation
8                   Lufthansa Cargo
9                         Lufthansa
10                Porter Airlines Inc
11                  Porter Airlines
12   Republic Airline United Express
13                 SkyWest Airlines
14                  Spirit Airlines
15                  United Airlines
16                          Westjet
17                   American Eagle
Name: Name, dtype: object
```

Airlines

Before Cleaning

Name	country_c	iata_code	iata_prefi	iata_acco	callsign	is internat	iosa_regis	iosa_expir	is_passen	is_cargo	is_schedu	total_aircr	average_fl	accidents	crashes_l	website	twitter	facebook	instagram	linkedin
10 Tanker	US					N/A														
135 Airwa	US				GENERAL	N/A														
1903 Aviat	SE				HIGHSCOF	N/A			1											
Air 1st Avi	US				ROUGHRI	N/A														
2 Sqn, No	UK				WYTON	N/A														
21 Air	US	2I*		681	CARGOSO	N/A	0			1		1	33	0	0					linkedin.com/company/21air/
213 Flight	RU					N/A														
223rd Flig	RU				CHKALOV	N/A						32	32.1							
224th Flig	RU				CARGOUN	N/A														
247 Aviati	UK				NIGHTING	N/A			1	1										

```
airlines_data.info()
```
✓ 0.6s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6461 entries, 0 to 6460
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                6461 non-null   object
 1   country_code        6397 non-null   object
 2   iata_code           1157 non-null   object
 3   iata_prefix         563 non-null    float64
 4   iata_accounting     629 non-null    float64
 5   callsign            5696 non-null   object
 6   is international    0 non-null      float64
 7   iosa_registered     1352 non-null   float64
 8   iosa_expiry         376 non-null    object
 9   is_passenger        2085 non-null   float64
 10  is_cargo            1017 non-null   float64
 11  is_scheduled        1176 non-null   float64
 12  total_aircrafts     1557 non-null   float64
 13  average_fleet_age   1557 non-null   float64
 14  accidents_last_5y   1289 non-null   float64
 15  crashes_last_5y     1262 non-null   float64
 16  website             0 non-null      float64
 17  twitter             0 non-null      float64
 18  facebook            1267 non-null   object
 19  instagram           999 non-null    object
 20  linkedin            786 non-null    object
dtypes: float64(13), object(8)
memory usage: 1.0+ MB
```

After cleaning

airline_na	country_c	iata	iosa_regis	iosa_expi	is_airline_	total_aircr	average_fl	accidents_last_5y
Alaska Air	US	AS	TRUE	2025-01-1	TRUE	158	8.1	1
American	US	AA	TRUE	2025-07-2	TRUE	684	10.2	26
Air Canad	CA	AC	TRUE	2024-04-1	TRUE	98	11.7	0
Delta Air L	US	DL	TRUE	2024-10-1	TRUE	591	12.5	22
Envoy Air	US	MQ	TRUE	2025-11-1	TRUE	161	9.6	0
Frontier Ai	US	F9	TRUE	2024-01-0	TRUE	83	3	4
GOL Linha	BR	G3	TRUE	2025-11-2	TRUE	103	10.7	0
Jazz Aviati	CA	QK	TRUE	2024-10-2	TRUE	85	13.8	3
Lufthansa	DE	LH	TRUE	2026-05-1	TRUE	131	8.9	1

```
airlines_df.info()
```
✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
Index: 17 entries, 0 to 17
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   airline_name         17 non-null     object
 1   country_code         17 non-null     object
 2   iata                 17 non-null     object
 3   iosa_registered      17 non-null     bool
 4   iosa_expiry          15 non-null     datetime64[ns, UTC]
 5   is_airline_passenger 17 non-null     bool
 6   total_aircrafts      17 non-null     int64
 7   average_fleet_age    17 non-null     float64
 8   accidents_last_5y    17 non-null     int64
dtypes: bool(2), datetime64[ns, UTC](1), float64(1), int64(2), object(3)
memory usage: 1.1+ KB
```

Data Validation

- Python Libaries used:
  - ydata_profiling
  - Pandera

Data Validation by ydata_profiling

Flight Price Profiling Report

Overview (1) Variables (2) Interactions (3) Correlations (4) Missing values (5) Sample (6) Duplicate rows (7)

Overview

Brought to you by YData

Overview | Alerts 11 | Reproduction

Dataset statistics

Number of variables	13
Number of observations	11,127
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	33
Duplicate rows (%)	0.3%
Total size in memory	6.5 MiB
Average record size in memory	612.1 B

Variable types

Text	1
Categorical	6
Numeric	3
DateTime	2
Boolean	1

Data Validation by ydata_profiling cont.

