Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №1 з дисципліни
«Аналіз даних в інформаційних системах»

„Створення сховища даних ”

Виконав(ла) *ІП-11 Тарасьонок Дмитро Євгенович*
(шифр, прізвище, ім'я, по батькові)

Перевірив *Олійний Ю. О.*
(прізвище, ім'я, по батькові)

Київ 2023

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – ознайомитись з підходами до створення сховищ даних

2 ЗАВДАННЯ

Навчитися створювати процедури завантаження даних до сховища.

1. Самостійно обрати не менше 3-х джерел відкритих даних.
2. Спроектувати модель Stage зони для ETL процесів.
3. Спроектувати модель основного сховища за типом «зірка» або «сніжинка».
4. Створити ETL засоби:
    - завантажити дані до Stage зони
    - створити набір процедур/функцій для перетворення та завантаження даних до основного сховища (або створити засобами програмних ETL засобів). Передбачити можливість завантаження змінених та додаткових даних.
5. Завантажити дані до основного сховища даних.

3 ПОСТАНОВКА ЗАДАЧІ

3.1 Обрати та завантажити дані

Для виконання даної лабораторної роботи мною було обрано дані щодо польотів у світі з сервісу FlightRadar24. Через API сервісу було отримано історію польотів, інформацію про літаки за їх реєстраційними номерами. Для коректного представлення даних також було завантажено список аеропортів світу, список континентів, країн, регіонів країн, а також моделей літаків. Потім для зберігання дат було згенеровано список дат, починаючи з 1 січня 1900 року. Такі старі дати необхідні для зберігання дати виготовлення літаків.

3.2 Опис джерел даних

Загалом було завантажено наступні набори даних:

1. Інформація про польоти – була отримана в ході багаторічного збору даних
2. Історія польотів за номером рейсу: https://api.flightradar24.com/common/v1/flight/list.json?enc=API_KEY&query=ROUTE_NUM&fetchBy=flight&limit=100&page=1&token=API_KEY
3. Інформація про літаки за реєстраційним номером: https://api.flightradar24.com/common/v1/flight/list.json?enc=API_KEY&query=REGISTRATION&fetchBy=reg&limit=1&page=1&filterBy=&token=API_KEY&client=ios_freemium&version=9.2.1
4. Список авіакомпаній: https://raw.githubusercontent.com/jpatokal/openflights/master/data/airlines.dat
5. Список аеропортів: https://davidmegginson.github.io/ourairports-data/airports.csv
6. Список країн: https://davidmegginson.github.io/ourairports-data/countries.csv

7. Список регіонів у країнах: https://davidmegginson.github.io/ourairports-data/regions.csv
8. База даних моделей літаків ICAO: https://www.icao.int/publications/doc8643/pages/search.aspx
9. База даних спеціальних позначень типів літаків ICAO: https://www.icao.int/publications/DOC8643/Pages/SpecialDesignators.aspx

Завантаження файлів є автоматизованим і виконується за допомогою функцій мовою Python. Також функції завантаження проводять певну обробку даних. Було сформовано наступні файли:

1. aircrafts_filtered.csv – літаки
2. airlines.csv – авіакомпанії
3. airports.csv – аеропорти
4. continents.csv – континенти
5. countries.csv – країни
6. dates.csv – дати
7. designators.csv – моделі літаків
8. flights.csv – польоти
9. regions.csv – регіони
10. special_designators – спеціальні позначення типів літаків, конкатенується з файлом designators.csv

Нижче наведена таблиця полів вищезазначених файлів:

aircrafts_filtered.csv	registration	Реєстраційний номер літака
	model_code	Код моделі
	model_text	Назва моделі
	country_code	Код країни реєстрації
	production_date	Дата виробництва університету

	owner_icao	ICAO код авіакомпанії-власника
airlines.csv	airline_id	Ідентифікатор авіакомпанії
	name	Назва
	alias	Псевдонім
	code	ICAO код авіакомпанії
	callsign	Позивний
	country	Країна реєстрації
	active	Активна?
airports.csv	code	ICAO код аеропорту
	type	Тип
	name	Назва
	latitude	Широта
	longitude	Довгота
	elevation	Висота над рівнем моря (фт)
	region	Регіон
	gps_code	Код GPS
continents.csv	code	ISO код континенту
	name	Назва
countries.csv	code	ISO код країни
	name	Назва
	continent	ISO код континенту
dates.csv	the_date	Дата
	weekday	День тижня (Понеділок – 0)
	month	Місяць

	year	Рік
	quarter	Квартал
	day_of_year	День року (0-365)
	weekend	Вихідний?
	week_of_year	Номер тижня
designators.csv	name	Назва моделі
	description	Опис
	turbulence_category	Категорія турбулентності
	designator	Позначення
	manufacturer	Виробник
	type	Тип
	engine_count	Кількість двигунів
	engine_type	Тип двигунів
flights.csv	flight_id	Ідентифікатор польоту
	route_number	Номер рейсу
	aircraft_registration	Реєстраційний номер літака
	airline_icao	ICAO код авіакомпанії
	airport_origin	Аеропорт вильоту
	airport_destination	Аеропорт призначення
	scheduled_departure	Запланований час вильоту
	scheduled_arrival	Запланований час прильоту

	real_departure	Фактичний час вильоту
	real_arrival	Фактичний час прильоту
regions.csv	code	ISO код регіону
	name	Назва
	country	ISO код країни
special_designators	ModelFullName	Назва моделі
	Designator	Позначення

4 РОЗРОБКА МОДЕЛІ СХОВИЩА ДАНИХ

Сховище даних вміщатиме всю вищезазначену інформацію. Фактова таблиця міститиме інформацію про номер рейсу, авіакомпанію, літак, заплановані та реальні часи вильоту та прильоту.

4.1 Модель stage зони

У stage зоні будуть наступні таблиці:

1. aircrafts - літаки
2. airlines - авіакомпанії
3. airports - аеропорти
4. flights - польоти
5. designators – моделі літаків
6. dates - дати
7. regions - регіони
8. countries - країни
9. continents – континенти

Загалом усі поля цієї моделі співпадають з полями CSV файлів, але варто зауважити, що в кожній таблиці первинним ключем є автоінкремент. Зроблено це для того, щоб при додаванні значень у неочищену stage зону не виникало проблем. На рисунку 4.1 наведена схема stage зони

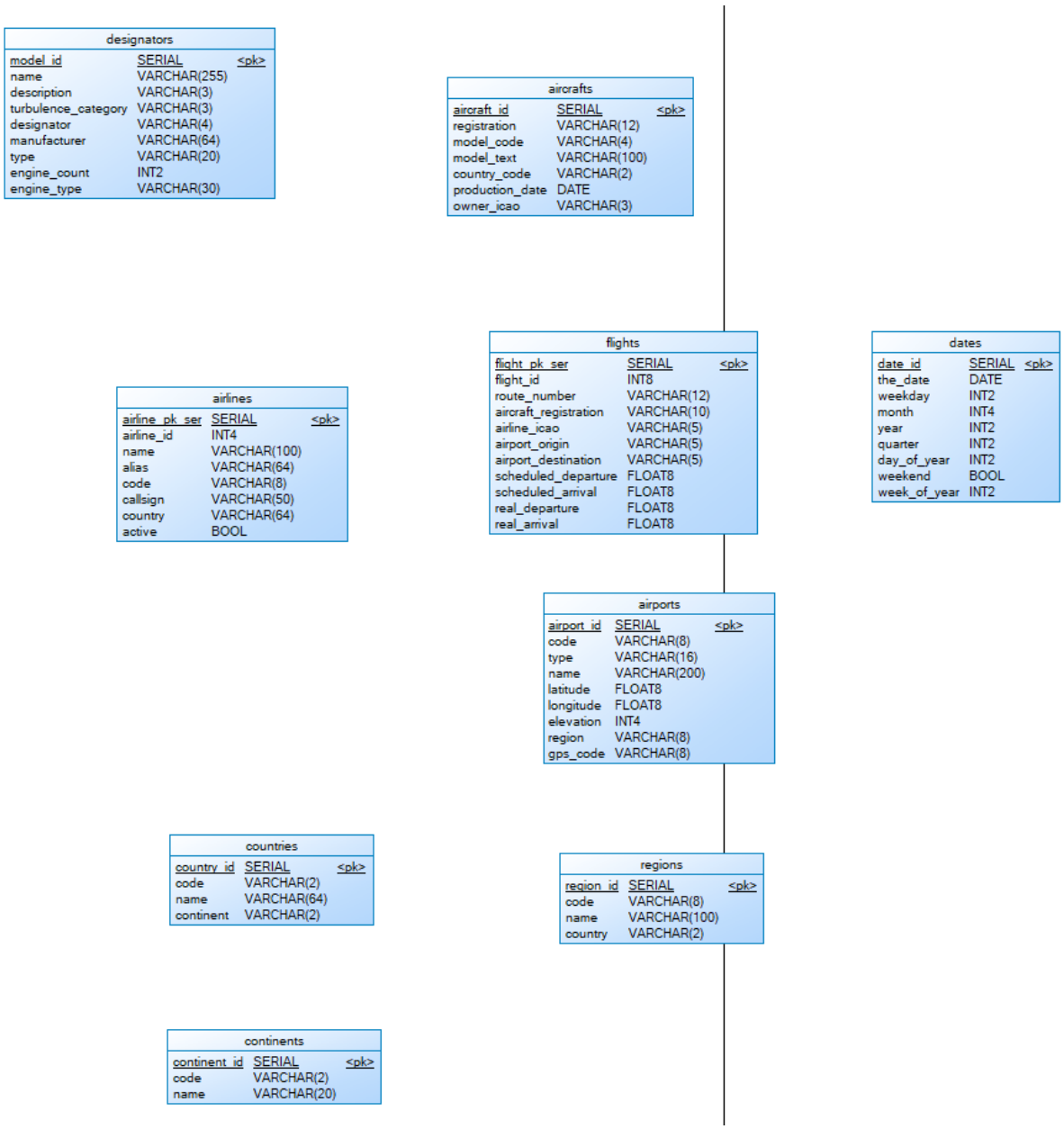

Рисунок 4.1 – Модель stage зони

4.2 Модель основного сховища даних

В основному сховищі всі значення, що можуть повторюватися, були винесені в окремі таблиці. Це: виробники літаків, категорії турбулентності літаків, типи літаків, описи літаків, типи двигунів, маршрути, типи аеропортів.

