Project - Selecting Content for Data Science Company

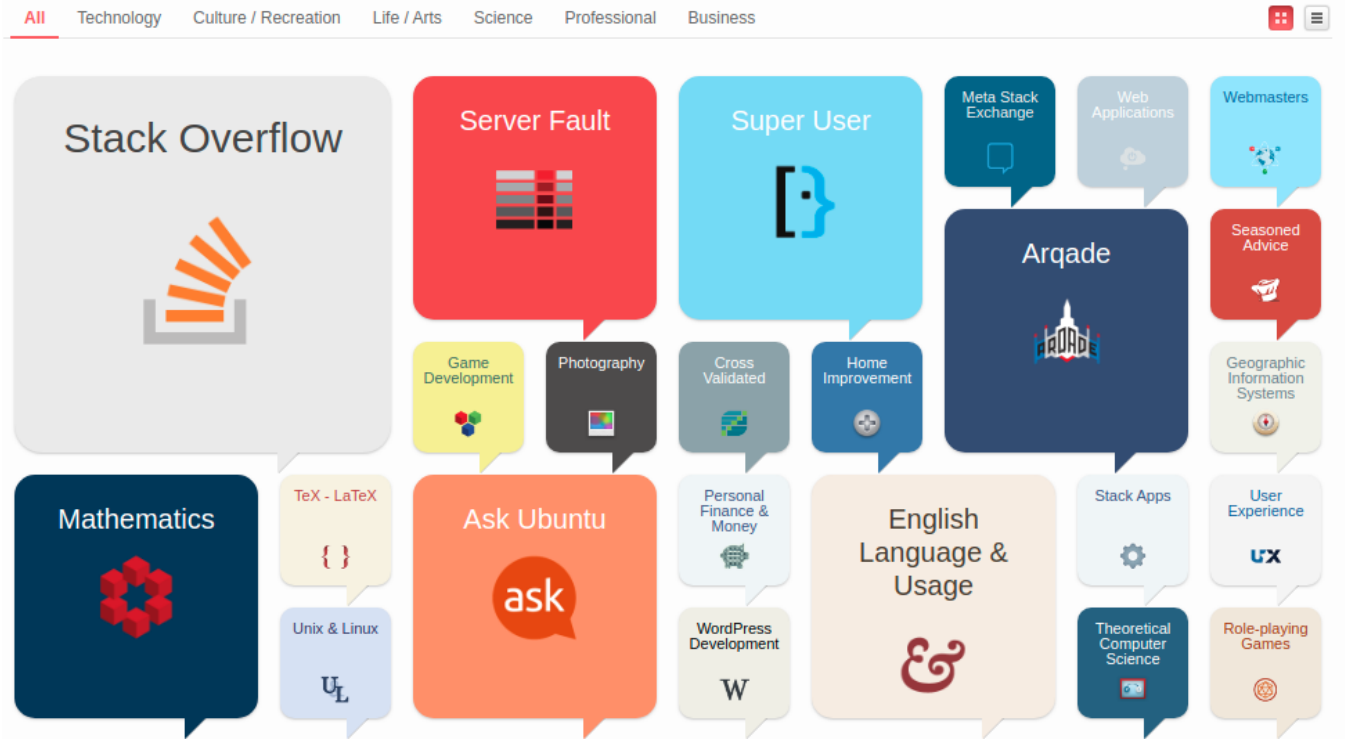
In this project, we will try to answer the question: "What do people want to learn in data science", and in so doing evaluate the available content for our company which deals with data science education.

Scenario

We're working for a company that creates data science content, be it books, online articles, videos or interactive text-based platforms. We're tasked with figuring out what is best content to write about. We realize that if we wanted to figure out what programming content to write, we could consult Stack Overflow (https://stackoverflow.com/) (a question and answer website about programming) and see what kind of content is more popular. After investigating Stack Overflow in depth, we find out that it is part of a question and answer website network called Stack Exchange (https://en.wikipedia.org/wiki/Stack_Exchange).

Stack Exchange

Stack Exchange hosts sites on a multitude of fields and subjects, including mathematics, physics, philosophy, and data science! Here's a sample of the most popular sites:



Stack Exchange employs a reputation award system for its questions and answers. Each post — each question/answer — is a post that is subject to upvotes and downvotes. This ensures that good posts are easily identifiable. More details are available on this tour (https://stackexchange.com/tour)

Being a multidisciplinary field, there a few Stack Exchange websites there are relevant to our goal here:

- Data Science (https://datascience.stackexchange.com/)
- Cross Validated (https://stats.stackexchange.com/) — a statistics site
- Artificial Intelligence (https://ai.stackexchange.com/)
- Mathematics (https://math.stackexchange.com/)
- Stack Overflow (https://stackoverflow.com/)

And if we want to include Data Engineering, we can also consider:

- Database Administrators (https://dba.stackexchange.com/);
- Unix & Linux (https://unix.stackexchange.com/);
- Software Engineering (https://softwareengineering.stackexchange.com/);

At the time of writing, in terms of users Data Science Stack Exchange (DSSE) is among top 40 sites with 85K users, however, in terms of unanswered questions it is among bottom 7 with 64% unanswered questions. Further statistics are available here (https://stackexchange.com/sites?view=list#questionsperday). This makes it quite attractive for exploring data science content.

Data Science Stack Exchange (DSSE)

The site is organized into following sections:

- Home (https://datascience.stackexchange.com/): Displays the `top questions` and contains a side bar menu with links to other sections of the site.
- [Questions]: Features all questions which can be filtered by `No answers` and `No accepted answers` . Currently, approximately 25,000 questions are featured of which over 15,000 are unanswered or with no accepted answer (63%).
- Tags (https://datascience.stackexchange.com/tags) are key words which organize questions as topics and facilitate search.
- Users (https://datascience.stackexchange.com/users) features the data on users. Users with highest `reputation` are featured before the others. Users can be searched and filtered by `user name` , `reputation` , `New user` etc.
- Unanswered (https://datascience.stackexchange.com/unanswered) features unanswered questions which can be filtered by votes, tags and newest.

The footer menu has links to Tour (https://datascience.stackexchange.com/tour) and Help (https://datascience.stackexchange.com/help) sections along with links to other Stack Exchange sites and features.

Questions and Answers

Data Science Stack Exchange(DSSE) is a question and answer site for Data science professionals, Machine Learning specialists, and those interested in learning more about the field.The method of asking questions, getting answers, and getting promoted to higher levels is given here (https://datascience.stackexchange.com/tour):

- The site is all about getting answers. It's not a discussion forum. There's no chit-chat.
- Good answers are voted up and rise to the top. The best answers show up first so that they are always easy to find.
- The person who asked can mark one answer as "accepted". Accepting doesn't mean it's the best answer, it just means that it worked for the person who asked.
- Focus on questions about an actual problem you have faced. Include details about what you have tried and exactly what you are trying to do.
- Avoid questions that are primarily opinion-based, or that are likely to generate discussion rather than answers. Questions that need improvement may be closed until someone fixes them.
- All questions are tagged with their subject areas. Each can have up to 5 tags. We can click any tag to see a list of questions with that tag, or go to the tag list to browse for topics 0f interest.
- User reputation score goes up when others vote up on their questions, answers and edits.

Getting Data

Stack Exchange sites have a number of convenient ways of getting data:

- **Scraping** : We can scrape a page for relevant information.

In this case, we will be retrieving tags on the first page of the tags link (https://datascience.stackexchange.com/tags)

In [1]:

```
import requests
# request data from "https://datascience.stackexchange.com/tags"
response = requests.get("https://datascience.stackexchange.com/tags")

content = response.content


from bs4 import BeautifulSoup

# Initialize the parser, and pass in the content we grabbed earlier.

parser = BeautifulSoup(content, 'html.parser')

tag = parser.select(".post-tag")

tags_scraping = []

for i in range(0,36):
    tag_text = tag[i].text
    tags_scraping.append(tag_text)
print(tags_scraping)
```

```
['machine-learning', 'python', 'deep-learning', 'neural-network', 'classif
ication', 'keras', 'nlp', 'scikit-learn', 'tensorflow', 'time-series', 're
gression', 'r', 'dataset', 'clustering', 'cnn', 'pandas', 'data-mining',
'predictive-modeling', 'lstm', 'statistics', 'feature-selection', 'data',
'random-forest', 'machine-learning-model', 'linear-regression', 'data-clea
ning', 'image-classification', 'rnn', 'convolutional-neural-network', 'dec
ision-trees', 'pytorch', 'logistic-regression', 'xgboost', 'visualizatio
n', 'training', 'data-science-model']
```

API

Obviously, scraping is a tedious process and getting even the names of tags will involve a lot of coding.

Next, we can try the Stack Exchange API (https://api.stackexchange.com/)

First, we register the app for access token

- App: Py Lesson
- Get client_id = 18238
- redirect_uri = `https://stackoverflow.com/oauth/login_success`

In [2]:

```
# access token requested
import requests
import requests.auth

response = requests.post("https://stackoverflow.com/oauth/dialog?client_id=18238&redirec

print(response.status_code)
```

```
200
```

In [3]:

```
# access_token is only required for increased quota, write and access to private info
## Following access token is now expired

headers = {'access_token': '6QkGLWxpVNzS6XWzfOFXGw))', 'key': 'LCyb3nlOflFZqImiVbfZog(('

response = requests.get("https://api.stackexchange.com/2.2/tags?page=1&pagesize=36&order
tags = response.json()
```

In [4]:

```
# Parse the json for extracting names of tags on page-1
import pandas as pd

tags_list = tags['items']

tags_df = pd.DataFrame(tags_list)

tags_api = tags_df["name"]
print(tags_api)
```

```
0                 machine-learning
1                           python
2                    deep-learning
3                   neural-network
4                   classification
5                            keras
6                              nlp
7                     scikit-learn
8                       tensorflow
9                      time-series
10                      regression
11                               r
12                         dataset
13                      clustering
14                             cnn
15                          pandas
16                     data-mining
17              predictive-modeling
18                            lstm
19                      statistics
20               feature-selection
21                            data
22                   random-forest
23           machine-learning-model
24                linear-regression
25                    data-cleaning
26             image-classification
27                             rnn
28    convolutional-neural-network
29                   decision-trees
30                          pytorch
31              logistic-regression
32                          xgboost
33                   visualization
34                        training
35               data-science-model
Name: name, dtype: object
```

In [5]:

```
# Check whether tag list obtained from scraping and from api are equal
tags_api = list(tags_df["name"])

equal = tags_api==tags_scraping
print(equal) # Are equal
```

```
True
```

Stack Exchange Data Explorer [(SEDE) (https://data.stackexchange.com/help)](https://data.stackexchange.com/help)

SEDE is an open source tool for running arbitrary queries against public data from the Stack Exchange network. Features include collaborative query editing for all graduated and public beta Stack Exchange sites.

