# NoSQL Assignment 1

Abhinav Kumar - IMT2022079
Siddhesh Deshpande - IMT2022080
Krish Patel - IMT2022097
Jinesh Pagaria - IMT2022044

February 13, 2025

## Section A

1. a) Marines
**Reasoning:** Here Marines scored 5 while their opponent Hawks scored 3 so $5 > 3$ and $5 <= 3 + 2$

2. d) Bay Stars, Monday
**Reasoning:** Here Bay Stars' opponent on Monday is NULL

3. a) Bay Stars, Tigers, 2
**Reasoning:** Maximum scores by Bay Stars is 2 and Tiger is their only opponent

4. a) Giants, Sunday, NULL, NULL could appear seventh through twelfth.
**Reasoning:** In MySQL NULL is considered smaller than other numbers hence Giants, Sunday, NULL, NULL is placed 12th but in PostgreSQL NULL is considered larger than a number hence it is placed third

5. c) Dragons, Giants
**Reasoning:** Dragons and Giants have the same opponent, Carp.

## Section B

### Problem 1

Q1. We created two databases `Assignment1_fk` with appropriate foreign key relations and Assingnment1_withoutfk without any foreign key constraints. For doing this we used the **create database** `<database_name>` command. We then used the following queries to create respective tables in the `Assignment1_withoutfk` database. The Queries are as follows.

1. **create table keywords(**
   **id int not null,**
   **term text not null,**
   **score real not null,**
   **primary key(id,term)**
   **);**

2. **create table revisionuri(**
   **id int not null,**
   **uri text not null,**
   **primary key(id)**
   **);**

The Following Queries were used to create tables inside the `Assignment1_fk` database. Here we have introduced foreign key constraints.

1. **create table revisionuri(**
   **id int not null,**
   **uri text not null,**
   **primary key(id)**
   **);**

2. **create table keywords(**
   **id int not null,**
   **term text not null,**
   **score real not null,**
   **primary key(id,term),**
   **constraint id_constraint foreign key (id) references revisionuri(id)**
   **);**

So here the logic for introducing this foreign key constraint was that we wanted to ensure that an id exist in the keywords table if and only if we have a url with the corresponding id in the REVISION_URIS file or in other words in the revisionuri table.

Q2.Now we had to bulk load the data preserving the UTF-8 encoding format of the data .So for this we have used the following queries .

1. ```
   \copy keywords from '/home/siddhesh/Documents/Nosql Assingment 1
   /Wikipedia-EN-20120601_KEYWORDS.TSV'
   DELIMITER E'\t'
   CSV ENCODING 'UTF8';
   ```

2. ```
   \copy revisionuri from '/home/siddhesh/Documents/Nosql Assingment 1
   /Wikipedia-EN-20120601_REVISION_URIS.TSV' DELIMITER E'\t'
   CSV ENCODING 'UTF8';
   ```

Q3.

