CS 327E Lecture 13

Shirley Cohen

November 21, 2016

Plan for Today

- Reading Quiz
- MySQL + JSON
- Final Project Assignment

Readings for Today

- JSON Data Type from the MySQL Reference Manual (section 12.6)
- JSON Functions from the MySQL Reference Manual (section 13.16)

Question 1

Which of the following statements is **false** about MySQL support for JSON:

A. MySQL has a native JSON datatype

B. MySQL converts JSON documents to a binary format

C. MySQL indexes JSON documents directly

D. MySQL supplies a number of functions for operating on JSON data

E. MySQL automatically validates JSON documents that are stored in JSON columns

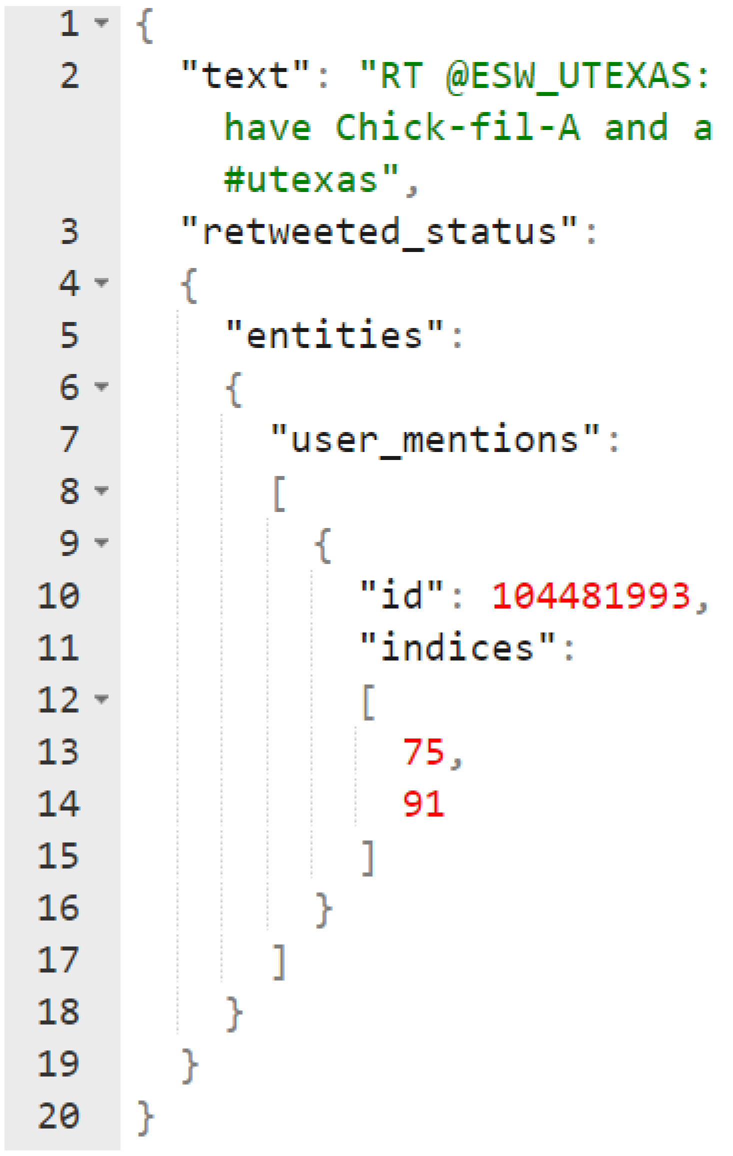
Question 2

In the context of JSON, `normalization` means the process of removing duplicate keys (or names) from a document.

A. True

B. False

Question 3



```
{
  "text": "RT @ESW_UTEXAS: Meeting tonight at 7pm in JGB 2.216! We'll
    have Chick-fil-A and a rep from @airliquidegroup will present!
    #utexas",
  "retweeted_status":
  {
    "entities":
    {
      "user_mentions":
      [
        {
          "id": 104481993,
          "indices":
          [
            75,
            91
          ]
        }
      ]
    }
  }
}
```

What does the path expression below evaluate to?

```
$.retweeted_status.entities.
user_mentions[0].indices[0]
```