The data is updated early every Sunday morning around 3:00 UTC.

Apart from web-scraping and use of API, SEDE provides access to the database and entertains T-SQL queries.

[Following query (https://data.stackexchange.com/datascience/query/1259433/schema)](https://data.stackexchange.com/datascience/query/1259433/schema) run at SEDE gives the same results as obtained from web-scraping and API.

```
SELECT TOP 36
            TagName, COUNT
  FROM Tags
ORDER BY COUNT DESC;
```

The query results can be downloaded as a csv file, and a permalink can also be created to the query (as done above). We will now compare results from the query with those of web_scraping and API.

In [6]:

```
import pandas as pd
query = pd.read_csv("top_36_tags.csv")
tags_query = query["TagName"]
tags_query = list(tags_query)
print(tags_query)
```

```
['machine-learning', 'python', 'neural-network', 'deep-learning', 'classif
ication', 'keras', 'scikit-learn', 'tensorflow', 'nlp', 'r', 'time-serie
s', 'dataset', 'regression', 'data-mining', 'clustering', 'cnn', 'predicti
ve-modeling', 'pandas', 'lstm', 'statistics', 'feature-selection', 'data',
'random-forest', 'image-classification', 'decision-trees', 'linear-regress
ion', 'text-mining', 'data-cleaning', 'visualization', 'reinforcement-lear
ning', 'rnn', 'xgboost', 'logistic-regression', 'convnet', 'bigdata', 'sv
m']
```

In [7]:

```
# Check whether these results are same as those obtained from web-scraping and API

equal = tags_api==tags_scraping==tags_query

print(equal) # All results are equal
```

```
False
```

In [8]:

```
# Determine top ten popular topics from tags
top_ten = query[:10]
print(top_ten)
```

```
            TagName  COUNT
0  machine-learning   7879
1            python   4612
2    neural-network   3304
3     deep-learning   3214
4    classification   2182
5             keras   2058
6      scikit-learn   1542
7        tensorflow   1475
8               nlp   1393
9                 r   1224
```

Getting Posts Data from SEDE

Above, we can see that maximum questions are contained in about top 10 or so tags. Same content (top 36 tags) is displayed on the tags home page (https://datascience.stackexchange.com/tags) Now we will focus on the `Posts` table in the database `SEDE` to create a database for a year (2019) and carry out a more granular analysis.

We will run a query against the SEDE DSSE database that extracts the following columns for all the questions in 2019: `Id` : An identification number for the post. `PostTypeId` : An identification number for the type of post. `CreationDate` : The date and time of creation of the post. `Score` : The post's score. `ViewCount` : How many times the post was viewed. `Tags` : What tags were used. `AnswerCount` : How many answers the question got (only applicable to question posts). `FavoriteCount` : How many times the question was favored (only applicable to question posts).

Using the API in Tandem

We will use the API in tandem with query in order to compare some of our results.

In [9]:

```
# Find total number of questions in 2019 using API

response = requests.get("https://api.stackexchange.com/2.2/questions?pagesize=100&fromda
questions_api = response.json()


questions_list = questions_api['items']
questions_100 = pd.DataFrame(questions_list)
print(questions_100.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   tags            100 non-null    object
 1   view_count      100 non-null    int64 
 2   favorite_count  100 non-null    int64 
 3   answer_count    100 non-null    int64 
 4   score           100 non-null    int64 
 5   creation_date   100 non-null    int64 
 6   question_id     100 non-null    int64 
 7   title           100 non-null    object
dtypes: int64(6), object(2)
memory usage: 6.4+ KB
None
```

Find total number of questions from query

We can see from above that the api is limited to returning 100 results per page and getting all the data would need running a loop several times consuming computational resources. We will, therefore, get the complete data from the database running [following query (https://data.stackexchange.com/datascience/query/1259625/questions-in-2019)](https://data.stackexchange.com/datascience/query/1259625/questions-in-2019)

```
    SELECT Id, CreationDate,
           Score, ViewCount, Tags,
           AnswerCount, FavoriteCount
      FROM posts
     WHERE PostTypeId = 1 AND YEAR(CreationDate) = 2019;
```

Note that of the various post types **Post TypeID** for "questions" is **1**.

We download the results as `2019_questions.csv` from SEDE.

In [10]:

```
# Convert the results into dataframe

questions = pd.read_csv("2019_questions.csv", parse_dates=["CreationDate"])

print(questions.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8839 entries, 0 to 8838
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             8839 non-null   int64
 1   CreationDate   8839 non-null   datetime64[ns]
 2   Score          8839 non-null   int64
 3   ViewCount      8839 non-null   int64
 4   Tags           8839 non-null   object
 5   AnswerCount    8839 non-null   int64
 6   FavoriteCount  1407 non-null   float64
dtypes: datetime64[ns](1), float64(1), int64(4), object(1)
memory usage: 483.5+ KB
None
```

Data Cleaning

We want to eventually focus on the `tags` column as well as other popularity, so we will carry out following data-cleaning steps:

- Fill missing values in `FavoriteCount` with `0` as the missing values indicate that the question was not voted upon.
- Convert `FavoriteCount` into `int`
- Convert `Tags` string into a more readable format

In [11]:

```
# Fill in missing values for the "FavoriteCount" column

questions.fillna(0, inplace=True)
questions["FavoriteCount"] = questions["FavoriteCount"].astype(int)

print(questions.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8839 entries, 0 to 8838
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Id             8839 non-null   int64         
 1   CreationDate   8839 non-null   datetime64[ns]
 2   Score          8839 non-null   int64         
 3   ViewCount      8839 non-null   int64         
 4   Tags           8839 non-null   object        
 5   AnswerCount    8839 non-null   int64         
 6   FavoriteCount  8839 non-null   int32         
dtypes: datetime64[ns](1), int32(1), int64(4), object(1)
memory usage: 449.0+ KB
None
```

In [12]:

```
# Convert format of tags string

print(questions["Tags"].sample(5))
```

```
1652                  <python><nlp><feature-construction>
6839                             <reinforcement-learning>
4198    <machine-learning><predictive-modeling><data><...
8675                         <python><logistic-regression>
6980                    <scikit-learn><linear-regression>
Name: Tags, dtype: object
```

In [13]:

```
questions['Tags'] = questions['Tags'].str.replace('><',',').str.replace('<', '')\
 .str.replace('>', '')\
 .str.split(',')
print(questions["Tags"].sample(5))
```

```
5144    [machine-learning, classification, svm, normal...
7534                 [python, scikit-learn, pandas, numpy]
6691                 [machine-learning, linear-regression]
1973                       [python, pandas, data-cleaning]
380     [neural-network, feature-selection, feature-en...
Name: Tags, dtype: object
```

Most Used and Most Viewed Tags

We will focus on `Tags` to determine:

- Count how many times each tag was used.
- Count how many times each tag was viewed.
- Create visualizations for the top tags of each of the above results.

In [14]:

```
print(questions.sample(5))
```

```
         Id        CreationDate  Score  ViewCount  \
8372  55087 2019-07-04 18:06:07      0        123
2038  46809 2019-03-06 18:09:43      1         43
4846  51450 2019-05-05 18:20:00      1        148
2661  47278 2019-03-14 06:01:30      2         82
4759  62183 2019-10-24 18:29:23      0          7

                                                   Tags  AnswerCount  \
8372  [machine-learning, visualization, matplotlib, ...            0
2038                     [visualization, data, heatmap]            1
4846  [time-series, lstm, rnn, recurrent-neural-net,...            0
2661  [deep-learning, nlp, recurrent-neural-net, mac...            1
4759  [machine-learning, regression, machine-learnin...            0

      FavoriteCount
8372              0
2038              0
4846              1
2661              1
4759              0
```

In [15]:

```
tag_no = {}
for tags in questions["Tags"]:
    for tag in tags:
        if tag not in tag_no:
            tag_no[tag] = 1
        else:
            tag_no[tag] += 1
```

In [16]:

```
tag_no = pd.DataFrame.from_dict(data=tag_no, orient="index")
tag_no.rename(columns={0: "No"}, inplace=True)
most_used=tag_no.sort_values(by='No', ascending=False, axis=0)

most_used.head(20)
```

Out[16]:

	No
machine-learning	2693
python	1814
deep-learning	1220
neural-network	1055
keras	935
classification	685
tensorflow	584
scikit-learn	540
nlp	493
cnn	489
time-series	466
lstm	402
pandas	354
regression	347
dataset	340
r	268
predictive-modeling	265
clustering	257
statistics	234
machine-learning-model	224