Загалом основне сховище даних включає такі таблиці:

1. airlines_dim – вимір авіакомпаній
2. airports_dim – вимір аеропортів
3. aircraft_types_dim – вимір типів літаків
4. countries_dim – вимір країн

5. airports_types_dim – вимір типів аеропортів
6. continents_dim – вимір континентів
7. engine_types_dim – вимір типів двигунів
8. models_descriptions_dim – вимір описів моделей літаків
9. manufacturers_dim – вимір виробників літаків
10. flights_fact – вимір польотів
11. dates_dim – вимір дат
12. aircrafts_dim – вимір літаків
13. designators_dim – вимір моделей літаків
14. regions_dim – вимір регіонів
15. turbulence_categories_dim – вимір категорій турбулентності
16. routes_dim – вимір маршрутів

На рисунку 4.2 наведено модель основного сховища даних

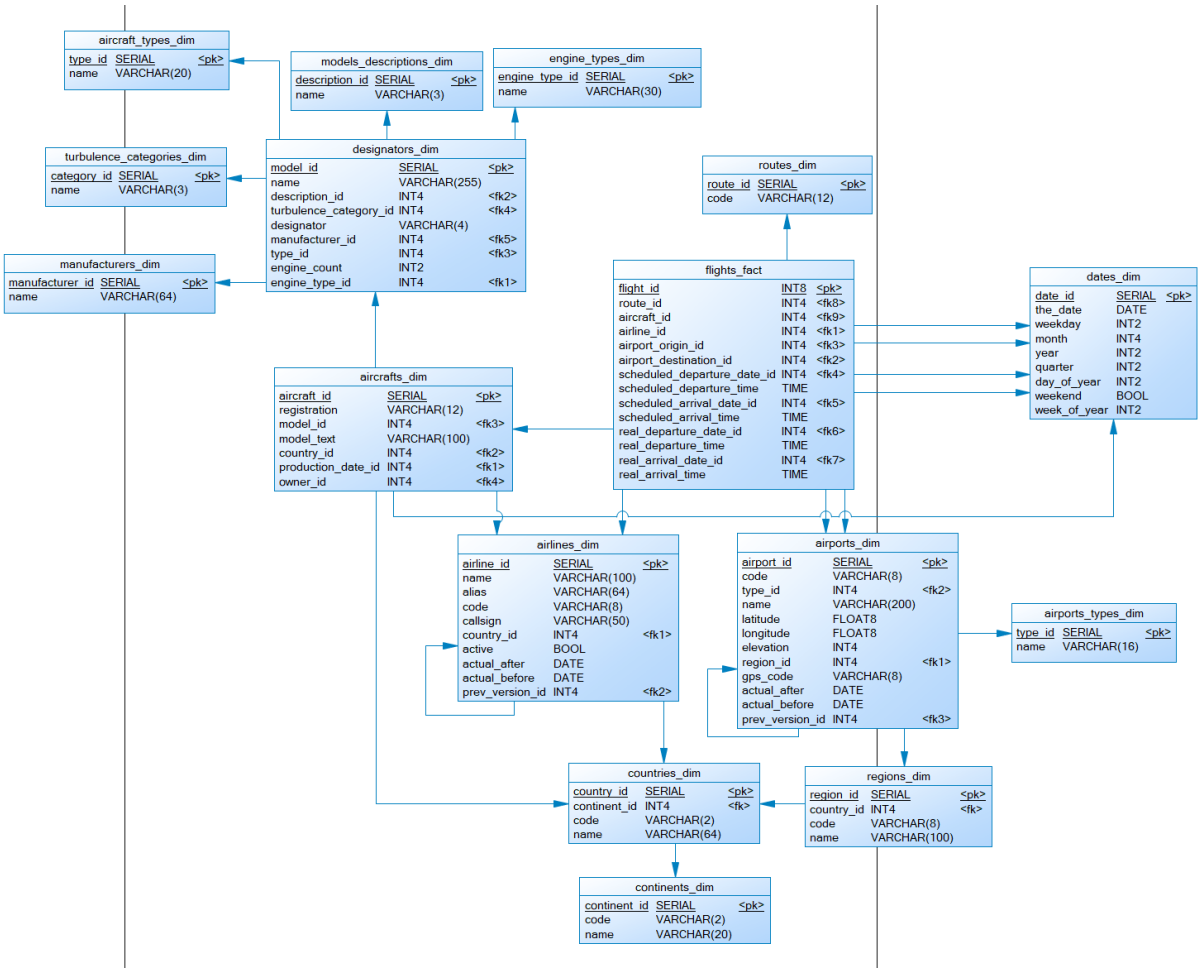

Рисунок 4.2 – Модель основного сховища даних

5 АЛГОРИТМ ЗАВАНТАЖЕННЯ ДАНИХ

Для початкового завантаження даних мною було створено такий алгоритм:

1. Переглядається історія польотів, отримуються номери рейсів
2. За номерами рейсів завантажуємо інформацію про них у json форматі
3. Об’єднуємо файли json в один файл CSV
4. Переглядаємо отриманий файл CSV та отримуємо унікальні реєстраційні номери літаків
5. Завантажуємо інформацію про літаки у json форматі
6. Об’єднуємо файли json в один файл CSV
7. Завантажуємо іншу зазначену інформацію
8. Генеруємо дати

Варто вказати, що цей алгоритм включає відкидання зайвих стовпців у файлах даних, а тому вихідні CSV файли вже мають такий же вигляд, як і таблиці stage зони, через що надалі з ними не потрібно проводити ніяких маніпуляцій.

Після створення CSV файлів дані завантажуються у stage зону, яка перед цим повністю очищається, але у випадку, якщо цього зроблено не було, зона спроектована таким чином, щоб це не викликало проблем.

Коли дані завантажено в stage зону, є декілька варіантів подальших дій з перенесення даних зі stage зони до основного сховища

1. Якщо таблиця виміру не містить зовнішніх ключів, треба завантажити всі унікальні значення цієї таблиці в stage зоні до таблиці вимірів при цьому ігноруючи повтори значень
2. Якщо таблиця виміру містить зовнішні ключі, треба отримати всі значення за унікальним полем, відсортовані за спаданням індексу (для того щоб додавалися новіші значення), а потім об’єднати за допомогою LEFT JOIN там, де значення може бути нульовим та INNER JOIN, де не може
3. Якщо це таблиця aircrafts, необхідно викликати функцію на мові Python, яка знайде назву моделі з таблиці моделей, а потім об’єднати ці таблиці

В останніх двох випадках при завантаженні одних і тих самих даних треба перевіряти цей момент і оновлювати значення для цього поля (ON CONFLICT DO UPDATE)

ДОДАТОК А (ЗАВАНТАЖЕННЯ ДАНИХ)

У даному додатку наведено код на мові Python для завантаження даних. Варто зауважити, що автоматичне завантаження файлів continents.csv, special_designators.csv не передбачене.

