#
##

StackOverflow

##

Store StackOverflow Q/A on EdgeDB

### Development & Build Instructions: - Make sure you create a directory that has all files that are needed for this Assignment in the same directory - From the Assignment Directory, execute the following commands: - edgedb project init - make a note the following two files: - edgedb.TOML - dbschema/default.esdl - edit/update dbschema/default.esdl - edgedb migration create - edgedb migrate - edgedb list types - edgedb info - edgedb ui - edgedb help

### How to fetch StackOverflow Data
#### Users, Badges, Post, Tags For different tags Docker, Kubernetes, Python, GoLang, Angular, React, etc.

Following are official URLs for stackoverflow ## References: - https://stackoverflow.com/tags - https://api.stackexchange.com/docs - https://api.stackexchange.com/docs/throttle

### Restrictions on the Stackexchange API: 1. It has throttles to limit 30 requests per second/application 2. Each object has a number of fields you could access - https://api.stackexchange.com/docs/types/post - https://api.stackexchange.com/docs/types/question - https://api.stackexchange.com/docs/types/answer 3. Pagesize can be any value between 0 and 100 and defaults to 30; you could use "total" or "has_more" property to get all results - https://api.stackexchange.com/docs/paging
### See below the details for Throttles (https://api.stackexchange.com/docs/throttle) ![image.png](attachment:image.png)

Filter: to see the body of Question or Answer add **filter=withbody** to your http/request

Example: Question with Body Filter - https://api.stackexchange.com/2.3/questions/77334532?order=desc&sort=activity&site=stackoverflow&filter=withbody ![image.png](attachment:image.png)

Example: Answer with Body Filter - https://api.stackexchange.com/2.3/answers/77338398?order=desc&sort=activity&site=stackoverflow&filter=withbody ![image-2.png](attachment:image-2.png)

### Exploring Stack Exchange API In order to understand how to work with Stack Exchange data and utilize their API, you can explore the available API endpoints provided by Stack Exchange. Here's how you can get started: 1. **Go to Stack Exchange API Documentation: Visit the Stack Exchange API documentation at api.stackexchange.com. 2. **Browse API Endpoints:** The API documentation provides a list of available endpoints, each with its purpose and usage. You can browse through these endpoints to understand what data you can retrieve. 3. **Authentication:** Depending on your use case, you might need to create an application on Stack Apps and obtain an API key for authentication. Here's how you can create an API key: - Visit the Stack Exchange API documentation at [Stackexchange](https://api.stackexchange.com/). - Click on "Register for an app key" .**

![image-2.png](attachment:image-2.png)

- Sign in to your account or create a new account if you don't have one.

![image.png](attachment:image.png)

- Fill out the required information about your application.

![image-4.png](attachment:image-4.png)

- Once your application is registered, you'll get an API key that you can use for authentication in your requests.

4. **Testing Endpoints:** You can use tools like `curl`, Python's `requests` library, or specialized API testing tools to interact with the endpoints and see the responses. 5. **Experiment and Learn:** Experiment with different endpoints, query parameters, and filters to tailor your requests to your specific needs. Feel free to explore the API and see how you can use it.

# EdgeDB - Download __[EdgeDB](https://www.edgedb.com/install)__ to your laptop - Getting Started with __[EdgeDB](https://www.edgedb.com/docs/intro/index)__ - Quick Start with __[Quickstart](https://www.edgedb.com/docs/intro/quickstart#ref-quickstart)__ - Data modeling in EdgeDB __[Data Model](https://www.edgedb.com/showcase/data-modeling)__ - EdgeDB's Schema Definition Language __[ESDL](https://www.edgedb.com/docs/datamodel/index)__ ## The three major platofrms are supported: 1. Windows 2. MacOS 3. Linux ## EdgeDB instance - A quickstart tutorial will walk you through the entire process of creating a simple EdgeDB-powered application __[Quick Start Tutorial](https://www.edgedb.com/docs/intro/quickstart)__ - Make sure to initialize your EdgeDB project from your current project directory and type from the terminal/command prompt the following command: - edgedb project init ## EdgeDB Migration - After you update your default.esdl in your current project **dbschema** directory, execute from the terminal/command prompt the following commands in the sequence listed below: - edgedb migration create - edgedb migrate ## EdgeDB Instance Destroy - Once your are done coding/testing, you could get a list of EdgeDB instance names and destroy/kill them using the following commands: - edgedb instance list - edgedb instance destroy -I "your_instance_name" --force ## EdgeDB UI - You can use the **EdgeDB UI**, the admin dashboard baked into every EdgeDB instance when you are instrumenting with your queries and requirements - Type the following command from a terminal/window to start the edgedb ui in a browser: - edgedb ui ## EdgeQL - https://www.edgedb.com/docs/edgeql/index
## GraphQL - https://www.edgedb.com/docs/clients/graphql/index

## EdgeDB Python Driver - Install the package (https://www.edgedb.com/docs/clients/python/installation#edgedb-python-installation) - pip install edgedb - Examples:
- Basic Usage :(https://www.edgedb.com/docs/clients/python/usage)
- AsyncIO API: (https://www.edgedb.com/docs/clients/python/api/asyncio_client#edgedb-python-asyncio-api-reference)
- Blocking API: (https://www.edgedb.com/docs/clients/python/api/blocking_client#edgedb-python-blocking-api-reference)