Using SEDE Query

We can use the [SEDE Query (https://data.stackexchange.com/datascience/query/1259649/questions-in-2019)](https://data.stackexchange.com/datascience/query/1259649/questions-in-2019) to obtain the same results as above.

```
SELECT TagName, COUNT(TagName)
  FROM Posts AS p
               INNER JOIN PostTags AS pt ON pt.PostId=p.id
               INNER JOIN Tags AS t ON t.id=pt.TagId
 WHERE PostTypeId = 1 AND YEAR(CreationDate) = 2019
 GROUP BY TagName
 ORDER BY COUNT(TagName) DESC;
```

In [17]:

```
tags_2019 = pd.read_csv("tags_2019.csv")
tags_2019.rename(columns={"Unnamed: 1": "No"}, inplace=True)
print(tags_2019.head(20))
```

```
                   TagName    No
0         machine-learning  2443
1                   python  1652
2            deep-learning  1082
3           neural-network   960
4                    keras   841
5           classification   629
6               tensorflow   515
7             scikit-learn   491
8                      nlp   450
9                      cnn   439
10             time-series   405
11                    lstm   341
12                  pandas   340
13              regression   316
14                 dataset   301
15              clustering   249
16                       r   243
17    predictive-modeling   242
18              statistics   211
19  machine-learning-model   206
```

In [18]:

```
# Results obtained from query and dataset are equal
equal = list(most_used["No"]) == list(tags_2019["No"])
print(equal)
```

```
False
```

In [19]:

```
# Determining Viewcounts for Tags in 2019

view_no = {}
for index, row in questions.iterrows():
    for tag in row['Tags']:
        if tag not in view_no:
            view_no[tag] = row['ViewCount']
        else:
            view_no[tag] += row['ViewCount']
```

In [20]:

```
view_no = pd.DataFrame.from_dict(data=view_no, orient="index")
view_no.rename(columns={0: "No"}, inplace=True)
most_viewed=view_no.sort_values(by='No', ascending=False, axis=0).head(20)

most_viewed
```

Out[20]:

	No
python	537585
machine-learning	388499
keras	268608
deep-learning	233628
pandas	201787
neural-network	185367
scikit-learn	128110
tensorflow	121369
classification	104457
dataframe	89352
lstm	74458
nlp	71382
cnn	70349
time-series	64134
numpy	49767
regression	49451
dataset	43151
pytorch	40240
csv	38654
clustering	33928

Using SEDE Query

We can use the [SEDE Query (https://data.stackexchange.com/datascience/query/1259696/viewcount-for-tags-created-in-2019-and-active-in-2019)](https://data.stackexchange.com/datascience/query/1259696/viewcount-for-tags-created-in-2019-and-active-in-2019) to obtain the same results as above.

```
SELECT TagName, SUM(ViewCount)
  FROM Posts AS p
               INNER JOIN PostTags AS pt ON pt.PostId=p.id
               INNER JOIN Tags AS t ON t.id=pt.TagId
               WHERE (PostTypeId = 1 AND YEAR(CreationDate) = 2019)
 GROUP BY TagName
 ORDER BY 2 DESC;
```

In [21]:

```
views_2019 = pd.read_csv("views_2019.csv")
views_2019.rename(columns={"Unnamed: 1": "No"}, inplace=True)
print(views_2019.head(20))
```

```
             TagName       No
0             python  1196833
1   machine-learning   773935
2              keras   544628
3             pandas   527831
4      deep-learning   453475
5     neural-network   380012
6       scikit-learn   280534
7         tensorflow   257580
8     classification   209486
9          dataframe   203979
10               nlp   152420
11               cnn   151930
12              lstm   143433
13       time-series   130870
14             numpy   116095
15        regression    94623
16               csv    88767
17           pytorch    88365
18           dataset    83026
19           seaborn    74060
```

In [22]:

```
# Results obtained from query and dataset are equal
equal = list(most_viewed["No"]) == list(views_2019["No"].head(20))
print(equal)
```

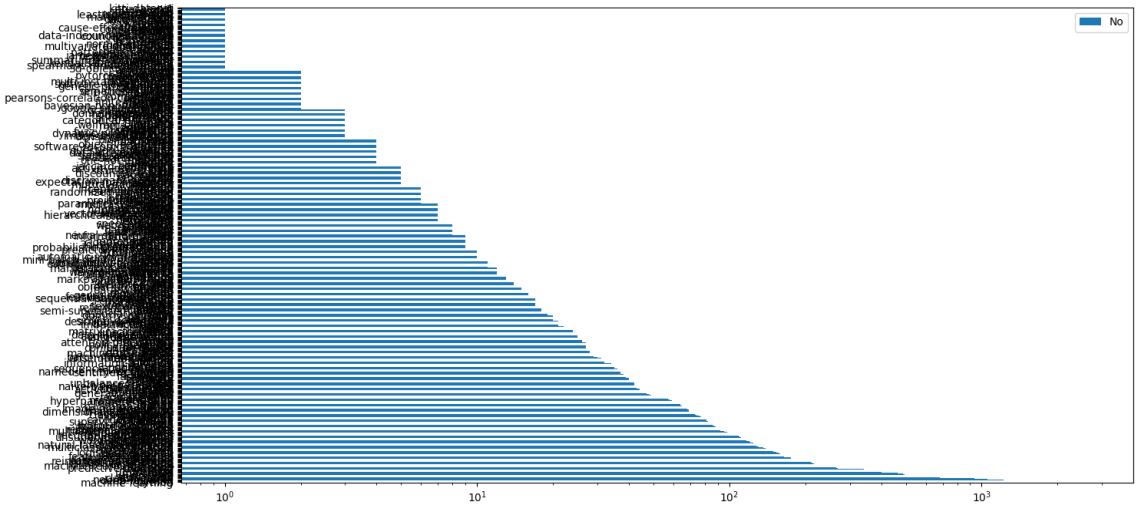
```
False
```

In [23]:

```
import matplotlib.pyplot as plt
%matplotlib inline
most_used.plot(kind="barh", figsize=(16,8),logx=True) #log scaling on x axis due to very
```

Out[23]:

<Axes: >

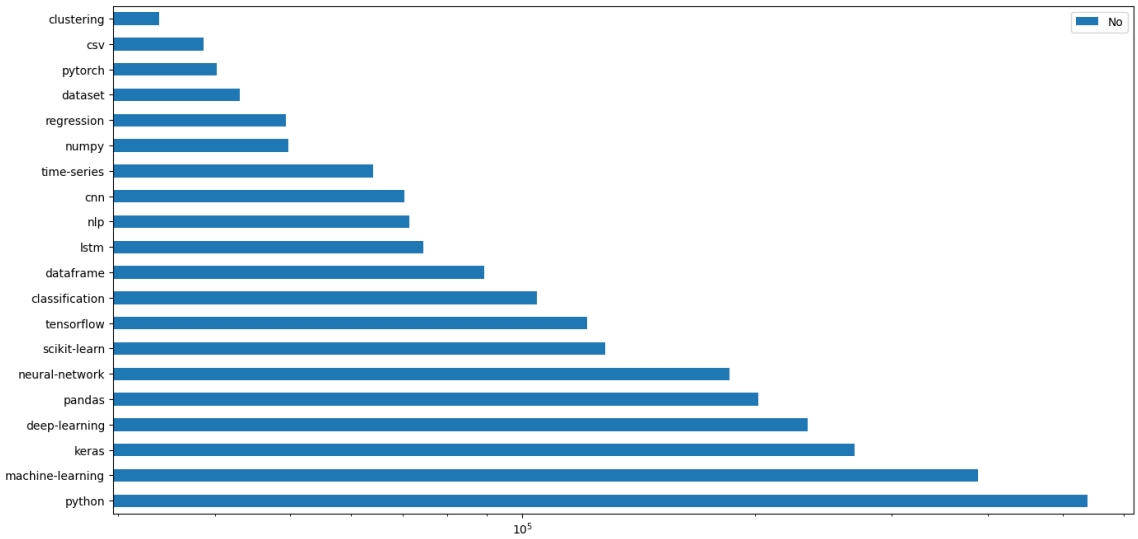


In [24]:

```
most_viewed.plot(kind="barh", figsize=(16,8),logx=True)#log scaling on x axis due to ver
```

Out[24]:

<Axes: >

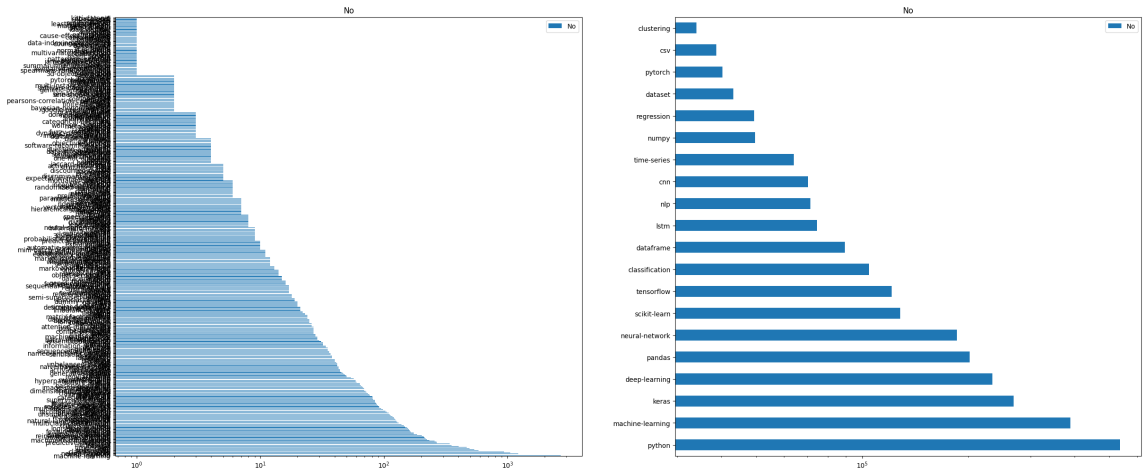


In [25]:

```
fig, axes = plt.subplots(1, 2)
fig.set_size_inches((28, 12))
#Log scaling on x axis due to very large numbers
most_used.plot(kind="barh", ax=axes[0], subplots=True, logx=True)
most_viewed.plot(kind="barh", ax=axes[1], subplots=True, logx=True)
```

Out[25]:

```
array([<Axes: title={'center': 'No'}>], dtype=object)
```



Relations Between Tags

If we want to see what tags are in `most_used`, but not in `most_viewed`, We can identify them by the missing values in ViewCount.

