ZAPYTANIA MARIADB

1. Airports

**Zwykłe zapytania**

Wyświetlenie wszystkich lotów z opóźnieniem przylotu większym niż 60 minut:

SELECT \*

FROM flights

WHERE ARRIVAL\_DELAY > 60;

Wyświetlenie nazw linii lotniczych, których kod IATA to AA:

SELECT AIRLINE

FROM airlines

WHERE IATA\_CODE = 'AA';

Wyświetlenie lotnisk znajdujących się w stanie CA (Kalifornia):

SELECT AIRPORT, CITY

FROM airports

WHERE STATE = 'CA';

**Zapytania z grupowaniem**

Wyświetlenie średniego opóźnienia przylotu dla każdego dnia tygodnia:

SELECT DAY\_OF\_WEEK, AVG(ARRIVAL\_DELAY) AS avg\_arrival\_delay

FROM flights

GROUP BY DAY\_OF\_WEEK;

Zliczenie liczby lotów anulowanych z każdej przyczyny:

SELECT CANCELLATION\_REASON, COUNT(\*) AS cancel\_count

FROM flights

WHERE CANCELLED = 1

GROUP BY CANCELLATION\_REASON;

Wyświetlenie liczby lotów każdego dnia dla linii lotniczej UA:

SELECT YEAR, MONTH, DAY, COUNT(\*) AS flight\_count

FROM flights

WHERE AIRLINE = 'UA'

GROUP BY YEAR, MONTH, DAY;

**Zapytania z join’ami**

Wyświetlenie nazw linii lotniczych oraz liczby lotów dla każdej z nich:

SELECT a.AIRLINE, COUNT(f.FLIGHT\_NUMBER) AS flight\_count

FROM flights f

JOIN airlines a ON f.AIRLINE = a.IATA\_CODE

GROUP BY a.AIRLINE;

Wyświetlenie nazw miast dla lotów, które miały opóźnienie przylotu większe niż 100 minut, łącząc dane z lotniska początkowego:

SELECT ap.CITY AS origin\_city, f.ARRIVAL\_DELAY

FROM flights f

JOIN airports ap ON f.ORIGIN\_AIRPORT = ap.IATA\_CODE

WHERE f.ARRIVAL\_DELAY > 100;

Wyświetlenie nazw lotnisk, do których miały miejsce loty opóźnione o więcej niż 2 godziny, wraz z nazwami linii lotniczych:

SELECT ap.AIRPORT AS destination\_airport, a.AIRLINE AS airline\_name

FROM flights f

JOIN airlines a ON f.AIRLINE = a.IATA\_CODE

JOIN airports ap ON f.DESTINATION\_AIRPORT = ap.IATA\_CODE

WHERE f.ARRIVAL\_DELAY > 120;

**Zapytania z podzapytaniami**

Wyświetlenie lotów z największym opóźnieniem przylotu dla każdego dnia:

SELECT \*

FROM flights f1

WHERE ARRIVAL\_DELAY = (

SELECT MAX(ARRIVAL\_DELAY)

FROM flights f2

WHERE f1.YEAR = f2.YEAR AND f1.MONTH = f2.MONTH AND f1.DAY = f2.DAY

);

Wyświetlenie nazw linii lotniczych, które miały więcej niż 1000 opóźnionych lotów:

SELECT AIRLINE

FROM airlines

WHERE IATA\_CODE IN (

SELECT AIRLINE

FROM flights

WHERE ARRIVAL\_DELAY > 0

GROUP BY AIRLINE

HAVING COUNT(\*) > 1000

);

Wyświetlenie miast, z których najwięcej lotów zostało anulowanych:

SELECT CITY

FROM airports

WHERE IATA\_CODE IN (

SELECT ORIGIN\_AIRPORT

FROM flights

WHERE CANCELLED = 1

GROUP BY ORIGIN\_AIRPORT

ORDER BY COUNT(\*) DESC

LIMIT 1

);

1. Bikes

**Zwykłe zapytania**

**Wybór wszystkich podróży, które trwały dłużej niż 30 minut**:

SELECT \* FROM TripUsers

WHERE tripduration > 30 \* 60;

