CONESTOGA
Connect Life and Learning

PROG73040
Big Data Integration and Processing
Winter 2024

W7 - Hands-On: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure

Instructor: Dr. Jasleen Kaur

Outline

- **Data Extraction from Big Data Systems (Week 6)**
- **Data Extraction Methods and Tools (Week 6)**
  - Extracting Data through APIs
    - Overview of APIs
    - Demonstration: API Data Extraction
    - Hands-on Activity
  - Web Scrapping for Data Extraction
    - Introduction to Web Scraping
    - Tools and Libraries
    - Demonstration and Hands-on Activity

- **Hands-On Exercises (Week 6 & 7)**
  - Creating a Data Extraction Pipeline in ADF to extract data via API and save it in Azure Blob Storage
  - **Hands-On: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure**

In Class Hands-On Activity: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure

Exercise 1: Analyzing YouTube Video Statistics using YouTube API

Objective: Fetch data for videos from a specific YouTube channel using the YouTube API, ingest this data into Azure Databricks, and perform basic analysis to find the most viewed and most liked videos.

Step 1: Set Up YouTube API Access

1) Go to the Google Developers Console, create a new project, and enable the YouTube Data API v3 for your project.

2) **Generate an API Key**: In the credentials section, create an API key. This key will be used to authenticate your requests to the YouTube API.

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 2: Fetch YouTube Data Using Python

1) Install Required Libraries: ‘google-api-python-client’

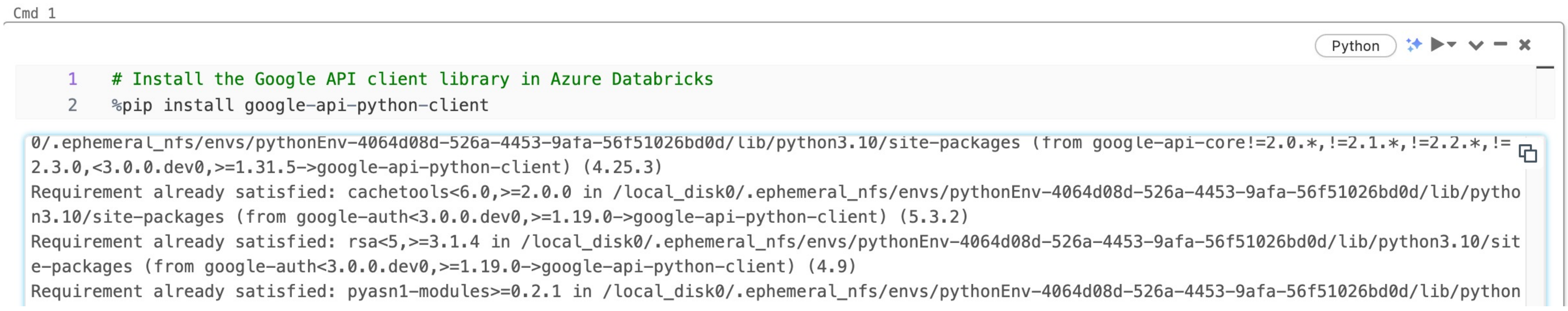
2) Fetch Data from the YouTube API: Use the API key to fetch data for a specific YouTube channel (example: TED Talks).

For simplicity fetch the latest n videos from a specific channel

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 2: Fetch YouTube Data Using Python (Cont.)
(Code with the description shared in separate file)

1) Install Required Libraries: ‘google-api-python-client’


Cmd 1

Python

```
# Install the Google API client library in Azure Databricks
%pip install google-api-python-client
```