Similarly if we want to know which tags are in the latter, but not the former we can use a similar approach.

In [26]:

```
in_used = pd.merge(most_used, most_viewed, how="left", left_index=True, right_index=True
print(in_used)
```

```
                  No_x      No_y
machine-learning  2693  388499.0
python            1814  537585.0
deep-learning     1220  233628.0
neural-network    1055  185367.0
keras              935  268608.0
...                ...       ...
cs231n               1       NaN
statsmodels          1       NaN
rdkit                1       NaN
apache-nifi          1       NaN
kitti-dataset        1       NaN

[526 rows x 2 columns]
```

In [27]:

```
pd.merge(most_used, most_viewed, how="right", left_index=True, right_index=True)
```

Out[27]:

	No_x	No_y
python	1814	537585
machine-learning	2693	388499
keras	935	268608
deep-learning	1220	233628
pandas	354	201787
neural-network	1055	185367
scikit-learn	540	128110
tensorflow	584	121369
classification	685	104457
dataframe	81	89352
lstm	402	74458
nlp	493	71382
cnn	489	70349
time-series	466	64134
numpy	117	49767
regression	347	49451
dataset	340	43151
pytorch	175	40240
csv	27	38654
clustering	257	33928

- `clustering` , `r` , `predictive-modeling` , `statistics` and `machine-learning-model` are in `most_used` but not in `most_viewed` .
- `dataframe` , `numpy` , `csv` , `pytorch` and `seaborn` are in `most_viewed` but not in `most-used` .

** Simpler Approach using Sets**

A far simpler appraoch is to use set methods `intersection` and `difference`

- As is obvious below, it gives the same results as above.

In [28]:

```
set(most_used.index).intersection(set(most_viewed.index))
```

Out[28]:

```
{'classification',
 'clustering',
 'cnn',
 'csv',
 'dataframe',
 'dataset',
 'deep-learning',
 'keras',
 'lstm',
 'machine-learning',
 'neural-network',
 'nlp',
 'numpy',
 'pandas',
 'python',
 'pytorch',
 'regression',
 'scikit-learn',
 'tensorflow',
 'time-series'}
```

In [29]:

```
set(most_used.index).difference(set(most_viewed.index))
```

Out[29]:

```
{'.net',
 '3d-object-detection',
 '3d-reconstruction',
 'ab-test',
 'accuracy',
 'activation',
 'activation-function',
 'active-learning',
 'activity-recognition',
 'actor-critic',
 'adaboost',
 'aggregation',
 'ai',
 'alex-net',
 'algorithms',
 'allennlp',
 'amazon-ml',
 'anaconda',
```

In [30]:

```
set(most_viewed.index).difference(set(most_used.index))
```

Out[30]:

```
set()
```

Identifying Potential Data Science Content using Domain Knowledge

As practitioners, we are aware that data science is a multi-disciplinary field with broad interlinked disciplines as under:

- Computer programming
- Data management
- Calculus, Maths and Algebra
- Statistics and Probability
- Artificial Intelligence

Our strategy will be:

- Identify most popular (most_used) tags associated with each dsicipline using domain knowledge.
- Run queries in the DSDE database to get other tags most frequently used with most popular tags.
- Create a superset of tags for each domain (discipline) and make them unique through set operations
- classify `tags` column in `all_questions` dataframe against each superset for classification as a subset.

Since general coding questions are dealt with at [stack overflow (https://stackoverflow.com/)](https://stackoverflow.com/), we will leave this part out.

- From Data management, we will focus on the tag `dataset` and create superset `ss_data`
- From the mathematical domains, we will focus on the tag `statistics` and create superset `ss_stat`
- From AI domains, we will focus on tags `machine-learning` and `deep-learning` and create superset `ss_ai`

Superset Data

Run following query at SEDE

```
SELECT Tags, TagName
  FROM Posts AS p
               INNER JOIN PostTags AS pt ON pt.PostId=p.id
               INNER JOIN Tags AS t ON t.id=pt.TagId
 WHERE TagName = 'dataset';
```

In [31]:

```
# Create a set 'ss_data'
ss_data_df = pd.read_csv("ss_data.csv")

ss_data_df["Tags"] = ss_data_df["Tags"].str.replace('><',',').str.replace('<', '')\
 .str.replace('>', '')\
 .str.split(',') # clean dataframe

print(ss_data_df.head(5),'\n')



ss_data_list = []
for index, row in ss_data_df.iterrows():
    for tag in row['Tags']:
        ss_data_list.append(tag)
print(ss_data_list[:10], '\n')

print(len(ss_data_list),'\n')

ss_data = set(ss_data_list)

print(len(ss_data))
```

```
                                                Tags  TagName
0                            [open-source, dataset]  dataset
1                  [open-source, dataset, crawling]  dataset
2  [machine-learning, classification, dataset, cl...  dataset
3                  [visualization, dataset, graphs]  dataset
4                         [knowledge-base, dataset]  dataset

['open-source', 'dataset', 'open-source', 'dataset', 'crawling', 'machine-
learning', 'classification', 'dataset', 'clustering', 'text-mining']

3491

271
```

Superset Statistics

Run following query at SEDE

``` SELECT Tags, TagName FROM Posts AS p INNER JOIN PostTags AS pt ON pt.PostId=p.id INNER JOIN Tags AS t ON t.id=pt.TagId WHERE TagName = 'statistics';
```

In [32]:

```
# Create a set 'ss_stat'
ss_stat_df = pd.read_csv("ss_stats.csv")

ss_stat_df["Tags"] = ss_stat_df["Tags"].str.replace('><',',').str.replace('<', '')\
 .str.replace('>', '')\
 .str.split(',') # clean dataframe

print(ss_stat_df.head(5),'\n')



ss_stat_list = []
for index, row in ss_stat_df.iterrows():
    for tag in row['Tags']:
        ss_stat_list.append(tag)
print(ss_stat_list[:10], '\n')

print(len(ss_stat_list),'\n')

ss_stat = set(ss_stat_list)

print(len(ss_stat))
```

```
                                                Tags     TagName
0                            [bigdata, statistics]  statistics
1                            [statistics, bigdata]  statistics
2  [bigdata, machine-learning, databases, statist...  statistics
3  [machine-learning, statistics, feature-selection]  statistics
4                    [statistics, reference-request]  statistics

['bigdata', 'statistics', 'statistics', 'bigdata', 'bigdata', 'machine-lea
rning', 'databases', 'statistics', 'education', 'machine-learning']

2535

220
```

Superset machine-learning

Run following query at SEDE

``` SELECT Tags, TagName FROM Posts AS p INNER JOIN PostTags AS pt ON pt.PostId=p.id INNER JOIN Tags AS t ON t.id=pt.TagId WHERE TagName = 'machine-learning';
```

In [33]:

```
# Create a set 'ss_ml'
ss_ml_df = pd.read_csv("ss_ml.csv")

ss_ml_df["Tags"] = ss_ml_df["Tags"].str.replace('><',',').str.replace('<', '')\
 .str.replace('>', '')\
 .str.split(',') # clean dataframe

print(ss_ml_df.head(5),'\n')



ss_ml_list = []
for index, row in ss_ml_df.iterrows():
    for tag in row['Tags']:
        ss_ml_list.append(tag)
print(ss_ml_list[:10], '\n')

print(len(ss_ml_list),'\n')

ss_ml = set(ss_ml_list)

print(len(ss_ml))
```

```
                                                Tags           TagName
0                                  [machine-learning]  machine-learning
1                [machine-learning, bigdata, libsvm]  machine-learning
2            [machine-learning, predictive-modeling]  machine-learning
3                    [data-mining, machine-learning]  machine-learning
4  [machine-learning, dimensionality-reduction, p...  machine-learning

['machine-learning', 'machine-learning', 'bigdata', 'libsvm', 'machine-lea
rning', 'predictive-modeling', 'data-mining', 'machine-learning', 'machine
-learning', 'dimensionality-reduction']

27613

476
```

In [34]:

```
# Create a set 'ss_dl'
ss_dl_df = pd.read_csv("ss_dl.csv")

ss_dl_df["Tags"] = ss_dl_df["Tags"].str.replace('><',',').str.replace('<', '')\
 .str.replace('>', '')\
 .str.split(',') # clean dataframe

print(ss_dl_df.head(5),'\n')



ss_dl_list = []
for index, row in ss_dl_df.iterrows():
    for tag in row['Tags']:
        ss_dl_list.append(tag)
print(ss_dl_list[:10], '\n')

print(len(ss_dl_list),'\n')

ss_dl = set(ss_dl_list)

print(len(ss_dl))
```

```
                                                Tags        TagName
0  [machine-learning, neural-network, deep-learni...  deep-learning
1                    [neural-network, deep-learning]  deep-learning
2  [machine-learning, data-mining, neural-network...  deep-learning
3  [machine-learning, classification, deep-learning]  deep-learning
4                  [machine-learning, deep-learning]  deep-learning

['machine-learning', 'neural-network', 'deep-learning', 'optimization', 'h
yperparameter', 'neural-network', 'deep-learning', 'machine-learning', 'da
ta-mining', 'neural-network']

12249

369
```