Znajdź wszystkie podróże, które zakończyły się na stacji "Central Park":

SELECT \* FROM TripUsers

WHERE end\_station\_name = 'Central Park';

Wybierz wszystkie unikalne typy użytkowników, którzy skorzystali z usługi:

SELECT DISTINCT usertype FROM TripUsers;

**Zapytania z grupowaniem**

Policz liczbę podróży dla każdego typu użytkownika:

SELECT usertype, COUNT(\*) AS trip\_count

FROM TripUsers

GROUP BY usertype;

Policz całkowity czas podróży dla każdego roku urodzenia użytkowników:

SELECT birth\_year, SUM(tripduration) AS total\_tripduration

FROM TripUsers

GROUP BY birth\_year;

Oblicz średnią długość podróży w zależności od płci użytkowników:

SELECT gender, AVG(tripduration) AS average\_tripduration

FROM TripUsers

GROUP BY gender;

**Zapytania z join’ami**

Pobierz szczegóły podróży oraz informacje o stacji startowej:

SELECT t.\*, s.station\_name AS start\_station\_name, s.latitude AS start\_lat, s.longitude AS start\_long

FROM TripUsers t

JOIN Stations s ON t.start\_station\_id = s.station\_id;

Znajdź podróże z informacją o stacji końcowej:

SELECT t.\*, s.station\_name AS end\_station\_name, s.latitude AS end\_lat, s.longitude AS end\_long

FROM TripUsers t

JOIN Stations s ON t.end\_station\_id = s.station\_id;

Wszystkie podróże oraz stacje startowe i końcowe (dwa połączenia JOIN):

SELECT t.trip\_id, t.tripduration, t.starttime, start\_station.station\_name AS start\_station, end\_station.station\_name AS end\_station

FROM TripUsers t

JOIN Stations start\_station ON t.start\_station\_id = start\_station.station\_id

JOIN Stations end\_station ON t.end\_station\_id = end\_station.station\_id;

**Zapytania z podzapytaniami**

Znajdź podróże, których długość była większa od średniej długości wszystkich podróży:

SELECT \* FROM TripUsers

WHERE tripduration > (SELECT AVG(tripduration) FROM TripUsers);

Wybierz stacje, które były stacją końcową dla przynajmniej jednej podróży:

SELECT \* FROM Stations

WHERE station\_id IN (SELECT DISTINCT end\_station\_id FROM TripUsers);

Znajdź podróże użytkowników urodzonych przed 1980 rokiem, które trwały dłużej niż średnia:

SELECT \* FROM TripUsers

WHERE birth\_year < 1980

AND tripduration > (SELECT AVG(tripduration) FROM TripUsers WHERE birth\_year < 1980);

1. Doctors\_Appointments

**Zwykłe zapytania**

Wyświetlenie wszystkich lekarzy o specjalizacji Cardiology:

SELECT \*

FROM doctors

WHERE specialization = 'Cardiology';

Wyświetlenie pacjentów urodzonych po 1 stycznia 1980 roku:

SELECT \*

FROM patients

WHERE birthdate > '1980-01-01';

Wyświetlenie wszystkich wizyt, które miały miejsce w ciągu ostatniego miesiąca:

SELECT \*

FROM patients

WHERE birthdate > '1980-01-01';

**Zapytania z grupowaniem**

Zliczenie liczby pacjentów w każdym roku urodzenia:

SELECT YEAR(birthdate) AS birth\_year, COUNT(\*) AS patient\_count

FROM patients

GROUP BY birth\_year;

Wyświetlenie liczby wizyt dla każdej specjalizacji lekarza:

SELECT d.specialization, COUNT(a.appointment\_id) AS appointment\_count

FROM appointments a

JOIN doctors d ON a.doctor\_id = d.doctor\_id

GROUP BY d.specialization;

Zliczenie liczby wizyt, które miały określoną diagnozę:

SELECT diagnosis, COUNT(\*) AS diagnosis\_count

FROM appointments

GROUP BY diagnosis;