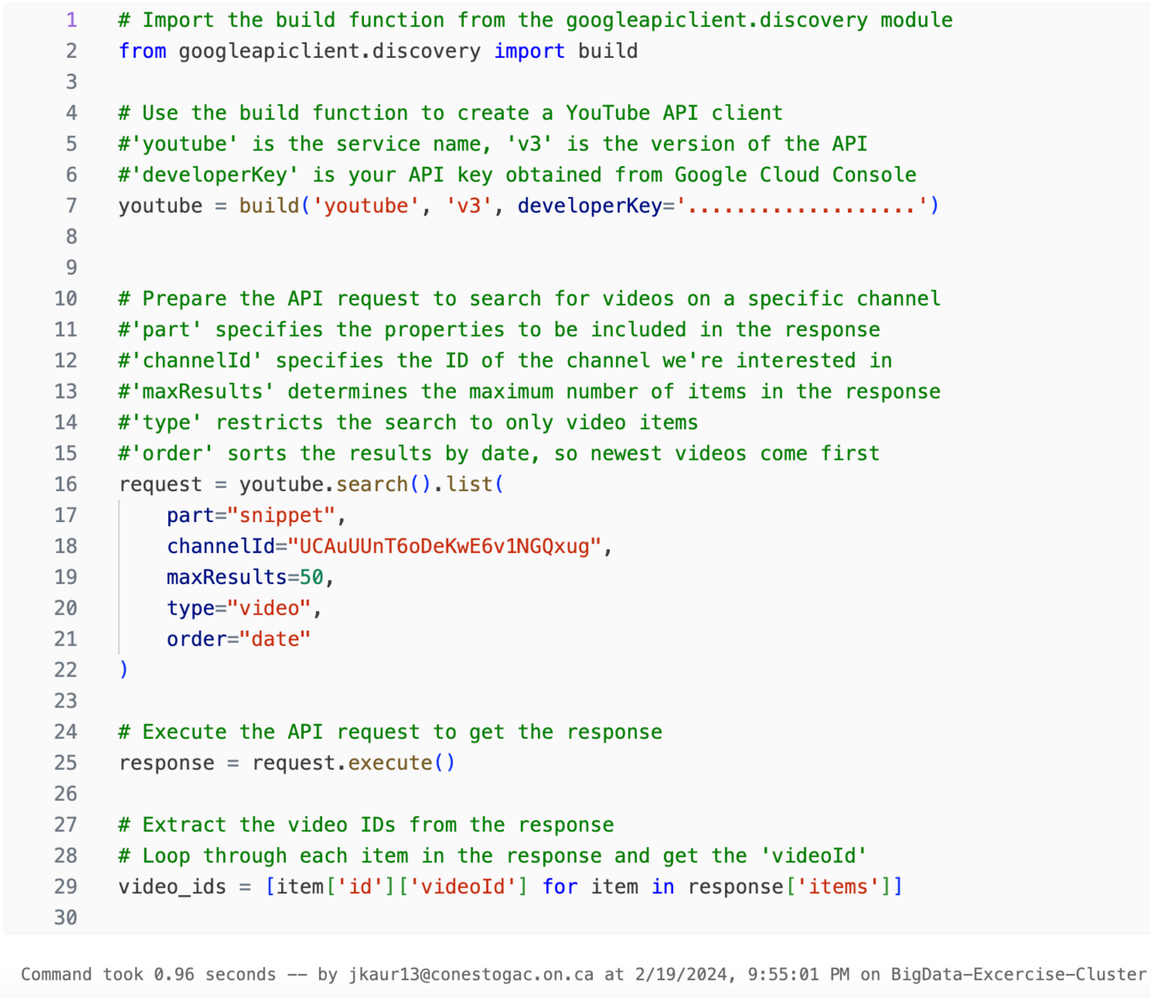
```
0/.ephemeral_nfs/envs/pythonEnv-4064d08d-526a-4453-9afa-56f51026bd0d/lib/python3.10/site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=
2.3.0,<3.0.0.dev0,>=1.31.5->google-api-python-client) (4.25.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-4064d08d-526a-4453-9afa-56f51026bd0d/lib/pytho
n3.10/site-packages (from google-auth<3.0.0.dev0,>=1.19.0->google-api-python-client) (5.3.2)
Requirement already satisfied: rsa<5,>=3.1.4 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-4064d08d-526a-4453-9afa-56f51026bd0d/lib/python3.10/sit
e-packages (from google-auth<3.0.0.dev0,>=1.19.0->google-api-python-client) (4.9)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-4064d08d-526a-4453-9afa-56f51026bd0d/lib/python
```

In Class Hands-On Activity: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 2: Fetch YouTube Data Using Python (Cont.)
(Code with the description shared in separate file)

2) Fetch Data from the YouTube using API Key



```
# Import the build function from the googleapiclient.discovery module
from googleapiclient.discovery import build

# Use the build function to create a YouTube API client
#'youtube' is the service name, 'v3' is the version of the API
#'developerKey' is your API key obtained from Google Cloud Console
youtube = build('youtube', 'v3', developerKey='..................')


# Prepare the API request to search for videos on a specific channel
#'part' specifies the properties to be included in the response
#'channelId' specifies the ID of the channel we're interested in
#'maxResults' determines the maximum number of items in the response
#'type' restricts the search to only video items
#'order' sorts the results by date, so newest videos come first
request = youtube.search().list(
    part="snippet",
    channelId="UCAuUUnT6oDeKwE6v1NGQxug",
    maxResults=50,
    type="video",
    order="date"
)

# Execute the API request to get the response
response = request.execute()

# Extract the video IDs from the response
# Loop through each item in the response and get the 'videoId'
video_ids = [item['id']['videoId'] for item in response['items']]

```

```
Command took 0.96 seconds -- by jkaur13@conestogac.on.ca at 2/19/2024, 9:55:01 PM on BigData-Excercise-Cluster
```

In Class Hands-On Activity: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 3: Fetch Statistics for Each Video and Load Data into a DataFrame

1) Retrieve Statistics for each video ID collected

2) Create a DataFrame: Convert the collected data into a Pandas DataFrame

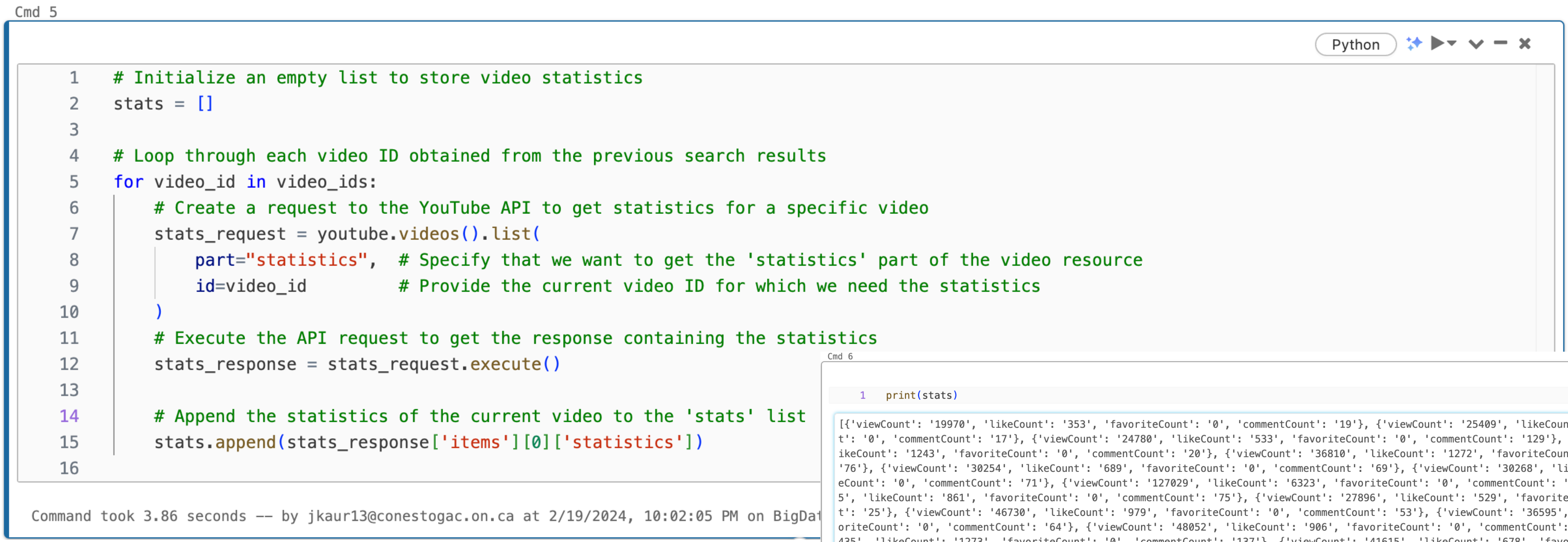
In Class Hands-On Activity: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 3: Fetch Statistics for Each Video and Load Data into a DataFrame