А.1. Загальний код

```
import csv
import json
import re
import time
import uuid
from pathlib import Path

import numpy as np
import pandas as pd
import pytz
import requests
import datetime as dt
import tqdm
import psycopg2
from difflib import SequenceMatcher
#%%
DATA_DIR = Path.cwd().joinpath('data')
SQL_DIR = Path.cwd().joinpath('sql')
AIRCRAFTS_DIR = DATA_DIR.joinpath('aircrafts')
AIRCRAFTS_FILTERED_DIR = DATA_DIR.joinpath('aircrafts_filtered')
ROUTES_DIR = DATA_DIR.joinpath('routes')
ROUTES_FILTERED_DIR = DATA_DIR.joinpath('routes_filtered')
#%%
DB_HOST = 'localhost'
DB_PORT = 5432
DB_USER = 'postgres'
DB_PASSWORD = r'zWKHqx1N3%Gt'
DB_NAME = 'data_analysis_lab1'
#%%
FR24_DTYPES = {
    'flight_id': object,
    'icao24': object,
    'latitude': float,
    'longitude': float,
    'heading': int,
    'height': int,
    'airspeed': int,
    'squawk': float,
    'locator': object,
    'aircraft': object,
    'registration': object,
    'unixtime': int,
    'departure': object,
    'arrival': object,
    'ticket_route': object,
    'status': int,
    'vertical_speed': int,
    'transponder_route': object,
    'airline': object,
}
#%%
SYMBOL_REPLACES = {
```

```
    '\u00c0': 'A',
    '\u00c1': 'A',
    '\u00c2': 'A',
    '\u00c3': 'A',
    '\u00c4': 'A',
    '\u00c5': 'A',
    '\u00c6': 'A',
    '\u00c7': 'C',
    '\u00c8': 'E',
    '\u00c9': 'E',
    '\u00ca': 'E',
    '\u00cb': 'E',
    '\u00cc': 'I',
    '\u00cd': 'I',
    '\u00ce': 'I',
    '\u00cf': 'I',
    '\u00d1': 'N',
    '\u00d2': 'O',
    '\u00d3': 'O',
    '\u00d4': 'O',
    '\u00d5': 'O',
    '\u00d6': 'O',
    '\u00d8': 'O',
    '\u00d9': 'U',
    '\u00da': 'U',
    '\u00db': 'U',
    '\u00dc': 'U',
    '\u00dd': 'Y',
    '\u00df': 'S',
    '\u00e0': 'a',
    '\u00e1': 'a',
    '\u00e2': 'a',
    '\u00e3': 'a',
    '\u00e4': 'a',
    '\u00e5': 'a',
    '\u00e6': 'a',
    '\u00e7': 'c',
    '\u00e8': 'e',
    '\u00e9': 'e',
    '\u00ea': 'e',
    '\u00eb': 'e',
    '\u00ec': 'i',
    '\u00ed': 'i',
    '\u00ee': 'i',
    '\u00ef': 'i',
    '\u00f0': 'd',
    '\u00f1': 'n',
    '\u00f2': 'o',
    '\u00f3': 'o',
    '\u00f4': 'o',
    '\u00f5': 'o',
    '\u00f6': 'ö',
    '\u00f8': 'o',
    '\u00f9': 'u',
    '\u00fa': 'u',
    '\u00fb': 'u',
    '\u00fc': 'u',
    '\u00fd': 'y',
    '\u00ff': 'y',
    '\u200b': '',
    '\xa0': ' '
}
```

А.2. get_unique_data() – отримання унікальних номерів рейсів та реєстраційних номерів літаків

```
def get_unique_data():
    registrations = []
    routes = []
    for index, file in enumerate(
            list(DATA_DIR.joinpath('fr24').iterdir())[0:]):
        try:
            data = pd.read_csv(file, names=FR24_DTYPES.keys(),
on_bad_lines='skip', dtype=FR24_DTYPES)
        except ValueError:
            continue
        unique_registrations = data.registration.dropna().unique()
        registrations.append(unique_registrations)

        unique_routes = data.ticket_route.dropna().unique()
        routes.append(unique_routes)
    registrations = np.unique(np.concatenate(registrations))
    routes = np.unique(np.concatenate(routes))
    return registrations, routes
```

А.3. get_aircraft_designators() – отримання моделей літаків

```
def get_aircraft_designators():
    response = requests.post(
        url='https://www4.icao.int/doc8643/External/AircraftTypes'
    )

    content = response.content.decode(response.encoding)
    json_content = json.loads(content)
    designators_data = pd.DataFrame.from_records(json_content)
    special_designators =
pd.read_csv(DATA_DIR.joinpath('special_designators.csv'))

    designators = pd.concat([designators_data, special_designators])

    designators.columns = ['name', 'description', 'turbulence_category',
'WTG', 'designator', 'manufacturer', 'type',
                           'engine_count', 'engine_type']
    designators.engine_count =
designators.engine_count.str.replace(r'[^\d]+', '1', regex=True)

    designators.to_csv(DATA_DIR.joinpath('designators.csv'), index=False,
                       columns=['name', 'description', 'turbulence_category',
'designator', 'manufacturer', 'type',
                                'engine_count', 'engine_type'])
```

А.4. get_aircrafts_info() – отримання інформації щодо літаків із файлу

```
def get_aircrafts_info(filename):
    registrations = pd.read_csv(filename, names=['registration'])

    registrations = registrations.sample(frac=1)

    progress_bar = tqdm.tqdm(registrations.registration)
    for registration in progress_bar:
        progress_bar.set_description(f'Fetching {registration}')
        try:
            response = requests.get(

url='https://api.flightradar24.com/common/v1/flight/list.json?'
```

```
                    'enc=IKQGxn3NR31_n-55iS2uKcuzjmvSFrtJX6mpRJYT7oI&'
                    f'query={registration}&'
                    'fetchBy=reg&'
                    'limit=1&'
                    'timestamp=0&'
                    'page=1&'
                    'filterBy=&'
                    'token=IKQGxn3NR31_n-55iS2uKcuzjmvSFrtJX6mpRJYT7oI&'
                    'client=ios_freemium&'
                    'version=9.2.1',
                headers={
                    'User-Agent': 'FlightradarFree/2023021501
CFNetwork/1404.0.5 Darwin/22.3.0'
                }
            )
            with open(f'aircrafts\\{registration}.json', 'w',
encoding=response.encoding) as file:
                file.write(response.content.decode(response.encoding))
            time.sleep(0.6)
        except:
            ...
```

A.5. get_routes_info() – отримання інформації щодо польотів із файлу

```
def get_routes_info(filename: 'str | Path'):
    routes = pd.read_csv(filename, names=['route'])

    routes = routes.sample(frac=1)

    progress_bar = tqdm.tqdm(routes.route)

    for route in progress_bar:
        progress_bar.set_description(f'Fetching {routes}')
        progress_bar.refresh()
        page = 1
        last = None
        while True:
            time.sleep(0.6)

            response = requests.get(

url='https://api.flightradar24.com/common/v1/flight/list.json?'
                    'enc=6pDfb2KZPxots_3kFVasmNL1WJ7rQXvJ5yJb4NhegjA&'
                    f'query={route}&'
                    'fetchBy=flight&'
                    'limit=100&'
                    f'page={page}&'
                    'token=6pDfb2KZPxots_3kFVasmNL1WJ7rQXvJ5yJb4NhegjA',
                headers={
                    'User-Agent': 'FlightradarFree/2023021501
CFNetwork/1404.0.5 Darwin/22.3.0'
                }
            )
            content = response.content.decode(response.encoding)

            debug_name = DATA_DIR.joinpath(f'debug\\{uuid.uuid4()}.json')

            with open(debug_name, 'w') as file:
                file.write(content)

            if '402 Payment Required' in content or 'Error reference number'
in content:
                break
```

```
            json_data = json.loads(content)
            if 'errors' in json_data:
                break
            elif 'result' not in json_data:
                break
            elif 'response' not in json_data['result']:
                break

            result = json_data['result']['response']

            respath = ROUTES_DIR.joinpath(f'{route}_{page}.json')

            with open(respath, 'w') as file:
                json.dump(json_data, file)

            if 'page' not in result:
                break

            data = result['data']

            if data is None or (last is not None and data[0] == last):
                respath.unlink()
                break

            if not result['page']['more']:
                break

            page += 1
            last = data[0]
```

A.6. filter_aircrafts() – фільтрація файлів із літаками для обробки помилок

```
def filter_aircrafts():
    progress_bar = tqdm.tqdm(AIRCRAFTS_DIR.iterdir())
    for filename in progress_bar:
        progress_bar.set_description(f'Checking {filename.name}')
        if filename.suffix == '.json':
            with open(filename) as file:
                content = file.read()
            if '402 Payment Required' in content or 'Cloudflare Location' in
content:
                filename.unlink()
            else:
                json_data = json.loads(content)
                if 'errors' not in json_data:
                    aircraftInfo =
json_data['result']['response']['aircraftInfo']
                    if aircraftInfo:
                        with
open(DATA_DIR.joinpath('aircrafts_filtered').joinpath(filename.name), 'w') as
file:

json.dump(json_data['result']['response']['aircraftInfo'], file)
        else:
            filename.unlink()
```

A.7. filter_routes() – фільтрація файлів із польотами для обробки помилок

```
def filter_routes():
    progress_bar = tqdm.tqdm(list(ROUTES_DIR.iterdir()))
    for filename in progress_bar:
        progress_bar.set_description(f'Checking {filename.name}')
        if filename.suffix == '.json':
```

```
            with open(filename) as file:
                content = file.read()
            json_data = json.loads(content)
            if 'errors' not in json_data:
                response = json_data['result']['response']
                if 'data' in response and response['data']:
                    with open(ROUTES_FILTERED_DIR.joinpath(filename.name),
'w') as file:
                        json.dump(response['data'], file)
```