A. {}

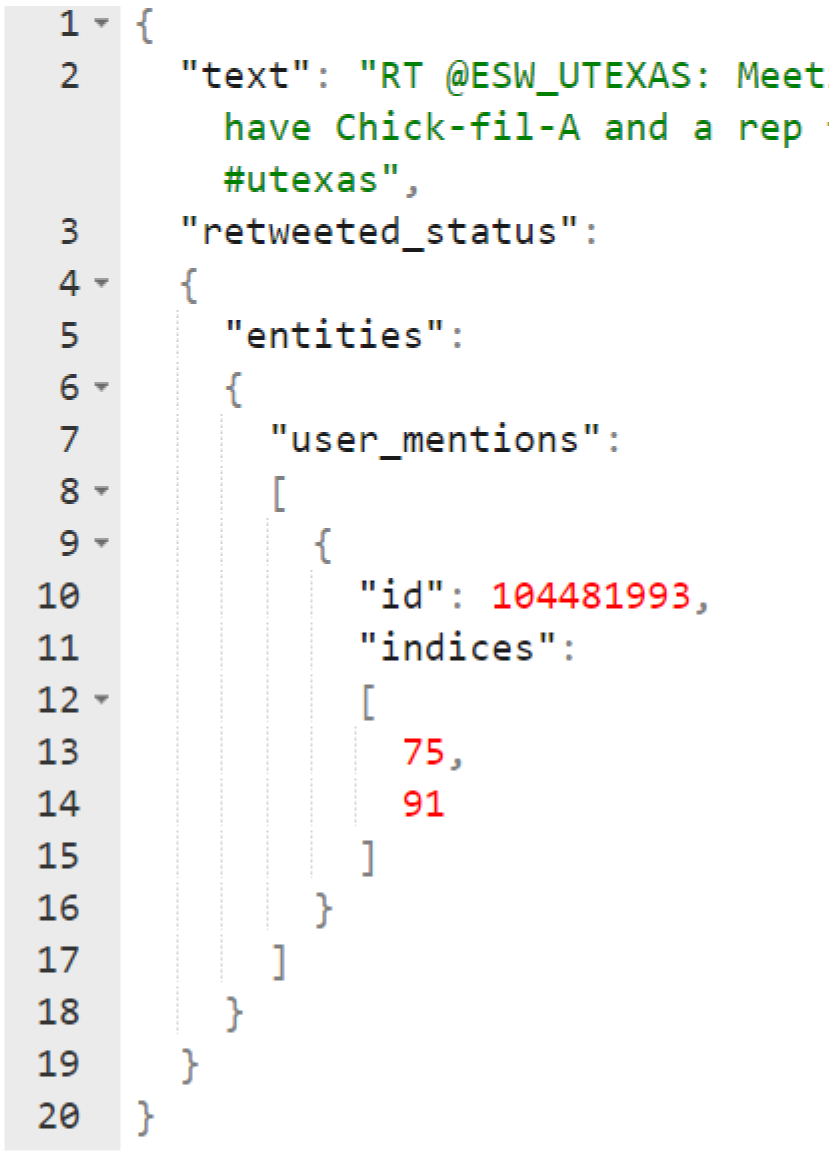
B. 75

C. NULL

D. 91

E. “id” : 104481993

Question 4



```
{
  "text": "RT @ESW_UTEXAS: Meeting tonight at 7pm in JGB 2.216! We'll
    have Chick-fil-A and a rep from @airliquidegroup will present!
    #utexas",
  "retweeted_status":
  {
    "entities":
    {
      "user_mentions":
      [
        {
          "id": 104481993,
          "indices":
          [
            75,
            91
          ]
        }
      ]
    }
  }
}
```

Let `j` be to the JSON document shown. What does the `select` statement below evaluate to?

```
select JSON_EXTRACT(j,
'$.retweeted_status.entities.
user_mentions[0].id');
```