(Code with the description shared in separate file)

1) Retrieve Statistics for each video ID collected



Cmd 5

Python

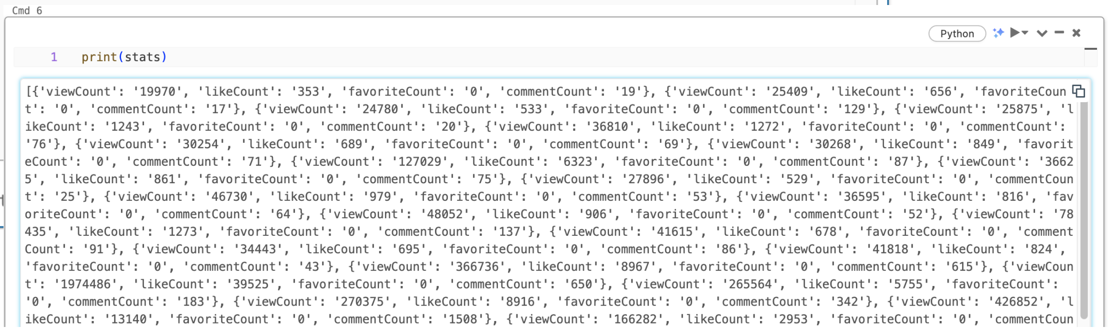
```python
# Initialize an empty list to store video statistics
stats = []

# Loop through each video ID obtained from the previous search results
for video_id in video_ids:
    # Create a request to the YouTube API to get statistics for a specific video
    stats_request = youtube.videos().list(
        part="statistics",  # Specify that we want to get the 'statistics' part of the video resource
        id=video_id         # Provide the current video ID for which we need the statistics
    )
    # Execute the API request to get the response containing the statistics
    stats_response = stats_request.execute()

    # Append the statistics of the current video to the 'stats' list
    stats.append(stats_response['items'][0]['statistics'])

```

```
Command took 3.86 seconds -- by jkaur13@conestogac.on.ca at 2/19/2024, 10:02:05 PM on BigDat
```



Cmd 6

Python

```python
print(stats)
```

```
[{'viewCount': '19970', 'likeCount': '353', 'favoriteCount': '0', 'commentCount': '19'}, {'viewCount': '25409', 'likeCount': '656', 'favoriteCoun
t': '0', 'commentCount': '17'}, {'viewCount': '24780', 'likeCount': '533', 'favoriteCount': '0', 'commentCount': '129'}, {'viewCount': '25875', 'l
ikeCount': '1243', 'favoriteCount': '0', 'commentCount': '20'}, {'viewCount': '36810', 'likeCount': '1272', 'favoriteCount': '0', 'commentCount':
'76'}, {'viewCount': '30254', 'likeCount': '689', 'favoriteCount': '0', 'commentCount': '69'}, {'viewCount': '30268', 'likeCount': '849', 'favorit
eCount': '0', 'commentCount': '71'}, {'viewCount': '127029', 'likeCount': '6323', 'favoriteCount': '0', 'commentCount': '87'}, {'viewCount': '3662
5', 'likeCount': '861', 'favoriteCount': '0', 'commentCount': '75'}, {'viewCount': '27896', 'likeCount': '529', 'favoriteCount': '0', 'commentCoun
t': '25'}, {'viewCount': '46730', 'likeCount': '979', 'favoriteCount': '0', 'commentCount': '53'}, {'viewCount': '36595', 'likeCount': '816', 'fav
oriteCount': '0', 'commentCount': '64'}, {'viewCount': '48052', 'likeCount': '906', 'favoriteCount': '0', 'commentCount': '52'}, {'viewCount': '78
435', 'likeCount': '1273', 'favoriteCount': '0', 'commentCount': '137'}, {'viewCount': '41615', 'likeCount': '678', 'favoriteCount': '0', 'comment
Count': '91'}, {'viewCount': '34443', 'likeCount': '695', 'favoriteCount': '0', 'commentCount': '86'}, {'viewCount': '41818', 'likeCount': '824',
'favoriteCount': '0', 'commentCount': '43'}, {'viewCount': '366736', 'likeCount': '8967', 'favoriteCount': '0', 'commentCount': '615'}, {'viewCoun
t': '1974486', 'likeCount': '39525', 'favoriteCount': '0', 'commentCount': '650'}, {'viewCount': '265564', 'likeCount': '5755', 'favoriteCount':
'0', 'commentCount': '183'}, {'viewCount': '270375', 'likeCount': '8916', 'favoriteCount': '0', 'commentCount': '342'}, {'viewCount': '426852', 'l
ikeCount': '13140', 'favoriteCount': '0', 'commentCount': '1508'}, {'viewCount': '166282', 'likeCount': '2953', 'favoriteCount': '0', 'commentCoun
```

In Class Hands-On Activity: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure

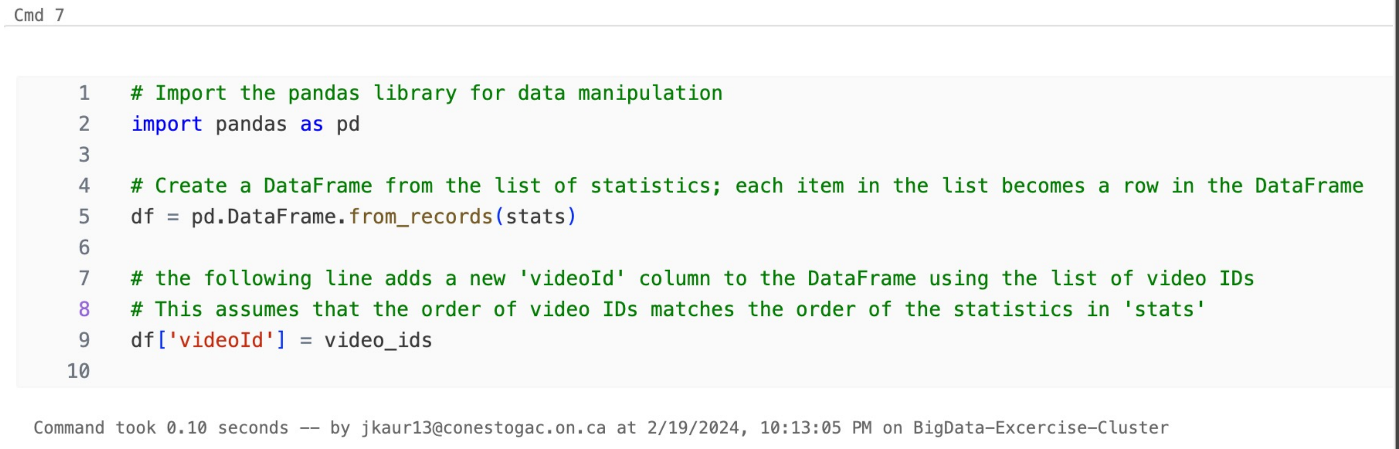
Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 3: Fetch Statistics for Each Video and Load Data into a DataFrame

(Code with the description shared in separate file)

2) Create a DataFrame

Convert the collected data into a Pandas DataFrame



Cmd 7

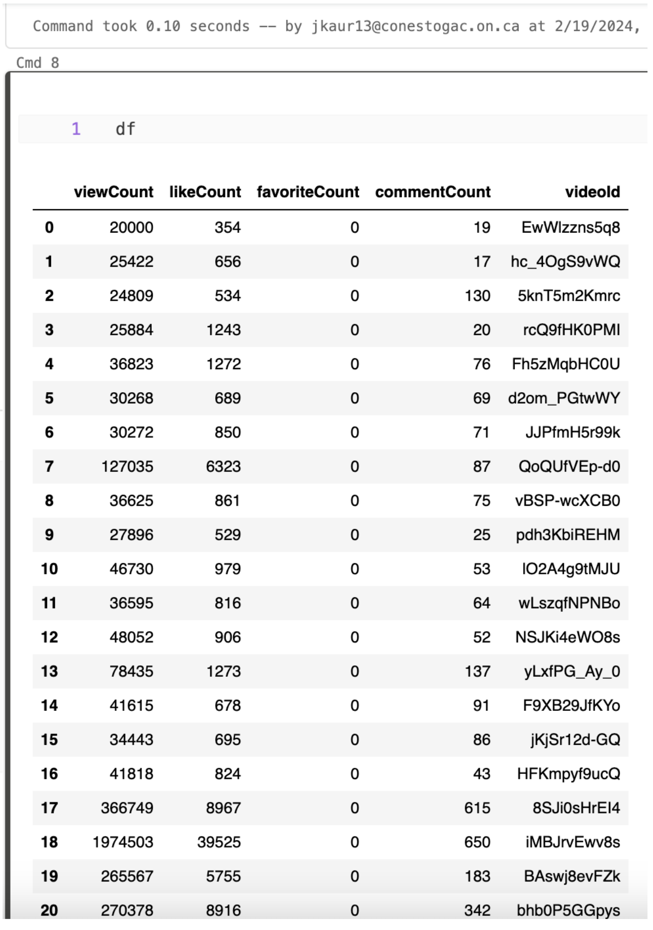
```
# Import the pandas library for data manipulation
import pandas as pd

# Create a DataFrame from the list of statistics; each item in the list becomes a row in the DataFrame
df = pd.DataFrame.from_records(stats)

# the following line adds a new 'videoId' column to the DataFrame using the list of video IDs
# This assumes that the order of video IDs matches the order of the statistics in 'stats'
df['videoId'] = video_ids

```

```
Command took 0.10 seconds -- by jkaur13@conestogac.on.ca at 2/19/2024, 10:13:05 PM on BigData-Excercise-Cluster
```



```
Command took 0.10 seconds -- by jkaur13@conestogac.on.ca at 2/19/2024,
```

Cmd 8

```
df
```

	viewCount	likeCount	favoriteCount	commentCount	videoId
0	20000	354	0	19	EwWlzzns5q8
1	25422	656	0	17	hc_4OgS9vWQ
2	24809	534	0	130	5knT5m2Kmrc
3	25884	1243	0	20	rcQ9fHK0PMI
4	36823	1272	0	76	Fh5zMqbHC0U
5	30268	689	0	69	d2om_PGtwWY
6	30272	850	0	71	JJPfmH5r99k
7	127035	6323	0	87	QoQUfVEp-d0
8	36625	861	0	75	vBSP-wcXCB0
9	27896	529	0	25	pdh3KbiREHM
10	46730	979	0	53	lO2A4g9tMJU
11	36595	816	0	64	wLszqfNPNBo
12	48052	906	0	52	NSJKi4eWO8s
13	78435	1273	0	137	yLxfPG_Ay_0
14	41615	678	0	91	F9XB29JfKYo
15	34443	695	0	86	jKjSr12d-GQ
16	41818	824	0	43	HFKmpyf9ucQ
17	366749	8967	0	615	8SJi0sHrEI4
18	1974503	39525	0	650	iMBJrvEwv8s
19	265567	5755	0	183	BAswj8evFZk
20	270378	8916	0	342	bhb0P5GGpys