In [35]:

```
# We will join ss_ml and ss_dl to create one superset ss_ai
ss_ai = ss_ml.union(ss_dl)

print(len(ss_ai))
```

```
495
```

Classification of Questions into Data Science Disciplines

- We have now created 3 supersets i.e. `ss_data` , `ss_stat` and `ss_ai` containing unique values of tags related to three broad disciplines: "Data Management (dm)", "Statistics (stat)" and "Artificial Intelligence (ai)"
- We will now see, how many questions ever asked on SEDE (Data Science) fall under these disciplines.
- Run following query at SEDE to get all_questions:

```
    SELECT Id, CreationDate, Tags
      FROM posts
     WHERE PostTypeId = 1;
```

In [36]:

```
# Create dataframe aq
aq = pd.read_csv("all_questions.csv")

aq["Tags"] = aq["Tags"].str.replace('><',',').str.replace('<', '')\
 .str.replace('>', '')\
 .str.split(',') # clean dataframe



print(aq.head(10),'\n')
```

```
      Id         CreationDate  \
0  12954  2016-07-23 07:05:30
1  12956  2016-07-23 09:46:42
2  12958  2016-07-23 12:34:34
3  12959  2016-07-23 21:02:17
4  12960  2016-07-23 22:33:04
5  12961  2016-07-24 08:33:43
6  12964  2016-07-24 10:17:35
7  12974  2016-07-25 04:31:08
8  12978  2016-07-25 13:00:12
9  12979  2016-07-25 14:22:25

                                          Tags
0       [python, clustering, unsupervised-learning]
1  [deep-learning, rnn, normalization, batch-norm...
2  [classification, clustering, statistics, missi...
3  [machine-learning, markov-process, audio-recog...
4           [text-mining, feature-extraction, text]
5                     [classification, svm, graphs]
6                [neural-network, tensorflow, rnn]
7                             [python, tensorflow]
8                             [recommender-system]
9  [python, feature-extraction, image-classificat...
```

- We will create three additional columns in the aq dataframe `dm`, `stat`, `ai` and populate them with 0
- if the tags list is a subset of any of the supersets respective column will change to 1

In [37]:

```
zeros = []
for index, row in aq.iterrows():
    zeros.append(0)
aq["dm"] = zeros
aq["stat"] = zeros
aq["ai"] = zeros

print(aq.head(5),'\n')
```

```
      Id         CreationDate  \
0  12954  2016-07-23 07:05:30
1  12956  2016-07-23 09:46:42
2  12958  2016-07-23 12:34:34
3  12959  2016-07-23 21:02:17
4  12960  2016-07-23 22:33:04

                                                Tags  dm  stat  ai
0        [python, clustering, unsupervised-learning]   0     0   0
1  [deep-learning, rnn, normalization, batch-norm...   0     0   0
2  [classification, clustering, statistics, missi...   0     0   0
3  [machine-learning, markov-process, audio-recog...   0     0   0
4            [text-mining, feature-extraction, text]   0     0   0
```

In [38]:

```
print(ss_data)
```

{'stanford-nlp', 'google', 'faster-rcnn', 'logistic-regression', 'image-re
cognition', 'data-stream-mining', 'machine-learning', 'machine-learning-mo
del', 'word-embeddings', 'marketing', 'probability', 'numerical', 'predict
ion', 'transformer', 'ngboost', 'excel', 'dataframe', 'interpolation', 'ls
tm', 'twitter', 'data', 'csv', 'error-handling', 'ngrams', 'generalizatio
n', 'domain-adaptation', 'language-model', 'q-learning', 'active-learnin
g', 'programming', 'data-science-model', 'sensors', 'epochs', 'computer-vi
sion', 'word2vec', 'graphical-model', 'fastai', 'mlp', 'tsne', 'categorica
l-encoding', 'discriminant-analysis', 'features', 'processing', 'tools',
'sql', 'metric', 'annotation', 'batch-normalization', 'data-analysis', 'im
age', 'ocr', 'bert', 'anomaly-detection', 'generative-models', 'scikit-lea
rn', 'apache-spark', 'categorical-data', 'image-classification', 'distribu
tion', 'tableau', 'rnn', 'predictive-modeling', 'caffe', 'random-forest',
'distance', 'bias', 'svm', 'etl', 'python-3.x', 'theory', 'serialisation',
'orange3', 'text-mining', 'preprocessing', 'variance', 'noisification', 'v
ersion-control', 'feature-engineering', 'sequence', 'k-means', 'multilabel
-classification', 'information-retrieval', 'privacy', 'audio-recognition',
'java', 'labelling', 'reinforcement-learning', 'aggregation', 'convnet',
'loss-function', 'counts', 'cross-validation', 'boosting', 'windows', 'ter
minology', 'fuzzy-logic', 'pandas', 'sas', 'mnist', 'xgboost', 'python',
'feature-extraction', 'association-rules', 'data-imputation', 'unbalanced-
classes', 'neural', 'deep-network', 'imbalanced-learn', 'data.table', 'bin
ary', 'activation-function', 'named-entity-recognition', 'career', 'traini
ng', 'bayesian-networks', 'speech-to-text', 'math', 'implementation', 'sam
pling', 'learning', 'multi-instance-learning', 'dataset', 'naive-bayes-cla
ssifier', 'sentiment-analysis', 'parameter', 'crawling', 'mean-shift', 'sm
ote', 'heatmap', 'tensorflow', 'data-wrangling', 'performance', 'object-de
tection', 'ml', 'web-scraping', 'genetic-algorithms', 'multitask-learnin
g', 'beginner', 'json', 'parsing', 'noise', 'k-nn', 'multiclass-classifica
tion', 'text', 'torch', 'ai', 'dbscan', 'definitions', 'statistics', 'dime
nsionality-reduction', 'descriptive-statistics', 'evaluation', 'orange',
'encoding', 'linear-regression', 'regression', 'matlab', 'data-augmentatio
n', 'dplyr', 'knowledge-base', 'yolo', 'optimization', 'anonymization', 'r
ecurrent-neural-net', 'normalization', 'research', 'aws', 'methodology',
'normal-equation', 'topic-model', 'self-driving', 'colab', 'experiments',
'feature-selection', 'apache-hadoop', 'keras', 'cnn', 'feature-scaling',
'similarity', 'missing-data', 'methods', 'pyspark', 'finance', 'clusters',
'gan', 'convolution', 'infographics', 'neural-network', 'attention-mechani
sm', 'randomized-algorithms', 'autoencoder', 'outlier', 'data-cleaning',
'ensemble-learning', 'accuracy', 'labels', 'powerbi', 'scipy', 'multivaria
te-distribution', 'decision-trees', 'plotting', 'pytorch', 'machine-transl
ation', 'activity-recognition', 'freebase', 'ranking', 'weighted-data', 'i
python', 'one-hot-encoding', 'recommender-system', 'search', 'unsupervised
-learning', 'pca', 'nlp', 'model-selection', 'numpy', 'visualization', 'ge
ospatial', 'scraping', 'perceptron', 'gradient-descent', 'bioinformatics',
'rstudio', 'non-parametric', 'project-planning', 'clustering', 'ab-test',
'simulation', 'class-imbalance', 'mutual-information', 'ensemble-modelin
g', 'matrix', 'r', 'dummy-variables', 'reference-request', 'metadata', 'ma
tplotlib', 'feature-construction', 'databases', 'object-recognition', 'for
ecasting', 'separable', 'matrix-factorisation', 'classification', 'overfit
ting', 'data-mining', 'scoring', 'weka', 'software-recommendation', 'graph
s', 'image-preprocessing', 'hyperparameter-tuning', 'time-series', 'superv
ised-learning', 'linearly-separable', 'ridge-regression', 'automatic-summa
rization', 'books', 'bigdata', 'kaggle', 'gridsearchcv', 'correlation', 'n
atural-language-process', 'tesseract', 'open-source', 'text-classificatio
n', 'deep-learning', 'algorithms', 'social-network-analysis', 'reshape',
'data-formats'}

In [39]:

```
# Convert "Tags" column to set using apply method
aq["Tags"] = aq["Tags"].apply(set)

print(aq.head(5),'\n')
```

```
      Id         CreationDate  \
0  12954  2016-07-23 07:05:30
1  12956  2016-07-23 09:46:42
2  12958  2016-07-23 12:34:34
3  12959  2016-07-23 21:02:17
4  12960  2016-07-23 22:33:04

                                                Tags  dm  stat  ai
0       {python, unsupervised-learning, clustering}   0     0   0
1  {batch-normalization, normalization, deep-lear...   0     0   0
2  {classification, statistics, missing-data, clu...   0     0   0
3  {markov-process, machine-learning, audio-recog...   0     0   0
4           {text-mining, feature-extraction, text}   0     0   0
```