### StackOverflow EdgeDB Database - Create the Graph-Relational Data Model for StackOverFlow using **EdgeDB** - Insert the Post's objects in the EdgeDB database - Your Object Data Model must consider the following objects/attributes - User - Badge - Post - Question - Answer - Comments - Tags - TagSynonyms - Etc.

module default { type Badge { required property user_id -> int32; property badge_id -> int32; property name -> str; property award_count -> int32; property rank -> str; property badge_type -> str; link to_user -> User; }; type User { property account_id -> int32; property is_employee -> bool; property last_modified_date -> str; property last_access_date -> str; property reputation_change_year -> int32; property reputation_change_quarter -> int32; property reputation_change_month -> int32; property reputation_change_week -> int32; property reputation_change_day -> int32; property reputation -> int32; property creation_date -> str; property user_type -> str; required property user_id -> int32; property location -> str; property website_url -> str; property display_name -> str; multi link has_posts := . int64; property score -> int32; property last_activity_date -> int64; property creation_date -> int64; property post_type -> str; required property user_id ->int32; link to_Question -> Question; link to_Answer -> Answer; link post_by_user -> User; multi link has_comments -> Comments } type Question{ property post_id -> int64; required property user_id ->int32; property body -> str; property title -> str; property question_id -> str; property creation_date -> str; property is_answered -> str; property score -> int32; multi link has_answer := . User; multi link has_Tags -> QuestionsRelatedTagsInfo; } type QuestionsRelatedTagsInfo{ required property question_id ->str; property tag -> str; link tagInformation -> Tags; } type Answer{ required property user_id ->int32; property post_id -> int64; property answer_id -> str; property question_id -> str; property body -> str; property account_id -> str; property reputation -> str; property score -> int32; property creation_date -> str; link to_Question -> Question; link by_user -> User; } type Comments{ property user_id ->int32; property user_type -> str; property score -> int32; property edited -> str; property creation_date -> str; property post_id -> int64; property comment_id -> int64; } type UserRelatedTagInfo{ required property user_id ->int32; property tagName -> str; multi link related_users -> User; } type Tags{ name -> str; count -> int64; is_require -> str; is_moderator_only -> str; multi link used_by -> UserRelatedTagInfo; multi link has_synonyms -> TagSynonyms; } type TagSynonyms{ creation_date -> str; last_applied_date -> str; applied_count -> int32; to_tag -> str; from_tag -> str; } }

Use networkx package to create the graph data model of the knowledge graph (Network) for StackOverflow https://networkx.org/documentation/stable/install.html

In [55]:
```
# ## Installing Dependencies

!pip install pandas
!pip install datetime
```

```
Requirement already satisfied: pandas in /Users/anushasp/anaconda3/lib/python3.11/site-packages (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /Users/anushasp/anaconda3/lib/python3.11/site-package
s (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/anushasp/anaconda3/lib/python3.11/site-packages (from pa
ndas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in /Users/anushasp/anaconda3/lib/python3.11/site-packages (from
pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in /Users/anushasp/anaconda3/lib/python3.11/site-packages (from p
andas) (1.24.3)
Requirement already satisfied: six>=1.5 in /Users/anushasp/anaconda3/lib/python3.11/site-packages (from python
-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: datetime in /Users/anushasp/anaconda3/lib/python3.11/site-packages (5.2)
Requirement already satisfied: zope.interface in /Users/anushasp/anaconda3/lib/python3.11/site-packages (from
datetime) (5.4.0)
Requirement already satisfied: pytz in /Users/anushasp/anaconda3/lib/python3.11/site-packages (from datetime)
(2023.3.post1)
Requirement already satisfied: setuptools in /Users/anushasp/anaconda3/lib/python3.11/site-packages (from zop
e.interface->datetime) (68.0.0)
```

Pull past-year data from stackexchange api

In [ ]:

In [56]:
```
# Import all the required packages
import pandas as pd
from dateutil.relativedelta import relativedelta
import datetime
import time
from datetime import timedelta
from datetime import datetime
import json
import requests
import edgedb

import warnings


warnings.filterwarnings('ignore')
```

In [ ]:

In [57]:
```
#In order to get the API key please refer to authentication section in Exploring Stack Exchange API
key = 'li8fkKefhocLtXvITPqPGQ(('
```

In [ ]:

In [58]:
```
# Set UNIT_TESTING to True when running unit tests.
UNIT_TESTING = True

# Make sure to switch UNIT_TESTING to False when fetching the whole dataset.
# This variable helps control whether you are performing unit testing or fetching complete data.
# For unit testing, keep it True; for data retrieval, set it to False.
```

In [ ]:

In [59]:
```
# Create an EdgeDB client for database interactions
```

```
client = edgedb.create_client()
```

In [ ]:

In [60]:
```
# In order to fetch last year data calculate the 'fromdate' as the timestamp of today minus 13 months

# For unit-testing purposes, collect data for past month only, later change it to 13 months

# Note: it will take roughly 20 minutes for collecting 1 month data from StackOverflow.

fromdate = str(int(datetime.timestamp(datetime.today() - relativedelta(days=7))))

# Calculate the 'todate' as the timestamp of today's date
todate = str(int(datetime.timestamp(datetime.today() - relativedelta(days=0))))
print(fromdate, todate)
```

```
1699474612 1700079412
```

In [ ]:

In [61]:
```
# Retrieve user data from the Stack Exchange API.
# This function sends requests to the Stack Exchange API to fetch user data
# and aggregates it into a list. It supports pagination to retrieve multiple pages of user data.

def fetchUsersData():
    data = []
    page = 1
    response = requests.get('https://api.stackexchange.com/2.3/users?pagesize=100&page=' + str(page) + '&site=s
    responseJson = json.loads(response.text)
    if "items" in responseJson:
        data += responseJson['items']

    while 'has_more' in responseJson and responseJson['has_more']:
        if(UNIT_TESTING):
            break
        if page%28 == 0:
            time.sleep(2)
        if 'backoff' in responseJson:
            backoff_time = responseJson['backoff']
            time.sleep(backoff_time)
        time.sleep(2)
        page += 1
        response = requests.get('https://api.stackexchange.com/2.3/users?pagesize=100&page=' + str(page) + '&s
        responseJson = json.loads(response.text)
        if "items" in responseJson:
            data += responseJson['items']

    print("Here is the snippet of the Users Data fetched")
    if data is not None and len(data) > 0:
        print(data[0])
    else:
        print("Users Data is either None or empty.")
    print()
    return data
```

In [ ]:

In [62]:
```
# Extract and insert tags for each user from their 'userData' into the 'Tags' key.
def extractUserTagsData(userData):
    for user in userData:
        # Extract tags for each user
        badge_counts = user.get('badge_counts', {})
        collectives = user.get('collectives', [])
        tags = []

        for collective in collectives:
            collective_tags = collective.get('collective', {}).get('tags', [])
            tags.extend(collective_tags)

        # Insert the 'Tag' key into the user's data with the list of tags
        user['Tag'] = tags
```

In [ ]:

In [63]:
```
# Fetch and return badge data for a list of users from Stack Exchange API.

def fetchBadges(data):
    badgesData = []
    page = 1
    for user_data in data:
        response = requests.get('https://api.stackexchange.com/2.3/users/'+ str(user_data['user_id']) +'/badge
        responseJson = json.loads(response.text)
        #print(response.text)
        if "items" in responseJson:
            badgesData += responseJson['items']

        while 'has_more' in responseJson and responseJson['has_more']:
            if(UNIT_TESTING):
                break
            if page%28 == 0:
                time.sleep(2)
            if 'backoff' in responseJson:
                backoff_time = responseJson['backoff']
                time.sleep(backoff_time)
            time.sleep(2)
            page += 1
            response = requests.get('https://api.stackexchange.com/2.3/users/'+ str(user_data['user_id']) +'/ba
            responseJson = json.loads(response.text)
            if "items" in responseJson:
                badgesData += responseJson['items']

    print("Here is the snippet of the User Badges Data fetched")
    if badgesData is not None and len(badgesData) > 0:
        print(badgesData[0])
    else:
        print("Badges Data is either None or empty.")
    print()
    return badgesData
```

In [ ]:

In [64]:
```
# Insert badge data into the database for a list of users using EdgeQL queries.

def insertBadges(data):
    for badge_data in data:
        user_id = badge_data["user"]["user_id"]
        badge_id = badge_data.get('badge_id', None)
        name = badge_data.get('name', '')
        award_count = badge_data.get('award_count', 0)
        rank = badge_data.get('rank', "")
        badge_type = badge_data.get('badge_type', '')

        client.query("""
                INSERT Badge {
                    user_id := <int32>$user_id,
                    badge_id := <int32>$badge_id,
                    name := <str>$name,
                    award_count := <int32>$award_count,
                    rank := <str>$rank,
                    badge_type := <str>$badge_type,
                    to_user := (
                            select User
                            filter
                                .user_id = <int32>$user_id
                                limit 1
                        )
                };
                """,
                user_id=user_id,
                badge_id=badge_id,
                name=name,
                award_count=award_count,
                rank=rank,
                badge_type=badge_type
            );
```

In [ ]:

In [65]:
```
# Insert user data into the database for a list of users using EdgeQL queries.

def insertUsersIntoEdgeDb(data):
    for user_data in data:
        client.query("""
                INSERT User{
                    account_id := <int32>$account_id,
                    is_employee := <bool>$is_employee,
                    last_modified_date := <str>$last_modified_date,
                    last_access_date := <str>$last_access_date,
                    reputation_change_year := <int32>$reputation_change_year,
                    reputation_change_quarter := <int32>$reputation_change_quarter,
                    reputation_change_month := <int32>$reputation_change_month,
                    reputation_change_week := <int32>$reputation_change_week,
                    reputation_change_day := <int32>$reputation_change_day,
                    reputation := <int32>$reputation,
                    creation_date := <str>$creation_date,
                    user_type := <str>$user_type,
                    user_id := <int32>$user_id,
                    location := <str>$location,
                    website_url := <str>$website_url,
                    display_name := <str>$display_name

                } """,
                account_id=user_data.get('account_id', 0),
                is_employee=user_data.get('is_employee', False),
                last_modified_date=str(user_data.get('last_modified_date', "")),
                last_access_date=str(user_data.get('last_access_date', "")),
                reputation_change_year=user_data.get('reputation_change_year', 0),
                reputation_change_quarter=user_data.get('reputation_change_quarter', 0),
                reputation_change_month=user_data.get('reputation_change_month', 0),
                reputation_change_week=user_data.get('reputation_change_week', 0),
                reputation_change_day=user_data.get('reputation_change_day', 0),
                reputation=user_data.get('reputation', 0),
                creation_date=str(user_data.get('creation_date', "")),
                user_type=user_data.get('user_type', ''),
                user_id=user_data.get('user_id', 0),
                location=user_data.get('location', ''),  # Use an empty string if 'location' is not present
                website_url=user_data.get('website_url', ''),
                display_name=user_data.get('display_name', '')
                );

        #inserting Tags
        for tag in user_data['Tag']:
            client.query("""
                    INSERT UserRelatedTagInfo{
                        user_id := <int32>$user_id,
                        tagName := <str>$tagName,
                        related_users:= (
                        select User
                            filter
                                .user_id = <int32>$user_id
                        )
                    }
                """, user_id=user_data.get('user_id', 0), tagName = tag);
```

In [ ]:

In [66]:
```
# Fetches posts from the Stack Exchange API and returns the data.

def fetchPostData():
    data = []
    page = 1
    response = requests.get('https://api.stackexchange.com/2.3/posts?order=desc&sort=activity&pagesize=100&page
    responseJson = json.loads(response.text)
    if "items" in responseJson:
        data += responseJson['items']

    while 'has_more' in responseJson and responseJson['has_more']:
        if(UNIT_TESTING):
            break
        if page%28 == 0:
            time.sleep(2)
        if 'backoff' in responseJson:
            backoff_time = responseJson['backoff']
            time.sleep(backoff_time)
        time.sleep(2)
        page += 1
        response = requests.get('https://api.stackexchange.com/2.3/posts?order=desc&sort=activity&pagesize=100&
        responseJson = json.loads(response.text)
        if "items" in responseJson:
            data += responseJson['items']

    print("Here is the snippet of the Posts Data fetched")
    if data is not None and len(data) > 0:
        print(data[0])
    else:
        print("Posts Data is either None or empty.")
    print()
    return data
```

In [ ]:

In [67]:
```
#fetching questions by using post_id's
def fetchQuestionsData(post_id):
    try:
        response = requests.get('https://api.stackexchange.com/2.3/questions/'+ str(post_id) + '?order=desc&so
        response_json = json.loads(response.text)
        return response_json
    except json.decoder.JSONDecodeError:
        # Handle the case where the response is not valid JSON
        print("Invalid JSON response for post_id:", post_id)
        return None
```

```
In [ ]:
```

```
In [68]: #fetching answers by using post_id's
def fetchAnswersData(post_id):
    try:
        response = requests.get('https://api.stackexchange.com/2.3/answers/'+ str(post_id) + '?order=desc&sort=
        response_json = json.loads(response.text)
        return response_json
    except json.decoder.JSONDecodeError:
        # Handle the case where the response is not valid JSON
        print("Invalid JSON response for post_id:", post_id)
        return None
```

```
In [ ]:
```

```
In [69]: def fetchQuestionsAndAnswersData(postData):
    questionsData = []
    answersData = []
    questionsCount = 0
    answersCount = 0
    questionIds = ""
    answerIds = ""
    for item in postData:
        POST_TYPE = item["post_type"]
        post_id = item["post_id"]
        if(POST_TYPE == "question"):
            questionIds+=str(post_id)+";"
            questionsCount +=1
            if(questionsCount == 100):
                questionsDatatemp = fetchQuestionsData(questionIds[:-1])
                if(questionsDatatemp is not None and "items" in questionsDatatemp and len(questionsDatatemp["i
                    for question in questionsDatatemp["items"]:
                        question["post_id"] = post_id
                        questionsData.append(question)
                questionIds = ""
                questionsCount = 0

        elif(POST_TYPE == "answer"):
            answerIds+=str(post_id)+";"
            answersCount +=1
            if(answersCount == 100):
                answersDatatemp = fetchAnswersData(answerIds[:-1])
                if(answersDatatemp is not None and "items" in answersDatatemp and len(answersDatatemp["items"]
                    for answer in answersDatatemp["items"]:
                        answer["post_id"] = post_id
                        answersData.append(answer)
                answerIds = ""
                answersCount = 0

    print("Here is the snippet of the Questions Data fetched")
    if questionsData is not None and len(questionsData) > 0:
        print(questionsData[0])
    else:
        print("questionsData is either None or empty.")
    print()

    print("Here is the snippet of the Answers Data fetched")
    if answersData is not None and len(answersData) > 0:
        print(answersData[0])
    else:
        print("answersData is either None or empty.")
    print()
    return questionsData, answersData
```

```
In [ ]:
```

```
In [70]: # Fetches comments from the Stack Exchange API and returns the data.

def fetchComments(postData):
    data = []
    page = 1
    post_ids = [str(post['post_id']) for post in postData]
    # Split into 100 batches
    batch_size = 100
    batches = [post_ids[i:i+batch_size] for i in range(0, len(post_ids), batch_size)]

    for i, batch in enumerate(batches):
        batch_str = ';'.join(batch)
        response = requests.get('https://api.stackexchange.com/2.3/posts/' + str(batch_str)+ '/comments?order=
        responseJson = json.loads(response.text)
        if "items" in responseJson:
            data += responseJson['items']
        while 'has_more' in responseJson and responseJson['has_more']:
            if(UNIT_TESTING):
                break
            if page%28 == 0:
                time.sleep(2)
            if 'backoff' in responseJson:
                backoff_time = responseJson['backoff']
                time.sleep(backoff_time)
            time.sleep(2)
            page += 1
            response = requests.get('https://api.stackexchange.com/2.3/posts/' + str(batch_str)+ '/comments?or
            responseJson = json.loads(response.text)
            if "items" in responseJson:
                data += responseJson['items']

    print("Here is the snippet of the Comments Data fetched")
    if data is not None and len(data) > 0:
        print(data[0])
    else:
        print("Comments Data is either None or empty.")
    print()
    return data
```

