DBS_2

1) Vradte zoznam vsetkych diskutujucich (users) prıspevku (posts) s ID :post id, pricom ich usporiadajte v zavislosti od casu vytvorenia ich komentara, zacınajuc od najnovsıch a konciac najstarsımi.

Volanie: /v2/posts/1819157/users

SQL dopyt:

```
GET_POST_USERS_QUERY = """
    SELECT users.id, users.reputation,
           TO_CHAR(users.creationdate AT TIME ZONE 'UTC+0', 'YYYY-MM-
DD"T"HH24:MI:SS.US+00:00') AS creationdate,
           users.displayname,
           TO_CHAR(users.lastaccessdate AT TIME ZONE 'UTC+0', 'YYYY-MM-
DD"T"HH24:MI:SS.US+00:00') AS lastaccessdate,
           users.websiteurl, users.location, users.aboutme, users.views,
users.upvotes, users.downvotes, users.profileimageurl, users.age,
users.accountid
        FROM users
    JOIN comments ON users.id = comments.userid
    WHERE comments.postid = $1
    ORDER BY comments.creationdate DESC;
"""
```

*Najprv vyberiem všetky užívateľské parametre a preformátujem parametre s časovými ukazovateľmi v UTC+0.

*Z tabuľky používateľov

*Kombinujem tabuľku komentárov a používateľov podľa ID, aby som dostal tabuľku komentárov a tvorcov

*Urobíme výber komentárov podľa ID príspevku(a získajte úplné informácie o používateľoch, ktorí ich vytvorili)

*Výstup organizujeme podľa dátumu vytvorenia komentárov

V kode:

```
@router.get('/v2/posts/{post_id}/users')      ### /v2/posts/1819157/users
async def get_post_users(post_id: int):
    users = await execute_query(GET_POST_USERS_QUERY, post_id)

    formatted_users = []
    for user in users:
        formatted_user = {
            "id": user["id"],
            "reputation": user["reputation"],
            "creationdate": str(user["creationdate"]),
            "displayname": user["displayname"],
            "lastaccessdate": str(user["lastaccessdate"]),
            "websiteurl": user["websiteurl"],
```

```
            "location": user["location"],
            "aboutme": user["aboutme"],
            "views": user["views"],
            "upvotes": user["upvotes"],
            "downvotes": user["downvotes"],
            "profileimageurl": user["profileimageurl"],
            "age": user["age"],
            "accountid": user["accountid"]
        }
        formatted_users.append(formatted_user)

    response_data = {"items": formatted_users}

    return response_data
```

Pomocou špecifickej značky zavolám požiadavku, ktorú potrebujem, a zobrazím ju naformátovanú

```
(post_id: int)    -    Premenlivý argument
```

```
async def execute_query(query, *args):
    conn = await connect_to_database()
    try:
        result = await conn.fetch(query, *args)
        return result
    finally:
        await conn.close()
```

Spoločný kód, ktorý sa používa vo všetkých 5 koncových bodoch, vykoná požiadavku a vráti potrebné dáta, ktoré sú naformátované výstupným kódom

2) Vypracujte zoznam diskutujucich pre pouzıvatela user id, obsahujuci pouzıvatelov, ktorı komentovali prıspevky, ktore dany pouzıvatel zalozil alebo na ktorych komentoval. Usporiadajte pouzıvatelov v zavislosti od datumu ich registracie, zacınajuc s tymi, ktoı sa zaregistrovali ako prvı.

Volanie: /v2/users/1076348/friends

SQL dopyt:

```
GET_FRIENDS_USERS_QUERY = """
    SELECT users.id, users.reputation,
           TO_CHAR(users.creationdate AT TIME ZONE 'UTC+0', 'YYYY-MM-
DD"T"HH24:MI:SS.US+00:00') AS creationdate,
           users.displayname,
```

```
           TO_CHAR(users.lastaccessdate AT TIME ZONE 'UTC+0', 'YYYY-MM-
DD"T"HH24:MI:SS.US+00:00') AS lastaccessdate,
           users.websiteurl, users.location, users.aboutme, users.views,
users.upvotes, users.downvotes, users.profileimageurl, users.age,
users.accountid
    FROM
    (SELECT comments.userid AS user_id
    FROM posts
    JOIN comments ON posts.id = comments.postid
    WHERE comments.userid = $1

    UNION

    SELECT comments.userid AS user_id
    FROM comments
    JOIN posts ON comments.postid = posts.id
    WHERE posts.owneruserid = $1
    ) AS user_ids
JOIN users ON users.id = user_ids.user_id ORDER BY users.creationdate;
"""
```