In [40]:

```
""" Identifies whether a set is a subset of a superset"""
def inset(x, ss):
    if x.issubset(ss):
        return 1
    else:
        return 0

aq["dm"] = aq["Tags"].apply(inset, ss=ss_data)
aq["stat"] = aq["Tags"].apply(inset, ss=ss_stat)
aq["ai"] = aq["Tags"].apply(inset, ss=ss_ai)

print(aq.head(20))
```

```
       Id         CreationDate  \
0   12954  2016-07-23 07:05:30
1   12956  2016-07-23 09:46:42
2   12958  2016-07-23 12:34:34
3   12959  2016-07-23 21:02:17
4   12960  2016-07-23 22:33:04
5   12961  2016-07-24 08:33:43
6   12964  2016-07-24 10:17:35
7   12974  2016-07-25 04:31:08
8   12978  2016-07-25 13:00:12
9   12979  2016-07-25 14:22:25
10  60250  2019-09-16 01:20:59
11  60251  2019-09-16 01:42:20
12  60256  2019-09-16 05:42:00
13  60257  2019-09-16 06:06:11
14  60258  2019-09-16 06:52:32
15  60260  2019-09-16 08:43:53
16  60261  2019-09-16 08:50:27
17  60262  2019-09-16 08:56:51
18  60264  2019-09-16 09:03:19
19  60266  2019-09-16 09:28:08

                                                 Tags  dm  stat  ai
0         {python, unsupervised-learning, clustering}   1     1   1
1   {batch-normalization, normalization, deep-lear...   1     0   1
2   {classification, statistics, missing-data, clu...   1     1   1
3   {markov-process, machine-learning, audio-recog...   0     0   1
4             {text-mining, feature-extraction, text}   1     0   1
5                       {classification, svm, graphs}   1     1   1
6                  {neural-network, tensorflow, rnn}   1     1   1
7                               {python, tensorflow}   1     1   1
8                               {recommender-system}   1     1   1
9   {python, image-recognition, feature-extraction...   1     1   1
10                            {linux, visualization}   0     0   1
11                     {svm, machine-learning, math}   1     1   1
12                                {pyspark, bigdata}   1     0   1
13    {object-detection, keras, tensorflow, pytorch}   1     1   1
14  {machine-translation, natural-language-process...   1     0   1
15                {loss-function, linear-regression}   1     1   1
16                                  {nlp, nlg, lstm}   0     0   1
17                              {data-science-model}   1     1   1
18                              {feature-extraction}   1     1   1
19                     {natural-language-process}   1     0   1
```

In [41]:

```
print(aq.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24891 entries, 0 to 24890
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Id            24891 non-null  int64 
 1   CreationDate  24891 non-null  object
 2   Tags          24891 non-null  object
 3   dm            24891 non-null  int64 
 4   stat          24891 non-null  int64 
 5   ai            24891 non-null  int64 
dtypes: int64(4), object(2)
memory usage: 1.1+ MB
None
```

Set membership

Let's recall that the supersets are collections of tags which have ever been used with a combination of marker tags `machine=learning` , `deep-learning` , `statistcs` and `datasets` of 3 main disciplines i.e. "AI", "Maths" and "Data Management" which we identified based on domain knowledge. The larger the superset the more likely it will be for any combination of tags or even a single tag to be identified as a subset of a particular superset.

As seen below, at present the membership criteria is quite lax and is solely contingent upon "size of the superset".

In [42]:

```
# Finding the size of membership
print ('ai members:', '\n', sum(aq["ai"]))

print ('dm members:', '\n', sum(aq["dm"]))

print ('stat members:', '\n', sum(aq["stat"]))
```

```
ai members:
 24420
dm members:
 19895
stat members:
 17729
```

Refining "Discipline" Membership Criteria

As seen above, out of approximately 22,000 questions, more than 21,000 are classified as belonging to AI disciplines such as `machine-learning` and `deep-learning` . So, it should not be enough to be a member of a superset.

Since, the supersets control which particular combination of `tags` are its members, we must impose additionsl conditions for membership.

We will amend the function `inset` to `indiscip` to reflect following additional/ existing conditions:

- **Length** of the set of tags must be at least 2, as a serious questioner homes in on topic of relevance from broad categories to specific topic or vice-versa. It is not necessary that, he will use the key tags such as `machine-learning` or `dataset` or `statistics` as people with different knowledge and skills are likely to have different conceptions of what is a broad category.
- **Unique Membership** We cannot implemenr unique membership, as there are lot of interlinks and cross-overs. A subset can be identified as belonging to more than one discipline. This is presently `True` . But we need to place some constraints on membership based on 'tags-combination'. For this, we will impose following condition through `indiscp` function: **Out of a given combination of 'tags' for a question which is a subset of a particular superset, at least one 'tag' out of the combination, should not be a member of one of the other two other supersets.**
- **Size of Superset** If we run following query at SEDE, we get the total number of tags used on the site.

```
SELECT COUNT(TagName)
  FROM Tags
596
```

In order to qualify as a discipline, the size of superset should be substantial enough. We fix it arbitrarily as **25%** (150) of unique tag names.

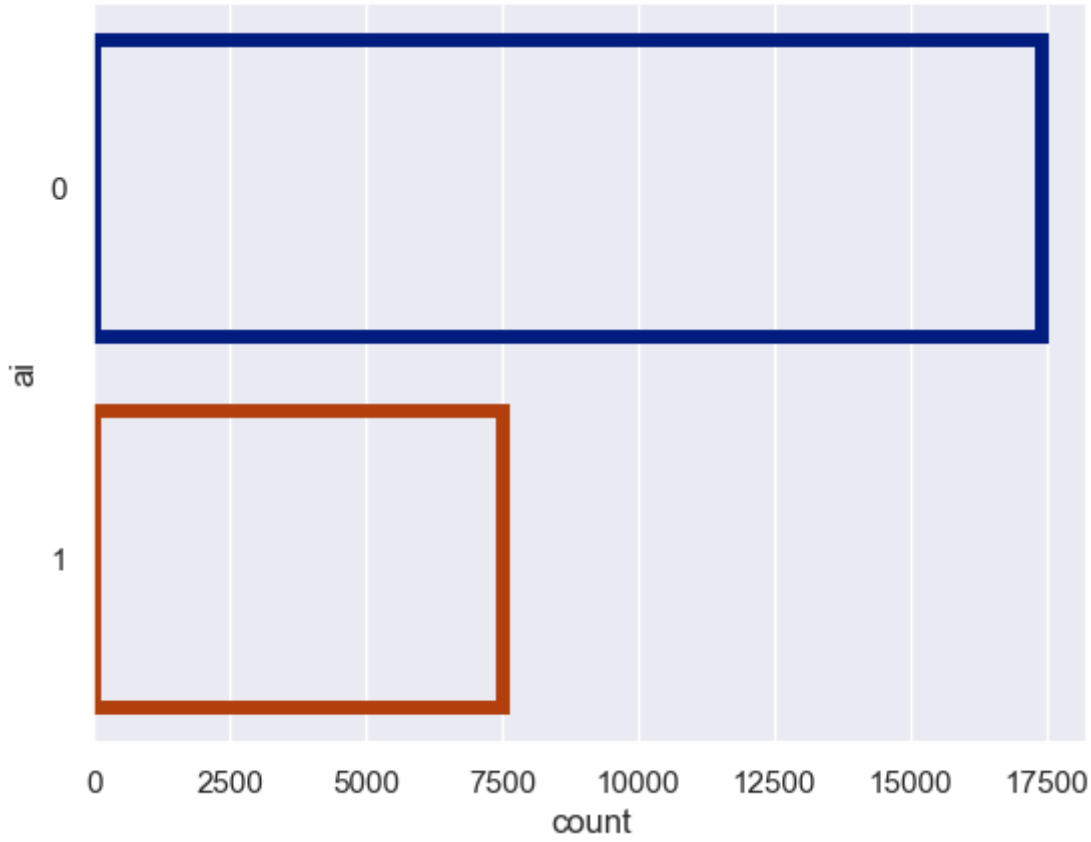
In [43]:

```
"""Identifies whether a set of 'tags' is a subset of a data science discipline"""
def indiscip(x, ss1, ss2, ss3):
    if (x.issubset(ss1) and len(x) >=2 and len(ss1) > 150) and (len(x.difference(ss2))>=
        return 1
    else:
        return 0

aq["dm"] = aq["Tags"].apply(indiscip, ss1=ss_data, ss2=ss_stat, ss3=ss_ai)
aq["stat"] = aq["Tags"].apply(indiscip, ss1=ss_stat, ss2=ss_data, ss3=ss_ai)
aq["ai"] = aq["Tags"].apply(indiscip, ss1=ss_ai, ss2=ss_data, ss3=ss_stat)


print ('ai questions:', '\n', sum(aq["ai"]))

print ('dm questions:', '\n', sum(aq["dm"]))

print ('stat questions:', '\n', sum(aq["stat"]))
```

```
ai questions:
 7504
dm questions:
 3134
stat questions:
 1155
```

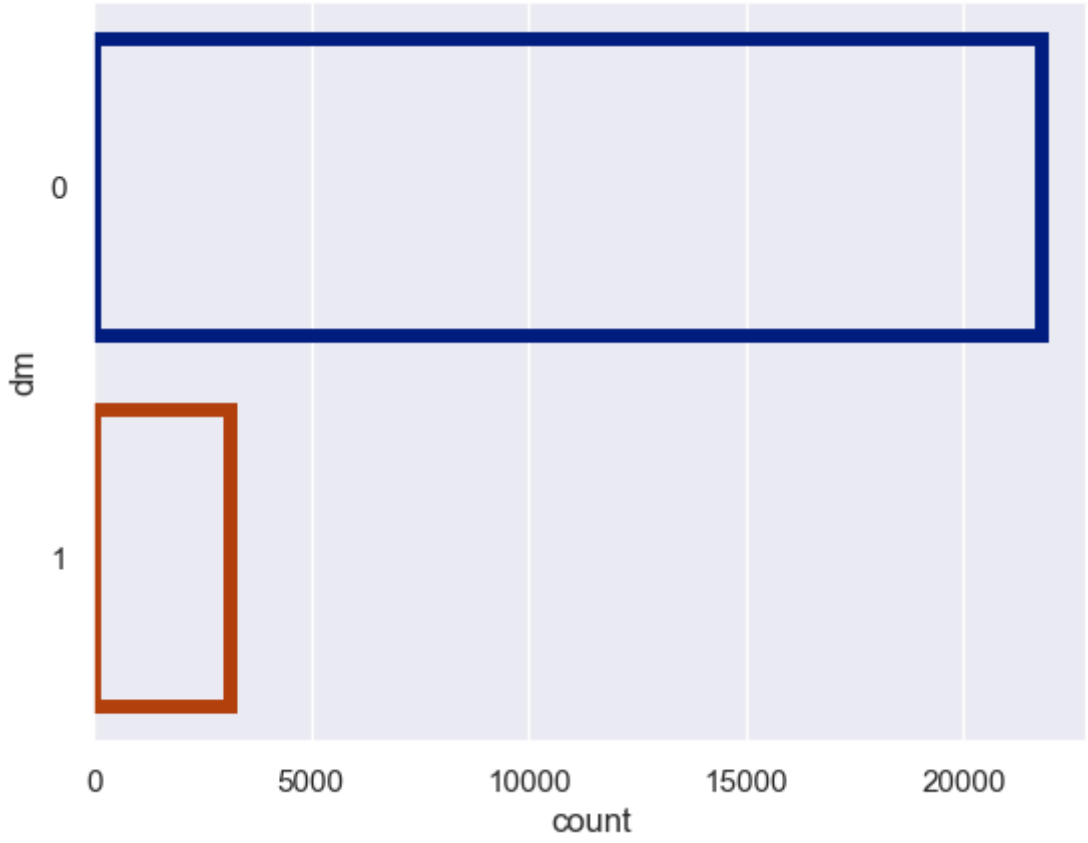
In [44]:

```
# Plot results for AI
import seaborn as sns
sns.set(style="darkgrid")

ax1= sns.countplot(y="ai", data=aq, facecolor=(0, 0, 0, 0),
                   linewidth=5,
                   edgecolor=sns.color_palette("dark", 3))
```