In Class Hands-On Activity: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 4: Basic Data Exploration and Analysis

1) Explore Data

2) Find the Most Viewed Video: video with the highest number of views

3) Find the Most Liked Video: video with the highest number of likes

4) Comment Engagement: Evaluate the engagement of viewers with the videos based on the comment counts

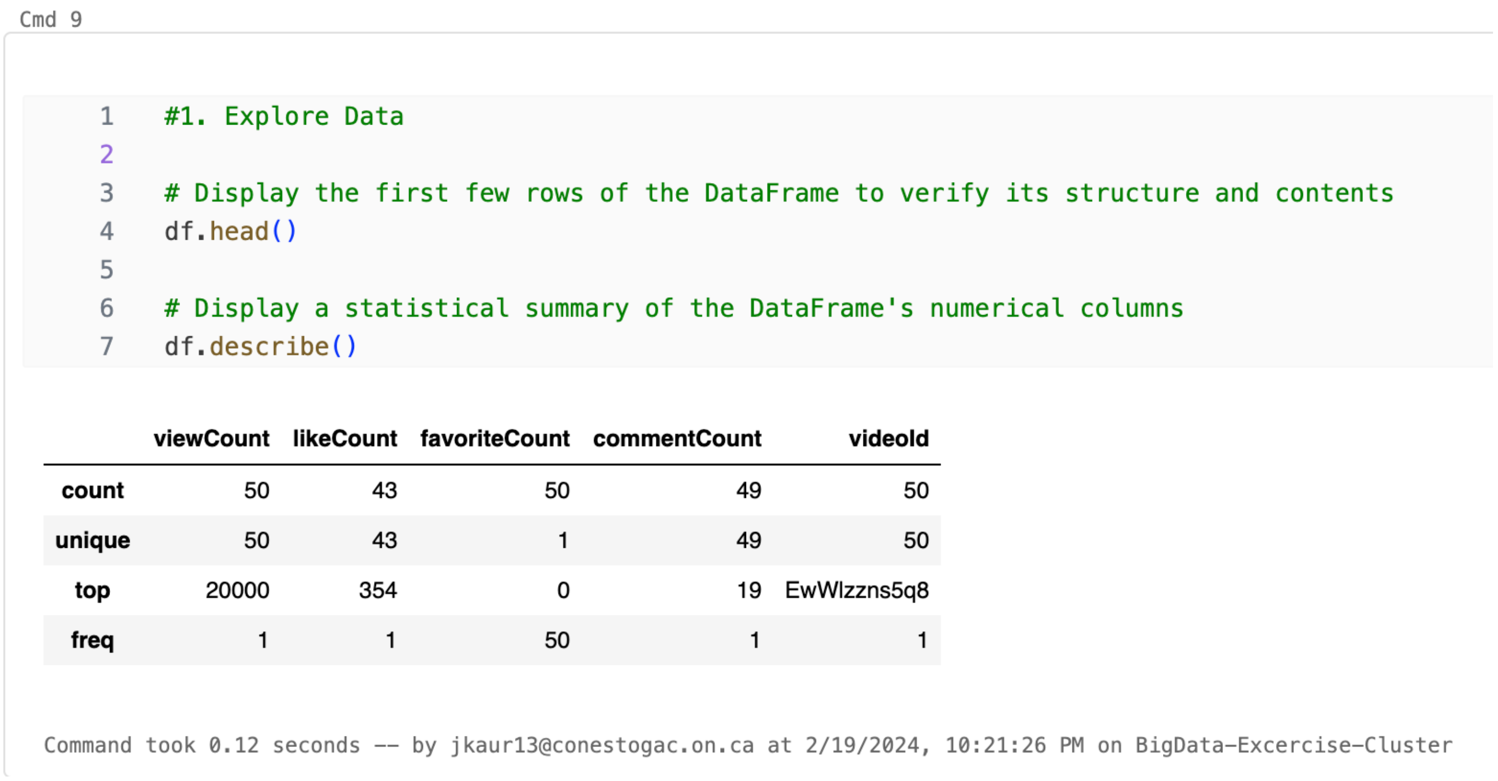
5) Engagement Ratio Analysis: Calculate the ratio of likes to views and comments to views to understand viewer engagement

In Class Hands-On Activity: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 4: Basic Data Exploration and Analysis
(Code with the description shared in separate file)

1) Explore Data



```
Cmd 9

#1. Explore Data

# Display the first few rows of the DataFrame to verify its structure and contents
df.head()

# Display a statistical summary of the DataFrame's numerical columns
df.describe()
```

	viewCount	likeCount	favoriteCount	commentCount	videoId
count	50	43	50	49	50
unique	50	43	1	49	50
top	20000	354	0	19	EwWlzzns5q8
freq	1	1	50	1	1