*Najprv vyberiem všetky užívateľské parametre a preformátujem parametre s časovými ukazovateľmi v UTC+0.
* Vyberieme ID používateľa osoby, ktorá komentár vytvorila, a zastúpime ho ako user_id. Z tabuľky, ktorá zahŕňa všetky komentáre pod príspevkom. Kde jeden z komentujúcich je identifikátor, ktorý sme vybrali. Týmto spôsobom získame zoznam používateľov, ktorí zanechali komentáre pod príspevkom, ktorý komentoval používateľ, ktorého sme vybrali.
* Vyberieme používateľa, ktorý príspevok vytvoril a zastupujeme ho ako user_id. Z tabuľky, ktorá obsahuje všetky komentáre a príspevky, pod ktorými boli komentáre zanechané. Vyberáme len tie príspevky, ktorých ID tvorcu sa zhoduje so zadanými(V súlade s tým dostaneme ID všetkých používateľov, ktorí zanechali komentáre pod príspevkom používateľa, ktorého ID bolo zadané)
* Skombinujeme tieto 2 tabuľky UNIONom a získame jedinečný zoznam ID používateľov
* Následne prepojíme výslednú tabuľku ID používateľov s tabuľkou používateľov podľa ID, aby sme získali informácie o požadovaných používateľoch
* Zoradiť používateľov podľa dátumu vytvorenia

V kode:
Všetko je rovnaké ako v prvom koncovom bode

3) Urcte, ake percentualne zastupenie maju prıspevky s konkretnym tagom v ramci celkoveho poctu prıspevkov vydanych v jednotlivych dnoch tyzdna (naprıklad pondelok, utorok), a to pre kazdy den tyzdna zvlast. Vysledky ukaze na skale od 0- 100 a zaokruhlite na dve desatinne miesta.

Volanie: /v2/tags/linux/stats

SQL dopyt:

```
GET_PERCENTAGE_TAGS_QUERY = """
    SELECT EXTRACT(DOW FROM posts.creationdate) AS day_of_week,
```

```
    ROUND((COUNT(CASE WHEN tags.tagname = $1 THEN 1 END) * 100.0 /
COUNT(DISTINCT posts.id)), 2) AS percent_linux

    FROM tags
    JOIN post_tags ON tags.id = post_tags.tag_id
    JOIN posts ON post_tags.post_id = posts.id

    GROUP BY day_of_week
    ORDER BY day_of_week ASC
"""
```

* Vyberieme deň v týždni (ktorý sa previedol z dátumu vytvorenia príspevku pomocou vstavanej funkcie) a percento príspevkov so značkou, ktorú sme vybrali (na počítanie príspevkov so značkami používame CASE WHEN, počítame celkovo počet príspevkov (používame DISTINCT, aby sme sa vyhli duplicitám). Potom pomocou COUNT vypočítame percentá pomocou bežnej matematickej operácie a výsledky zaokrúhlime pomocou ROUND)
* Vyberte zo zlúčenej tabuľky. Najprv skombinujeme značky príspevkov so značkami podľa ID značky, aby sme získali informácie o ID príspevkov, pod ktorými boli značky ponechané, a potom skombinujeme príspevky a značky príspevkov podľa ID príspevkov, aby sme získali prístup k dátumu vytvorenia príspevku podľa názvu požadovaného tagu
* Výstup zoskupujeme podľa dňa v týždni, aby sme získali celkový počet príspevkov pre konkrétny deň v týždni
* Dni v týždni sú usporiadané vzostupne

V kode:

```
day_of_week_names = {
    0: "sunday",
    1: "monday",
    2: "tuesday",
    3: "wednesday",
    4: "thursday",
    5: "friday",
    6: "saturday"
}
```

Keďže sa dni v týždni zobrazujú ako čísla, vďaka formátovaniu ich zobrazujem správne

```
formatted_percentage = {}
for row in statistics:
    day_of_week = row[0]
    percent_linux = row[1]

    day = day_of_week_names.get(day_of_week)

    formatted_percentage[day] = percent_linux

sunday_value = formatted_percentage.pop("sunday")
formatted_percentage["sunday"] = sunday_value
```

Keďže nedeľa je 0, presúvam ju zo začiatku na koniec zoznamu

4) Vystupom je zoznam :limit najnovsıch vyriesenych prıspevkov, ktore boli otvorene maximalne :duration in minutes minut (pocet minut medzi creationdate a closeddate). Trvanie otvorenia (duration) zaokruhlite na dve desatinne miesta.