Flight Price Profiling Report

Overview 1 | Variables 2 | Interactions 3 | Correlations 4 | Missing values 5 | Sample 6 | Duplicate rows 7

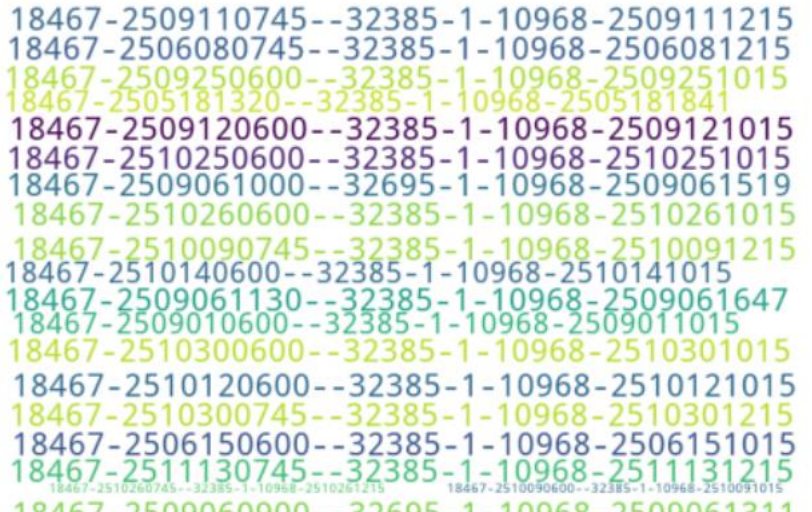
Variables

Select Columns

itinerary_id

Text

Distinct	6327
Distinct (%)	56.9%
Missing	0
Missing (%)	0.0%
Memory size	1.1 MiB



Pandera

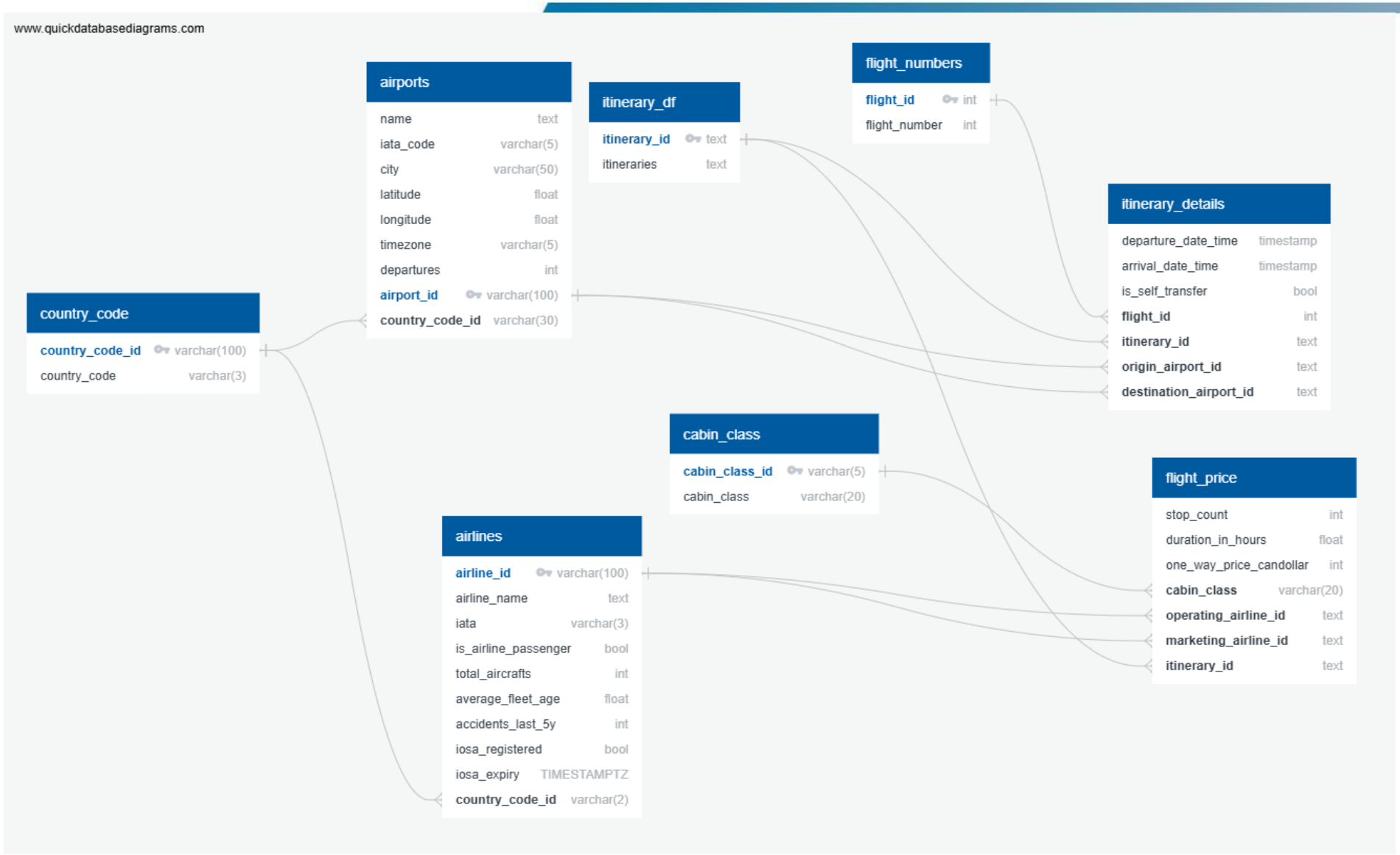
```python
# Dataframe to validate
# we have filtered_airports_df ready
# define schema
schema = pa.DataFrameSchema({
    "name": pa.Column(str),
    "iata code": pa.Column(str,unique=True),
    "country_code_id": pa.Column(str,checks=pa.Check.str_length(max_value=4)),
    "city": pa.Column(str, checks=Check.isin(['Toronto','Breslau','Montreal','Ottawa','Chicago',
                                              'Dallas-Ft Worth','Denver','Atlanta','Atlanta',
                                              'Los Angeles'])),
    "latitude": pa.Column(float, checks=[Check.ge(-90.0), Check.le(90.0)]),
    "longitude": pa.Column(float, checks=[Check.ge(-180.0), Check.le(180.0)]),
    "timezone": pa.Column(str, checks=pa.Check.str_length(3)),
    "departures": pa.Column(np.float64, nullable=True)
     })

# Validate the dataframe
try:
    schema.validate(airport_data)
    print("Data is valid!")
    airports_df = schema(airport_data)
except pa.errors.SchemaError as e:
    print(f"Data validation error: {e}")
```

✓ 0.0s

```
Data is valid!
