



Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Predmet / Subject

– Databázové systémy / Database systems –

- Dokumentácia / Documentation -

Zadanie č.2

Ak. Rok / Academic term: 2022/2023, letný semester

Cvičiaci / Instructors:

Ing. Jakub Dubec

Študent / Student:

Adam Grík



Bratislava, 2023.

Obsah

1	Úvod	- 2 -
2	Endpoint – zoznam cestujúcich	- 2 -
3	Endpoint – detail letu	- 3 -
4	Endpoint – neskoré odlety	- 4 -
5	Endpoint – linky, ktoré obslúžili najviac pasažierov	- 5 -
6	Endpoint – naplánované linky	- 6 -
7	Endpoint – všetky destinácie zo zadaného letiska	- 7 -
8	Endpoint – vyťaženosť letov pre konkrétnu linku	- 8 -
9	Endpoint – priemerná vyťaženosť linky pre jednotlivé dni v týždni	- 9 -

1 Úvod

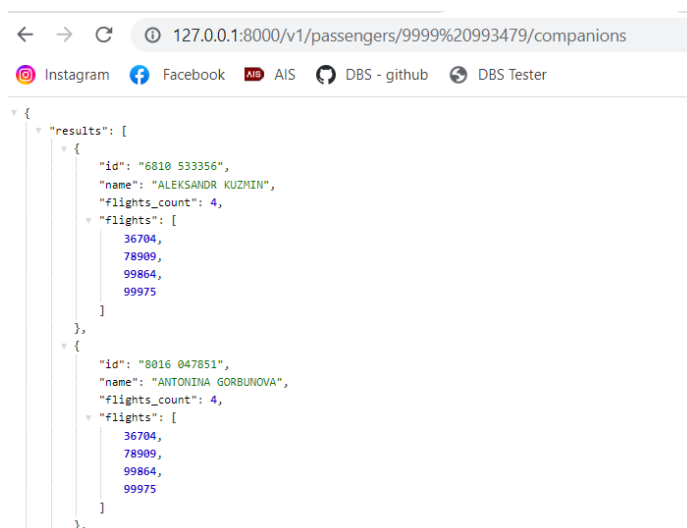
Úlohou tohto zadania bolo implementovať HTTP end-pointy podľa popisu zo zadania. Aplikácia číta dáta zo zadaného datasetu flights.sql. Riešenie implementujem v programovacom jazyku Python. Obsahom tejto dokumentácie je popis a zobrazenie výsledkov z jednotlivých SQL dopytov.

2 Endpoint – zoznam cestujúcich

```
SELECT
  t.passenger_id,
  t.passenger_name,
  COUNT(DISTINCT second_boarding_passes.flight_id) as flights_count,
  ARRAY_AGG(DISTINCT second_boarding_passes.flight_id ORDER BY second_boarding_passes.flight_id ASC) as flights
FROM bookings.tickets t
JOIN bookings.boarding_passes first_boarding_passes ON t.ticket_no = first_boarding_passes.ticket_no
JOIN (SELECT DISTINCT flight_id
      FROM bookings.boarding_passes bp
      JOIN bookings.tickets t ON t.ticket_no = bp.ticket_no
      WHERE t.passenger_id = %s)
as second_boarding_passes ON first_boarding_passes.flight_id = second_boarding_passes.flight_id
WHERE t.passenger_id != %s|
GROUP BY t.passenger_id, t.passenger_name
ORDER BY flights_count DESC, t.passenger_id ASC, flights ASC
```

V tomto endpointe si robím SELECT najprv na stĺpec *passenger_id*, *passenger_name*, potom pomocou funkcie COUNT spočítam počet letov. Posledné si “selectujem” pole všetkých *flight_id* z “joinutej” tabuľky *second_boarding_passes*. Následne vykonávam dva krát JOIN. Prvý je jednoduchý, jednoducho si iba “joinem” tabuľku *boarding_passes* pomocou stĺpca *ticket_no*. V druhom JOIN príkaze používam subquery, ktorá zabezpečuje to, že mi vytiahne všetky *flight_id*, ktoré absolvoval pasažier s daným *passenger_id*, vytiahnutým z URL adresy. Tento druhý JOIN spájam s tabuľkou z prvého JOIN-u, aby som získal všetky lety ktoré cestujúci absolvoval. Následne už vykonávam iba podmienku nato aby sa mi vo výsledku nevyskytol ten cestujúci, ktorý je zadaný v danej URL adrese. Ako posledné vykonávam už len potrebné GROUP BY a ORDER BY.