A. "indices" : [75, 91]

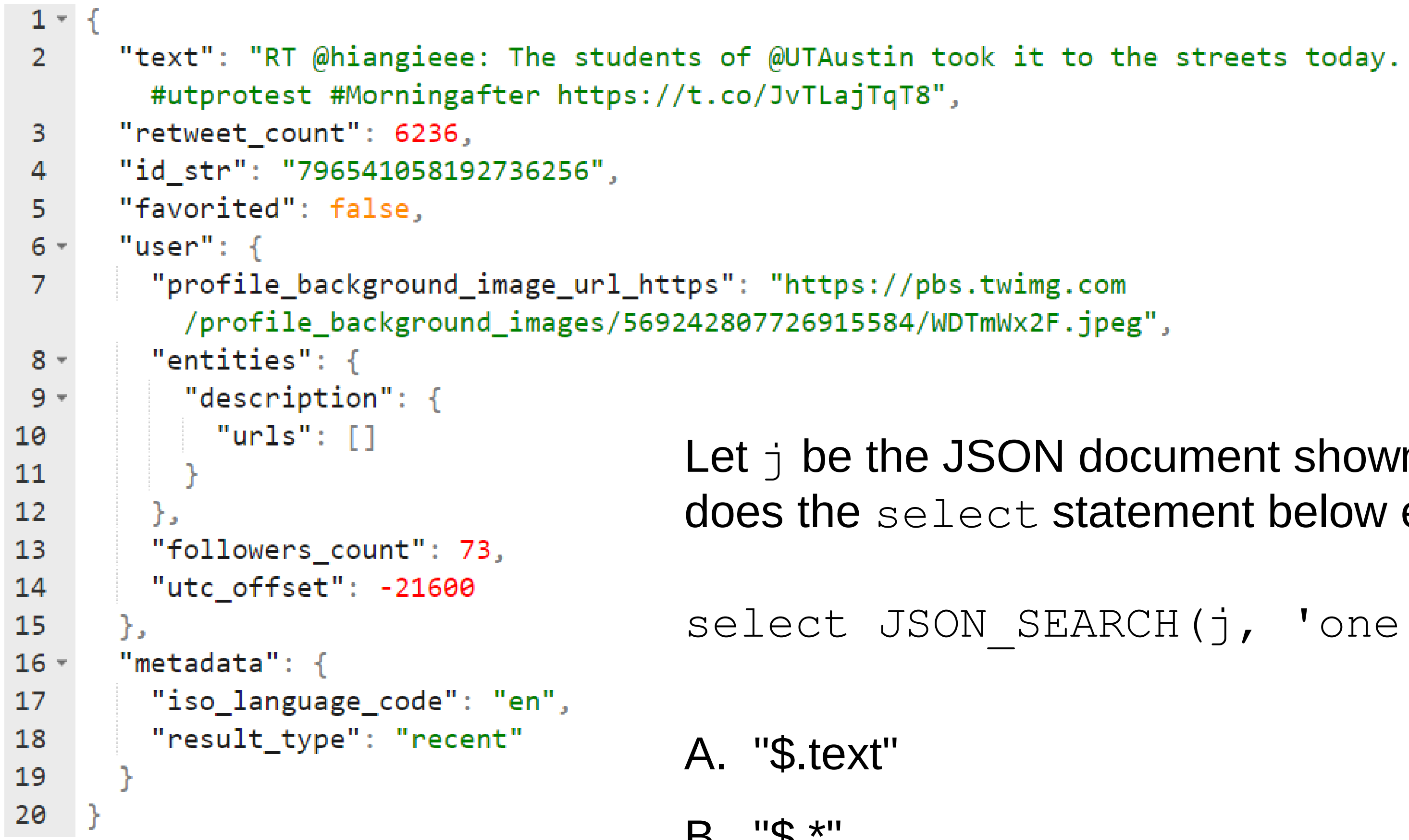
B. 75

C. 91

D. 104481993

E. None of the above

Question 5



```
{
  "text": "RT @hiangieee: The students of @UTAustin took it to the streets today.
    #utprotest #Morningafter https://t.co/JvTLajTqT8",
  "retweet_count": 6236,
  "id_str": "796541058192736256",
  "favorited": false,
  "user": {
    "profile_background_image_url_https": "https://pbs.twimg.com
      /profile_background_images/569242807726915584/WDTmWx2F.jpeg",
    "entities": {
      "description": {
        "urls": []
      }
    },
    "followers_count": 73,
    "utc_offset": -21600
  },
  "metadata": {
    "iso_language_code": "en",
    "result_type": "recent"
  }
}
```

Let `j` be the JSON document shown. What does the `select` statement below evaluate to?

```
select JSON_SEARCH(j, 'one', 'RT');
```

A. "$.text"

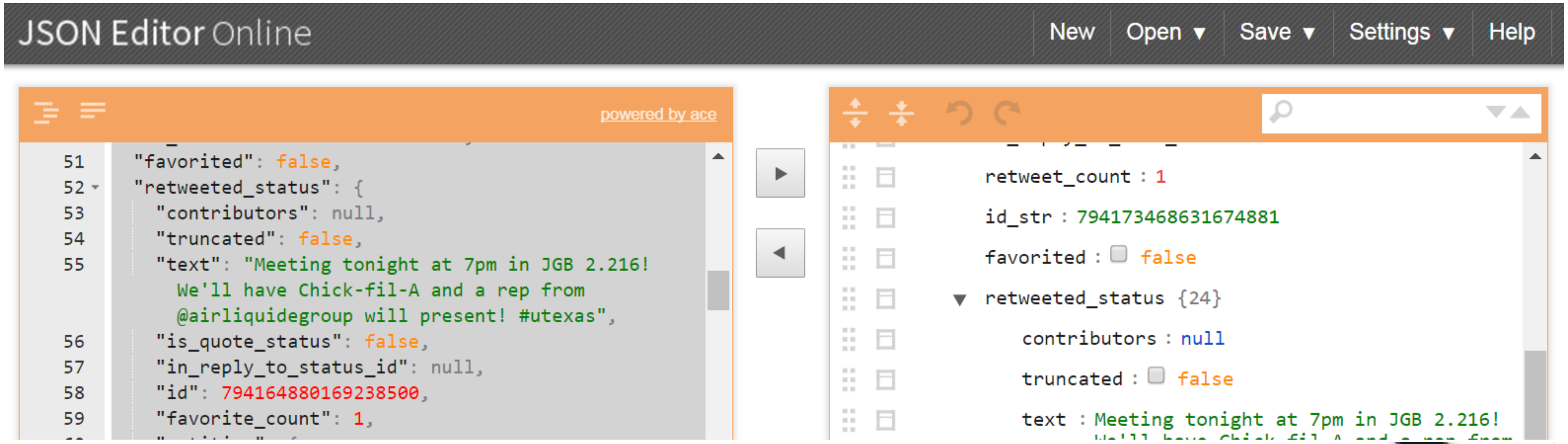
B. "$.*"