```

ERD


www.quickdatabasediagrams.com
airports
name text
iata_code varchar(5)
city varchar(50)
latitude float
longitude float
timezone varchar(5)
departures int
airport_id varchar(100)
country_code_id varchar(30)
itinerary_df
itinerary_id text
itineraries text
flight_numbers
flight_id int
flight_number int
itinerary_details
departure_date_time timestamp
arrival_date_time timestamp
is_self_transfer bool
flight_id int
itinerary_id text
origin_airport_id text
destination_airport_id text
country_code
country_code_id varchar(100)
country_code varchar(3)
cabin_class
cabin_class_id varchar(5)
cabin_class varchar(20)
flight_price
stop_count int
duration_in_hours float
one_way_price_candollar int
cabin_class varchar(20)
operating_airline_id text
marketing_airline_id text
itinerary_id text
airlines
airline_id varchar(100)
airline_name text
iata varchar(3)
is_airline_passenger bool
total_aircrafts int
average_fleet_age float
accidents_last_5y int
iosa_registered bool
iosa_expiry TIMESTAMPTZ
country_code_id varchar(2)

A sample of Postgres Tables

Merged Itinerary Details table

departure_date_time	arrival_date_time	is_self_tra	flight_id	itinerary_i	origin_air	destination_airport_id
4/1/2025 7:45	4/1/2025 10:17	TRUE	1	it1	ap5	ap9
4/1/2025 11:20	4/1/2025 12:49	TRUE	2	it1	ap9	ap7
4/1/2025 7:00	4/1/2025 7:59	TRUE	3	it2	ap1	ap4
4/1/2025 6:15	4/1/2025 8:49	TRUE	4	it3	ap5	ap9
4/1/2025 7:45	4/1/2025 10:17	TRUE	5	it4	ap5	ap9
4/1/2025 11:20	4/1/2025 12:49	TRUE	6	it4	ap9	ap7
4/1/2025 12:20	4/1/2025 14:40	TRUE	7	it5	ap5	ap9
4/1/2025 13:24	4/1/2025 16:14	TRUE	8	it6	ap5	ap7
4/1/2025 17:15	4/1/2025 20:10	TRUE	9	it7	ap5	ap7
4/1/2025 6:45	4/1/2025 9:40	TRUE	10	it8	ap5	ap7
4/1/2025 19:40	4/1/2025 22:14	TRUE	11	it9	ap5	ap7
4/1/2025 14:00	4/1/2025 16:34	TRUE	12	it10	ap5	ap7
4/1/2025 13:14	4/1/2025 14:09	TRUE	13	it11	ap5	ap6