Časť JSON výstupu z daného endpointu.



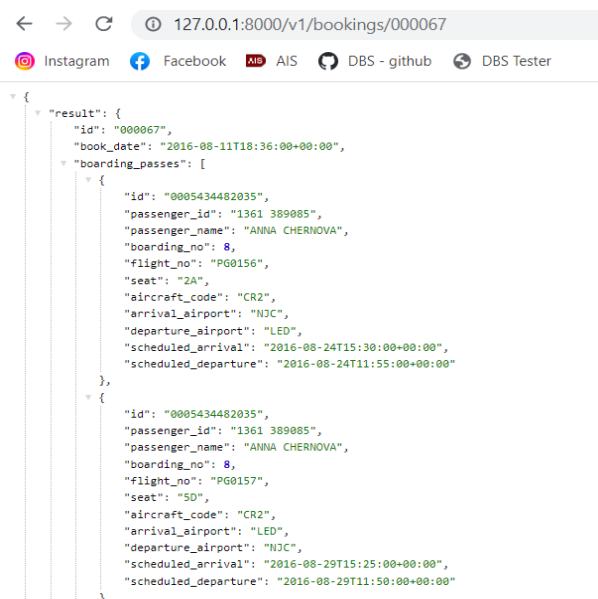
3 Endpoint – detail letu

```
SELECT
    bookings.book_ref,
    bookings.book_date,
    tickets.ticket_no,
    tickets.passenger_id,
    tickets.passenger_name,
    boarding_passes.boarding_no,
    flights.flight_no,
    boarding_passes.seat_no,
    flights.aircraft_code,
    flights.arrival_airport,
    flights.departure_airport,
    flights.scheduled_arrival,
    flights.scheduled_departure
FROM bookings.bookings
JOIN bookings.tickets ON (tickets.book_ref = bookings.book_ref)
JOIN bookings.boarding_passes ON (tickets.ticket_no = boarding_passes.ticket_no)
JOIN bookings.flights ON (boarding_passes.flight_id = flights.flight_id)

WHERE tickets.book_ref = %s
ORDER BY ticket_no ASC, boarding_no ASC
```

Ako prvé si v SELECT-e vytiahnem všetky potrebné stĺpce, ktoré potrebujem na splnenie tohto endpointu. Následne si spravím JOIN na všetky tabuľky ktoré potrebujem, a to tak že, tabuľka *bookings* má spoločný stĺpec *book_ref* v tabuľke *tickets* aby som si z nej mohol vytiahnuť stĺpce *ticket_no*, *passenger_id* a *passenger_name*. Následne si spravím JOIN na tabuľku *boarding_passes*, z ktorej potrebujem stĺpec *boarding_no* a *seat_no* pomocou stĺpca *ticket_no*. Ako posledný JOIN si spravím na tabuľku *flights* z ktorej potrebujem vytiahnuť údaje ohľadom daného letu, pomocou stĺpca *flight_id*. Nakoniec ešte pridám podmienku WHERE, kde zadávam hodnotu *booking_ref*, ktorú získavam z danej URL a už to len vďaka ORDER BY usporiadam na základe *ticket_no* a *boarding_no* ako bolo požadované v zadaní.

Časť JSON vystúpu z daného endpointu.