In [ ]:

In [71]:
```
# Insert answers data into EdgeDB.

def insertAnswersData(answersData):
    for answer in answersData:
        owner = answer.get("owner", {})
        client.query("""
            INSERT Answer {
                user_id:=<int32>$user_id,
                post_id :=<int64>$post_id,
                answer_id := <str>$answer_id,
                body:= <str>$body,
                question_id :=<str>$question_id,
                account_id := <str>$account_id,
                reputation := <str>$reputation,
                score :=<int32>$score,
                creation_date :=<str>$creation_date,
                to_Question := (
                    select Question
                    filter
                        .question_id = <str>$question_id
                        limit 1
                ),
                by_user := (
                    select User
                    filter
                        .user_id = <int32>$user_id
                    limit 1
                )
            }
                """,
                user_id = owner.get("user_id", 0),
                post_id = answer.get("post_id", 0),
                account_id = str(owner.get("account_id", None)),
                reputation = str(owner.get("reputation", None)),
                body = answer.get("body"),
                score = answer.get("score", 0),
                creation_date = str(answer.get("creation_date", None)),
                answer_id = str(answer.get("answer_id", None)),
                question_id = str(answer.get("question_id", None))

            );
```

In [ ]:

In [72]:
```
# Insert questions data into EdgeDB.

def insertQuestionsData(questionsData):
    for question in questionsData:
        owner = question.get("owner", {})
        for tag in question['tags']:
            tag_name = tag
            client.query("""
                INSERT QuestionsRelatedTagsInfo {
                    question_id:=<str>$question_id,
                    tag:=<str>$tag,
                    tagInformation := (
                        select Tags
                        filter
                            .name = <str>$tag
                        limit 1
                    )

                }
                """,
                question_id = str(question.get('question_id', 0)),
                tag = tag_name
            );
        client.query("""
            INSERT Question {
                post_id :=<int64>$post_id,
                user_id:=<int32>$user_id,
                title :=<str>$title,
                question_id :=<str>$question_id,
                creation_date :=<str>$creation_date,
                is_answered :=<str>$is_answered,
                body:=<str>$body,
                score:=<int32>$score,
                question_by_user := (
                    select User
                    filter
                        .user_id = <int32>$user_id
                    limit 1
                ),
                has_Tags :=(
                    select QuestionsRelatedTagsInfo
                    filter
                        .question_id = <str>$question_id
                    )
            }
                """,
                post_id = question.get('post_id', 0),
                user_id = owner.get("user_id", 0),
                title = question.get('title', 'Unknown Title'),
                body = question.get("body",""),
                question_id = str(question.get('question_id', '0')),
                creation_date = str(question.get('creation_date', '0')),
                is_answered = str(question.get('is_answered', 'False')),
                score = question.get('score', 0)

            );
```

In [ ]:

In [73]:
```
# Insert comments data into EdgeDB.

def insertCommentsData(commentsData):
    for comment in commentsData:
        owner = comment.get("owner", {})
        user_id = owner.get("user_id", 0)
        user_type = owner.get("user_type", None)
        score = comment.get("score", 0)
        edited = str(comment.get("edited", None))
        creation_date = str(comment.get("creation_date", None))
        post_id = comment.get("post_id", 0)
        comment_id = comment.get("comment_id", 0)
        client.query("""
            INSERT Comments {
                user_id:=<int32>$user_id,
                user_type := <str>$user_type,
                score:=<int32>$score,
                edited :=<str>$edited,
                creation_date :=<str>$creation_date,
                post_id :=<int64>$post_id,
                comment_id := <int64>$comment_id
            }
                """,
                 user_id = user_id,
                 user_type = user_type,
                 score = score,
                 edited =edited,
                 creation_date = creation_date,
                 post_id = post_id,
                 comment_id = comment_id

            );
```