4/1/2025 15:02	4/1/2025 17:41	TRUE	14	it11	ap6	ap7
4/1/2025 6:15	4/1/2025 7:30	TRUE	15	it12	ap5	ap6
4/1/2025 8:05	4/1/2025 10:47	TRUE	16	it12	ap6	ap7
4/1/2025 16:44	4/1/2025 17:44	TRUE	17	it13	ap5	ap6
4/1/2025 18:45	4/1/2025 21:33	TRUE	18	it13	ap6	ap7
4/2/2025 7:00	4/2/2025 7:59	TRUE	3	it14	ap1	ap4
4/2/2025 21:15	4/2/2025 22:54	TRUE	19	it15	ap9	ap7
4/2/2025 21:15	4/2/2025 22:54	TRUE	19	it16	ap9	ap7
4/2/2025 6:15	4/2/2025 8:49	TRUE	4	it17	ap5	ap9
4/2/2025 7:45	4/2/2025 10:17	TRUE	5	it18	ap5	ap9
4/2/2025 11:20	4/2/2025 12:49	TRUE	6	it18	ap9	ap7
4/2/2025 7:45	4/2/2025 10:17	TRUE	5	it19	ap5	ap9
4/2/2025 13:17	4/2/2025 14:40	TRUE	20	it19	ap9	ap7

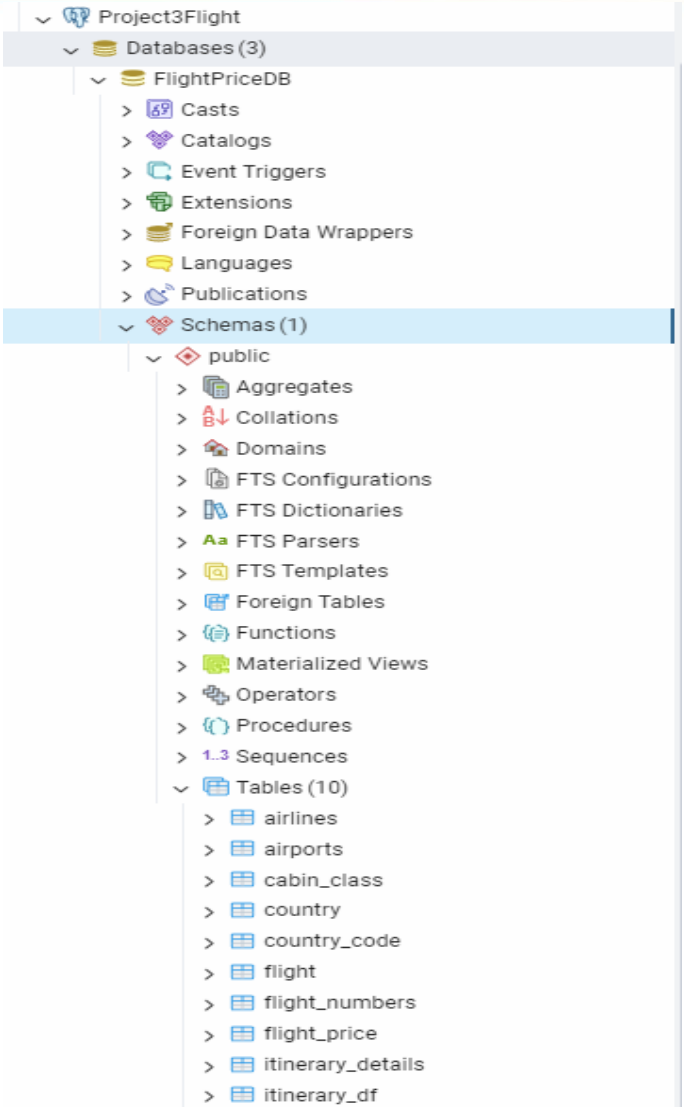
Merged Flight Price table

itinerary_i	stop_cour	duration_i	one_way_	cabin_cla	operating_	marketing_airline_id
18467-250	1	2.53	477	cc1	al4	al16
18467-250	1	2.48	477	cc1	al4	al16
18390-250	2	0.98	253	cc1	al10	al11
18467-250	1	2.57	581	cc2	al4	al4
18467-250	1	2.53	581	cc2	al4	al4
18467-250	1	2.48	581	cc2	al4	al4
18467-250	1	2.33	581	cc2	al4	al4
18467-250	0	3.83	649	cc3	al5	al2
18467-250	0	3.92	527	cc3	al2	al2
18467-250	0	3.92	649	cc3	al2	al2
18467-250	0	3.57	827	cc4	al3	al15
18467-250	0	3.57	1049	cc4	al3	al15
18467-250	1	1.92	670	cc4	al17	al2
18467-250	1	2.65	670	cc4	al2	al2
18467-250	1	2.25	549	cc4	al5	al2
18467-250	1	2.7	549	cc4	al2	al2
18467-250	1	2	549	cc4	al5	al2
18467-250	1	2.8	549	cc4	al2	al2
18390-250	2	0.98	253	cc1	al10	al11
18390-250	2	2.65	263	cc1	al14	al14
18390-250	2	2.65	263	cc1	al14	al14
18467-250	1	2.57	581	cc2	al4	al4
18467-250	1	2.53	581	cc2	al4	al4
18467-250	1	2.48	581	cc2	al4	al4
18467-250	1	2.53	560	cc2	al4	al4
18467-250	1	2.38	560	cc2	al4	al4

Storing Flight Data in PostgreSQL

- Why we chose Postgres?

- Efficiently handles large amounts of flight data while ensuring accuracy.
- Supports both structured tables and flexible data formats like JSON.