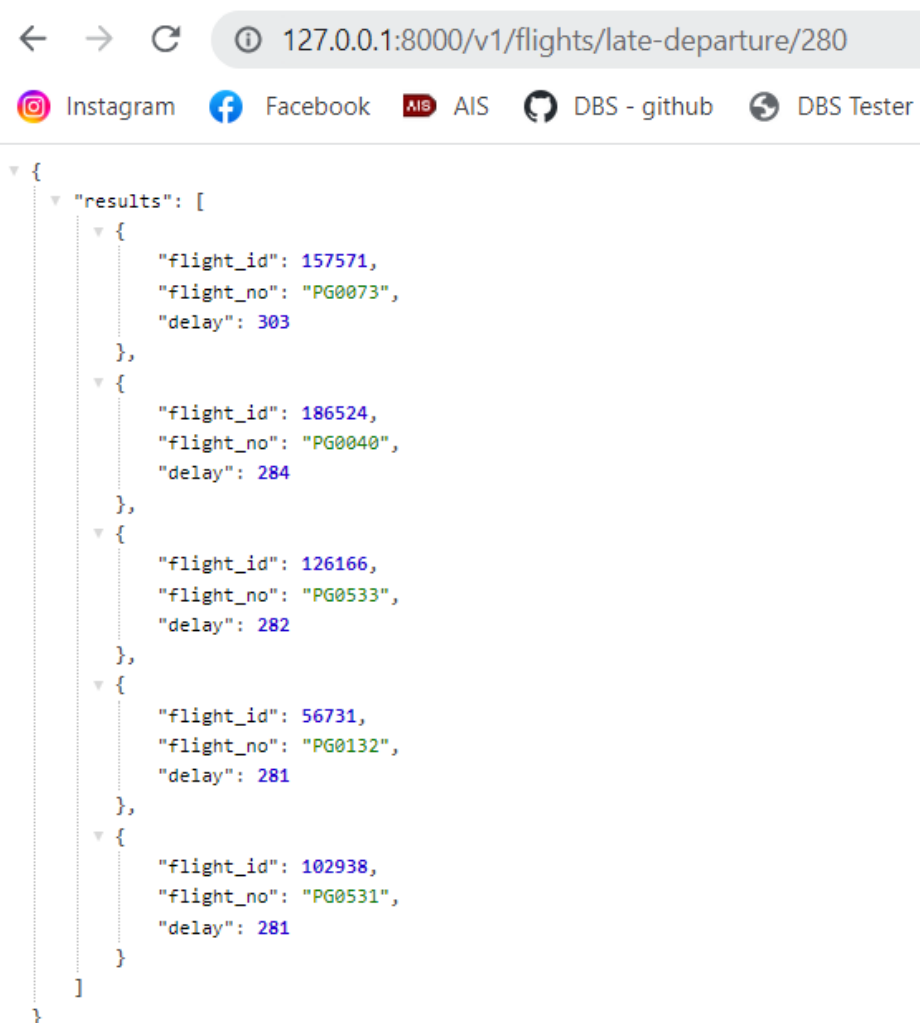


4 Endpoint – neskoré odlety

```
SELECT
    flight_id,
    flight_no,
    ((extract (epoch from ((actual_departure - scheduled_departure)/60))))::integer as delays
FROM bookings.flights
WHERE (((extract (epoch from ((actual_departure - scheduled_departure)/60))))::integer) > %s
ORDER BY delays DESC, flight_id ASC
```

V tomto endpointe si selectujem všetko iba z tabuľky flights, čiže som nepotreboval žiaden JOIN. Na výpočet meškanie využívam funkcie *extract* a *epoch* v ktorých odpočítavam dátum naplánovného priletu od dátumu aktuálneho priletu a následne to ešte delím číslom 60 aby som získal hodiny. Nakoniec pridám ešte podmienku vďaka ktorej si vyselectujem iba tie lety ktoré meškali viac minút ako hodnota ktorá je zadaná v URL adrese, a takisto už len zoradím na základe meškania od najväčšieho a ak majú rovnaké meškanie tak zoradím podľa *flight_id*.

JSON výstup z tohto http endpointu.