In [ ]:

In [74]:
```
# Insert user posts into EdgeDB

def insertPostData(data):
    for post_data in data:
        #print(post_data)
        post_id = post_data.get("post_id")
        score = post_data.get("score", 0)
        last_activity_date = post_data.get("last_activity_date", 0)
        creation_date = post_data.get("creation_date", 0)
        post_type = post_data.get("post_type", "unknown")
        user_data = post_data.get("owner")
        user_id = user_data.get("user_id", 0) if user_data is not None else 0

        #print(user_id)
        if not user_id == 0:
            client.query("""
                    INSERT Post {
                        post_id := <int64>$post_id,
                        score := <int32>$score,
                        last_activity_date := <int64>$last_activity_date,
                        creation_date := <int64>$creation_date,
                        post_type := <str>$post_type,
                        user_id := <int32>$user_id,
                        to_Question := (
                                select Question
                                filter
                                    .post_id = <int64>$post_id
                                    limit 1
                            ),
                        to_Answer := (
                                select Answer
                                filter
                                    .post_id = <int64>$post_id
                                    limit 1
                            ),
                        post_by_user := (
                                select User
                                filter
                                    .user_id = <int32>$user_id
                                    limit 1
                            ),
                        has_comments := (
                                select Comments
                                filter
                                    .post_id = <int64>$post_id
                            )
                        };
                    """,
                     post_id=post_id,
                     score=score,
                     last_activity_date=last_activity_date,
                     creation_date=creation_date,
                     post_type=post_type,
                     user_id=user_id

                );
```

```
In [ ]:
```

```
In [75]: def fetchTagsData():
             data = []
             page = 1
             response = requests.get('https://api.stackexchange.com/2.3/tags?order=desc&sort=popular&pagesize=100&page=
             responseJson = json.loads(response.text)
             #print(response.text)
             if "items" in responseJson:
                 data += responseJson['items']

             while 'has_more' in responseJson and responseJson['has_more']:
                 if(UNIT_TESTING):
                     break
                 if page%28 == 0:
                     time.sleep(2)
                 if 'backoff' in responseJson:
                     backoff_time = responseJson['backoff']
                     time.sleep(backoff_time)
                 time.sleep(2)
                 page += 1
                 response = requests.get('https://api.stackexchange.com/2.3/tags?order=desc&sort=popular&pagesize=100&pa
                 responseJson = json.loads(response.text)
                 if "items" in responseJson:
                     data += responseJson['items']

             print("Here is the snippet of the Tags Data fetched")
             if data is not None and len(data) > 0:
                 print(data[0])
             else:
                 print("Tags Data is either None or empty.")
             print()
             return data
```

```
In [ ]:
```

```
In [76]: def fetchTagSynonymsData(tagsData):
             data = []
             page = 1
             for tag in tagsData:
                 response = requests.get('https://api.stackexchange.com/2.3/tags/' + tag.get('name')+ '/synonyms?order=
                 responseJson = json.loads(response.text)
                 #print(response.text)
                 if "items" in responseJson:
                     data += responseJson['items']

                 while 'has_more' in responseJson and responseJson['has_more']:
                     if(UNIT_TESTING):
                         break
                     if page%28 == 0:
                         time.sleep(2)
                     if 'backoff' in responseJson:
                         backoff_time = responseJson['backoff']
                         time.sleep(backoff_time)
                     time.sleep(2)
                     page += 1
                     response = requests.get('https://api.stackexchange.com/2.3/tags/' + tag.get('name')+ '/synonyms?or
                     responseJson = json.loads(response.text)
                     if "items" in responseJson:
                         data += responseJson['items']

             print("Here is the snippet of the Tag Synonyms Data fetched")
             if data is not None and len(data) > 0:
                 print(data[0])
             else:
                 print("Tag Synonyms Data is either None or empty.")
             print()
             return data
```

```
In [ ]:
```

```
In [77]: def insertTagsData(tagsData):
             for tag in tagsData:
                 client.query("""
                     INSERT Tags {
                         name :=<str>$name,
                         count:=<int64>$count,
                         is_require :=<str>$is_require,
                         is_moderator_only :=<str>$is_moderator_only,
                         used_by := (
                                         select UserRelatedTagInfo
                                         filter
                                             .tagName = <str>$name
                                     ),
                         has_synonyms := (
                                         select TagSynonyms
```

```
                                filter
                                    .to_tag = <str>$name
                            )
                }
                """,
                name = tag.get('name', None),
                count = tag.get('count', 0),
                is_require = tag.get('is_require', "False"),
                is_moderator_only = str(tag.get('is_moderator_only', "False"))
            );
```