- It integrates well with AWS RDS and pgAdmin, making it easy to manage and analyze our data

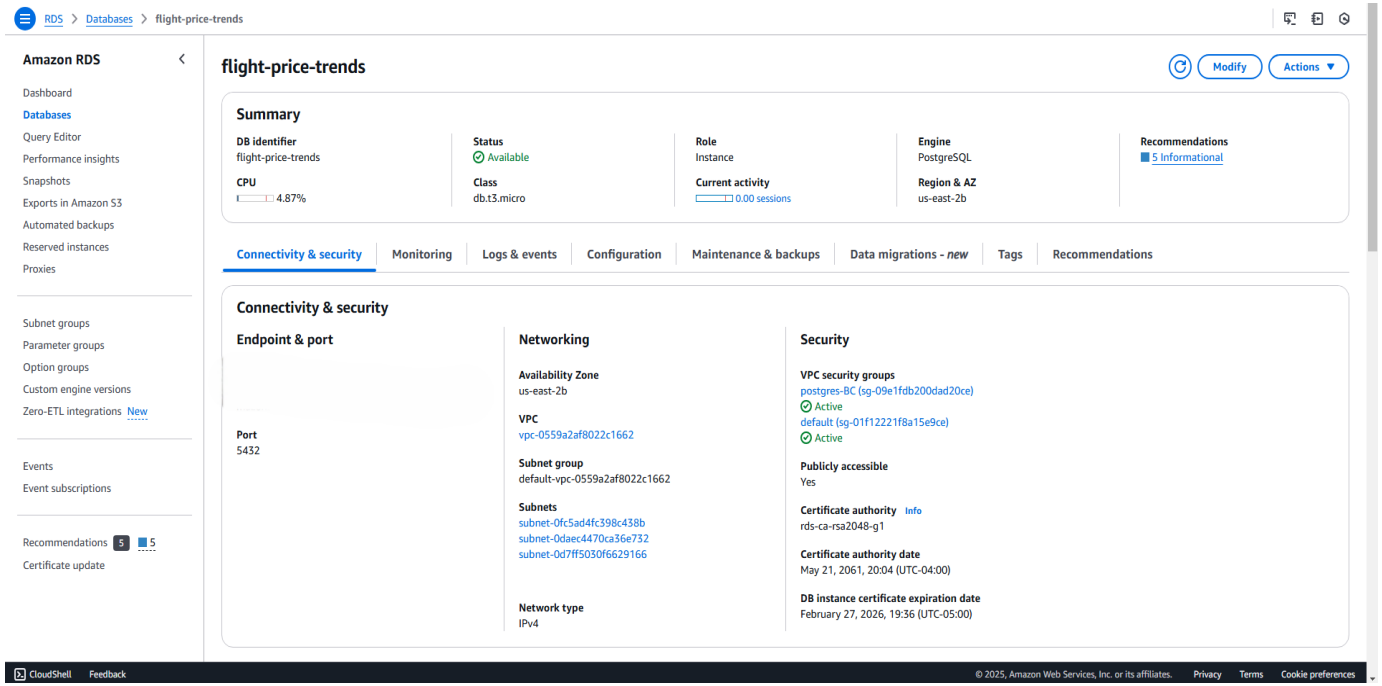


Flask API

- Developed a Flask API to analyze one-way flight prices.