5 Endpoint – linky, ktoré obslúžili najviac pasažierov

```
SELECT
    flights.flight_no,
    COUNT(boarding_passes) as counter
FROM bookings.flights
JOIN bookings.boarding_passes ON (flights.flight_id = boarding_passes.flight_id)
WHERE flights.status = 'Arrived'
GROUP BY flight_no
ORDER BY counter DESC
LIMIT %s|
```

V SELECTE tohto endpointu si ako prvé vyberiem stĺpec *flight_no* z tabuľky *flights*, ako ďalšie vďaka funkcii COUNT spočítam ako keby všetky tabuľky *boarding_passes*, ktoré sa viažu na daný let, čo mi zabezpečí to že dostanem počet pasažierov. Následne si musím spraviť JOIN na danú *boarding_passes* tabuľku pomocou *flight_id*. Ako ďalšie pridávam podmienku WHERE, ktorá mi zabezpečuje to, aby som počítal pasažierov iba z tých letov, ktoré už sú uskutočnené a nie len naplánované. Následne si spravím GROUP BY podľa *flight_no*, usporiadam podľa najväčšieho počtu prepravených pasažierov a nakoniec ešte pridám LIMIT, ktorý získavam z URL adresy.

Príklad JSON výstupu z tohto endpointu.

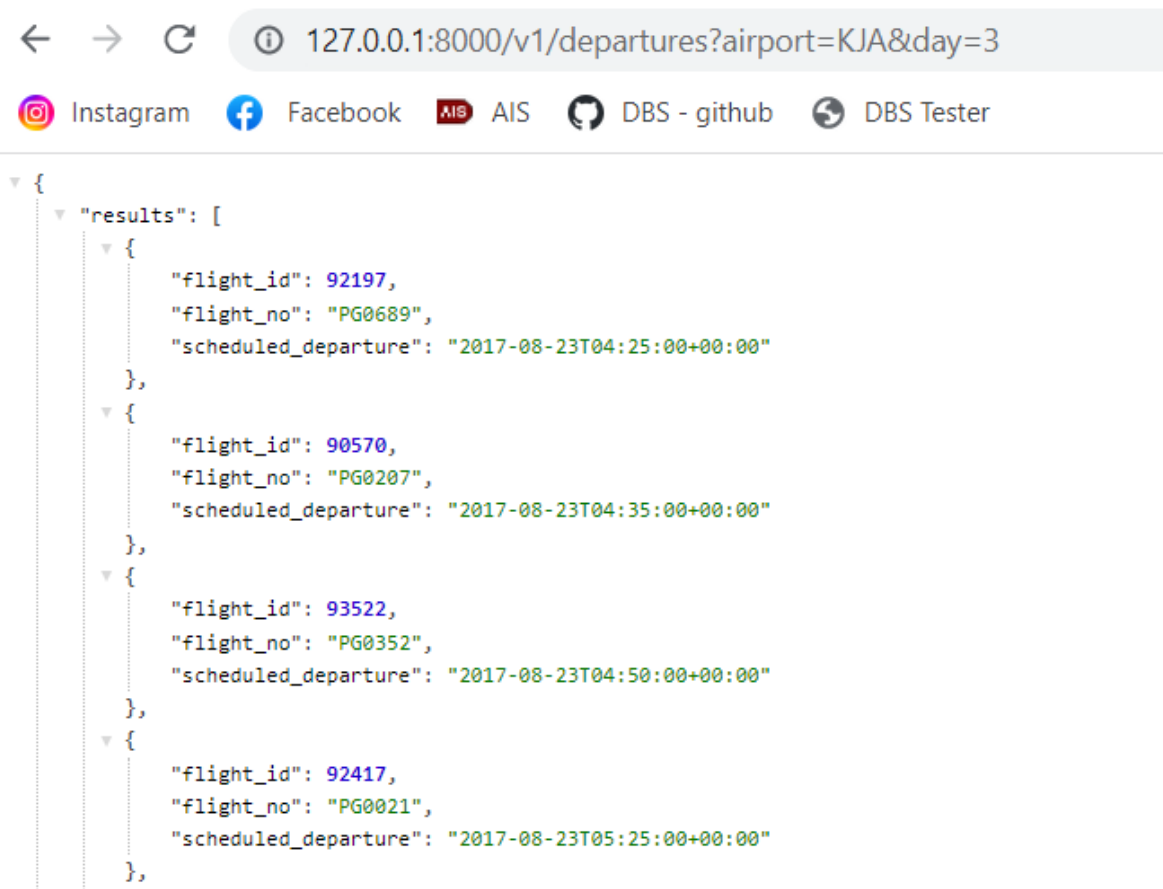


6 Endpoint – naplánované linky

```
SELECT
    flight_id,
    flight_no,
    scheduled_departure
FROM bookings.flights
WHERE (flights.status = 'Scheduled') AND(extract(dow from scheduled_departure) = %s)
AND(departure_airport= %s)
ORDER BY scheduled_departure, flight_id ASC'
```