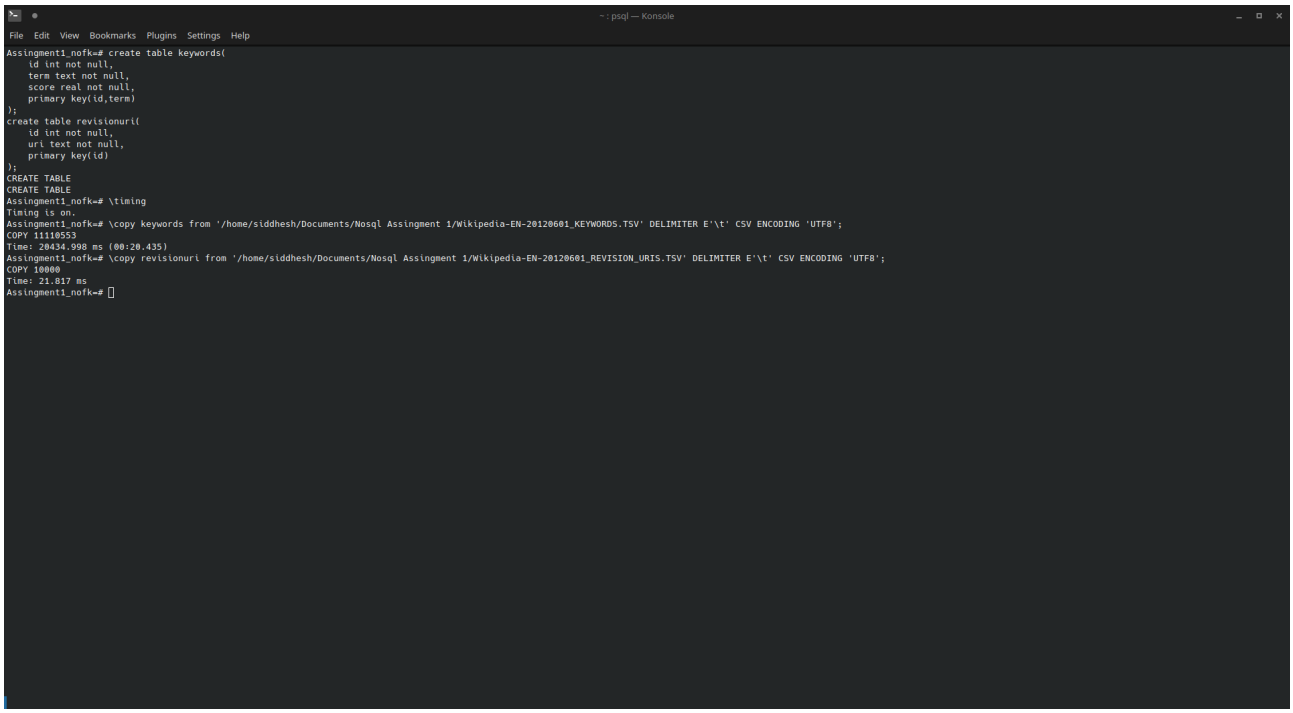


Figure 1: bulk loading data without foreign key constraint

1. **Time to load the uri data :** 21.817 milliseconds.
2. **Time to load the keyword data :** 20.434 seconds.

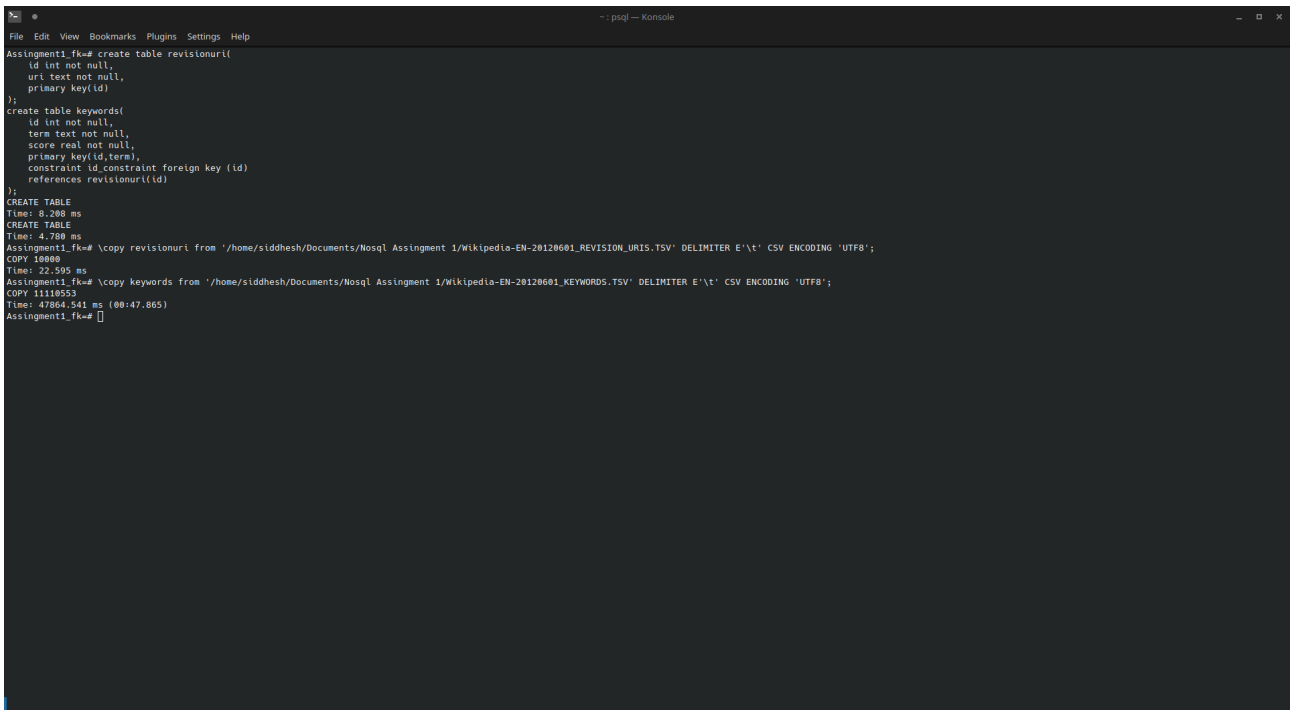


Figure 2: bulk loading data with foreign key constraint

1. **Time to load the uri data :** 22.595 milliseconds
2. **Time to load the keywords data :** 47.865 seconds.

**Reasoning :** When the foreign key constraint is introduced, the loading of data is slower as PostgreSQL has to check for the existence of the corresponding primary key in another table for each row inserted in one table.