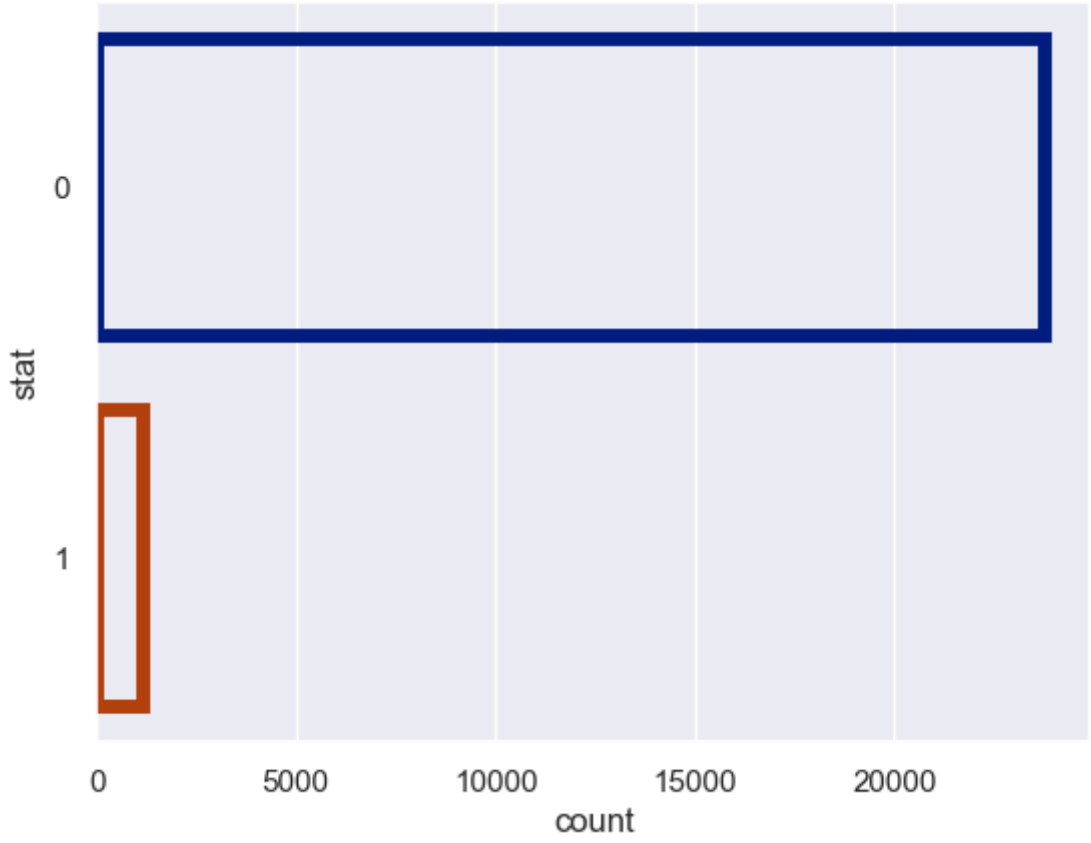
In [45]:

```
# Plot results for Data Management
ax= sns.countplot(y="dm", data=aq, facecolor=(0, 0, 0, 0),
                  linewidth=5,
                  edgecolor=sns.color_palette("dark", 3))
```



In [46]:

```
# Plot results from Statistics
ax=sns.countplot(y="stat", data=aq, facecolor=(0, 0, 0, 0),
                 linewidth=5,
                 edgecolor=sns.color_palette("dark", 3))
```



Exploring Time-Series for all Disciplines

From above it can be seen that **AI** and its components `machine-learning` and subfield `deep-learning` form the major thrust of questions at SEDS. However, other disciplines are not insignificant, and We will explore them further as a time series.

- We have already classified all `ai` questions as `deep-learning` questions. We will now use the dataframe `aq` to:
- Count how many questions per discipline including `deep-learning` are asked per time period.
- The total amount of questions per time period.
- How many `deep learning` and other disciplines' questions there are relative to the total amount of questions per time period.

Analysis Strategy

- We will change `CreationDate` to string in `yymm` format
- Define a function `get_qtr` which uses string functions to extract quarters from string in format `yyQn`, where `n` is the quarter number
- `apply` function `get_qtr` to `CreationDate` and convert it to `yyQn` format
- `groupby` `yyQn` and aggregate count for all disciplines `ai`, `dm` and `stat`

In [47]:

```
# change "CreationDate" to string
aq_ts = aq
aq_ts["qtr"] = aq_ts["CreationDate"].astype(str).str[2:7].str.replace('-','')
aq_ts.drop(labels="CreationDate", axis=1, inplace=True)
```

In [48]:

```
aq_ts=aq_ts.sort_values(by='qtr', ascending=True)
print(aq_ts.head())
```

```
       Id                                                Tags  dm  stat  ai
\
3464  184                                           {tools}   0     0   0
1948   71                               {statistics, bigdata}   0     0   0
3455  134  {map-reduce, scalability, mongodb, apache-hadoop}   0     0   1
3456  138                {bigdata, performance, efficiency}   0     1   1
3457  143  {.net, nosql, data-indexing-techniques, effici...   0     0   0

       qtr
3464  1405
1948  1405
3455  1405
3456  1405
3457  1405
```

In [49]:

```
# Define function get_qtr
""" extracts quarters from yymm"""
def get_qtr(string):
    year = int(string[0:2])*100
    month = int(string)-year
    qtr = int(((month)-1)/3)+1
    return '{y}Q{q}'.format(y=string[0:2], q=qtr)
```

In [50]:

```
# apply 'get_qtr' to 'aq_ts["qtr"]'
aq_ts["qtr"] = aq_ts["qtr"].apply(get_qtr)
```

In [51]:

```
print(aq_ts.tail())
```

```
          Id                                               Tags  dm  stat
ai  \
23500  76923        {graphical-model, machine-learning, graphs}   0     0
0
23499  76922        {time-series, data-analysis, visualization}   0     0
0
23498  76921  {ndcg, feature-engineering, recommender-system...   0     0
0
23557  76972              {training, xgboost, machine-learning}   0     0
0
24025  77093  {vector-space-models, attention-mechanism, dee...   0     0
1

        qtr
23500  20Q3
23499  20Q3
23498  20Q3
23557  20Q3
24025  20Q3
```

In [52]:

```
import numpy as np
aq_ts_pt = aq_ts.pivot_table(values = ["ai","dm","stat"], index = "qtr", aggfunc=(np.sum
```

In [53]:

```
print(aq_ts_pt)
```

```
        ai   dm  stat
qtr
14Q2    51   11    19
14Q3    34   11     9
14Q4    29    9     6
15Q1    26   16     6
15Q2    57   24    12
15Q3    48   17    16
15Q4    74   24    13
16Q1   120   42    21
16Q2   132   47    19
16Q3   126   42    32
16Q4   124   43    33
17Q1   168   86    33
17Q2   168   63    39
17Q3   171   79    28
17Q4   235   88    42
18Q1   335  166    44
18Q2   402  177    83
18Q3   412  178    61
18Q4   367  154    61
19Q1   534  231    85
19Q2   585  248    87
19Q3   735  310    99
19Q4   732  322    87
20Q1   793  319    86
20Q2  1005  410   130
20Q3    41   17     4
```

In [54]:

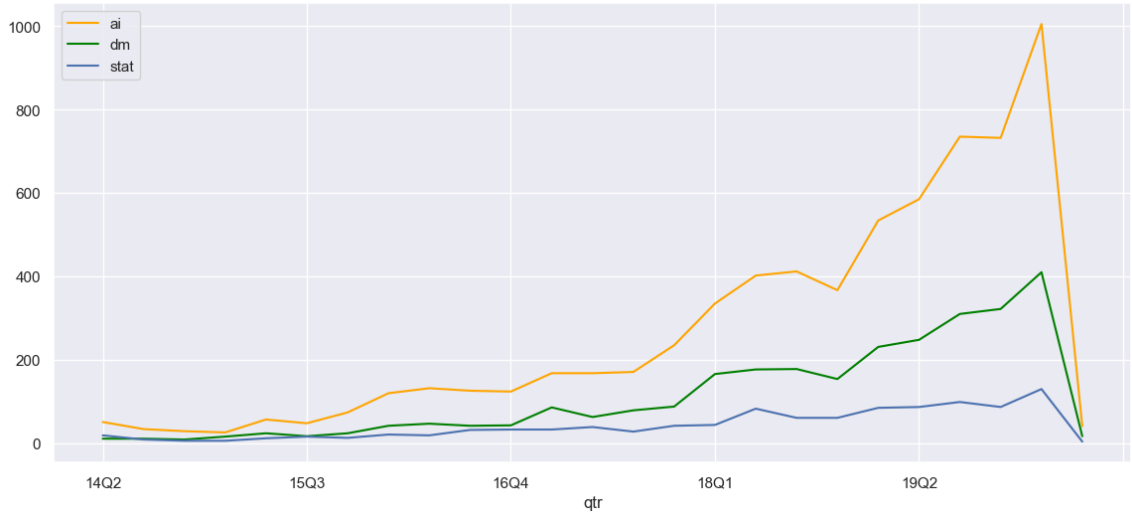
```
aq_ts_pt.drop(labels="20Q1", inplace=True)
```