V tomto endpointe si najprv spravím SELECT na všetky potrebné stĺpce, a to *flight_id*, *flight_no* a *scheduled_departure*. Všetky stĺpce sa nachádzajú v jednej tabuľke *flights*, takže nepotrebujem žiaden JOIN. Následne si spravím niekoľko podmienok, ako prvá podmienka je to, že let musí byť ešte len naplánovaný, následne si vďaka funkcii *extract* vytiahnem z dátumu *scheduled_departure* číslo dňa v týždni, ktorý porovnávam z hodnotou získanou z URL adresy a ako posledné si ešte porovnávam *departure_airport*, ktorého hodnotu tiež porovnávam s hodnotou získanou z URL adresy. Ako posledné už len usporiadam podľa dátumu a ak majú dátum rovnaký tak usporiadam podľa nižšieho *flight_id*.

Časť JSON výstupu daného endpointu.



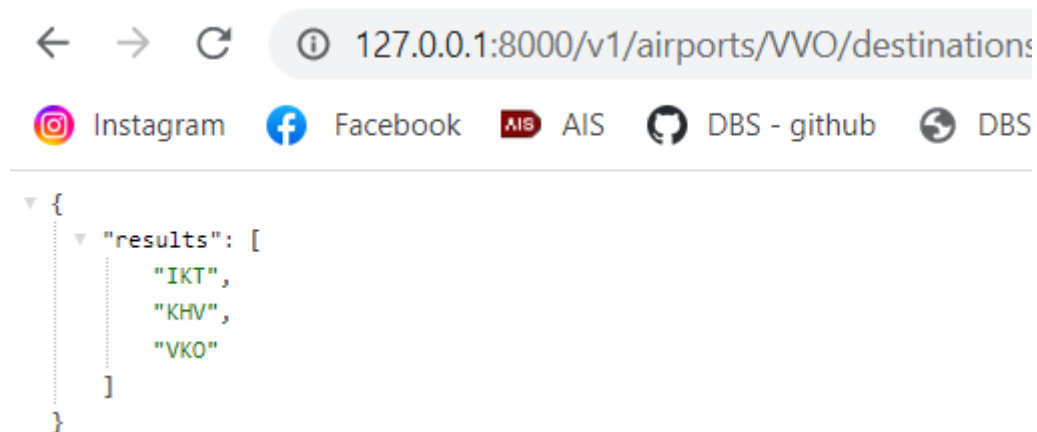
```
{
  "results": [
    {
      "flight_id": 92197,
      "flight_no": "PG0689",
      "scheduled_departure": "2017-08-23T04:25:00+00:00"
    },
    {
      "flight_id": 90570,
      "flight_no": "PG0207",
      "scheduled_departure": "2017-08-23T04:35:00+00:00"
    },
    {
      "flight_id": 93522,
      "flight_no": "PG0352",
      "scheduled_departure": "2017-08-23T04:50:00+00:00"
    },
    {
      "flight_id": 92417,
      "flight_no": "PG0021",
      "scheduled_departure": "2017-08-23T05:25:00+00:00"
    }
  ]
}
```

7 Endpoint – všetky destinácie zo zadaného letiska

```
SELECT
  DISTINCT arrival_airport
FROM bookings.flights
WHERE departure_airport = %s
ORDER BY arrival_airport ASC
```

V tomto endpointe si robím select na stĺpec *arrival_airport* z tabuľky *flights*, kde pridávam iba podmienky že stĺpec *departure_airport* sa rovná hodnote zadanej v URL adrese.