A.8. routes_to_csv() – об'єднання json файлів із польотами в один CSV файл

```
def routes_to_csv():
    def check_data(data) -> bool:
        if 'identification' in data:
            if 'id' in data['identification']:
                return data['identification']['id'] is not None
        return False

    csv_path = DATA_DIR.joinpath('flights.csv')
    if csv_path.exists():
        csv_path.unlink()

    progress_bar = tqdm.tqdm(list(ROUTES_FILTERED_DIR.iterdir()))

    for filename in progress_bar:
        progress_bar.set_description(f'Adding {filename.name}')
        progress_bar.refresh()
        with open(filename) as json_file:
            json_data = json.load(json_file)
            airline_icao = ''
            flights = []
            for index, row in enumerate(filter(check_data, json_data)):
                if 'status' in row:
                    if row['status']['live']:
                        continue
                route_number_regex = re.match('(?P<number>.+)_.+',
filename.name)
                route_number = route_number_regex.group('number')
                flight_id = row['identification']['id']
                aircraft_registration = ''
                airport_origin = ''
                airport_destination = ''
                scheduled_departure = ''
                scheduled_arrival = ''
                real_departure = ''
                real_arrival = ''

                if 'aircraft' in row:
                    aircraft = row['aircraft']
                    if aircraft and 'registration' in aircraft:
                        aircraft_registration =
row['aircraft']['registration']
                if not airline_icao and 'airline' in row:
                    airline = row['airline']
                    if airline and 'code' in airline:
                        code = airline['code']
                        if code and 'icao' in code:
                            airline_icao = code['icao']
                if 'airport' in row:
                    airport = row['airport']
                    if airport:
                        if 'origin' in airport:
                            origin = airport['origin']
```

```
                            if origin and 'code' in origin:
                                code = origin['code']
                                if code and 'icao' in code:
                                    airport_origin = code['icao']
                            if 'destination' in airport:
                                destination = airport['destination']
                                if destination and 'code' in destination:
                                    code = destination['code']
                                    if code and 'icao' in code:
                                        airport_destination = code['icao']
                    if 'time' in row:
                        route_times = row['time']
                        if route_times:
                            if 'scheduled' in route_times:
                                scheduled_times = route_times['scheduled']
                                if scheduled_times:
                                    if 'departure' in scheduled_times:
                                        scheduled_departure =
scheduled_times['departure']
                                    if 'arrival' in scheduled_times:
                                        scheduled_arrival =
scheduled_times['arrival']
                            if 'real' in route_times:
                                real_times = route_times['real']
                                if real_times:
                                    if 'departure' in real_times:
                                        real_departure = real_times['departure']
                                    if 'arrival' in real_times:
                                        real_arrival = real_times['arrival']
                    flights.append([int(flight_id, 16), route_number,
aircraft_registration, airline_icao,
                                    airport_origin, airport_destination,
                                    scheduled_departure, scheduled_arrival,
real_departure,
                                    real_arrival])
                if airline_icao:
                    for flight in flights:
                        flight[3] = airline_icao
                with open(csv_path, 'a', newline='') as file:
                    csv_writer = csv.writer(file)
                    csv_writer.writerows(flights)

    dtypes = {
        'flight_id': int,
        'route_number': object,
        'aircraft_registration': object,
        'airline_icao': object,
        'airport_origin': object,
        'airport_destination': object,
        'scheduled_departure': object,
        'scheduled_arrival': object,
        'real_departure': float,
        'real_arrival': float
    }
    data = pd.read_csv(
        csv_path,
        names=dtypes.keys()
    )

    data = data.drop_duplicates(subset='flight_id')
    data.to_csv(csv_path, index=False)
```

A.9. get_missing_aircrafts() – отримання недостаючих літаків

```
def get_missing_aircrafts():
    dtypes = {
        'flight_id': str,
        'route_number': str,
        'aircraft_registration': str,
        'airline_icao': str,
        'airport_origin': str,
        'airport_destination': str,
        'scheduled_departure': str,
        'scheduled_arrival': str,
        'real_departure': float,
        'real_arrival': float
    }
    data = pd.read_csv(
        DATA_DIR.joinpath('routes_filtered.csv'),
        names=dtypes.keys(),
        dtype=dtypes,
        index_col='flight_id'
    )

    aircrafts_aircrafts = set(map(lambda p: p.stem,
DATA_DIR.joinpath('aircrafts_filtered').iterdir()))
    routes_aircrafts = set(data.aircraft_registration.unique())

    with open(DATA_DIR.joinpath('missed_aircrafts.csv'), 'w') as file:
        file.write('\n'.join(map(str, routes_aircrafts -
aircrafts_aircrafts)))

    get_aircrafts_info(DATA_DIR.joinpath('missed_aircrafts.csv'))
```

A.10. aircrafts_to_csv() – об’єднання json файлів із літаками в один CSV файл

```
def aircrafts_to_csv():
    csv_path = DATA_DIR.joinpath('aircrafts_filtered.csv')
    with open(csv_path, 'w', newline='') as csv_file:
        csw_writer = csv.writer(csv_file)
        csw_writer.writerow([
            'registration',
            'model_code',
            'model_text',
            'country_code',
            'production_date',
            'owner_icao'
        ])

    progress_bar = tqdm.tqdm(list(AIRCRAFTS_FILTERED_DIR.iterdir()))

    for filename in progress_bar:
        # for filename in [AIRCRAFTS_FILTERED_DIR.joinpath('D-EEEH.json')]:
        progress_bar.set_description(f'Adding {filename.name}')
        progress_bar.refresh()
        with open(filename) as json_file:
            try:
                json_data = json.load(json_file)
            except:
                continue
            registration = filename.stem

            model_code = None
```

```
            model_text = None
            country_code = None
            production_date = None
            owner_icao = None

            if 'model' in json_data:
                model = json_data['model']
                if model:
                    if 'code' in model:
                        model_code = model['code']
                    if 'text' in model:
                        model_text = model['text']
                        if model_text:
                            for original, to_replace in
SYMBOL_REPLACES.items():
                                model_text = model_text.replace(original,
to_replace)

            if 'country' in json_data:
                country = json_data['country']
                if country and 'alpha2' in country:
                    country_code = country['alpha2']

            if 'age' in json_data:
                age = json_data['age']
                if age and 'date' in age and age['date']:
                    production_date = dt.datetime.strptime(age['date'], '%b
%Y').date()

            if 'owner' in json_data:
                owner = json_data['owner']
                if owner and 'code' in owner:
                    code = owner['code']
                    if 'icao' in code:
                        owner_icao = code['icao']
            with open(csv_path, 'a', newline='') as file:
                csv_writer = csv.writer(file)
                csv_writer.writerow([
                    registration,
                    model_code,
                    model_text,
                    country_code,
                    production_date,
                    owner_icao
                ])
```

A.11. get_airports() – отримання інформації щодо аеропортів

```
def get_airports():
    airports = pd.read_csv('https://davidmegginson.github.io/ourairports-
data/airports.csv', header=0, names=[
        'airport_id',
        'code',
        'type',
        'name',
        'latitude',
        'longitude',
        'elevation',
        'continent',
        'country',
        'region',
        'municipality',
        'scheduled_service',
        'gps_code',
```

```
        'iata_code',
        'local_code',
        'home_link',
        'wikipedia_link',
        'keywords'
    ], keep_default_na=False)
    airports.to_csv(DATA_DIR.joinpath('airports.csv'), columns=[
        'code',
        'type',
        'name',
        'latitude',
        'longitude',
        'elevation',
        'region',
        'gps_code'
    ], index=False)
```

A.12. get_countries() – отримання інформації щодо країн

```
def get_countries():
    countries = pd.read_csv('https://davidmegginson.github.io/ourairports-
data/countries.csv', header=0, names=[
        'country_id',
        'code',
        'name',
        'continent',
        'wikipedia_link',
        'keywords'
    ], keep_default_na=False)
    countries.to_csv(DATA_DIR.joinpath('countries.csv'), columns=[
        'code',
        'name',
        'continent'
    ], index=False)
```

A.13. get_regions() – отримання інформації щодо регіонів

```
def get_regions():
    regions = pd.read_csv('https://davidmegginson.github.io/ourairports-
data/regions.csv', header=0, names=[
        'region_id',
        'code',
        'local_coe',
        'name',
        'continent',
        'country',
        'wikipedia_link',
        'keywords'
    ], keep_default_na=False)
    regions.to_csv(DATA_DIR.joinpath('regions.csv'), columns=[
        'code',
        'name',
        'country'
    ], index=False)
```

A.14. get_airlines() – отримання інформації щодо авіакомпаній

```
def get_airlines():
    airlines =
pd.read_csv('https://raw.githubusercontent.com/jpatokal/openflights/master/da
ta/airlines.dat',
```

```
                                header=None, names=[
            'airline_id',
            'name',
            'alias',
            'iata_code',
            'code',
            'callsign',
            'country',
            'active'
        ], keep_default_na=False)
    airlines.active = airlines.active.str.upper().map({'Y': True, 'N':
False})
    airlines.to_csv(DATA_DIR.joinpath('airlines.csv'), columns=[
        'airline_id',
        'name',
        'alias',
        'code',
        'callsign',
        'country',
        'active'
    ], index=False)
```

A.15. generate_dates() – генерація дат

```
def generate_dates(
        start_date=dt.datetime(year=1900, month=1, day=1, tzinfo=pytz.UTC),
        last_date=dt.datetime.now(tz=pytz.UTC)
):
    if start_date.tzinfo != pytz.UTC or last_date.tzinfo != pytz.UTC:
        raise ValueError('Time zone must be UTC')

    rows = [
        ['the_date', 'weekday', 'month', 'year', 'quarter', 'day_of_year',
'weekend', 'week_of_year']
    ]

    first_unix_date = dt.datetime(year=1970, month=1, day=1, tzinfo=pytz.UTC)
    current_date = start_date

    while current_date < last_date:
        the_date = current_date.date()
        weekday = current_date.weekday()
        month = current_date.month
        year = current_date.year
        quarter = (current_date.month - 1) // 3
        day_of_year = current_date.timetuple().tm_yday
        weekend = 5 <= weekday
        week_of_year = current_date.isocalendar()[1]
        rows.append([the_date, weekday, month, year, quarter, day_of_year,
weekend, week_of_year])

    with open(DATA_DIR.joinpath('dates.csv'), 'w', newline='') as csv_file:
        csw_writer = csv.writer(csv_file)
        csw_writer.writerows(rows)
```