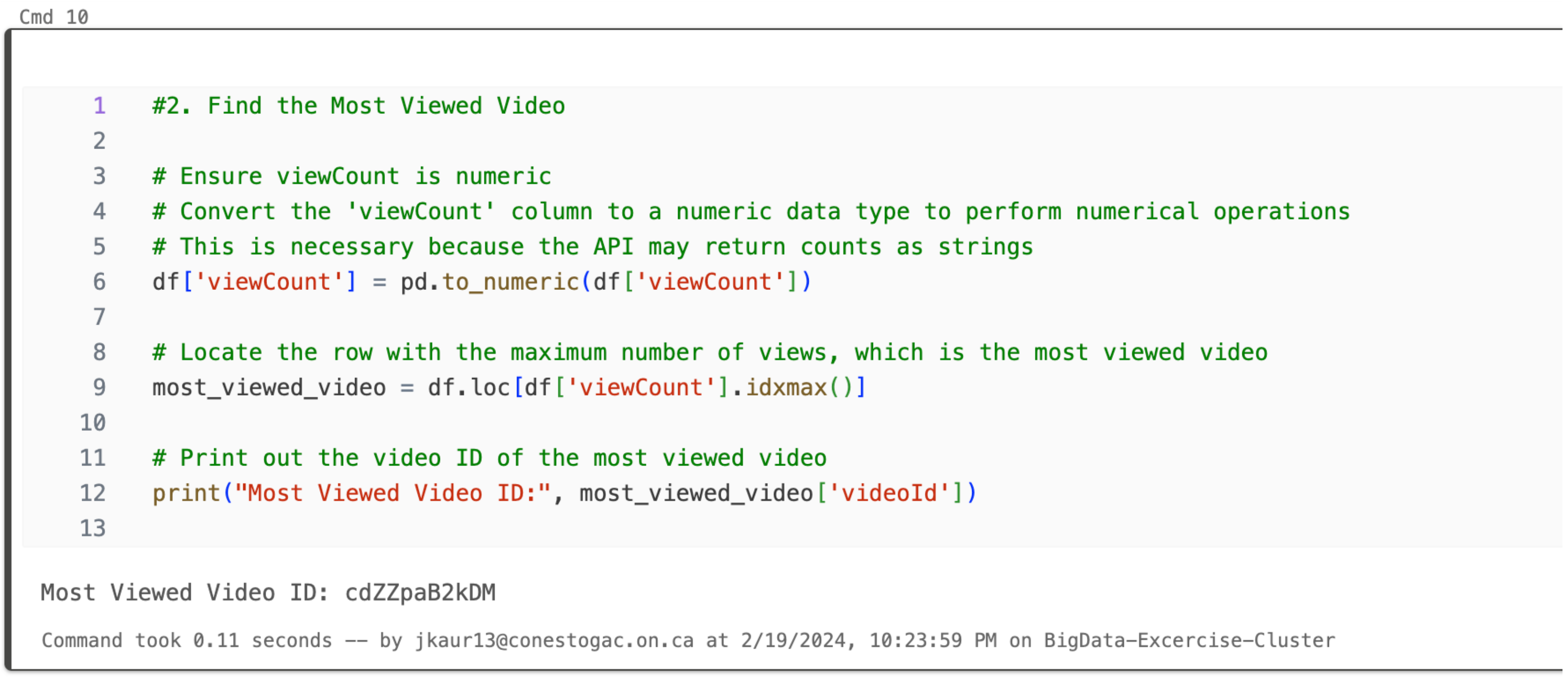
```
Command took 0.12 seconds -- by jkaur13@conestogac.on.ca at 2/19/2024, 10:21:26 PM on BigData-Excercise-Cluster
```

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 4: Basic Data Exploration and Analysis

(Code with the description shared in separate file)

2) Find the Most Viewed Video: video with the highest number of views



```
Cmd 10

#2. Find the Most Viewed Video

# Ensure viewCount is numeric
# Convert the 'viewCount' column to a numeric data type to perform numerical operations
# This is necessary because the API may return counts as strings
df['viewCount'] = pd.to_numeric(df['viewCount'])

# Locate the row with the maximum number of views, which is the most viewed video
most_viewed_video = df.loc[df['viewCount'].idxmax()]

# Print out the video ID of the most viewed video
print("Most Viewed Video ID:", most_viewed_video['videoId'])

Most Viewed Video ID: cdZZpaB2kDM

Command took 0.11 seconds -- by jkaur13@conestogac.on.ca at 2/19/2024, 10:23:59 PM on BigData-Excercise-Cluster
```

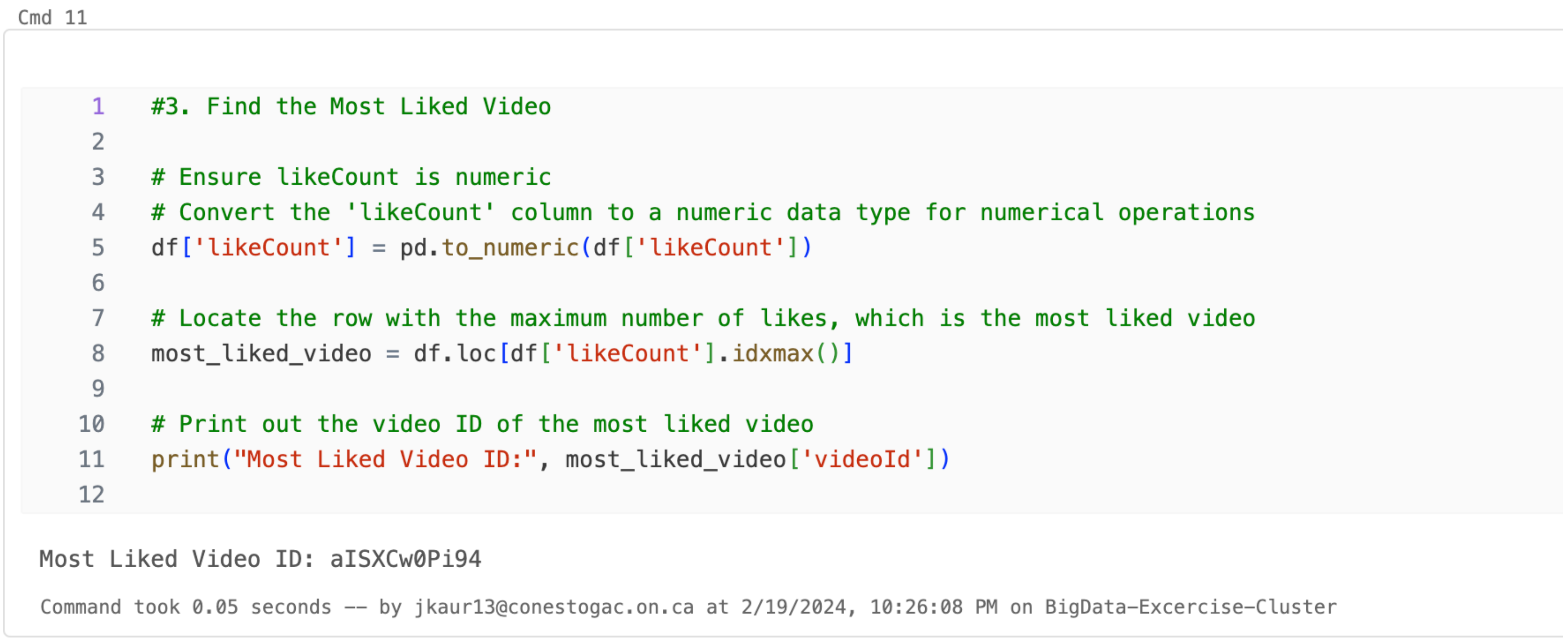
In Class Hands-On Activity: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 4: Basic Data Exploration and Analysis