Príklad JSON výstupu.

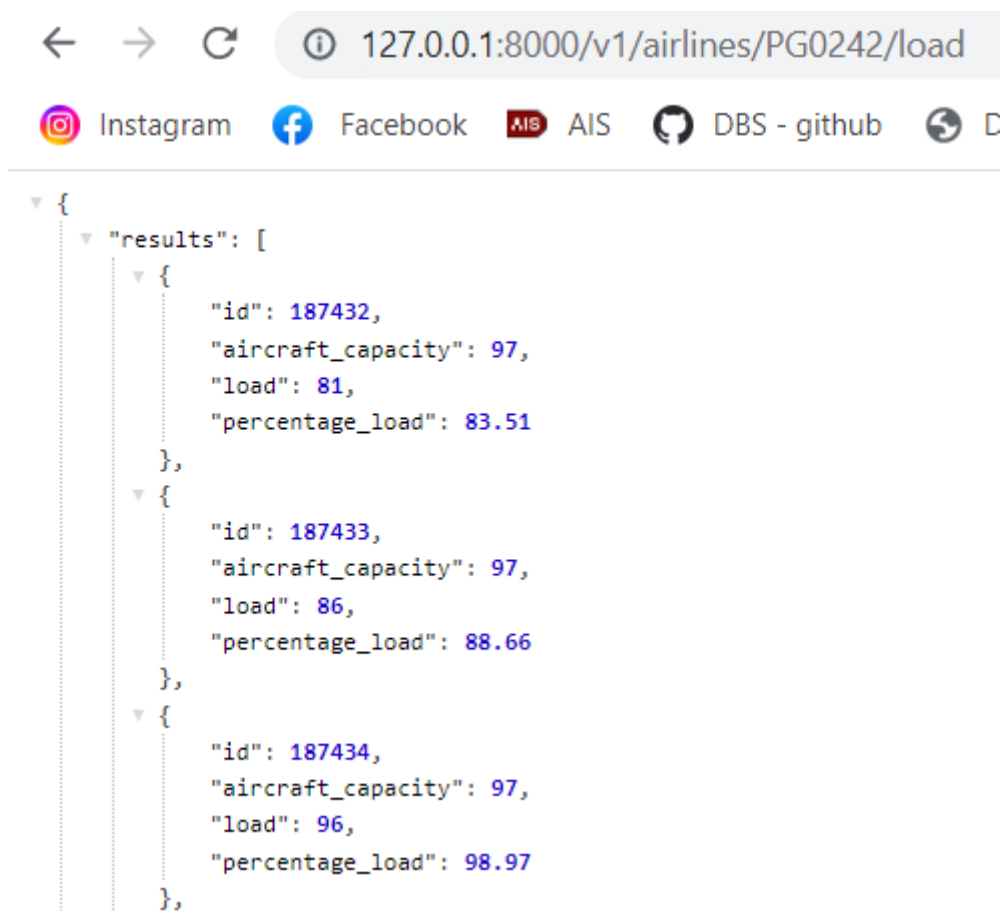


8 Endpoint – vyt'áženosť letov pre konkrétnu linku

```
SELECT
  flights.flight_id,
  COUNT(DISTINCT(seats.seat_no)) as aircraft_capacity,
  COUNT(DISTINCT(ticket_flights.ticket_no)) as "load",
  ROUND(100.0 * (COUNT(DISTINCT(ticket_flights.ticket_no))) / (COUNT(DISTINCT(seats.seat_no))),2) as percentage
FROM bookings.flights
JOIN bookings.seats ON(flights.aircraft_code = seats.aircraft_code)
JOIN bookings.ticket_flights ON(flights.flight_id = ticket_flights.flight_id)
WHERE flight_no = %s
GROUP BY flights.flight_id
ORDER BY flights.flight_id
```