A.16. generate_data() – виклик усіх функцій, необхідних для створення CSV файлів

```
def generate_data():
    aircrafts_to_csv()
    get_airlines()
    get_airports()
```

```
get_countries()
generate_dates()
get_aircraft_designators()
routes_to_csv()
get_regions()
```

ДОДАТОК Б (СКРИПТИ ДЛЯ СТВОРЕННЯ STAGE ЗОНИ)

```
/*==============================================================*/
/* DBMS name:      PostgreSQL 9.x                               */
/* Created on:     07.03.2023 0:44:41                           */
/*==============================================================*/


drop table if exists stage.aircrafts;

drop table if exists stage.airlines;

drop table if exists stage.airports;

drop table if exists stage.continents;

drop table if exists stage.countries;

drop table if exists stage.dates;

drop table if exists stage.designators;

drop table if exists stage.flights;

drop table if exists stage.regions;

/*==============================================================*/
/* Table: aircrafts                                             */
/*==============================================================*/
create table stage.aircrafts
(
    aircraft_id     SERIAL       not null,
    registration    VARCHAR(12)  not null,
    model_code      VARCHAR(4)   null,
    model_text      VARCHAR(100) null,
    country_code    VARCHAR(2)   null,
    production_date DATE         null,
    owner_icao      VARCHAR(3)   null,
    constraint PK_AIRCRAFTS primary key (aircraft_id)
);

/*==============================================================*/
/* Table: airlines                                              */
/*==============================================================*/
create table stage.airlines
(
    airline_pk_ser SERIAL       not null,
    airline_id     INT4         not null,
    name           VARCHAR(100) not null,
    alias          VARCHAR(64)  null,
    code           VARCHAR(8)   null,
    callsign       VARCHAR(50)  null,
    country        VARCHAR(64)  null,
    active         BOOL         not null,
    constraint PK_AIRLINES primary key (airline_pk_ser)
);

/*==============================================================*/
/* Table: airports                                              */
/*==============================================================*/
create table stage.airports
(
    airport_id SERIAL       not null,
    code       VARCHAR(8)   null,
```

```
    type        VARCHAR(16)  not null,
    name        VARCHAR(200) not null,
    latitude    FLOAT8       not null,
    longitude   FLOAT8       not null,
    elevation   INT4         null,
    region      VARCHAR(8)   not null,
    gps_code    VARCHAR(8)   null,
    constraint PK_AIRPORTS primary key (airport_id)
);

/*==============================================================*/
/* Table: continents                                            */
/*==============================================================*/
create table stage.continents
(
    continent_id SERIAL      not null,
    code         VARCHAR(2)  not null,
    name         VARCHAR(20) not null,
    constraint PK_CONTINENTS primary key (continent_id)
);

/*==============================================================*/
/* Table: countries                                             */
/*==============================================================*/
create table stage.countries
(
    country_id SERIAL      not null,
    code       VARCHAR(2)  not null,
    name       VARCHAR(64) not null,
    continent  VARCHAR(2)  not null,
    constraint PK_COUNTRIES primary key (country_id)
);

/*==============================================================*/
/* Table: dates                                                 */
/*==============================================================*/
create table stage.dates
(
    date_id      SERIAL not null,
    the_date     DATE   not null,
    weekday      INT2   not null,
    month        INT4   not null,
    year         INT2   not null,
    quarter      INT2   not null,
    day_of_year  INT2   not null,
    weekend      BOOL   not null,
    week_of_year INT2   not null,
    constraint PK_DATES primary key (date_id)
);

/*==============================================================*/
/* Table: designators                                           */
/*==============================================================*/
create table stage.designators
(
    model_id            SERIAL       not null,
    name                VARCHAR(255) not null,
    description         VARCHAR(3)   null,
    turbulence_category VARCHAR(3)   null,
    designator          VARCHAR(4)   not null,
    manufacturer        VARCHAR(64)  null,
    type                VARCHAR(20)  null,
    engine_count        INT2         null,
    engine_type         VARCHAR(30)  null,
```

```
    constraint PK_DESIGNATORS primary key (model_id)
);

/*==============================================================*/
/* Table: flights                                               */
/*==============================================================*/
create table stage.flights
(
    flight_pk_ser         SERIAL      not null,
    flight_id             INT8        not null,
    route_number          VARCHAR(12) not null,
    aircraft_registration VARCHAR(10) null,
    airline_icao          VARCHAR(5)  null,
    airport_origin        VARCHAR(5)  null,
    airport_destination   VARCHAR(5)  null,
    scheduled_departure   FLOAT8      null,
    scheduled_arrival     FLOAT8      null,
    real_departure        FLOAT8      null,
    real_arrival          FLOAT8      null,
    constraint PK_FLIGHTS primary key (flight_pk_ser)
);

/*==============================================================*/
/* Table: regions                                               */
/*==============================================================*/
create table stage.regions
(
    region_id SERIAL       not null,
    code      VARCHAR(8)   not null,
    name      VARCHAR(100) not null,
    country   VARCHAR(2)   not null,
    constraint PK_REGIONS primary key (region_id)
);
```

