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

Комп’ютерний практикум №1

Програмування інтелектулаьних систем

Виконав
студент групи ІП-11:
Панченко С. В.

Перевірила:
Барчишин Л.М.

Київ 2023

ЗМІСТ

1 ЗАВДАННЯ

1) Створити стовпчикову і звичайну бд https://github.com/mariadb-corporation/mariadb-columnstore-sample-data (приклад застосунку https://github.com/orgs/mariadb-developers/repositories?q=flights-app&language=)

2) Розрахувати сумарну затримку по містах

3) Порахувати кількість польотів по містах

4) Знайти місто з найменшою і найбільшою затримкою

5) Знайти всі польоти з затримкою більше за середній час затримки

6) Заміряти вбудованими методами об”єм БД та швидкість виконання запитів. Порівняти звичайну і стовпчикову

2 ЗАПИТИ

2.1 total_flights_by_cities

```
async fn total_flights_by_cities(conn: &mut PgConnection, suffix: &str) ->
Result<u32, MyError> {
    let now = std::time::SystemTime::now();

    let query = format!("
        select a.city as city, SUM(f.id) as total_flights from flights{} as f
        inner join airports as a on a.iata_code = f.dest
        group by city
    ", suffix);

    sqlx::query(&query).execute(conn).await?;

    Ok(std::time::SystemTime::now().duration_since(now)?.as_millis() as u32)
}
```

2.2 minmax_delay_by_cities

```
async fn minmax_delay_by_cities(conn: &mut PgConnection, is_max: bool, suffix:
&str) -> Result<u32, MyError> {
    let minmax = if is_max { "MAX" } else { "MIN" };

    let now = std::time::SystemTime::now();

    let query = format!("
        with t as (
            select a.city as city, SUM(f.late_aircraft_delay) as total_delay
from flights{} as f
            inner join airports{} as a on a.iata_code = f.dest
            group by city
        )
        select city, total_delay from t where total_delay = (SELECT {}
(total_delay) from t LIMIT 1)
    ", suffix, suffix, minmax);

    sqlx::query(&query).execute(conn).await?;

    Ok(std::time::SystemTime::now().duration_since(now)?.as_millis() as u32)
}

async fn min_delay_by_cities(conn: &mut PgConnection, suffix: &str) ->
Result<u32, MyError> {
```

```
    minmax_delay_by_cities(conn, false, suffix).await
}

async fn max_delay_by_cities(conn: &mut PgConnection, suffix: &str) ->
Result<u32, MyError> {
    minmax_delay_by_cities(conn, true, suffix).await
}
```

2.3 more_than_median

```
async fn more_than_median(conn: &mut PgConnection, suffix: &str) -> Result<u32,
MyError> {
    let now = std::time::SystemTime::now();

    let query = format!("
        select f.id from flights{} as f
        where f.id > (SELECT PERCENTILE_DISC(0.5) WITHIN GROUP(ORDER BY
flights{}.late_aircraft_delay) FROM flights{})
    ", suffix, suffix, suffix);

    sqlx::query(&query).execute(conn).await?;

    Ok(std::time::SystemTime::now().duration_since(now)?.as_millis() as u32)
}
```

3 РЕЗУЛЬТАТИ


```
{
  "results": [
    {
      "func": "total_delay_by_cities",
      "millis": {
        "column_based": 33.8,
        "row_based": 30.1
      }
    },
    {
      "func": "total_flights_by_cities",
      "millis": {
        "column_based": 33.4,
        "row_based": 28.4
      }
    },
    {
      "func": "min_delay_by_cities",
      "millis": {
        "column_based": 32.4,
        "row_based": 27.4
      }
    },
    {
      "func": "max_delay_by_cities",
      "millis": {
        "column_based": 32.3,
        "row_based": 27.4
      }
    },
    {
      "func": "total_flights_by_cities",
      "millis": {
        "column_based": 33.0,
        "row_based": 24.1
      }
    }
  ]
}
```