V tomto endpointe si spravím SELECT na *flight_id*, následne si spočítam pomocou COUNT a stĺpca *seat_no* v tabuľke *seats* kapacitu lietadla. Obsadenosť lietadla si vypočítam tiež pomocou funkcie COUNT a stĺpca *ticket_no* v tabuľke *ticket_flights*. Percentuálnu vyt'áženosť daného letu vypočítam tak, že si videlím obsadenosť lietadla s kapacitou lietadla. Následne si ešte spravím dva krát JOIN na potrebné tabuľky, pridám podmienku WHERE kde porovnávam stĺpec *flight_no* s hodnotou zadanou v URL adrese. Ako posledné už len spravím GROUP BY, a usporiadam podľa *flight_id* od najmenšieho.

Časť JSON výstupu daného endpointu.



9 Endpoint – priemerná vyt'áženosť linky pre jednotlivé dni v týždni

```
SELECT
    flights.flight_no,
    ROUND(AVG(CASE WHEN EXTRACT(dow FROM flights.scheduled_departure) = 1 THEN tf_count * 100.0 / seats_count END), 2) AS monday,
    ROUND(AVG(CASE WHEN EXTRACT(dow FROM flights.scheduled_departure) = 2 THEN tf_count * 100.0 / seats_count END), 2) AS tuesday,
    ROUND(AVG(CASE WHEN EXTRACT(dow FROM flights.scheduled_departure) = 3 THEN tf_count * 100.0 / seats_count END), 2) AS wednesday,
    ROUND(AVG(CASE WHEN EXTRACT(dow FROM flights.scheduled_departure) = 4 THEN tf_count * 100.0 / seats_count END), 2) AS thursday,
    ROUND(AVG(CASE WHEN EXTRACT(dow FROM flights.scheduled_departure) = 5 THEN tf_count * 100.0 / seats_count END), 2) AS friday,
    ROUND(AVG(CASE WHEN EXTRACT(dow FROM flights.scheduled_departure) = 6 THEN tf_count * 100.0 / seats_count END), 2) AS saturday,
    ROUND(AVG(CASE WHEN EXTRACT(dow FROM flights.scheduled_departure) = 0 THEN tf_count * 100.0 / seats_count END), 2) AS sunday
FROM bookings.flights
JOIN (SELECT flight_id,
        COUNT(*) AS tf_count
        FROM bookings.ticket_flights
        GROUP BY flight_id) AS tf ON tf.flight_id = flights.flight_id
JOIN (SELECT aircraft_code,
        COUNT(*) AS seats_count
        FROM bookings.seats GROUP BY aircraft_code) AS s ON s.aircraft_code = flights.aircraft_code
WHERE flights.flight_no = %s
GROUP BY flights.flight_no;
```

V tomto endpointe si spravím SELECT najprv na *flight_no*, následne si postupne vďaka funkciám ROUND, AVG, EXTRACT vypočítam priemernú vyt'áženosť letu pre každý deň zvlášť vďaka podmienkam a vďaka premenným *tf_count* a *seats_count*, ktoré si získavam z jednotlivých subqueries nižšie. Následne v dvoch JOIN-och a subqueries, získavam najprv počet predaných leteniek, ktoré sa viažu k danému *flight_id*, a tak isto v ďalšej subquery získavam vďaka spočítaniu všetkých záznamov z tabuliek *seats*, ktoré sa viažu na daný *aircraft_code*. Ako posledné už len spravím podmienku kde porovnávam hodnotu *flight_no* s hodnotou získanou z URL adresy.

Príklad JSON výstupu pre daný endpoint.

←
→
↻

i
127.0.0.1:8000/v1/airlines/PG0242/load-week

Instagram

Facebook

AIS

DBS - github

DBS Tester

```
{
  "result": {
    "flight_no": "PG0242",
    "monday": 81.17,
    "tuesday": 82.66,
    "wednesday": 84.81,
    "thursday": 79.8,
    "friday": 82.25,
    "saturday": 80.25,
    "sunday": 82.88
  }
}
```