ДОДАТОК В (СКРИПТИ ДЛЯ СТВОРЕННЯ ОСНОВНОГО СХОВИЩА ДАНИХ)

```
/*==============================================================*/
/* DBMS name:      PostgreSQL 9.x                               */
/* Created on:     07.03.2023 0:45:27                           */
/*==============================================================*/


drop index if exists aircraft_type_name_unique cascade;

drop table if exists aircraft_types_dim cascade;

drop index if exists aircraft_registration_unique cascade;

drop table if exists aircrafts_dim cascade;

drop index if exists airline_code_unique cascade;

drop table if exists airlines_dim cascade;

drop index if exists airport_code_unique cascade;

drop table if exists airports_dim cascade;

drop index if exists airport_type_name_unique cascade;

drop table if exists airports_types_dim cascade;

drop index if exists continent_code_unique cascade;

drop table if exists continents_dim cascade;

drop index if exists country_code_unique cascade;

drop table if exists countries_dim cascade;

drop index if exists time_date_unique cascade;

drop table if exists dates_dim cascade;

drop index if exists designator_unique cascade;

drop table if exists designators_dim cascade;

drop index if exists engine_type_name_unique cascade;

drop table if exists engine_types_dim cascade;

drop table if exists flights_fact cascade;

drop index if exists manufacturer_name_unique cascade;

drop table if exists manufacturers_dim cascade;

drop index if exists model_description_name_unique cascade;

drop table if exists models_descriptions_dim cascade;

drop index if exists region_code_unique cascade;

drop table if exists regions_dim cascade;
```

```
drop index if exists route_code_unique cascade;

drop table if exists routes_dim cascade;

drop index if exists turbulence_name cascade;

drop table if exists turbulence_categories_dim cascade;

/*==============================================================*/
/* Table: aircraft_types_dim                                    */
/*==============================================================*/
create table aircraft_types_dim (
                                    type_id              SERIAL
not null,
                                    name                 VARCHAR(20)
not null,
                                    constraint PK_AIRCRAFT_TYPES_DIM primary
key (type_id)
);

/*==============================================================*/
/* Index: aircraft_type_name_unique                             */
/*==============================================================*/
create unique index aircraft_type_name_unique on aircraft_types_dim (
                                                                    name
    );

/*==============================================================*/
/* Table: aircrafts_dim                                         */
/*==============================================================*/
create table aircrafts_dim (
                                aircraft_id          SERIAL               not
null,
                                registration         VARCHAR(12)          not
null,
                                model_id             INT4
null,
                                model_text           VARCHAR(100)
null,
                                country_id           INT4
null,
                                production_date_id   INT4
null,
                                owner_id             INT4
null,
                                constraint PK_AIRCRAFTS_DIM primary key
(aircraft_id)
);

/*==============================================================*/
/* Index: aircraft_registration_unique                          */
/*==============================================================*/
create unique index aircraft_registration_unique on aircrafts_dim (

registration
    );

/*==============================================================*/
/* Table: airlines_dim                                          */
/*==============================================================*/
create table airlines_dim (
                               airline_id           SERIAL               not
null,
                               name                 VARCHAR(100)         null,
```

```
                                alias                VARCHAR(64)          null,
                                code                 VARCHAR(8)           null,
                                callsign             VARCHAR(50)          null,
                                country_id           INT4                 null,
                                active               BOOL                 null,
                                constraint PK_AIRLINES_DIM primary key
(airline_id)
);

/*==============================================================*/
/* Index: airline_code_unique                                   */
/*==============================================================*/
create unique index airline_code_unique on airlines_dim (
                                                        code
    );

/*==============================================================*/
/* Table: airports_dim                                          */
/*==============================================================*/
create table airports_dim (
                                airport_id           SERIAL               not
null,
                                code                 VARCHAR(8)           null,
                                type_id              INT4                 null,
                                name                 VARCHAR(200)         null,
                                latitude             FLOAT8               null,
                                longitude            FLOAT8               null,
                                elevation            INT4                 null,
                                region_id            INT4                 null,
                                gps_code             VARCHAR(8)           null,
                                constraint PK_AIRPORTS_DIM primary key
(airport_id)
);

/*==============================================================*/
/* Index: airport_code_unique                                   */
/*==============================================================*/
create unique index airport_code_unique on airports_dim (
                                                        code
    );

/*==============================================================*/
/* Table: airports_types_dim                                    */
/*==============================================================*/
create table airports_types_dim (
                                    type_id              SERIAL
not null,
                                    name                 VARCHAR(16)
not null,
                                    constraint PK_AIRPORTS_TYPES_DIM primary
key (type_id)
);

/*==============================================================*/
/* Index: airport_type_name_unique                              */
/*==============================================================*/
create unique index airport_type_name_unique on airports_types_dim (
                                                                   name
    );

/*==============================================================*/
/* Table: continents_dim                                        */
/*==============================================================*/
create table continents_dim (
```

```
                                continent_id         SERIAL               not
null,
                                code                 VARCHAR(2)           not
null,
                                name                 VARCHAR(20)          not
null,
                                constraint PK_CONTINENTS_DIM primary key
(continent_id)
);

/*==============================================================*/
/* Index: continent_code_unique                                 */
/*==============================================================*/
create unique index continent_code_unique on continents_dim (
                                                                code
    );

/*==============================================================*/
/* Table: countries_dim                                         */
/*==============================================================*/
create table countries_dim (
                                country_id           SERIAL               not
null,
                                continent_id         INT4                 not
null,
                                code                 VARCHAR(2)           not
null,
                                name                 VARCHAR(64)          not
null,
                                constraint PK_COUNTRIES_DIM primary key
(country_id)
);

/*==============================================================*/
/* Index: country_code_unique                                   */
/*==============================================================*/
create unique index country_code_unique on countries_dim (
                                                              code
    );

/*==============================================================*/
/* Table: dates_dim                                             */
/*==============================================================*/
create table dates_dim (
                            date_id              SERIAL               not
null,
                            the_date             DATE                 not
null,
                            weekday              INT2                 not
null,
                            month                INT4                 not
null,
                            year                 INT2                 not
null,
                            quarter              INT2                 not
null,
                            day_of_year          INT2                 not
null,
                            weekend              BOOL                 not
null,
                            week_of_year         INT2                 not
null,
                            constraint PK_DATES_DIM primary key (date_id)
);
```

```
/*==============================================================*/
/* Index: time_date_unique                                      */
/*==============================================================*/
create unique index time_date_unique on dates_dim (
                                                    the_date
    );

/*==============================================================*/
/* Table: designators_dim                                       */
/*==============================================================*/
create table designators_dim (
                                    model_id              SERIAL
not null,
                                    name                  VARCHAR(255)
null,
                                    description_id        INT4
null,
                                    turbulence_category_id INT4
null,
                                    designator            VARCHAR(4)
null,
                                    manufacturer_id       INT4
null,
                                    type_id               INT4
null,
                                    engine_count          INT2
null,
                                    engine_type_id        INT4
null,
                                    constraint PK_DESIGNATORS_DIM primary key
(model_id)
);

/*==============================================================*/
/* Index: designator_unique                                     */
/*==============================================================*/
create unique index designator_unique on designators_dim (
                                                           designator,
                                                           name
    );

/*==============================================================*/
/* Table: engine_types_dim                                      */
/*==============================================================*/
create table engine_types_dim (
                                     engine_type_id        SERIAL
not null,
                                     name                  VARCHAR(30)
not null,
                                     constraint PK_ENGINE_TYPES_DIM primary key
(engine_type_id)
);

/*==============================================================*/
/* Index: engine_type_name_unique                               */
/*==============================================================*/
create unique index engine_type_name_unique on engine_types_dim (
                                                                  name
    );

/*==============================================================*/
/* Table: flights_fact                                          */
/*==============================================================*/
```

```
create table flights_fact (
                                flight_id            INT8                 not
null,
                                route_id             INT4                 null,
                                aircraft_id          INT4                 null,
                                airline_id           INT4                 null,
                                airport_origin_id    INT4                 null,
                                airport_destination_id INT4
null,
                                scheduled_departure_date_id INT4
null,
                                scheduled_departure_time TIME
null,
                                scheduled_arrival_date_id INT4
null,
                                scheduled_arrival_time TIME
null,
                                real_departure_date_id INT4
null,
                                real_departure_time  TIME                 null,
                                real_arrival_date_id INT4                 null,
                                real_arrival_time    TIME                 null,
                                constraint PK_FLIGHTS_FACT primary key
(flight_id)
);

/*==============================================================*/
/* Table: manufacturers_dim                                     */
/*==============================================================*/
create table manufacturers_dim (
                                    manufacturer_id      SERIAL
not null,
                                    name                 VARCHAR(64)
not null,
                                    constraint PK_MANUFACTURERS_DIM primary
key (manufacturer_id)
);

/*==============================================================*/
/* Index: manufacturer_name_unique                              */
/*==============================================================*/
create unique index manufacturer_name_unique on manufacturers_dim (
                                                                    name
    );

/*==============================================================*/
/* Table: models_descriptions_dim                               */
/*==============================================================*/
create table models_descriptions_dim (
                                        description_id       SERIAL
not null,
                                        name                 VARCHAR(3)
not null,
                                        constraint
PK_MODELS_DESCRIPTIONS_DIM primary key (description_id)
);

/*==============================================================*/
/* Index: model_description_name_unique                         */
/*==============================================================*/
create unique index model_description_name_unique on models_descriptions_dim
(

name
```

```
    );

/*==============================================================*/
/* Table: regions_dim                                           */
/*==============================================================*/
create table regions_dim (
                              region_id             SERIAL                not
null,
                              country_id            INT4                  not
null,
                              code                  VARCHAR(8)            not
null,
                              name                  VARCHAR(100)          not
null,
                              constraint PK_REGIONS_DIM primary key
(region_id)
);

/*==============================================================*/
/* Index: region_code_unique                                    */
/*==============================================================*/
create unique index region_code_unique on regions_dim (
                                                        code
    );

/*==============================================================*/
/* Table: routes_dim                                            */
/*==============================================================*/
create table routes_dim (
                             route_id              SERIAL                not
null,
                             code                  VARCHAR(12)           not
null,
                             constraint PK_ROUTES_DIM primary key (route_id)
);

/*==============================================================*/
/* Index: route_code_unique                                     */
/*==============================================================*/
create unique index route_code_unique on routes_dim (
                                                      code
    );

/*==============================================================*/
/* Table: turbulence_categories_dim                             */
/*==============================================================*/
create table turbulence_categories_dim (
                                          category_id           SERIAL
not null,
                                          name                  VARCHAR(3)
not null,
                                          constraint
PK_TURBULENCE_CATEGORIES_DIM primary key (category_id)
);

/*==============================================================*/
/* Index: turbulence_name                                       */
/*==============================================================*/
create unique index turbulence_name on turbulence_categories_dim (
                                                                   name
    );

alter table aircrafts_dim
    add constraint FK_AIRCRAFT_REFERENCE_DATES_DI foreign key
```

```
(production_date_id)
        references dates_dim (date_id)
        on delete restrict on update cascade;

alter table aircrafts_dim
    add constraint FK_AIRCRAFT_REFERENCE_COUNTRIE foreign key (country_id)
        references countries_dim (country_id)
        on delete restrict on update restrict;

alter table aircrafts_dim
    add constraint FK_AIRCRAFT_REFERENCE_DESIGNAT foreign key (model_id)
        references designators_dim (model_id)
        on delete restrict on update restrict;

alter table aircrafts_dim
    add constraint FK_AIRCRAFT_REFERENCE_AIRLINES foreign key (owner_id)
        references airlines_dim (airline_id)
        on delete restrict on update restrict;

alter table airlines_dim
    add constraint FK_AIRLINES_REFERENCE_COUNTRIE foreign key (country_id)
        references countries_dim (country_id)
        on delete restrict on update restrict;

alter table airports_dim
    add constraint FK_AIRPORTS_REFERENCE_REGIONS_ foreign key (region_id)
        references regions_dim (region_id)
        on delete restrict on update restrict;

alter table airports_dim
    add constraint FK_AIRPORTS_REFERENCE_AIRPORTS foreign key (type_id)
        references airports_types_dim (type_id)
        on delete restrict on update restrict;

alter table countries_dim
    add constraint FK_COUNTRIE_REFERENCE_CONTINEN foreign key (continent_id)
        references continents_dim (continent_id)
        on delete restrict on update restrict;

alter table designators_dim
    add constraint FK_DESIGNAT_REFERENCE_ENGINE_T foreign key
(engine_type_id)
        references engine_types_dim (engine_type_id)
        on delete restrict on update restrict;

alter table designators_dim
    add constraint FK_DESIGNAT_REFERENCE_MODELS_D foreign key
(description_id)
        references models_descriptions_dim (description_id)
        on delete restrict on update restrict;

alter table designators_dim
    add constraint FK_DESIGNAT_REFERENCE_AIRCRAFT foreign key (type_id)
        references aircraft_types_dim (type_id)
        on delete restrict on update restrict;

alter table designators_dim
    add constraint FK_DESIGNAT_REFERENCE_TURBULEN foreign key
(turbulence_category_id)
        references turbulence_categories_dim (category_id)
        on delete restrict on update restrict;

alter table designators_dim
    add constraint FK_DESIGNAT_REFERENCE_MANUFACT foreign key
```

```
(manufacturer_id)
        references manufacturers_dim (manufacturer_id)
        on delete restrict on update restrict;

alter table flights_fact
    add constraint FK_FLIGHTS__REFERENCE_ROUTES_D foreign key (route_id)
        references routes_dim (route_id)
        on delete restrict on update restrict;

alter table flights_fact
    add constraint FK_FLIGHTS__REFERENCE_AIRCRAFT foreign key (aircraft_id)
        references aircrafts_dim (aircraft_id)
        on delete restrict on update cascade;

alter table flights_fact
    add constraint FK_FLIGHTS__REFERENCE_AIRLINES foreign key (airline_id)
        references airlines_dim (airline_id)
        on delete restrict on update cascade;

alter table flights_fact
    add constraint FK_FLIGHTS__AIRPORT_D_AIRPORTS foreign key
(airport_destination_id)
        references airports_dim (airport_id)
        on delete restrict on update cascade;

alter table flights_fact
    add constraint FK_FLIGHTS__FLIGHT_AI_AIRPORTS foreign key
(airport_origin_id)
        references airports_dim (airport_id)
        on delete restrict on update cascade;

alter table flights_fact
    add constraint FK_FLIGHTS__FLIGHT_RE_ARR_t foreign key
(real_arrival_date_id)
        references dates_dim (date_id)
        on delete restrict on update cascade;

alter table flights_fact
    add constraint FK_FLIGHTS__FLIGHT_RE_DATES_DI foreign key
(real_departure_date_id)
        references dates_dim (date_id)
        on delete restrict on update cascade;

alter table flights_fact
    add constraint FK_FLIGHTS__FLIGHT_SC_ARR foreign key
(scheduled_arrival_date_id)
        references dates_dim (date_id)
        on delete restrict on update cascade;

alter table flights_fact
    add constraint FK_FLIGHTS__FLIGHT_SC_DEP foreign key
(scheduled_departure_date_id)
        references dates_dim (date_id)
        on delete restrict on update cascade;

alter table regions_dim
    add constraint FK_REGIONS__REFERENCE_COUNTRIE foreign key (country_id)
        references countries_dim (country_id)
        on delete restrict on update restrict;
```