**Zapytania z join’ami**

Wyświetlenie imion i nazwisk lekarzy oraz imion i nazwisk ich pacjentów dla wszystkich wizyt:

SELECT d.first\_name AS doctor\_first\_name, d.last\_name AS doctor\_last\_name,

p.first\_name AS patient\_first\_name, p.last\_name AS patient\_last\_name

FROM appointments a

JOIN doctors d ON a.doctor\_id = d.doctor\_id

JOIN patients p ON a.patient\_id = p.patient\_id;

Wyświetlenie specjalizacji lekarzy oraz liczby wizyt, które wykonali:

SELECT d.specialization, COUNT(a.appointment\_id) AS appointment\_count

FROM doctors d

LEFT JOIN appointments a ON d.doctor\_id = a.doctor\_id

GROUP BY d.specialization;

Wyświetlenie nazwisk pacjentów, którzy mieli więcej niż jedną wizytę:

SELECT p.first\_name, p.last\_name, COUNT(a.appointment\_id) AS visit\_count

FROM patients p

JOIN appointments a ON p.patient\_id = a.patient\_id

GROUP BY p.patient\_id

HAVING visit\_count > 1;

**Zapytania z podzapytaniami**

Wyświetlenie pacjentów, którzy mieli wizytę u specjalisty Cardiology:

SELECT first\_name, last\_name

FROM patients

WHERE patient\_id IN (

SELECT patient\_id

FROM appointments a

JOIN doctors d ON a.doctor\_id = d.doctor\_id

WHERE d.specialization = 'Cardiology'

);

Wyświetlenie lekarzy, którzy mieli co najmniej jedną wizytę w ciągu ostatnich 7 dni:

SELECT first\_name, last\_name

FROM doctors

WHERE doctor\_id IN (

SELECT doctor\_id

FROM appointments

WHERE appointment\_date >= CURDATE() - INTERVAL 7 DAY

);

Wyświetlenie pacjentów, którzy mają co najmniej dwie wizyty na tę samą diagnozę:

SELECT first\_name, last\_name

FROM patients

WHERE patient\_id IN (

SELECT patient\_id

FROM appointments

GROUP BY patient\_id, diagnosis

HAVING COUNT(appointment\_id) >= 2

);

-------------------------------------------------------------------------------------------------------------------------------------------

ZAPYTANIA MONGODB

1. Airpotrs

**Zwykłe zapytania**

Wyświetlenie wszystkich lotów z opóźnieniem przylotu większym niż 60 minut:

db.flights.find({ ARRIVAL\_DELAY: { $gt: 60 } });

Wyświetlenie nazw linii lotniczych, których kod IATA to AA:

db.airlines.find({ IATA\_CODE: "AA" }, { AIRLINE: 1, \_id: 0 });

Wyświetlenie lotnisk znajdujących się w stanie CA (Kalifornia):

db.airports.find({ STATE: "CA" }, { AIRPORT: 1, CITY: 1, \_id: 0 });

**Zapytania z grupowaniem**

Wyświetlenie średniego opóźnienia przylotu dla każdego dnia tygodnia:

db.flights.aggregate([

{ $group: { \_id: "$DAY\_OF\_WEEK", avg\_arrival\_delay: { $avg: "$ARRIVAL\_DELAY" } } }

]);

Zliczenie liczby lotów anulowanych z każdej przyczyny:

db.flights.aggregate([

{ $match: { CANCELLED: 1 } },

{ $group: { \_id: "$CANCELLATION\_REASON", cancel\_count: { $sum: 1 } } }

]);

Wyświetlenie liczby lotów każdego dnia dla linii lotniczej UA:

db.flights.aggregate([

{ $match: { AIRLINE: "UA" } },

{ $group: { \_id: { year: "$YEAR", month: "$MONTH", day: "$DAY" }, flight\_count: { $sum: 1 } } }

]);

**Zapytania z join’ami**

Wyświetlenie nazw linii lotniczych oraz liczby lotów dla każdej z nich:

db.flights.aggregate([

{ $lookup: { from: "airlines", localField: "AIRLINE", foreignField: "IATA\_CODE", as: "airline\_info" } },