# Problem 2

Q1.

```
select r.uri
from revisionuri as r
inner join keywords as k on k.id = r.id
group by r.id
having
(
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
);
```

Figure 3: Problem 1

**Reasoning**:This query fetches the uri values from the revisionuri table by executing an inner join with the keywords table based on the id field. The results are aggregated based on r.id, and the HAVING clause selects these groups only where all four particular keywords (beginning with "infantri", "reinforc", "brigad", and "fire") are present for the respective r.id in the keywords table. The subqueries utilize EXISTS to query for the existence of these keywords, returning only r.id values with all four matching terms. Q2.

```sql
-- Problem 2
select r.uri
from revisionuri as r
inner join keywords as k on k.id = r.id
group by r.id
having
(
    (exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%'))
    or
    (not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%'))
    or
    (not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%'))
    or
    (not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%'))
);
```

Figure 4: Problem 2

**Reasoning:** This query selects uri values from the revisionuri table where the related id in the keywords table is equal to one of four criteria. Each criterion verifies the existence of a certain keyword (beginning with "infantri", "reinforc", "brigad", or "fire") and non existence of the others. It employs EXISTS to check for existence and NOT EXISTS to ensure the non-existence of similar keywords for every criterion.
Q3.

```sql
-- problem 3
select r.uri
from revisionuri as r
inner join keywords as k on k.id = r.id
group by r.id
having
(
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
    not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
);
```

Figure 5: Problem 3

**Reasoning:**This query returns uri values from the revisionuri table where the related id in the keywords table satisfies a particular condition: the id should not have a keyword that begins with "infantri", should have a keyword beginning with "reinforc", and should not have keywords beginning with "brigad" or "fire". The query employs EXISTS and NOT EXISTS to filter according to these patterns of keywords.

Q4.

```sql
-- problem 4
select r.uri
from revisionuri as r
inner join keywords as k on k.id = r.id
group by r.id
having
(
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
)
ORDER BY
(
    SELECT SUM(CASE
                WHEN k.term LIKE 'infantri%' THEN k.score
                WHEN k.term LIKE 'reinforc%' THEN k.score
                WHEN k.term LIKE 'brigad%' THEN k.score
                WHEN k.term LIKE 'fire%' THEN k.score
                ELSE 0
            END)
    FROM keywords k WHERE k.id = r.id
) desc;
```

Figure 6: Problem 4

**Reasoning:**This query returns uri values from the revisionuri table where the corresponding id in the keywords table has keywords that begin with "infantri", "reinforc", "brigad", and "fire". The HAVING clause requires that all four keywords exist for every id. The query also sorts the results based on the sum of the score values for these keywords, with greater scores first. The CASE expression in the ORDER BY clause combines the scores of matching keywords, so the results are ordered on the basis of the combined score for each id.
Q5.

```sql
-- problem 5
select r.uri
from revisionuri as r
inner join keywords as k on k.id = r.id
group by r.id
having
(
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') or
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') or
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') or
    exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
)
ORDER BY
(
    SELECT SUM(CASE
                WHEN k.term LIKE 'infantri%' THEN k.score
                WHEN k.term LIKE 'reinforc%' THEN k.score
                WHEN k.term LIKE 'brigad%' THEN k.score
                WHEN k.term LIKE 'fire%' THEN k.score
                ELSE 0
            END)
    FROM keywords k WHERE k.id = r.id
) desc ;
```

Figure 7: Problem 5

**Reasoning:**This query fetches uri values from the revisionuri table where the corresponding id in the keywords table contains any combination of those words beginning with "infantri", "reinforc", "brigad", or "fire". The HAVING clause verifies whether any of these keywords are present for every id. The results are also sorted by the sum of the score values for the matching keywords. The CASE expression within the ORDER BY clause returns the total score for the keywords that have matched the given patterns, with larger scores first in the result set.

Q6.

```sql
-- problem 6
select r.uri
from revisionuri as r
inner join keywords as k on k.id = r.id
group by r.id
having
(
    (
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
    ) or
    (
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
    ) or
    (
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
    ) or
    (
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
    ) or
    (
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
    ) or
    (
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
    ) or
    (
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'infantri%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'reinforc%') and
        not exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'brigad%') and
        exists(select keywords.id from keywords where keywords.id=r.id and keywords.term like 'fire%')
    )
)
ORDER BY
(
    SELECT SUM(CASE
                WHEN k.term LIKE 'reinforc%' THEN k.score
                ELSE 0
            END)
         - SUM(CASE
                WHEN k.term LIKE 'infantri%' THEN k.score
                WHEN k.term LIKE 'brigad%' THEN k.score
                WHEN k.term LIKE 'fire%' THEN k.score
                ELSE 0
            END)
    FROM keywords k WHERE k.id = r.id
) DESC;
```

Figure 8: Problem 6

**Reasoning:**This query gets the uri values of Wikipedia articles in the revisionuri table that include the stemmed keyword "reinforc" but do not include all of the keywords beginning with "infantri", "brigad", or "fire". The HAVING clause guarantees at least one of the keywords "reinforc" exists, while permitting all subsets of the keywords "infantri", "brigad", or "fire" other than the subset containing all this 3 words. In addition, the question sorts the results according to the keyword scores: if the score for "reinforc" is greater than the sum of the scores for the words "infantri", "brigad", and "fire", the article will be placed higher. The ranking is determined by subtracting the total scores of "infantri", "brigad", and "fire" from that of "reinforc" such that documents having a higher affinity towards "reinforc" are ranked higher in the results.