```
(base)
jagar@James-PC MINGW64 ~/flask_postgres
$ C:/Users/jagar/anaconda3/envs/dev/python.exe c:/Users/jagar/flask_postgres/app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 109-750-147
```

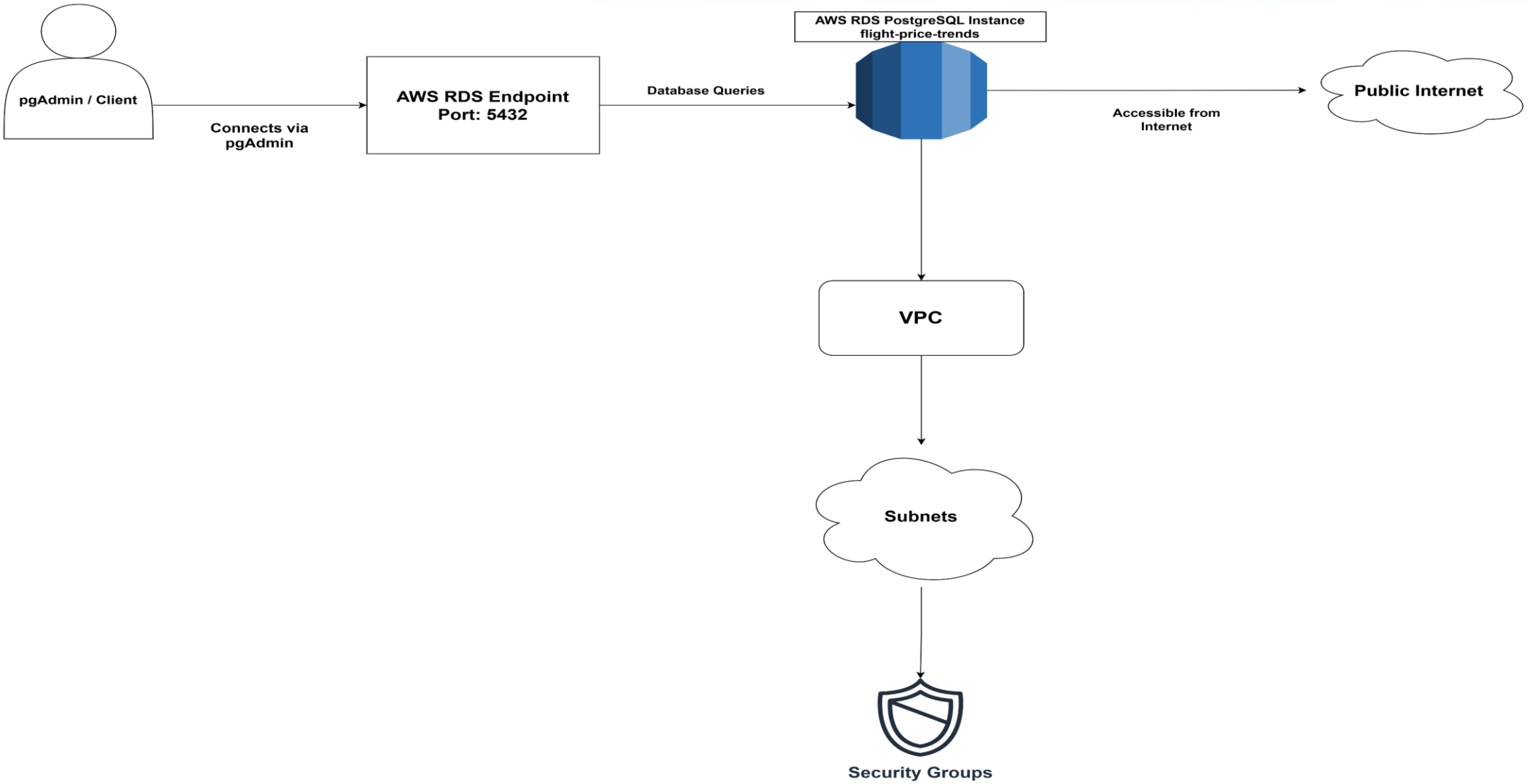
- Connected to AWS RDS PostgreSQL database (FlightPriceDB).