{ $unwind: "$airline\_info" },

{ $group: { \_id: "$airline\_info.AIRLINE", flight\_count: { $sum: 1 } } }

]);

Wyświetlenie nazw miast dla lotów, które miały opóźnienie przylotu większe niż 100 minut, łącząc dane z lotniska początkowego:

db.flights.aggregate([

{ $match: { ARRIVAL\_DELAY: { $gt: 100 } } },

{ $lookup: { from: "airports", localField: "ORIGIN\_AIRPORT", foreignField: "IATA\_CODE", as: "origin\_airport" } },

{ $unwind: "$origin\_airport" },

{ $project: { origin\_city: "$origin\_airport.CITY", ARRIVAL\_DELAY: 1, \_id: 0 } }

]);

Wyświetlenie nazw lotnisk, do których miały miejsce loty opóźnione o więcej niż 2 godziny, wraz z nazwami linii lotniczych:

db.flights.aggregate([

{ $match: { ARRIVAL\_DELAY: { $gt: 120 } } },

{ $lookup: { from: "airlines", localField: "AIRLINE", foreignField: "IATA\_CODE", as: "airline\_info" } },

{ $unwind: "$airline\_info" },

{ $lookup: { from: "airports", localField: "DESTINATION\_AIRPORT", foreignField: "IATA\_CODE", as: "destination\_airport" } },

{ $unwind: "$destination\_airport" },

{ $project: { destination\_airport: "$destination\_airport.AIRPORT", airline\_name: "$airline\_info.AIRLINE", \_id: 0 } }

]);

**Zapytania z podzapytaniami**

Wyświetlenie lotów z największym opóźnieniem przylotu dla każdego dnia:

db.flights.aggregate([

{ $group: { \_id: { year: "$YEAR", month: "$MONTH", day: "$DAY" }, max\_delay: { $max: "$ARRIVAL\_DELAY" } } },

{ $lookup: { from: "flights", localField: "max\_delay", foreignField: "ARRIVAL\_DELAY", as: "max\_delay\_flights" } },

{ $unwind: "$max\_delay\_flights" },

{ $match: { "$expr": { $and: [

{ $eq: [ "$\_id.year", "$max\_delay\_flights.YEAR" ] },

{ $eq: [ "$\_id.month", "$max\_delay\_flights.MONTH" ] },

{ $eq: [ "$\_id.day", "$max\_delay\_flights.DAY" ] }

] } }

]);

Wyświetlenie nazw linii lotniczych, które miały więcej niż 1000 opóźnionych lotów:

db.flights.aggregate([

{ $match: { ARRIVAL\_DELAY: { $gt: 0 } } },

{ $group: { \_id: "$AIRLINE", delayed\_flights: { $sum: 1 } } },

{ $match: { delayed\_flights: { $gt: 1000 } } },

{ $lookup: { from: "airlines", localField: "\_id", foreignField: "IATA\_CODE", as: "airline\_info" } },

{ $unwind: "$airline\_info" },

{ $project: { airline\_name: "$airline\_info.AIRLINE", \_id: 0 } }

]);

Wyświetlenie miast, z których najwięcej lotów zostało anulowanych:

db.flights.aggregate([

{ $match: { CANCELLED: 1 } },

{ $group: { \_id: "$ORIGIN\_AIRPORT", cancel\_count: { $sum: 1 } } },

{ $sort: { cancel\_count: -1 } },

{ $limit: 1 },

{ $lookup: { from: "airports", localField: "\_id", foreignField: "IATA\_CODE", as: "origin\_airport" } },

{ $unwind: "$origin\_airport" },

{ $project: { city: "$origin\_airport.CITY", \_id: 0 } }

]);

1. Bikes

**Zwykłe zapytania**

Wybór wszystkich podróży, które trwały dłużej niż 30 minut:

db.tripusers.find({ tripduration: { $gt: 30 \* 60 } });

Znajdź wszystkie podróże, które zakończyły się na stacji "Central Park":

db.tripusers.find({ end\_station\_name: "Central Park" });

Wybierz wszystkie unikalne typy użytkowników, którzy skorzystali z usługi:

db.tripusers.distinct("usertype");