In [ ]:

```

```

In [78]:

```
def insertTagSynonymsData(tagSynonymsData):
    for tagSynonym in tagSynonymsData:
        client.query("""
                INSERT TagSynonyms {
                    creation_date := <str>$creation_date,
                    last_applied_date := <str>$last_applied_date,
                    applied_count := <int64>$applied_count,
                    to_tag := <str>$to_tag,
                    from_tag := <str>$from_tag
                };
                """,
                creation_date= str(tagSynonym.get('creation_date', None)),
                last_applied_date= str(tagSynonym.get('last_applied_date', None)),
                applied_count= tagSynonym.get('applied_count', 0),
                to_tag= tagSynonym.get('to_tag', None),
                from_tag= tagSynonym.get('from_tag', None)
            );
```

In [ ]:

```

```

In [79]:

```
start_time = time.time()
userData = fetchUsersData()
print("fetchUsersData took", time.time() - start_time, "seconds")

start_time = time.time()
extractUserTagsData(userData)
print("extractUserTagsData took", time.time() - start_time, "seconds")

start_time = time.time()
badgesData = fetchBadges(userData)
print("fetchBadges took", time.time() - start_time, "seconds")

start_time = time.time()
tagsData = fetchTagsData()
print("fetchTagsData took", time.time() - start_time, "seconds")

start_time = time.time()
tagSynonymsData = fetchTagSynonymsData(tagsData)
print("fetchTagSynonymsData took", time.time() - start_time, "seconds")

start_time = time.time()
postData = fetchPostData()
print("fetchPostData took", time.time() - start_time, "seconds")

start_time = time.time()
questionsData, answersData = fetchQuestionsAndAnswersData(postData)
print("fetchQuestionsAndAnswersData took", time.time() - start_time, "seconds")

start_time = time.time()
commentsData = fetchComments(postData)
print("fetchComments took", time.time() - start_time, "seconds")
```

```
Here is the snippet of the Users Data fetched
{'badge_counts': {'bronze': 5, 'silver': 0, 'gold': 0}, 'account_id': 29861972, 'is_employee': False, 'last_ac
cess_date': 1700065233, 'reputation_change_year': 172, 'reputation_change_quarter': 172, 'reputation_change_mo
nth': 172, 'reputation_change_week': 20, 'reputation_change_day': 10, 'reputation': 173, 'creation_date': 1699
516667, 'user_type': 'registered', 'user_id': 22885423, 'link': 'https://stackoverflow.com/users/22885423/jtl3
13', 'profile_image': 'https://www.gravatar.com/avatar/3c17a090fe54eb10861d535d395bdcaf?s=256&d=identicon&r=PG
&f=y&so-version=2', 'display_name': 'jtl313'}

fetchUsersData took 0.6461150646209717 seconds
extractUserTagsData took 0.0001742839813232422 seconds
Here is the snippet of the User Badges Data fetched
{'user': {'account_id': 29861972, 'reputation': 173, 'user_id': 22885423, 'user_type': 'registered', 'profile_
image': 'https://www.gravatar.com/avatar/3c17a090fe54eb10861d535d395bdcaf?s=256&d=identicon&r=PG&f=y&so-versio
n=2', 'display_name': 'jtl313', 'link': 'https://stackoverflow.com/users/22885423/jtl313'}, 'badge_type': 'nam
ed', 'award_count': 1, 'rank': 'bronze', 'badge_id': 2, 'link': 'https://stackoverflow.com/badges/2/student',
'name': 'Student'}

fetchBadges took 16.209375143051147 seconds
Here is the snippet of the Tags Data fetched
{'has_synonyms': False, 'is_moderator_only': False, 'is_required': False, 'count': 27, 'name': 'mysql-error-10
68'}

fetchTagsData took 0.20665979385375977 seconds
Here is the snippet of the Tag Synonyms Data fetched
{'creation_date': 1677976681, 'last_applied_date': 1699511152, 'applied_count': 1, 'to_tag': 'do-not-use-typo-
in-tag', 'from_tag': 'state-managment'}

fetchTagSynonymsData took 11.59232211112976 seconds
Here is the snippet of the Posts Data fetched
{'owner': {'account_id': 7830559, 'reputation': 242, 'user_id': 5920776, 'user_type': 'registered', 'profile_i
mage': 'https://www.gravatar.com/avatar/f80dec7fca63c7799b1429a2fd85dda5?s=256&d=identicon&r=PG&f=y&so-version
=2', 'display_name': 'Dave R', 'link': 'https://stackoverflow.com/users/5920776/dave-r'}, 'score': 0, 'last_ac
tivity_date': 1700079420, 'creation_date': 1700078335, 'post_type': 'question', 'post_id': 77490577, 'content_
license': 'CC BY-SA 4.0', 'link': 'https://stackoverflow.com/q/77490577'}

fetchPostData took 0.2178349494934082 seconds
Here is the snippet of the Questions Data fetched
questionsData is either None or empty.

Here is the snippet of the Answers Data fetched
answersData is either None or empty.

fetchQuestionsAndAnswersData took 0.00015020370483398438 seconds
Here is the snippet of the Comments Data fetched
{'owner': {'account_id': 129540, 'reputation': 210371, 'user_id': 328193, 'user_type': 'registered', 'accept_r
ate': 91, 'profile_image': 'https://i.stack.imgur.com/FnwU5.jpg?s=256&g=1', 'display_name': 'David', 'link':
'https://stackoverflow.com/users/328193/david'}, 'edited': False, 'score': 0, 'creation_date': 1700079431, 'po
st_id': 77490639, 'comment_id': 136612054, 'content_license': 'CC BY-SA 4.0'}

fetchComments took 0.17219924926757812 seconds
```