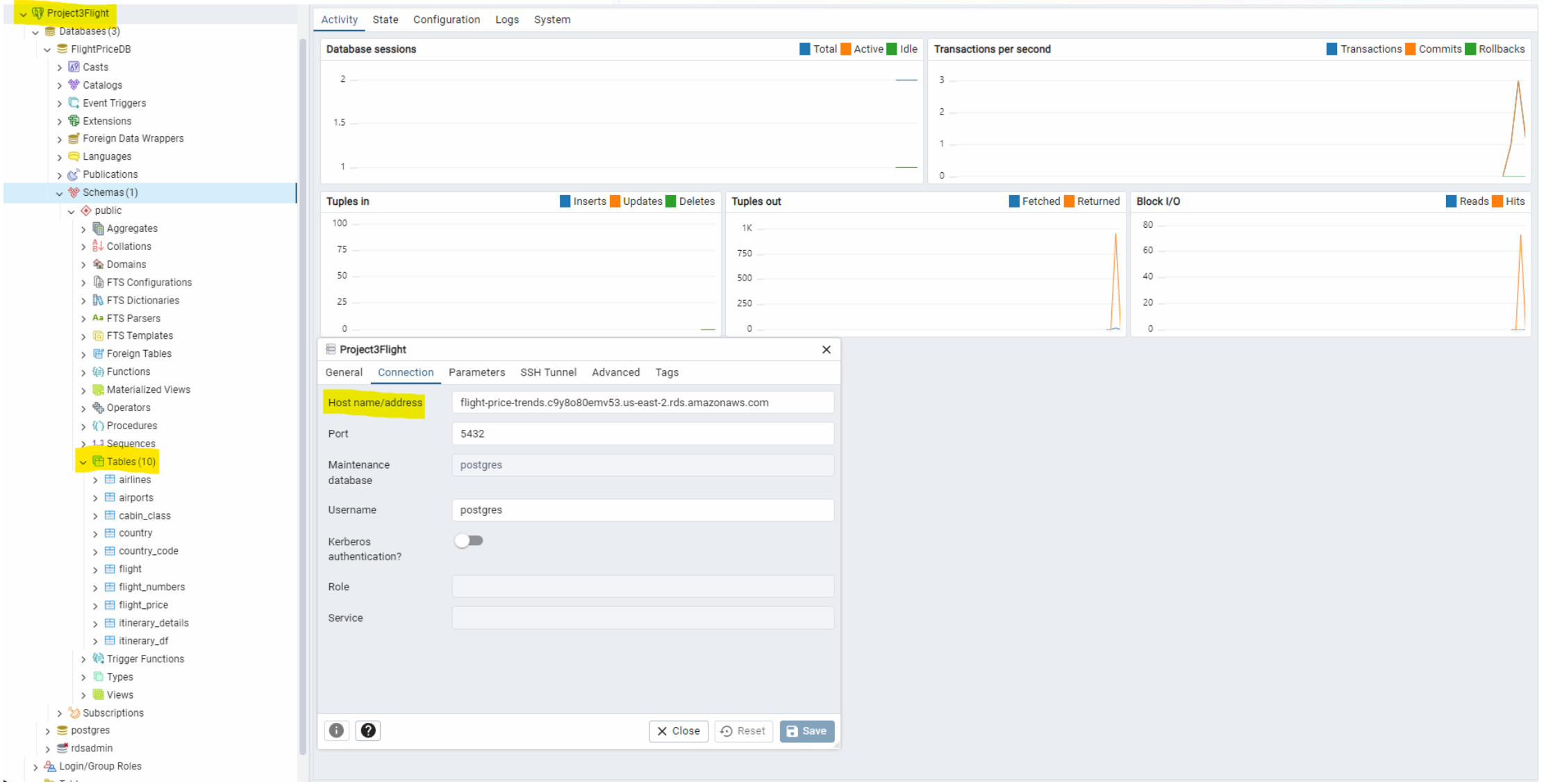
User Interaction

Now for your in-flight demonstration

Up In The Cloud



Up in the Cloud cont.



What’s next?

- Include Return Flights
- Price prediction using Machine Learning
- Update Flask API to show more information especially with the app routes

i.e More destinations, Other countries, More Routes

- Data Visualization

Resources

- [GoFlightLabs Flight Prices] https://www.goflightlabs.com/flight-prices
- How to Create and Connect PostgreSQL with Amazon RDS | S3 CloudHub = https://www.youtube.com/watch?v=0A-5ITILrMA

- **Dataset Licensing (under trial version):** The dataset can be used only for personal and trial - purposes. The trial version is not applicable for commercial purposes.
- [Terms](https://www.goflightlabs.com/terms)

Thank you for flying with us!