(Code with the description shared in separate file)

3) Find the Most Liked Video: video with the highest number of likes



Cmd 11

```python
#3. Find the Most Liked Video

# Ensure likeCount is numeric
# Convert the 'likeCount' column to a numeric data type for numerical operations
df['likeCount'] = pd.to_numeric(df['likeCount'])

# Locate the row with the maximum number of likes, which is the most liked video
most_liked_video = df.loc[df['likeCount'].idxmax()]

# Print out the video ID of the most liked video
print("Most Liked Video ID:", most_liked_video['videoId'])

```

```
Most Liked Video ID: aISXCw0Pi94
```

Command took 0.05 seconds -- by jkaur13@conestogac.on.ca at 2/19/2024, 10:26:08 PM on BigData-Excercise-Cluster

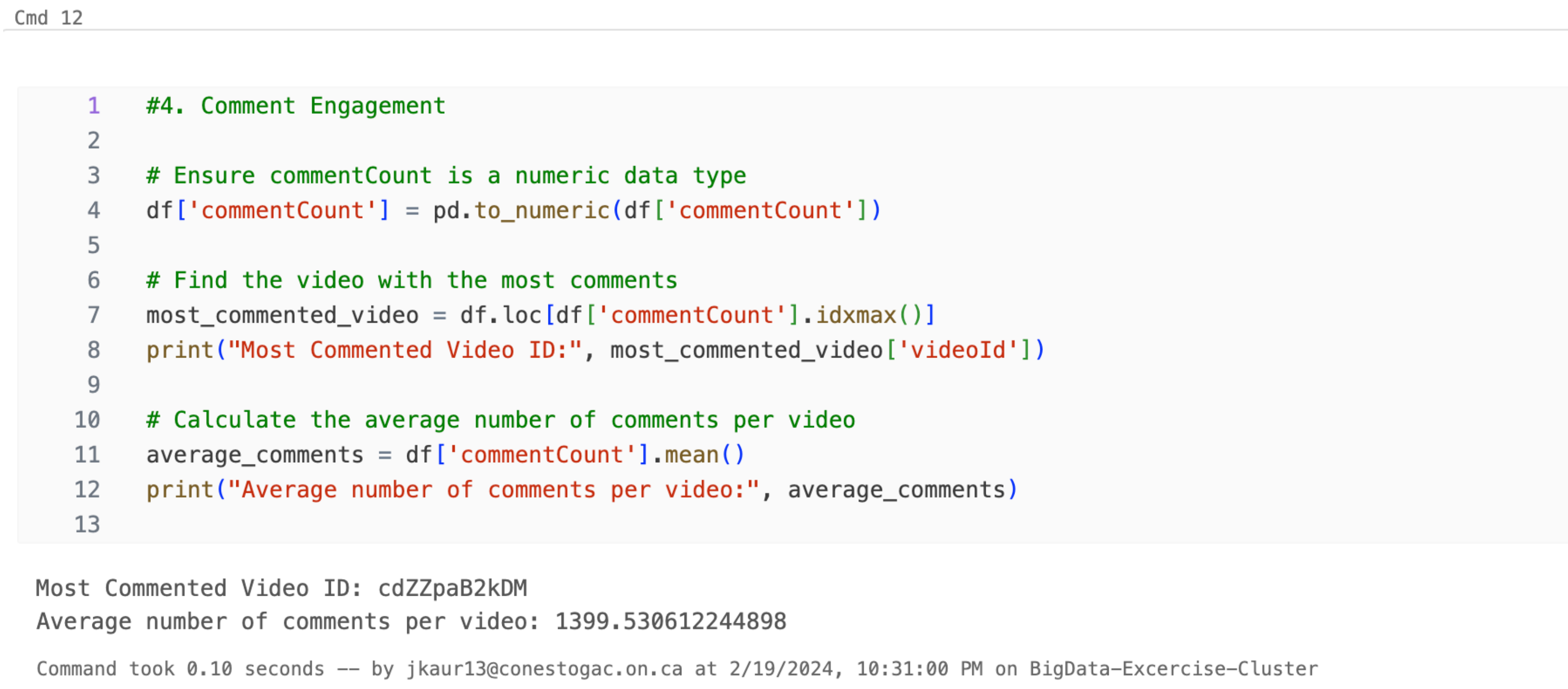
In Class Hands-On Activity: Big Data Extraction, Ingestion, Exploration and Simple Analysis using Microsoft Azure

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 4: Basic Data Exploration and Analysis