ДОДАТОК Г (ETL СКРИПТИ)

Г.1. Python-скрипт для завантаження даних у stage зону

```
def load_to_stage():
    conn = psycopg2.connect(host=DB_HOST, database=DB_NAME,
user=DB_USER, password=DB_PASSWORD)
    conn.autocommit = True
    cursor = conn.cursor()

    cursor.execute('TRUNCATE stage.aircrafts;')
    cursor.execute('TRUNCATE stage.airlines;')
    cursor.execute('TRUNCATE stage.airports;')
    cursor.execute('TRUNCATE stage.continents;')
    cursor.execute('TRUNCATE stage.countries;')
    cursor.execute('TRUNCATE stage.designators;')
    cursor.execute('TRUNCATE stage.flights;')
    cursor.execute('TRUNCATE stage.regions;')
    cursor.execute('TRUNCATE stage.dates;')

    cursor.execute(
        f"COPY stage.aircrafts (registration, model_code, model_text,
country_code, production_date, owner_icao) FROM
'{DATA_DIR.joinpath('aircrafts_filtered.csv').absolute()}' DELIMITER
',' CSV HEADER;")

    cursor.execute(
        f"COPY stage.airlines (airline_id, name, alias, code, callsign,
country, active) FROM '{DATA_DIR.joinpath('airlines.csv')}' DELIMITER
',' CSV HEADER;")

    cursor.execute(
        f"COPY stage.airports (code, type, name, latitude, longitude,
elevation, region, gps_code) FROM
'{DATA_DIR.joinpath('airports.csv').absolute()}' DELIMITER ',' CSV
HEADER;")

    cursor.execute(
        f"COPY stage.continents (code, name) FROM
'{DATA_DIR.joinpath('continents.csv').absolute()}' DELIMITER ',' CSV
HEADER;")

    cursor.execute(
        f"COPY stage.countries (code, name, continent) FROM
'{DATA_DIR.joinpath('countries.csv').absolute()}' DELIMITER ',' CSV
HEADER;")

    cursor.execute(
        f"COPY stage.designators (name, description,
turbulence_category, designator, manufacturer, type, engine_count,
engine_type) FROM '{DATA_DIR.joinpath('designators.csv').absolute()}'
DELIMITER ',' CSV HEADER;")

    cursor.execute(
        f"COPY stage.flights (flight_id, route_number,
aircraft_registration, airline_icao, airport_origin,
airport_destination, scheduled_departure, scheduled_arrival,
real_departure, real_arrival) FROM
'{DATA_DIR.joinpath('flights.csv').absolute()}' DELIMITER ',' CSV
HEADER;")

    cursor.execute(
        f"COPY stage.regions (code, name, country) FROM
```

```
'{DATA_DIR.joinpath('regions.csv').absolute()}' DELIMITER ',' CSV
HEADER;")

    cursor.execute(
        f"COPY stage.dates (the_date, weekday, month, year, quarter,
day_of_year, weekend, week_of_year) FROM
'{DATA_DIR.joinpath('dates.csv').absolute()}' DELIMITER ',' CSV
HEADER;")

    cursor.close()
    conn.close()
```

Г.2. SQL-скрипт для завантаження даних зі stage зони до основного сховища даних (окрім вимірів літаків та польотів)

```
-- Designators
INSERT INTO public.manufacturers_dim (name)
SELECT DISTINCT manufacturer
FROM stage.designators
WHERE manufacturer IS NOT NULL
ON CONFLICT (name) DO NOTHING;

INSERT INTO public.turbulence_categories_dim (name)
SELECT DISTINCT turbulence_category
FROM stage.designators
WHERE turbulence_category IS NOT NULL
ON CONFLICT (name) DO NOTHING;

INSERT INTO public.aircraft_types_dim (name)
SELECT DISTINCT type
FROM stage.designators
WHERE type IS NOT NULL
ON CONFLICT (name) DO NOTHING;

INSERT INTO public.models_descriptions_dim (name)
SELECT DISTINCT description
FROM stage.designators
WHERE description IS NOT NULL
ON CONFLICT (name) DO NOTHING;

INSERT INTO public.engine_types_dim (name)
SELECT DISTINCT engine_type
FROM stage.designators
WHERE designators.engine_type IS NOT NULL
ON CONFLICT (name) DO NOTHING;


-- Geo zones
INSERT INTO public.continents_dim (code, name)
SELECT DISTINCT ON (stage.continents.code) code, name
FROM stage.continents
ORDER BY code, stage.continents.continent_id DESC
ON CONFLICT (code) DO UPDATE SET name = excluded.name;

INSERT INTO public.countries_dim (code, continent_id, name)
SELECT DISTINCT ON (stage.countries.code) stage.countries.code,

public.continents_dim.continent_id,
                                          stage.countries.name
FROM stage.countries
         INNER JOIN public.continents_dim ON stage.countries.continent
= public.continents_dim.code
```

```
ORDER BY code, stage.countries.country_id DESC
ON CONFLICT (code) DO UPDATE SET continent_id = excluded.continent_id,
                                 name         = excluded.name;

INSERT INTO public.regions_dim (code, country_id, name)
SELECT DISTINCT ON (stage.regions.code) stage.regions.code,

public.countries_dim.country_id,
                                        stage.regions.name
FROM stage.regions
         INNER JOIN public.countries_dim ON stage.regions.country =
public.countries_dim.code
ORDER BY code, stage.regions.region_id DESC
ON CONFLICT (code) DO UPDATE SET country_id = excluded.country_id,
                                 name       = excluded.name;


-- Airports
INSERT INTO public.airports_types_dim (name)
SELECT DISTINCT stage.airports.type
FROM stage.airports
ON CONFLICT (name) DO NOTHING;

INSERT INTO public.airports_dim (code, type_id, name, latitude,
longitude, elevation, region_id, gps_code)
SELECT DISTINCT ON (stage.airports.code) stage.airports.code,

public.airports_types_dim.type_id,
                                        stage.airports.name,
                                        stage.airports.latitude,
                                        stage.airports.longitude,
                                        stage.airports.elevation,
                                        public.regions_dim.region_id,
                                        stage.airports.gps_code
FROM stage.airports
         LEFT JOIN public.airports_types_dim ON stage.airports.type =
public.airports_types_dim.name
         LEFT JOIN public.regions_dim ON stage.airports.region =
public.regions_dim.code
ORDER BY stage.airports.code, stage.airports.airport_id DESC
ON CONFLICT (code) DO UPDATE SET type_id   = excluded.type_id,
                                 name      = excluded.name,
                                 latitude  = excluded.latitude,
                                 longitude = excluded.longitude,
                                 elevation = excluded.elevation,
                                 region_id = excluded.region_id,
                                 gps_code  = excluded.gps_code;


-- Airlines
INSERT INTO public.airlines_dim (code, airline_id, name, alias,
callsign, country_id, active)
SELECT DISTINCT ON (stage.airlines.code) stage.airlines.code,
                                        stage.airlines.airline_id,
                                        stage.airlines.name,
                                        stage.airlines.alias,
                                        stage.airlines.callsign,

public.countries_dim.country_id,
                                        stage.airlines.active
FROM stage.airlines
         LEFT JOIN public.countries_dim ON stage.airlines.country =
public.countries_dim.name
ORDER BY stage.airlines.code, stage.airlines.airline_pk_ser DESC
```

```
ON CONFLICT (airline_id) DO UPDATE SET alias      = excluded.alias,
                                       code       = excluded.code,
                                       callsign   = excluded.callsign,
                                       country_id =
excluded.country_id,
                                       active     = excluded.active;


-- Routes
INSERT INTO public.routes_dim (code)
SELECT DISTINCT stage.flights.route_number
FROM stage.flights
ON CONFLICT (code) DO NOTHING;

-- Dates
INSERT INTO public.dates_dim (the_date, weekday, month, year, quarter,
day_of_year, weekend, week_of_year)
SELECT DISTINCT ON (stage.dates.the_date) stage.dates.the_date,
                                         stage.dates.weekday,
                                         stage.dates.month,
                                         stage.dates.year,
                                         stage.dates.quarter,
                                         stage.dates.day_of_year,
                                         stage.dates.weekend,
                                         stage.dates.week_of_year
FROM stage.dates
ORDER BY stage.dates.the_date, stage.dates.date_id DESC
ON CONFLICT (the_date) DO UPDATE SET weekend = excluded.weekend;


-- Designators
INSERT INTO designators_dim (name, designator, description_id,
turbulence_category_id, manufacturer_id, type_id,
                             engine_count, engine_type_id)
SELECT DISTINCT ON (stage.designators.name,
stage.designators.designator) stage.designators.name,

stage.designators.designator,

public.models_descriptions_dim.description_id,

public.turbulence_categories_dim.category_id,

public.manufacturers_dim.manufacturer_id,

public.aircraft_types_dim.type_id,

stage.designators.engine_count,

public.engine_types_dim.engine_type_id
FROM stage.designators
         LEFT JOIN public.models_descriptions_dim ON
public.models_descriptions_dim.name = stage.designators.description
         LEFT JOIN public.turbulence_categories_dim
                   ON public.turbulence_categories_dim.name =
stage.designators.turbulence_category
         LEFT JOIN public.manufacturers_dim ON
public.manufacturers_dim.name = stage.designators.manufacturer
         LEFT JOIN public.aircraft_types_dim ON
public.aircraft_types_dim.name = stage.designators.type
         LEFT JOIN public.engine_types_dim ON
public.engine_types_dim.name = stage.designators.engine_type
ORDER BY stage.designators.name, stage.designators.designator,
stage.designators.model_id DESC
```

```
ON CONFLICT (name, designator) DO UPDATE SET description_id         =
excluded.description_id,
                                              turbulence_category_id =
excluded.turbulence_category_id,
                                              manufacturer_id        =
excluded.manufacturer_id,
                                              type_id                =
excluded.type_id,
                                              engine_count           =
excluded.engine_count,
                                              engine_type_id         =
excluded.engine_type_id;

DROP TABLE IF EXISTS temp_aircrafts;
```