Volanie: /v2/posts?duration=5&limit=2

SQL dopyt:

```
GET_POST_DURATION_WITH_LIMIT_QUERY = """
    SELECT posts.id, TO_CHAR(posts.creationdate AT TIME ZONE 'UTC+0', 'YYYY-
MM-DD"T"HH24:MI:SS.US+00:00') AS creationdate, posts.viewcount,
TO_CHAR(posts.lasteditdate AT TIME ZONE 'UTC+0', 'YYYY-MM-
DD"T"HH24:MI:SS.US+00:00') AS lasteditdate, TO_CHAR(posts.lastactivitydate AT
TIME ZONE 'UTC+0', 'YYYY-MM-DD"T"HH24:MI:SS.US+00:00') AS lastactivitydate,
posts.title, TO_CHAR(posts.closeddate AT TIME ZONE 'UTC+0', 'YYYY-MM-
DD"T"HH24:MI:SS.US+00:00') AS closeddate, ROUND((EXTRACT(EPOCH FROM
(closeddate::timestamp - creationdate::timestamp)) / 60), 2) AS duration

    FROM posts
    WHERE closeddate IS NOT NULL AND EXTRACT(EPOCH FROM
(closeddate::timestamp - creationdate::timestamp)) / 60 < $1

    ORDER BY creationdate DESC
    LIMIT $2
"""
```

* Vyberieme všetky informácie o príspevkoch a preformátujeme čas, navyše pomocou funkcie EXTRACT(EPOCH) zistíme čas v sekundách medzi dátumom vytvorenia a dátumom uzávierky, potom vydelíme 60, aby sme dostali čas v minútach a zaokrúhliť výsledok.
* Potom vyberieme z tabulky príspevkov tie príspevky, v ktorých je čas medzi vytvorením a uzavretím kratší ako zadaný čas
* Zobrazujeme relevantné príspevky, počnúc tými najnovšími
* Zobrazte určený počet vhodných príspevkov

5) Navrhnite koncovy bod (endpoint), ktory poskytne zoznam prıspevkov usporiadanych od najnovsıch po najstarsie. Sucastou odpovede je aj kompletny zoznam priradenych tagov. Tento koncovy bod podporuje dva parametre: • limit: maximalny pocet prıspevkov v odpovedi, • query: retazec na vyhladavanie nad posts.title a posts.body. Vyhlad´avanie nie je citlive na diakritiku a male/ve˘ lke pısmena

Volanie: /v2/posts?limit=1&query=linux

SQL dopyt:

```
GET_POST_ON_KEYWORD_WITH_LIMIT_QUERY = """
SELECT posts.id, TO_CHAR(posts.creationdate AT TIME ZONE 'UTC+0', 'YYYY-MM-
DD"T"HH24:MI:SS.US+00:00') AS creationdate, posts.viewcount,
TO_CHAR(posts.lasteditdate AT TIME ZONE 'UTC+0', 'YYYY-MM-
DD"T"HH24:MI:SS.US+00:00') AS lasteditdate, TO_CHAR(posts.lastactivitydate AT
TIME ZONE 'UTC+0', 'YYYY-MM-DD"T"HH24:MI:SS.US+00:00') AS lastactivitydate,
posts.title, posts.body, posts.answercount, TO_CHAR(posts.closeddate AT TIME
ZONE 'UTC+0', 'YYYY-MM-DD"T"HH24:MI:SS.US+00:00') AS closeddate,
STRING_AGG(tags.tagname, ', ') AS tags_list
    FROM posts
    JOIN post_tags ON posts.id = post_tags.post_id JOIN tags ON
post_tags.tag_id = tags.id
    WHERE (posts.body ILIKE '%' || unaccent($2) || '%' OR posts.title ILIKE
'%' || unaccent($2) || '%')
    GROUP BY posts.id, posts.creationdate, posts.viewcount,
posts.lasteditdate, posts.title, posts.body, posts.answercount,
posts.closeddate, posts.lastactivitydate
    ORDER BY posts.creationdate DESC
    LIMIT $1
"""
```

* Vyberieme všetky informácie o príspevkoch a preformátujeme čas, okrem toho vytvoríme reťazec, v ktorom skombinujeme všetky značky spojené s týmto príspevkom pomocou funkcie STRING_AGG

* Tabuľku značiek príspevkov skombinujeme s príspevkami, aby sme vedeli, aké ID štítkov sú v tomto príspevku, a potom štítky zlúčime s štítkami príspevkov, aby ste mohli nájsť príspevok podľa názvu štítku a nie podľa jeho ID.

* Vyberáme len tie príspevky, v ktorých má nadpis alebo telo časť, ktorú sme vybrali (slovo/vety). Na to použite % na stranách vybranej časti, čo znamená, že pred a za týmto slovom môže byť čokoľvek. Ignorujeme veľké a malé písmená v slove, ktoré sme zadali pomocou funkcie ILIKE, a ignorujeme diakritiku pomocou unaccent().

* Tabuľku zoskupujeme okolo všetkých indikátorov príspevkov, aby sa blok značky zobrazoval správne

* Zobrazuje sa zvolený počet zodpovedajúcich príspevkov

V kode:

```
async def get_posts(limit: int, duration: Optional[int] = None, query:
Optional[str] = None):
```

```
if duration is not None:
    posts = await execute_query(GET_POST_DURATION_WITH_LIMIT_QUERY, duration,
limit)
```

```
elif query is not None:
```

Keďže požiadavka je rovnaká, ale argumenty sú odlišné, môžeme dostať odpoveď v závislosti od prijatých argumentov