(Code with the description shared in separate file)

4) Comment Engagement: Evaluate the engagement of viewers with the videos based on the comment counts



Cmd 12

```python
#4. Comment Engagement

# Ensure commentCount is a numeric data type
df['commentCount'] = pd.to_numeric(df['commentCount'])

# Find the video with the most comments
most_commented_video = df.loc[df['commentCount'].idxmax()]
print("Most Commented Video ID:", most_commented_video['videoId'])

# Calculate the average number of comments per video
average_comments = df['commentCount'].mean()
print("Average number of comments per video:", average_comments)

```

```
Most Commented Video ID: cdZZpaB2kDM
Average number of comments per video: 1399.530612244898
```

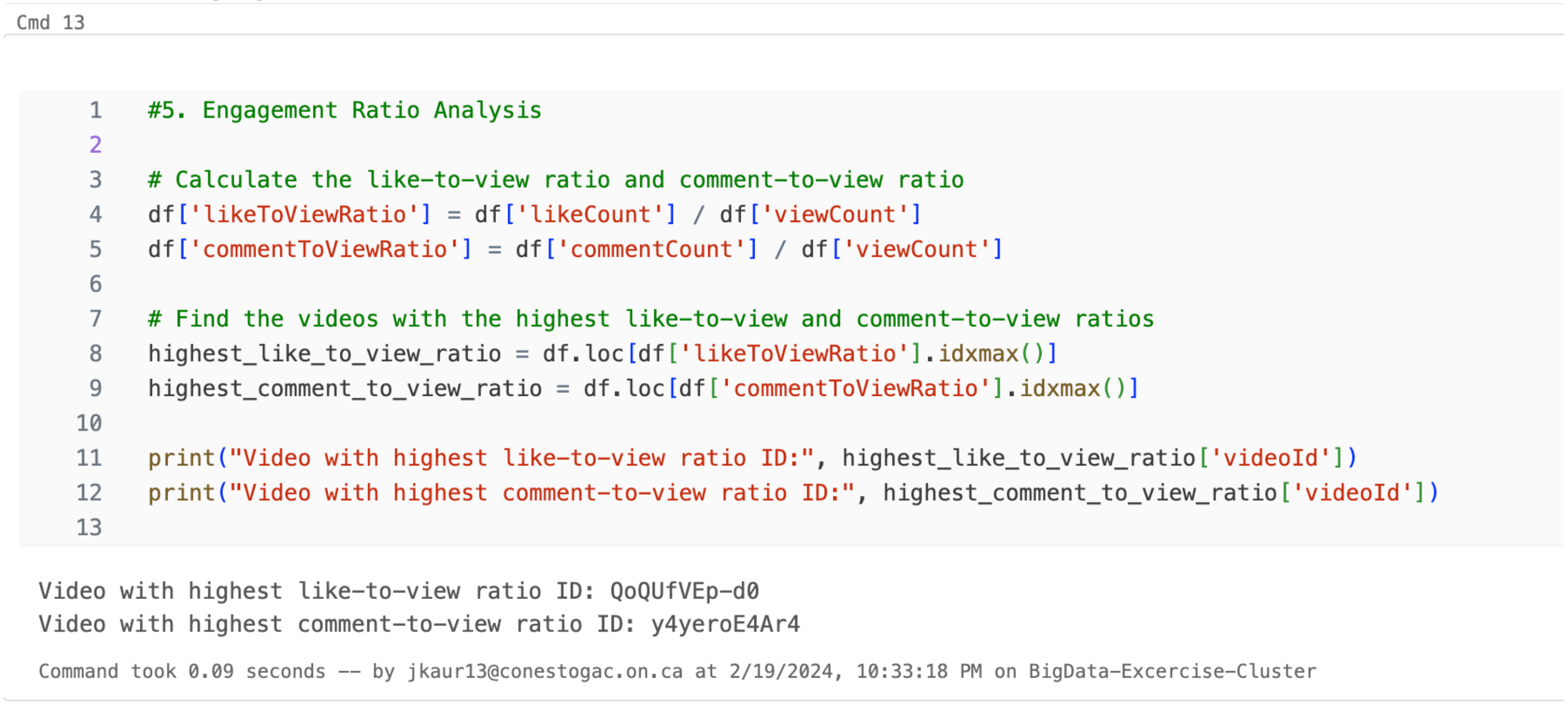
```
Command took 0.10 seconds -- by jkaur13@conestogac.on.ca at 2/19/2024, 10:31:00 PM on BigData-Excercise-Cluster
```

Exercise 1: Analyzing YouTube Video Statistics using YouTube API (Cont.)

Step 4: Basic Data Exploration and Analysis

(Code with the description shared in separate file)

5) Engagement Ratio Analysis: Calculate the ratio of likes to views and comments to views to understand viewer engagement



Cmd 13

```python
#5. Engagement Ratio Analysis

# Calculate the like-to-view ratio and comment-to-view ratio
df['likeToViewRatio'] = df['likeCount'] / df['viewCount']
df['commentToViewRatio'] = df['commentCount'] / df['viewCount']

# Find the videos with the highest like-to-view and comment-to-view ratios
highest_like_to_view_ratio = df.loc[df['likeToViewRatio'].idxmax()]
highest_comment_to_view_ratio = df.loc[df['commentToViewRatio'].idxmax()]

print("Video with highest like-to-view ratio ID:", highest_like_to_view_ratio['videoId'])
print("Video with highest comment-to-view ratio ID:", highest_comment_to_view_ratio['videoId'])

```

```
Video with highest like-to-view ratio ID: QoQUfVEp-d0
Video with highest comment-to-view ratio ID: y4yeroE4Ar4

Command took 0.09 seconds -- by jkaur13@conestogac.on.ca at 2/19/2024, 10:33:18 PM on BigData-Excercise-Cluster
```