C. {}

D. NULL

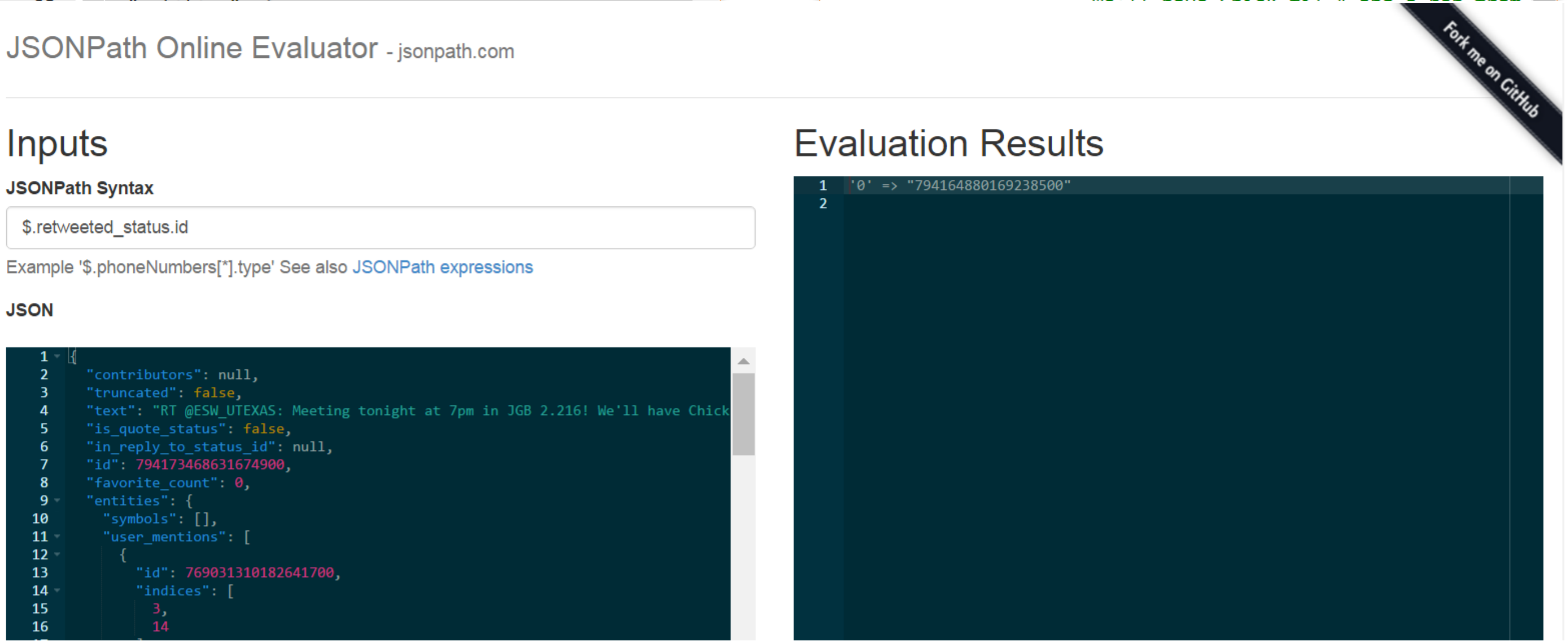
E. None of the above

Demo 1



```
"favorited": false,
"retweeted_status": {
  "contributors": null,
  "truncated": false,
  "text": "Meeting tonight at 7pm in JGB 2.216!
    We'll have Chick-fil-A and a rep from
    @airliquidegroup will present! #utexas",
  "is_quote_status": false,
  "in_reply_to_status_id": null,
  "id": 794164880169238500,
  "favorite_count": 1,
```

```
retweet_count : 1
id_str : 794173468631674881
favorited : ☐ false
▼ retweeted_status {24}
    contributors : null
    truncated : ☐ false
    text : Meeting tonight at 7pm in JGB 2.216!
```



JSONPath Online Evaluator - jsonpath.com

Inputs

JSONPath Syntax

$.retweeted_status.id

Example '$.phoneNumbers[*].type' See also JSONPath expressions

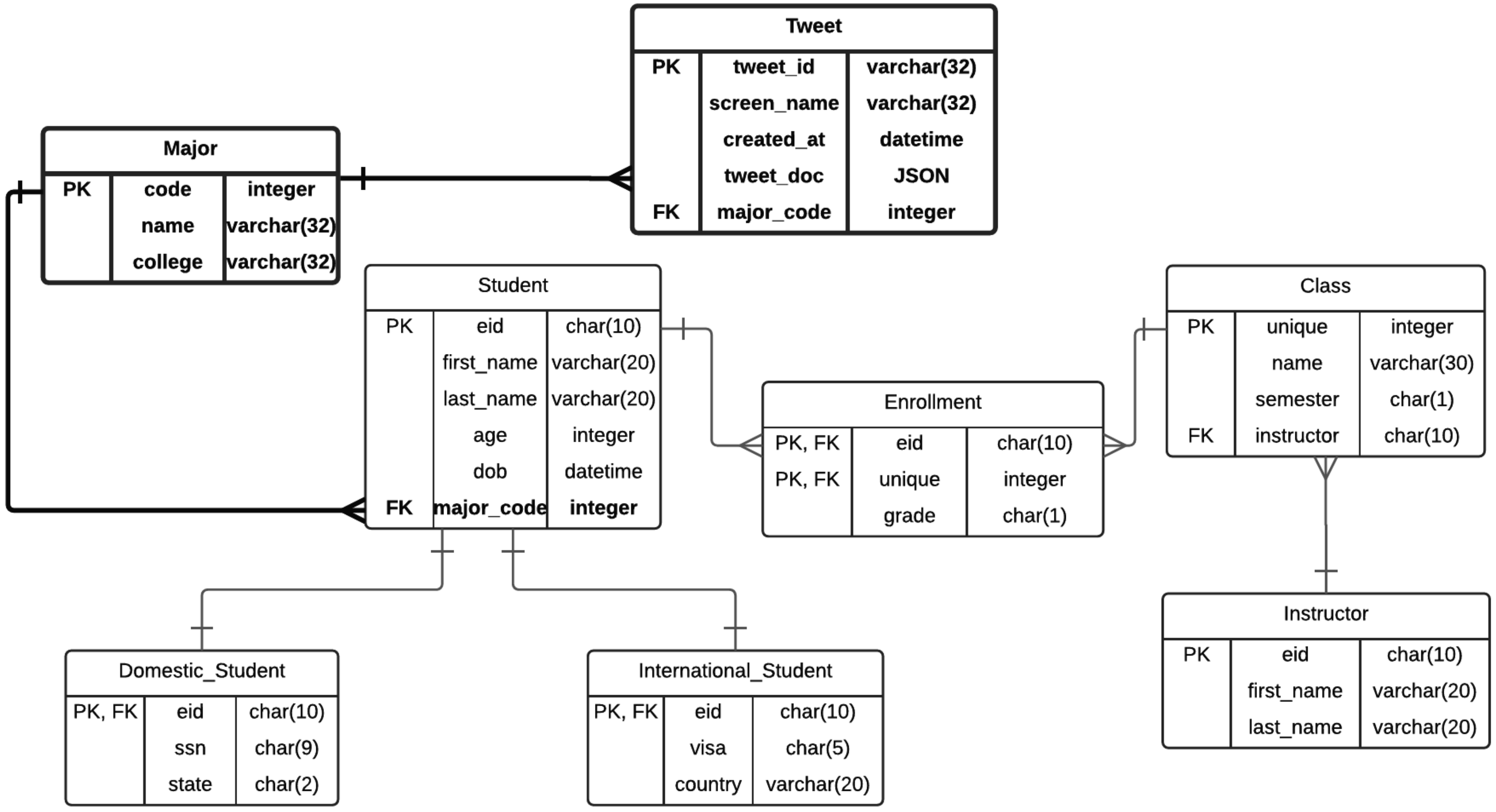
JSON

```
{
  "contributors": null,
  "truncated": false,
  "text": "RT @ESW_UTEXAS: Meeting tonight at 7pm in JGB 2.216! We'll have Chick
  "is_quote_status": false,
  "in_reply_to_status_id": null,
  "id": 794173468631674900,
  "favorite_count": 0,
  "entities": {
    "symbols": [],
    "user_mentions": [
      {
        "id": 769031310182641700,
        "indices": [
          3,
          14
```