**Zapytania z grupowaniem**

Policz liczbę podróży dla każdego typu użytkownika:

db.tripusers.aggregate([

{ $group: { \_id: "$usertype", trip\_count: { $sum: 1 } } }

]);

Policz całkowity czas podróży dla każdego roku urodzenia użytkowników:

db.tripusers.aggregate([

{ $group: { \_id: "$birth\_year", total\_tripduration: { $sum: "$tripduration" } } }

]);

Oblicz średnią długość podróży w zależności od płci użytkowników:

db.tripusers.aggregate([

{ $group: { \_id: "$gender", average\_tripduration: { $avg: "$tripduration" } } }

]);

**Zapytania z join’ami**

Pobierz szczegóły podróży oraz informacje o stacji startowej:

db.tripusers.aggregate([

{

$lookup: {

from: "stations",

localField: "start\_station\_id",

foreignField: "station\_id",

as: "start\_station\_info"

}

},

{ $unwind: "$start\_station\_info" }

]);

Znajdź podróże z informacją o stacji końcowej:

db.tripusers.aggregate([

{

$lookup: {

from: "stations",

localField: "end\_station\_id",

foreignField: "station\_id",

as: "end\_station\_info"

}

},

{ $unwind: "$end\_station\_info" }

]);

Wszystkie podróże oraz stacje startowe i końcowe (dwa połączenia $lookup):

db.tripusers.aggregate([

{

$lookup: {

from: "stations",

localField: "start\_station\_id",

foreignField: "station\_id",

as: "start\_station\_info"

}

},

{

$lookup: {

from: "stations",

localField: "end\_station\_id",

foreignField: "station\_id",

as: "end\_station\_info"

}

},

{ $unwind: "$start\_station\_info" },

{ $unwind: "$end\_station\_info" }

]);

**Zapytania z podzapytaniami**

Znajdź podróże, których długość była większa od średniej długości wszystkich podróży:

Najpierw obliczamy średnią długość podróży, potem zapytanie z warunkiem

const avgTripDuration = db.tripusers.aggregate([

{ $group: { \_id: null, avg\_tripduration: { $avg: "$tripduration" } } }

]).toArray()[0].avg\_tripduration;

// Następnie wykonujemy zapytanie z warunkiem

db.tripusers.find({ tripduration: { $gt: avgTripDuration } });

Wybierz stacje, które były stacją końcową dla przynajmniej jednej podróży:

db.stations.find({

station\_id: { $in: db.tripusers.distinct("end\_station\_id") }

});

Znajdź podróże użytkowników urodzonych przed 1980 rokiem, które trwały dłużej niż średnia:

const avgTripDuration1980 = db.tripusers.aggregate([

{ $match: { birth\_year: { $lt: 1980 } } },

{ $group: { \_id: null, avg\_tripduration: { $avg: "$tripduration" } } }

]).toArray()[0].avg\_tripduration;

db.tripusers.find({

birth\_year: { $lt: 1980 },

tripduration: { $gt: avgTripDuration1980 }

});

1. Doctors\_Appointments

**Zwykłe zapytania**

Wyświetlenie wszystkich lekarzy o specjalizacji Cardiology:

db.doctors.find({ specialization: "Cardiology" });

Wyświetlenie pacjentów urodzonych po 1 stycznia 1980 roku:

db.patients.find({ birthdate: { $gt: new Date("1980-01-01") } });

Wyświetlenie wszystkich wizyt, które miały miejsce w ciągu ostatniego miesiąca:

db.appointments.find({

appointment\_date: {

$gte: new Date(new Date().setMonth(new Date().getMonth() - 1))

}

});

**Zapytania z grupowaniem**

Zliczenie liczby pacjentów w każdym roku urodzenia:

db.patients.aggregate([

{ $group: { \_id: { $year: "$birthdate" }, patient\_count: { $sum: 1 } } }

]);

Wyświetlenie liczby wizyt dla każdej specjalizacji lekarza:

db.appointments.aggregate([

{ $lookup: { from: "doctors", localField: "doctor\_id", foreignField: "doctor\_id", as: "doctor\_info" } },