Г.3. Python-скрипт для створення нової таблиці з літаками та додаванням до неї стовпця назви моделі з таблиці моделей

```
def link_models(conn: str):
    aircrafts = pd.read_sql_table('aircrafts', conn, schema='stage',
index_col='aircraft_id')
    designators = pd.read_sql_table('designators_dim', conn,
index_col='model_id')
    results = []
    progress_bar = tqdm.tqdm(list(aircrafts.iterrows()))
    for row in progress_bar:
        _, model_code, model_text, *_ = row[1]
        progress_bar.set_description(f'{model_code} | {model_text}')
        progress_bar.refresh()
        variants = designators.query(f'designator ==
"{model_code}"').name
        result = None
        if model_text and not variants.empty:
            ratios = np.array([SequenceMatcher(None, model_text,
variant).ratio() for variant in variants])
            result = variants.iloc[ratios.argmax()]
        results.append(result)
    aircrafts['linked_model_text'] = results
    return aircrafts
```

Г.4. SQL-скрипт для завантаження даних зі стейдж зони до основного сховища даних (для виміру літаків та польотів)

```
-- Aircrafts
SELECT *
FROM temp_aircrafts;
INSERT INTO public.aircrafts_dim (registration, model_id, model_text,
country_id, production_date_id, owner_id)
SELECT DISTINCT ON (temp_aircrafts.registration)
temp_aircrafts.registration,

public.designators_dim.model_id,

temp_aircrafts.model_text,

public.countries_dim.country_id,
```

```
public.dates_dim.date_id,

public.airlines_dim.airline_id
FROM temp_aircrafts
         LEFT JOIN public.designators_dim ON temp_aircrafts.model_code
= public.designators_dim.designator AND

temp_aircrafts.linked_model_text = public.designators_dim.name
         LEFT JOIN public.countries_dim ON temp_aircrafts.country_code
= public.countries_dim.code
         LEFT JOIN public.dates_dim ON temp_aircrafts.production_date =
public.dates_dim.the_date
         LEFT JOIN public.airlines_dim ON temp_aircrafts.owner_icao =
public.airlines_dim.code
ORDER BY temp_aircrafts.registration, temp_aircrafts.aircraft_id DESC
ON CONFLICT (registration) DO UPDATE SET model_id           =
excluded.model_id,
                                         model_text         =
excluded.model_text,
                                         country_id         =
excluded.country_id,
                                         production_date_id =
excluded.production_date_id,
                                         owner_id           =
excluded.owner_id;
DROP TABLE IF EXISTS temp_aircrafts;

INSERT INTO public.flights_fact (flight_id, route_id, aircraft_id,
airline_id, airport_origin_id,
                                 airport_destination_id,
scheduled_departure_date_id, scheduled_departure_time,
                                 scheduled_arrival_date_id,
scheduled_arrival_time, real_departure_date_id,
                                 real_departure_time,
real_arrival_date_id, real_arrival_time)
SELECT flight_id,
       public.routes_dim.route_id,
       public.aircrafts_dim.aircraft_id,
       public.airlines_dim.airline_id,
       ap1.airport_id,
       ap2.airport_id,
       d1.date_id,
       scheduled_departure::TIME,
       d2.date_id,
       scheduled_arrival::TIME,
       d3.date_id,
       real_departure::TIME,
       d4.date_id,
       real_arrival::TIME
FROM (SELECT DISTINCT ON (stage.flights.flight_id)
stage.flights.flight_id,

stage.flights.route_number,

stage.flights.aircraft_registration,

stage.flights.airline_icao,

stage.flights.airport_origin,

stage.flights.airport_destination,

TO_TIMESTAMP(stage.flights.scheduled_departure) AS scheduled_departure,
```

```
    TO_TIMESTAMP(stage.flights.scheduled_arrival)   AS scheduled_arrival,

    TO_TIMESTAMP(stage.flights.real_departure)      AS real_departure,

    TO_TIMESTAMP(stage.flights.real_arrival)        AS real_arrival
          FROM stage.flights
          ORDER BY flight_id DESC) AS flights_subq
             LEFT JOIN public.routes_dim ON route_number =
public.routes_dim.code
             LEFT JOIN public.aircrafts_dim ON aircraft_registration =
public.aircrafts_dim.registration
             LEFT JOIN public.airlines_dim ON airline_icao =
public.airlines_dim.code
             LEFT JOIN public.airports_dim ap1 ON airport_origin = ap1.code
             LEFT JOIN public.airports_dim ap2 ON airport_destination =
ap2.code
             LEFT JOIN public.dates_dim d1 ON scheduled_departure::date =
d1.the_date
             LEFT JOIN public.dates_dim d2 ON scheduled_arrival::date =
d2.the_date
             LEFT JOIN public.dates_dim d3 ON real_departure::date =
d3.the_date
             LEFT JOIN public.dates_dim d4 ON real_arrival::date =
d4.the_date
    ON CONFLICT (flight_id) DO UPDATE SET route_id                    =
    excluded.route_id,
                                          aircraft_id                 =
    excluded.aircraft_id,
                                          airline_id                  =
    excluded.airline_id,
                                          airport_origin_id           =
    excluded.airport_origin_id,
                                          airport_destination_id      =
    excluded.airport_destination_id,
                                          scheduled_departure_date_id =
    excluded.scheduled_departure_date_id,
                                          scheduled_departure_time    =
    excluded.scheduled_departure_time,
                                          scheduled_arrival_date_id   =
    excluded.scheduled_arrival_date_id,
                                          scheduled_arrival_time      =
    excluded.scheduled_arrival_time,
                                          real_departure_date_id      =
    excluded.real_departure_date_id,
                                          real_departure_time         =
    excluded.real_departure_time,
                                          real_arrival_date_id        =
    excluded.real_arrival_date_id,
                                          real_arrival_time           =
    excluded.real_arrival_time;
```

Г.5. Python-скрипт для повного завантаження даних

```
def etl():
    load_to_stage()
    stage_to_fact()
```