Evaluation Results

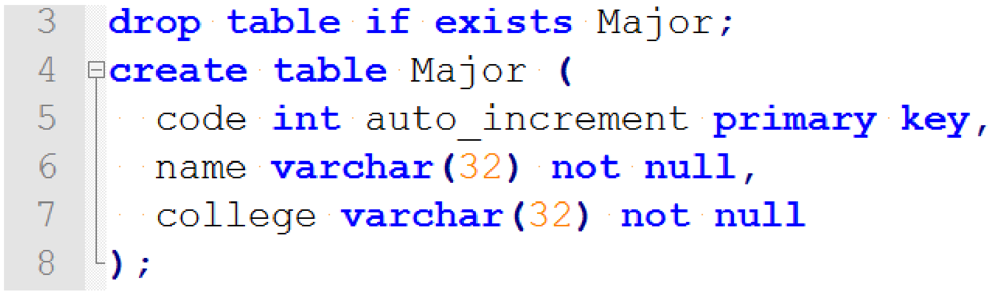
```
'0' => "794164880169238500"
```

UT Class Enrollment & Twitter

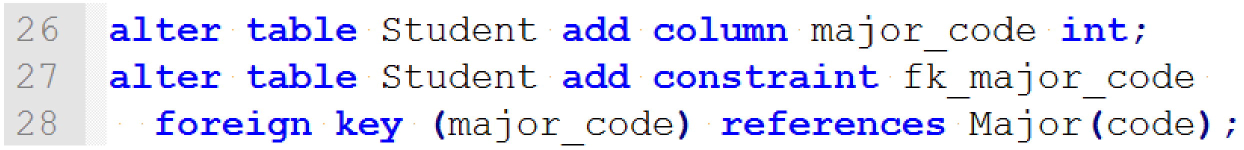


Logical ERD - UT Class Enrollment - CS 327E Fall 2016

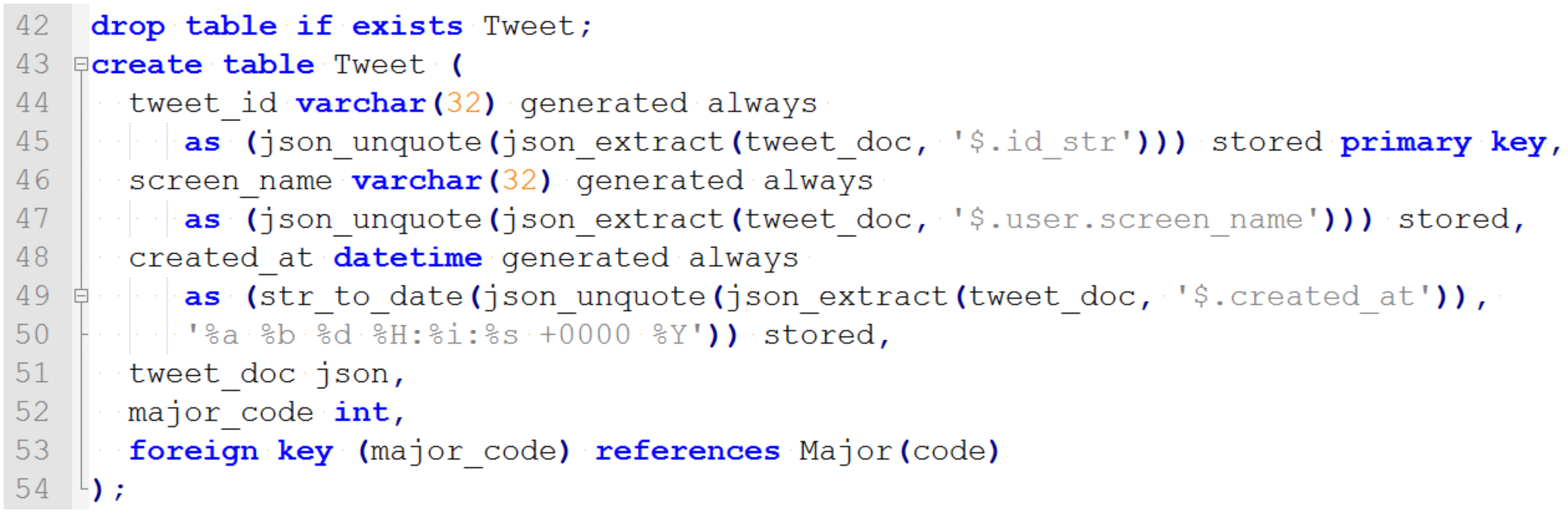
New DDL



```sql
drop table if exists Major;
create table Major (
  code int auto_increment primary key,
  name varchar(32) not null,
  college varchar(32) not null
);
```

