–Gruppe G Erklärung Abfragen CA 3 :

1.) Auflisten der Posts, die von einem Account gemacht wurden bzw. ihm zugeordnet wurden:

*query = "select * from Tweets._default.posts where user_id = " + str(current_user)*

Diese Query erlaubt es uns, sämtliche Informationen über Posts abzurufen, die von “current_User” verfasst wurden.

2.) Finden der 100 Accounts mit den meisten Followern:

query = "select user_id,ARRAY_LENGTH(followers_id) from Tweets._default.new_accounts order by ARRAY_LENGTH(followers_id) desc limit 100"

Unsere Datenstruktur erlaubt es, für jeden Account die Länge des Arrays auszugeben, in dem für die Accounts die IDs vermerkt sind, die dem jeweiligen Account folgen.

Danach lassen sich diese einfach dieser Länge nach sortieren.

Sortiert man in absteigender Reihenfolge, so erhält man die 100 Accounts, die die meisten Follower haben.

3.) Finden der 100 Accounts, die den meisten der Accounts folgen, die in 1) gefunden wurden:

Mit der folgenden Query werden wieder die user_id’s der Top 100 Accounts abgefragt.

sql_query_100 = "select user_id from Tweets._default.new_accounts order by ARRAY_LENGTH(followers_id) desc limit 100"

Diese werden dann in eine Liste konvertiert, welche dann in die folgende Abfrage integriert wird:

sql_query_followers = "select user_id, ARRAY_LENGTH(ARRAY_INTERSECT(" + str(following_accs) + ", following_id)) from Tweets._default.new_accounts order by ARRAY_LENGTH(ARRAY_INTERSECT("+ str(following_accs) + ", following_id))

Hier wird mit ARRAY_INTERSECT die Schnittmenge aus den verfolgten Accounts eines jeden Users und den Top 100 Accounts erstellt.
Der zurückgegebene Array enthält alle verfolgten Accounts der Top 100 des jeweiligen Users. Die Länge dieses Arrays wird mit *ARRAY_LENGTH* abgefragt. Sortiert man die “user_id´s” nach diese Länge, erhält man das gewünschte Ergebnis.

4.) Auflisten der Informationen für die persönliche Startseite eines beliebigen Accounts (am besten mit den in 2) gefundenen Accounts ausprobieren); die Startseite soll Folgendes beinhalten (als getrennte Queries umsetzen):

die Anzahl der Follower:
num_followers_query = "select ARRAY_LENGTH(followers_id) from Tweets._default.new_accounts where user_id = " + str(current_user)

die Anzahl der verfolgten Accounts:
num_following_query = "select ARRAY_LENGTH(following_id) from Tweets._default.new_accounts where user_id = " + str(current_user)

wahlweise die 25 neuesten oder die 25 beliebtesten Posts der verfolgten Accounts (per DB-Abfrage):
Hierzu wird zuerst, mit folgender Abfrage, die Liste der verfolgten Accounts abgerufen:
following_query = "select following_id from Tweets._default.new_accounts where user_id = " + str(current_user)

Das Ergebnis wird dann in Form einer Liste in die nächste Abfrage integriert, welche uns dann die gewünschten Ergebnisse liefert. Im gezeigten Fall gibt die Abfrage die Posts zurück, die die größte Anzahl an Likes besitzen.

*posts_query = "select * from Tweets._default.posts where user_id in " + str(following_accs) + "order by TO_NUMBER(number_of_likes) desc limit 25"*

Anmerkung: Da wir Caching in Aufgabe 5 betreiben, werden die Ergebnisse der beliebtesten Posts in den Cache abgelegt. Wird die Startseite für den selben Nutzer noch einmal angelegt, so wird keine komplette Abfrage an die gesamte Datenbank mehr gestellt, sondern nur noch der Inhalt des Caches geladen.

5.) Caching der Posts für die Startseite (vgl. 4), erfordert einen sog. Fan-Out in den Cache jedes Followers beim Schreiben eines neuen Posts:

Für die Erstellung eines neuen Posts wird in unserem System lediglich eine Nutzer-ID angegeben, von dieser der Post abgesetzt werden soll. Diese ID setzt dann einen “Standardpost” ab, der in unserem System hinterlegt ist. Dies dient lediglich zu Demonstrationszwecken, da der eigentliche Post für das “Fan-Out” beim Caching nicht relevant ist.
Dann wird mit der folgenden Abfrage die Liste der der ID folgenden Accounts geholt:

following_query = "select following_id from Tweets._default.new_accounts where user_id = " + str(user_id)

Daraufhin wird für die User, die dem Account folgen, der neue Post in den Cache angehängt.

Wir haben uns entschlossen, lediglich den Cache der Nutzer zu erweitern, für die bereits ein Cache besteht, da wir dies für ein realistischeres Szenario halten und sonst große unnötige Last auf dem System erzeugen würden.
Sonst müssten wir entweder nur die neusten Posts in den Cache schreiben, hätten dann aber alle alten Posts nicht im Cache und müssten dann nochmal eine komplette Abfrage fahren, wenn wir eine Startseite aufbauen wollen -> kein Gewinn durch Cache. Oder wir müssten für jeden Follower, der noch keinen Cache besitzt, einen Cache erzeugen, was bei Accounts mit großen Followerzahlen große Lastspitzen erzeugen würde.

6.) Auflisten der 25 beliebtesten Posts, die ein geg. Wort enthalten (falls möglich auch mit UND-Verknüpfung mehrerer Worte)

```
querystring = "select * from Tweets._default.posts where content LIKE"
 searchwords = query.split(" ")
 for i in range (len(searchwords)):
    if i == 0:
       querystring = querystring +  " \"%"+searchwords[i]+'%\"'
    else:
       querystring = querystring + " AND content LIKE" + " \"%"+searchwords[i]+'%\"'

 querystring = querystring + " order by TO_NUMBER(number_of_likes) desc limit 25"
```

Hierbei werden die gesuchten Wörter übergeben und dann mit dem “LIKE” Befehl und dem “AND” Befehl aneinander gehängt und dann die entsprechenden Posts nach Anzahl der Likes sortiert zurückgegeben.