In [55]:

```
# Comparative Time Series Curves for "all disciplines"

fig, axes = plt.subplots()
fig.set_size_inches((14, 6))
#Log scaling on x axis due to very large numbers
aq_ts_pt["ai"].plot(kind="line", color='orange', subplots=True, label="ai", legend=True)
aq_ts_pt["dm"].plot(kind="line", color='green', subplots=True, label="dm", legend=True)
aq_ts_pt["stat"].plot(kind="line", subplots=True, label="stat", legend=True)
```

Out[55]:

```
array([<Axes: xlabel='qtr'>], dtype=object)
```

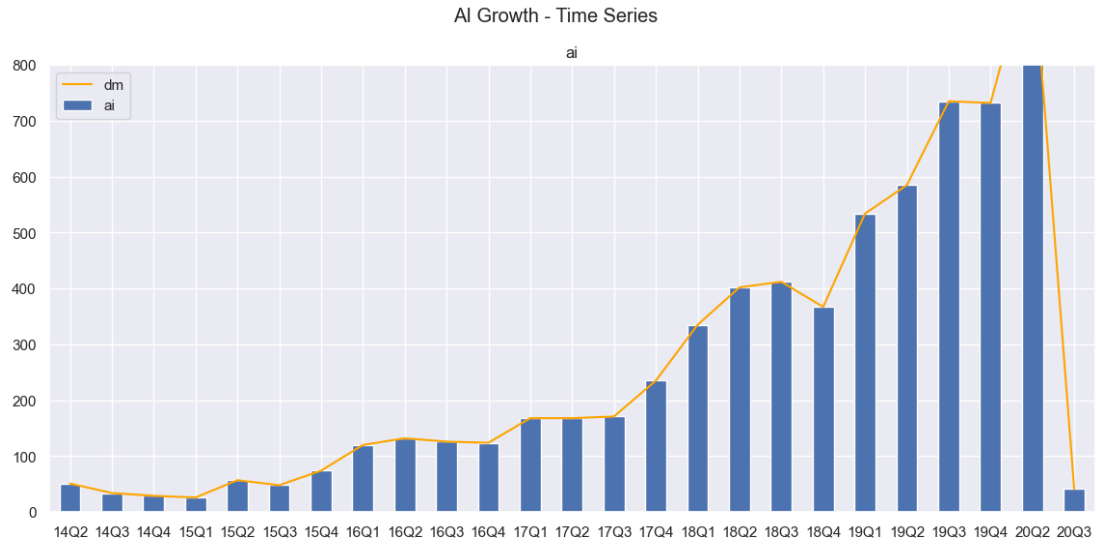


In [56]:

```
# Plotting Time Series: "AI Growth"

fig, axes = plt.subplots()
fig.set_size_inches((14, 6))
plt.ylim(0, 800)

aq_ts_pt["ai"].plot(kind="bar", subplots=True, label="ai", legend=True, title = "AI Grow
aq_ts_pt["ai"].plot(kind="line", color='orange', subplots=True, label="dm", legend=True)
```

Out[56]:

```
array([<Axes: title={'center': 'ai'}, xlabel='qtr'>], dtype=object)
```

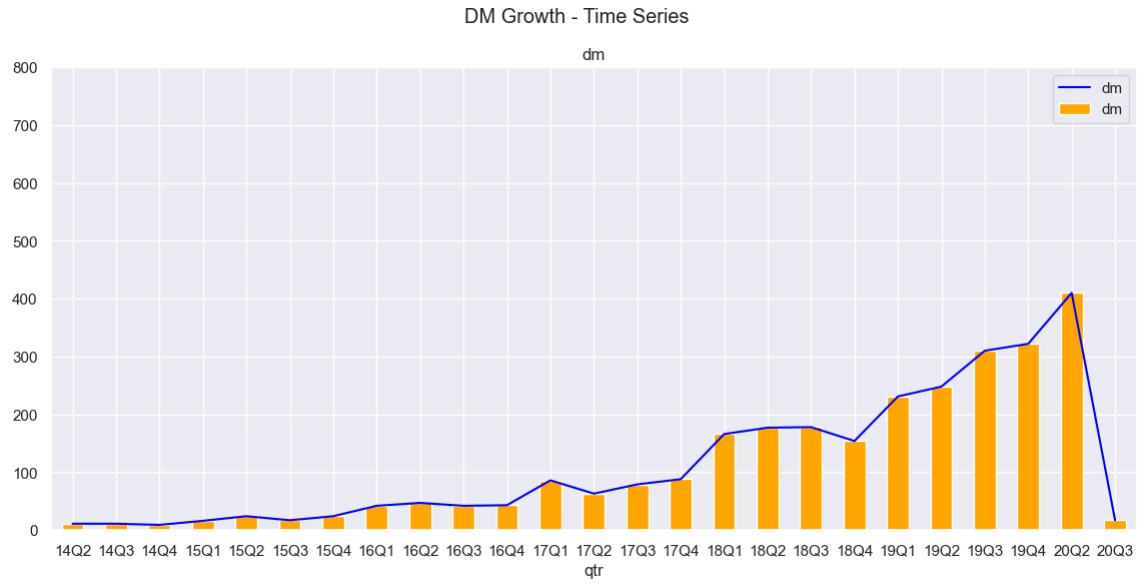


In [57]:

```
# Plotting Time Series: "DM Growth"

fig, axes = plt.subplots()
fig.set_size_inches((14, 6))
plt.ylim(0, 800)
aq_ts_pt["dm"].plot(kind="bar", color='orange', subplots=True, label="dm", legend=True,
aq_ts_pt["dm"].plot(kind="line", color='blue', subplots=True, label="dm", legend=True)
```

Out[57]:

```
array([<Axes: title={'center': 'dm'}, xlabel='qtr'>], dtype=object)
```

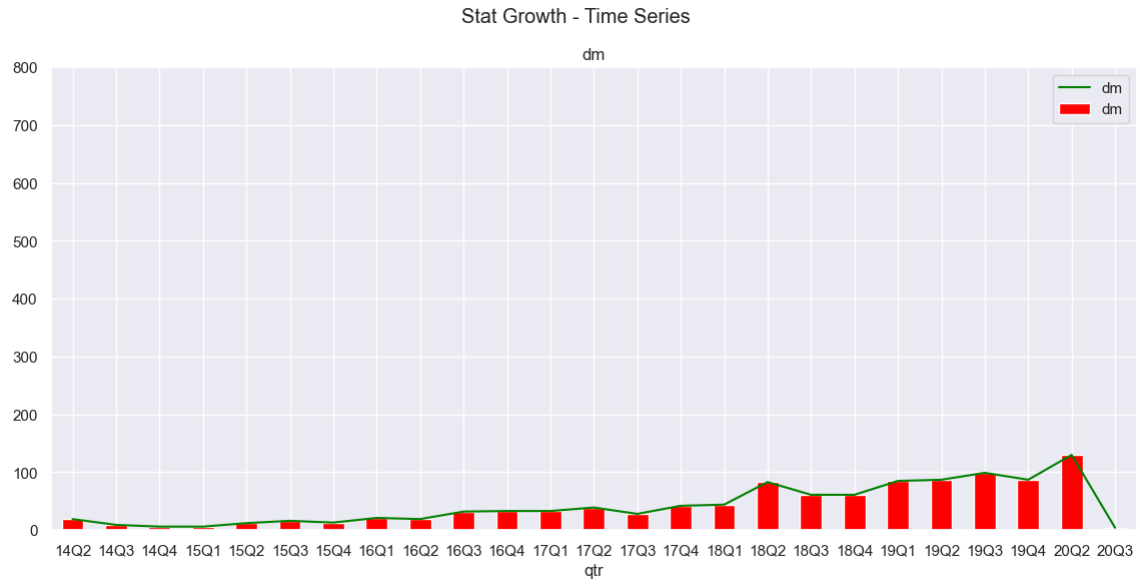


In [58]:

```
# Plotting Time Series: "Stat Growth"

fig, axes = plt.subplots()
fig.set_size_inches((14, 6))
plt.ylim(0, 800)
aq_ts_pt["stat"].plot(kind="bar", color='red', subplots=True, label="dm", legend=True, t
aq_ts_pt["stat"].plot(kind="line", color='green', subplots=True, label="dm", legend=True
```

Out[58]:

```
array([<Axes: title={'center': 'dm'}, xlabel='qtr'>], dtype=object)
```



Recommendations

- It is clear from above that the AI subfields including `deep-learning` and to an extent `data-management` subfields have shown marked growth since 4th quarter of 2018. Perhaps, this is indicative of the interest in the field in general. A major AI discipline `maths-statistics` has not registered much interset despite being a major requirement for data analysis. Perhaps, it is due to requirement of STEM education in this field.
- Our content, for present, should be focused on `deep-learning` and `data-management` .