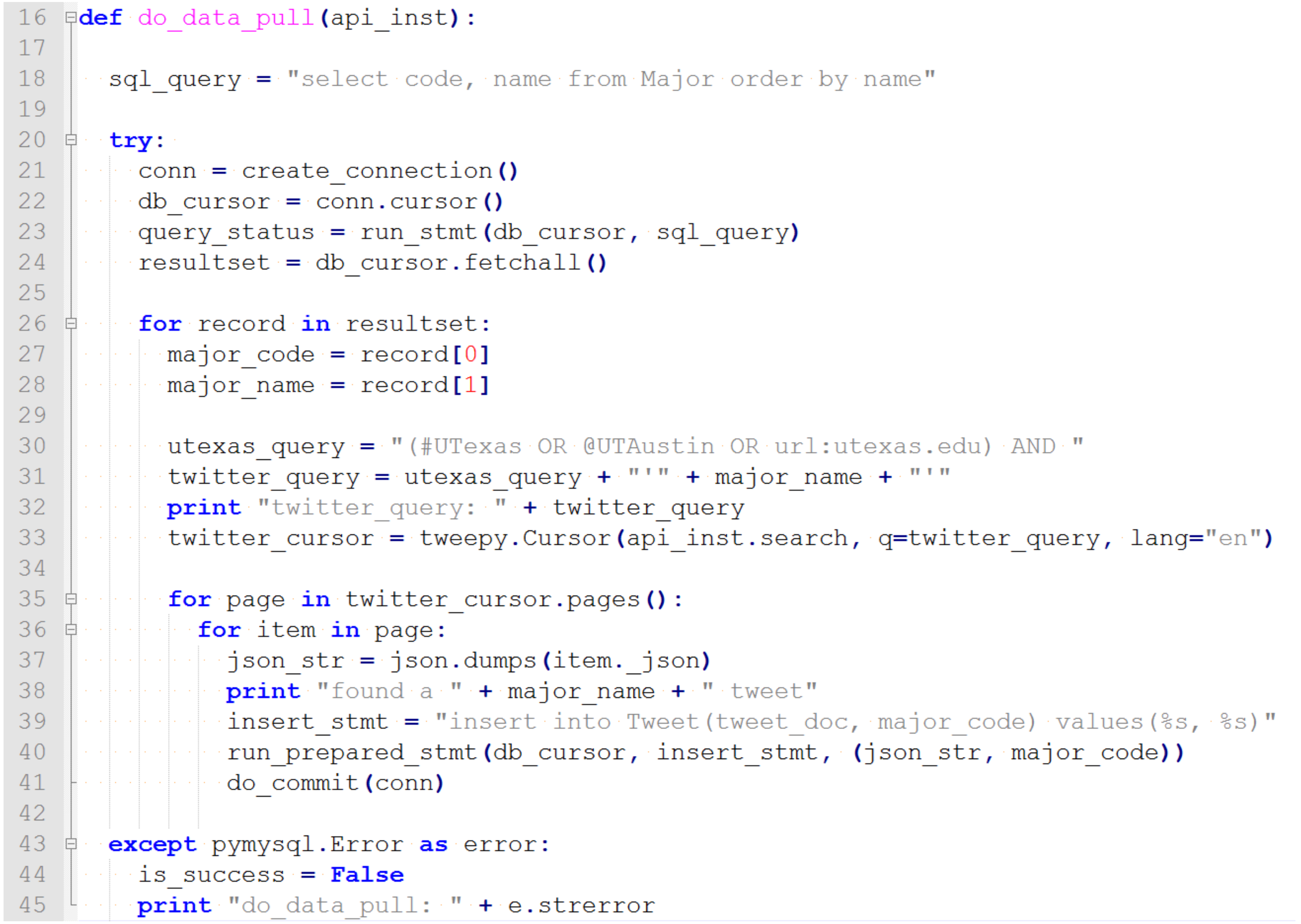


```sql
alter table Student add column major_code int;
alter table Student add constraint fk_major_code
  foreign key (major_code) references Major(code);
```



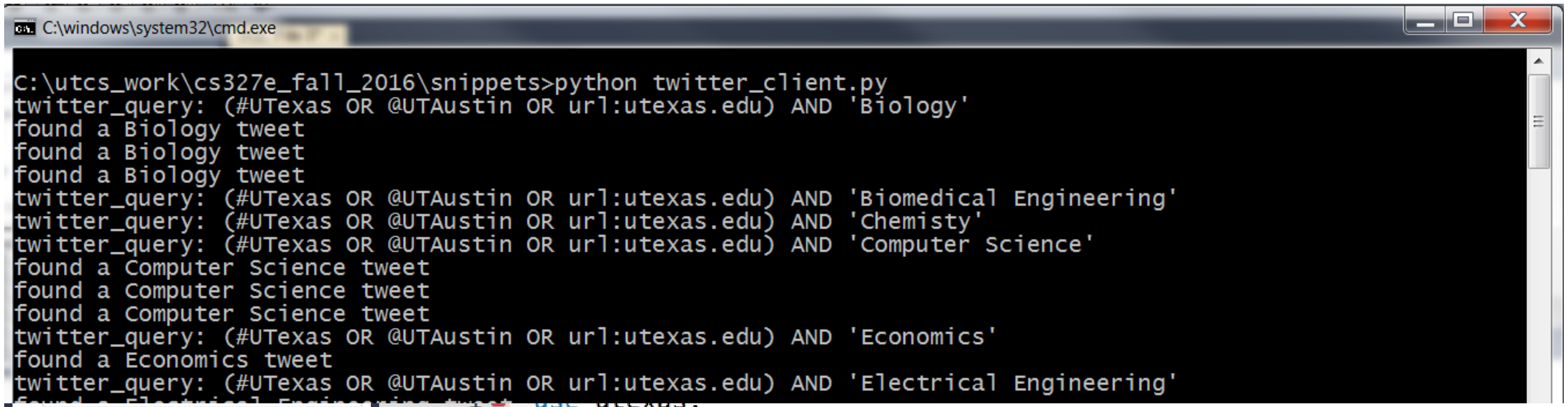
```sql
drop table if exists Tweet;
create table Tweet (
  tweet_id varchar(32) generated always
      as (json_unquote(json_extract(tweet_doc, '$.id_str'))) stored primary key,
  screen_name varchar(32) generated always
      as (json_unquote(json_extract(tweet_doc, '$.user.screen_name'))) stored,
  created_at datetime generated always
      as (str_to_date(json_unquote(json_extract(tweet_doc, '$.created_at')),
      '%a %b %d %H:%i:%s +0000 %Y')) stored,
  tweet_doc json,
  major_code int,
  foreign key (major_code) references Major(code)
);
```