{ $unwind: "$doctor\_info" },

{ $group: { \_id: "$doctor\_info.specialization", appointment\_count: { $sum: 1 } } }

]);

Zliczenie liczby wizyt, które miały określoną diagnozę:

db.appointments.aggregate([

{ $group: { \_id: "$diagnosis", diagnosis\_count: { $sum: 1 } } }

]);

**Zapytania z join’ami**

Wyświetlenie imion i nazwisk lekarzy oraz imion i nazwisk ich pacjentów dla wszystkich wizyt:

db.appointments.aggregate([

{ $lookup: { from: "doctors", localField: "doctor\_id", foreignField: "doctor\_id", as: "doctor\_info" } },

{ $unwind: "$doctor\_info" },

{ $lookup: { from: "patients", localField: "patient\_id", foreignField: "patient\_id", as: "patient\_info" } },

{ $unwind: "$patient\_info" },

{ $project: {

doctor\_first\_name: "$doctor\_info.first\_name",

doctor\_last\_name: "$doctor\_info.last\_name",

patient\_first\_name: "$patient\_info.first\_name",

patient\_last\_name: "$patient\_info.last\_name"

}

}

]);

Wyświetlenie specjalizacji lekarzy oraz liczby wizyt, które wykonali:

db.doctors.aggregate([

{ $lookup: { from: "appointments", localField: "doctor\_id", foreignField: "doctor\_id", as: "appointments" } },

{ $project: { specialization: 1, appointment\_count: { $size: "$appointments" } } }

]);

Wyświetlenie nazwisk pacjentów, którzy mieli więcej niż jedną wizytę:

db.appointments.aggregate([

{ $group: { \_id: "$patient\_id", visit\_count: { $sum: 1 } } },

{ $match: { visit\_count: { $gt: 1 } } },

{ $lookup: { from: "patients", localField: "\_id", foreignField: "patient\_id", as: "patient\_info" } },

{ $unwind: "$patient\_info" },

{ $project: { first\_name: "$patient\_info.first\_name", last\_name: "$patient\_info.last\_name" } }

]);

**Zapytania z podzapytaniami**

Wyświetlenie pacjentów, którzy mieli wizytę u specjalisty Cardiology:

db.appointments.aggregate([

{ $lookup: { from: "doctors", localField: "doctor\_id", foreignField: "doctor\_id", as: "doctor\_info" } },

{ $unwind: "$doctor\_info" },

{ $match: { "doctor\_info.specialization": "Cardiology" } },

{ $lookup: { from: "patients", localField: "patient\_id", foreignField: "patient\_id", as: "patient\_info" } },

{ $unwind: "$patient\_info" },

{ $project: { first\_name: "$patient\_info.first\_name", last\_name: "$patient\_info.last\_name" } }

]);

Wyświetlenie lekarzy, którzy mieli co najmniej jedną wizytę w ciągu ostatnich 7 dni:

db.appointments.aggregate([

{ $match: { appointment\_date: { $gte: new Date(new Date().setDate(new Date().getDate() - 7)) } } },

{ $group: { \_id: "$doctor\_id" } },

{ $lookup: { from: "doctors", localField: "\_id", foreignField: "doctor\_id", as: "doctor\_info" } },

{ $unwind: "$doctor\_info" },

{ $project: { first\_name: "$doctor\_info.first\_name", last\_name: "$doctor\_info.last\_name" } }

]);

Wyświetlenie pacjentów, którzy mają co najmniej dwie wizyty na tę samą diagnozę:

db.appointments.aggregate([

{ $group: { \_id: { patient\_id: "$patient\_id", diagnosis: "$diagnosis" }, appointment\_count: { $sum: 1 } } },

{ $match: { appointment\_count: { $gte: 2 } } },

{ $group: { \_id: "$\_id.patient\_id" } },

{ $lookup: { from: "patients", localField: "\_id", foreignField: "patient\_id", as: "patient\_info" } },

{ $unwind: "$patient\_info" },

{ $project: { first\_name: "$patient\_info.first\_name", last\_name: "$patient\_info.last\_name" } }

]);