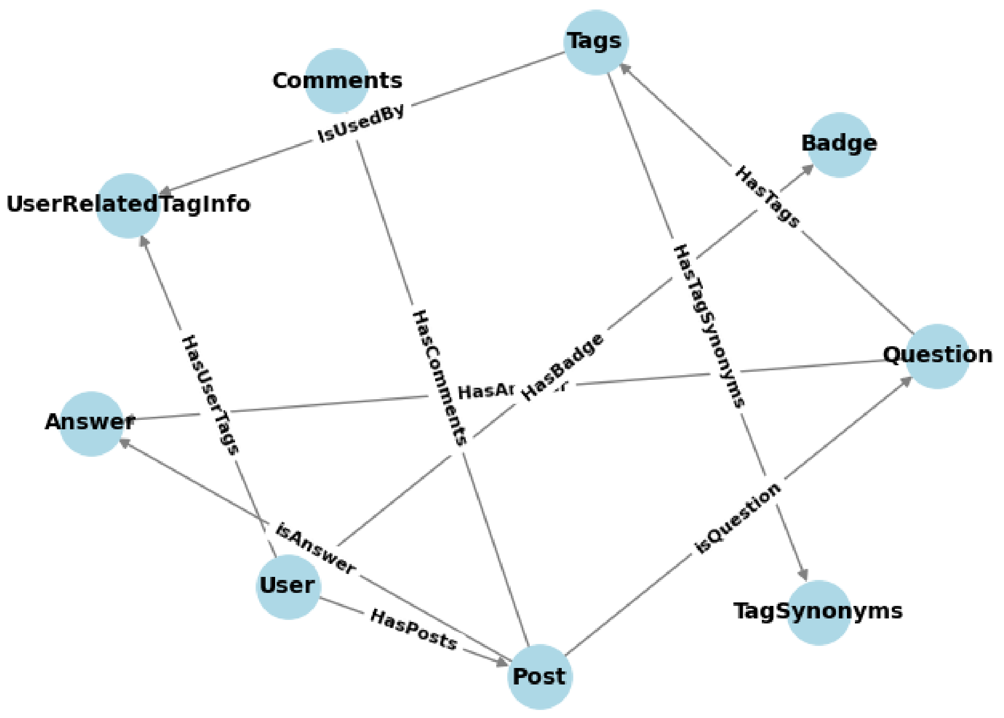
In [80]:
```python
# Insert user data and badges data into EdgeDB.
insertUsersIntoEdgeDb(userData)
insertBadges(badgesData)

#Insert Tags and TagSynonyms in EdgeDB
insertTagSynonymsData(tagSynonymsData)
insertTagsData(tagsData)

#Insert Questions, Answers, Comments and Post into EdgeDB
insertQuestionsData(questionsData)
insertAnswersData(answersData)
insertCommentsData(commentsData)
insertPostData(postData)
```

In [ ]:

In [81]:
```python
#Add your code for this requirement in this cell
import networkx as nx
import matplotlib.pyplot as plt

# Create a graph object
G = nx.DiGraph()

# Add nodes for each object
G.add_node('Tags')
G.add_node('Question')
G.add_node('Answer')
G.add_node('User')
G.add_node('Post')
G.add_node('Comments')
G.add_node('Badge')
G.add_node('TagSynonyms')
G.add_node('UserRelatedTagInfo')
```

```
# Add edges for each link
G.add_edge('Question', 'Tags', label='HasTags')
G.add_edge('Question', 'Answer', label='HasAnswer')
G.add_edge('User', 'Post', label = 'HasPosts')
G.add_edge('User', 'Badge', label = 'HasBadge')
G.add_edge('Post', 'Question', label = 'isQuestion')
G.add_edge('Post', 'Answer', label = 'isAnswer')
G.add_edge('Post', 'Comments', label = 'HasComments')
G.add_edge('Tags', 'TagSynonyms', label = 'HasTagSynonyms')
G.add_edge('User', 'UserRelatedTagInfo', label = 'HasUserTags')
G.add_edge('Tags', 'UserRelatedTagInfo', label = 'IsUsedBy')




# Draw the graph
pos = nx.spring_layout(G, k = 20, seed=20)
nx.draw(G, pos, with_labels=True, font_size=10, node_size=800, node_color='lightblue', edge_color='gray', font_

# Add edge labels
edge_labels = nx.get_edge_attributes(G, 'label')
nx.draw_networkx_edge_labels(G, pos, edge_labels, font_size=8, font_weight='bold')

plt.title("Graph data model for StackOverFlow")
plt.show()
```

Graph data model for StackOverFlow


Tags
Comments
IsUsedBy
Badge
HasTags
UserRelatedTagInfo
HasTagSynonyms
HasUserTags
HasComments
Question
HasAr
HasBadge
Answer
isAnswer
isQuestion
User
HasPosts
TagSynonyms
Post