Twitter Client

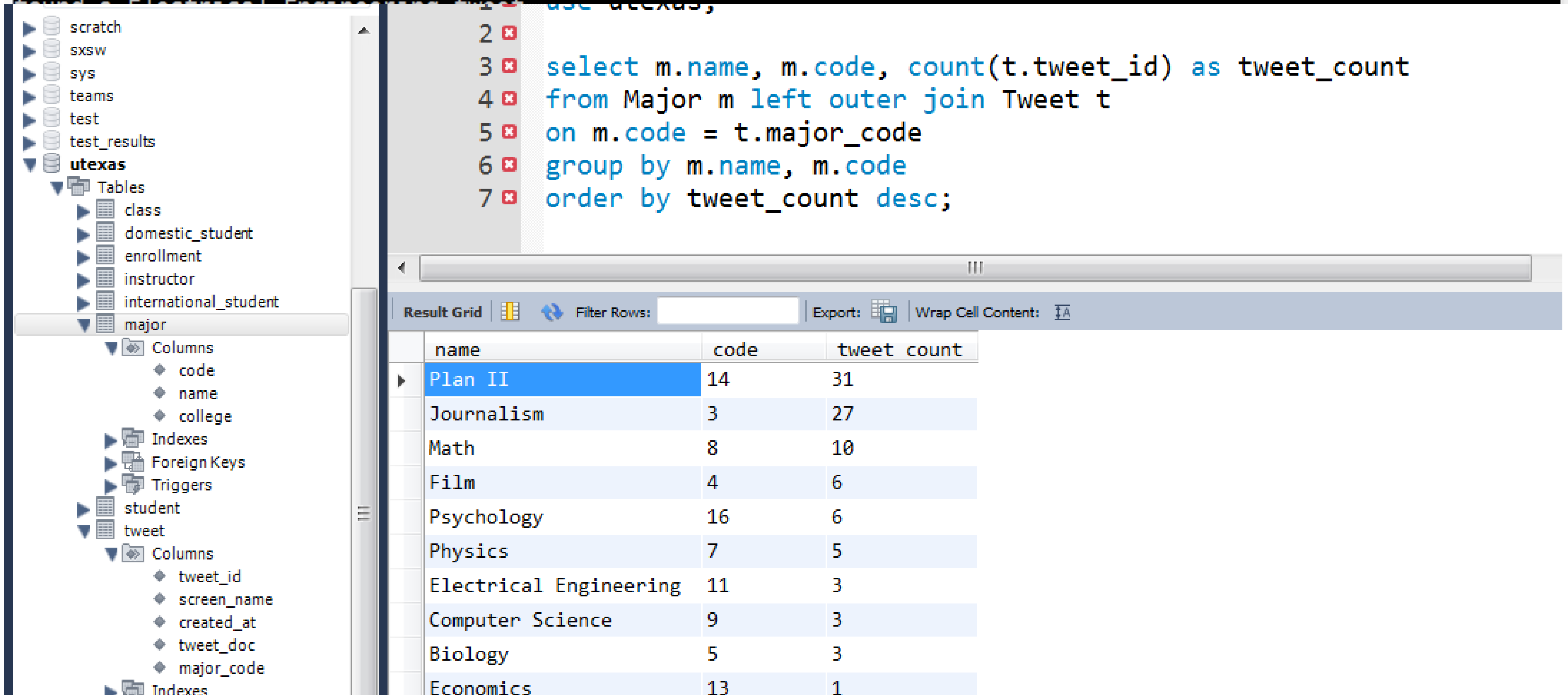


```python
def do_data_pull(api_inst):

  sql_query = "select code, name from Major order by name"

  try:
    conn = create_connection()
    db_cursor = conn.cursor()
    query_status = run_stmt(db_cursor, sql_query)
    resultset = db_cursor.fetchall()

    for record in resultset:
      major_code = record[0]
      major_name = record[1]

      utexas_query = "(#UTexas OR @UTAustin OR url:utexas.edu) AND "
      twitter_query = utexas_query + "'" + major_name + "'"
      print "twitter_query: " + twitter_query
      twitter_cursor = tweepy.Cursor(api_inst.search, q=twitter_query, lang="en")

      for page in twitter_cursor.pages():
        for item in page:
          json_str = json.dumps(item._json)
          print "found a " + major_name + " tweet"
          insert_stmt = "insert into Tweet(tweet_doc, major_code) values(%s, %s)"
          run_prepared_stmt(db_cursor, insert_stmt, (json_str, major_code))
          do_commit(conn)

  except pymysql.Error as error:
    is_success = False
    print "do_data_pull: " + e.strerror
```

Demo 2



```
C:\utcs_work\cs327e_fall_2016\snippets>python twitter_client.py
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Biology'
found a Biology tweet
found a Biology tweet
found a Biology tweet
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Biomedical Engineering'
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Chemisty'
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Computer Science'
found a Computer Science tweet
found a Computer Science tweet
found a Computer Science tweet
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Economics'
found a Economics tweet
twitter_query: (#UTexas OR @UTAustin OR url:utexas.edu) AND 'Electrical Engineering'
```

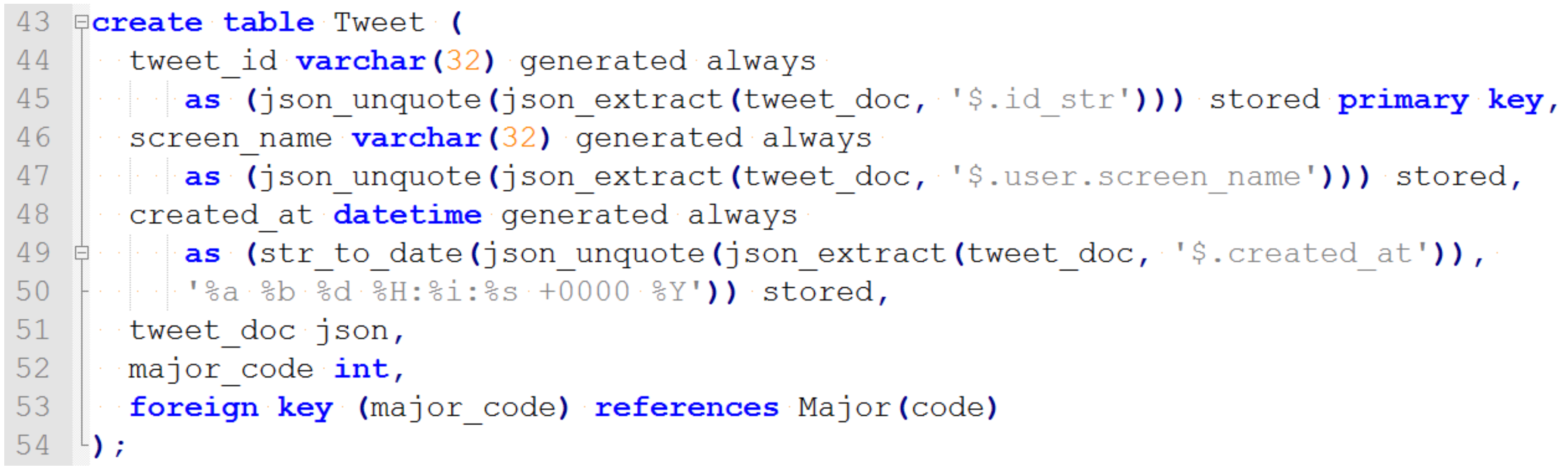


```sql
use utexas;

select m.name, m.code, count(t.tweet_id) as tweet_count
from Major m left outer join Tweet t
on m.code = t.major_code
group by m.name, m.code
order by tweet_count desc;
```

name	code	tweet_count
Plan II	14	31
Journalism	3	27
Math	8	10
Film	4	6
Psychology	16	6
Physics	7	5
Electrical Engineering	11	3
Computer Science	9	3
Biology	5	3
Economics	13	1

Concept Question 1

We want to extend the Twitter Client to check for duplicate tweets before doing the insert into MySQL. Assume that in Python we extract the id of the tweet and store the value in the variable `$id`. How can we formulate a SQL query that checks for duplicate tweets given `$id`?



```
create table Tweet (
  tweet_id varchar(32) generated always
      as (json_unquote(json_extract(tweet_doc, '$.id_str'))) stored primary key,
  screen_name varchar(32) generated always
      as (json_unquote(json_extract(tweet_doc, '$.user.screen_name'))) stored,
  created_at datetime generated always
      as (str_to_date(json_unquote(json_extract(tweet_doc, '$.created_at')),
      '%a %b %d %H:%i:%s +0000 %Y')) stored,
  tweet_doc json,
  major_code int,
  foreign key (major_code) references Major(code)
);
```

A. select count(*) from Tweet where tweet_id = `$id`

B. select * from Tweet order by tweet_id

C. select count(distinct tweet_id) from Tweet where tweet_id = `$id`

D. select * from Tweet where tweet_id = `$id`

E. select count(tweet_id) from Tweet where tweet_id = `$id`

Concept Question 2

We want to implement a more accurate count of tweets per UT major. More specifically, we want to filter out all retweets and only add up the origin tweets. Assume that for all tweets, we extract the origin tweet id from the tweet and we store this value in a new field called `Tweet.origin_tweet_id`. How can we modify the query below to only count unique tweets?

```
select m.name, m.code, count(t.tweet_id) as tweet_count
from Major m left outer join Tweet t
on m.code = t.major_code
group by m.name, m.code
order by tweet_count desc;
```

A. Replace: `count(t.tweet_id)` with: `count(t.origin_tweet_id)`
B. Replace: `count(t.tweet_id)` with: `count(distinct t.origin_tweet_id)`
C. Change the outer join to an inner join
D. Change the left outer join to a right outer join
E. None of the above

Final Project

http://www.cs.utexas.edu/~scohen/project/final_project.pdf