DataPrep - The easiest way to prepare data in Python

Jiannan Wang

Simon Fraser University

Jan 6, 2021, Databricks
Talk Outline

1. DataPrep Overview
2. Dive into DataPrep
   • DataPrep.EDA
   • DataPrep.Connector
3. Future Direction
Data Preparation Is Still the Bottleneck!!!

2014

The New York Times

For Big-Data Scientists, ‘Janitor Work’ Is Key Hurdle to Insights

Yet far too much handcrafted work — what data scientists call “data wrangling,” “data munging” and “data janitor work” — is still required. Data scientists, according to interviews and expert estimates, spend from 50 percent to 80 percent of their time mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets.


2020

The State of Data Science 2020

Moving from hype toward maturity

We were disappointed, if not surprised, to see that data wrangling still takes the lion’s share of time in a typical data professional’s day. Our respondents reported that almost half of their time is spent on the combined tasks of data loading and cleansing. Data

Why Is Data Preparation Hard?

1. Too many small problems (e.g., standardize date, dedup address, etc)
2. Humans have different levels of expertise (in data science and programming)
3. Domain specific (finance, social science, healthcare, economics, etc.)
Human-in-the-loop Data Preparation

Three Directions

• Spreadsheet GUI
• Workflow GUI
• Notebook GUI
### Spreadsheet GUI

#### Preview

<table>
<thead>
<tr>
<th>IMSI</th>
<th>CONTRACT_END</th>
<th>CONTRACT_START</th>
<th>SUBSCRIBER_AGE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3101792268112721</td>
<td>6/4/16</td>
<td>7/23/09</td>
<td>1</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>3101669956766700</td>
<td>2/22/15</td>
<td>10/6/13</td>
<td>7</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>31017948622541</td>
<td>9/23/16</td>
<td>1/9/07</td>
<td>13</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>31080532849238</td>
<td>5/29/15</td>
<td>2/14/01</td>
<td>4</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310026939721985</td>
<td>9/11/15</td>
<td>9/18/10</td>
<td>8</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>310269815466952</td>
<td>8/27/15</td>
<td>3/13/06</td>
<td>16</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>31071948724861</td>
<td>1/16/16</td>
<td>5/11/04</td>
<td>4</td>
<td>INACTIVE</td>
</tr>
<tr>
<td>310719766648471</td>
<td>8/11/06</td>
<td>9/11/06</td>
<td>13</td>
<td>ACTIVE</td>
</tr>
</tbody>
</table>

#### Hide Example Values

- 12.65k
- 5.37k

#### Trifacta

- 14-Nov-2012
- 11-Jul-2007
- 03-Jul-2013
Workflow GUI
In [1]:

```python
import qgrid
import numpy as np
import pandas as pd

df = pd.DataFrame(np.random.rand(3,5))
qu = qgrid.show_grid(df)
qu
```

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.09835</td>
<td>0.65988</td>
<td>0.24213</td>
<td>0.1324</td>
<td>0.21811</td>
</tr>
<tr>
<td>1</td>
<td>0.11348</td>
<td>0.17392</td>
<td>0.18452</td>
<td>0.25759</td>
<td>0.45406</td>
</tr>
<tr>
<td>2</td>
<td>0.77328</td>
<td>0.2921</td>
<td>0.37934</td>
<td>0.87462</td>
<td>0.93311</td>
</tr>
</tbody>
</table>
Which Direction To Go?

“Data Prep Market was valued at USD 3.29 Billion in 2019 and is projected to reach USD 18.11 Billion by 2027, growing at a CAGR of 25.64% from 2020 to 2027.”

Source: https://www.verifiedmarketresearch.com/product/data-prep-market/

Three Directions

• Spreadsheet GUI
  Targeted at non-programmers

• Workflow GUI

• Notebook GUI
  Targeted at data scientists
Our Vision

Machine Learning Made Easy

Deep Learning Made Easy

Big Data Made Easy

Visualization Made Easy

Data Preparation Made Easy
# DataPrep Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Dates</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataPrep.EDA</td>
<td>May 2019 - Now</td>
<td>Simplify Exploratory Data Analysis</td>
</tr>
<tr>
<td>DataPrep.Connector</td>
<td>Nov 2019 - Now</td>
<td>Simplify Web Data Collection</td>
</tr>
<tr>
<td>DataPrep.Clean</td>
<td>Sept 2020 - Now</td>
<td>Simplify Data Cleaning</td>
</tr>
<tr>
<td>DataPrep.Feature</td>
<td>Planning</td>
<td>Simplify Feature Engineering</td>
</tr>
<tr>
<td>DataPrep.Integrate</td>
<td>Planning</td>
<td>Simplify Data Integration</td>
</tr>
</tbody>
</table>
Understand your data with a few lines of code in seconds using DataPrep.eda

I'll look into it and get back to you. By the way what you guys are doing is amazing and have the potential to be a game-changer if it cut some time out of data prep.

This will save me so much time even just exploring my data, not to mention coding all of it up. You've done good in the world.

Would love a pyspark/koalas module

https://www.reddit.com/r/Python/comments/hlqnim/understand_your_data_with_a_few_lines_of_code_in/
Talk Outline

1. DataPrep Overview

2. Dive into DataPrep
   - DataPrep.EDA
   - DataPrep.Connector

3. Future Direction
DataPrep.EDA

Task-Centric Exploratory Data Analysis
Exploratory Data Analysis (EDA)

Understand data and discover insights via data visualization, data summarization, etc.

Understand “Age” column

<table>
<thead>
<tr>
<th>Minimum</th>
<th>0.42</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-th Percentile</td>
<td>4</td>
</tr>
<tr>
<td>Q1</td>
<td>20.125</td>
</tr>
<tr>
<td>Median</td>
<td>28</td>
</tr>
<tr>
<td>Q3</td>
<td>38</td>
</tr>
<tr>
<td>95-th Percentile</td>
<td>56</td>
</tr>
<tr>
<td>Maximum</td>
<td>80</td>
</tr>
<tr>
<td>Range</td>
<td>79.58</td>
</tr>
<tr>
<td>IQR</td>
<td>17.875</td>
</tr>
</tbody>
</table>
Current EDA Solutions in Python

Solution 1: Pandas + Matplotlib

😊 Hard to Use

- **Beginner**: Need to know how to write plotting code
- **Expert**: Need to write lengthy and repetitive code

Understand “Age” column

Write Code

<table>
<thead>
<tr>
<th>Minimum</th>
<th>0.42</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-th Percentile</td>
<td>4</td>
</tr>
<tr>
<td>Q1</td>
<td>20.125</td>
</tr>
<tr>
<td>Median</td>
<td>26</td>
</tr>
<tr>
<td>Q3</td>
<td>38</td>
</tr>
<tr>
<td>95-th Percentile</td>
<td>56</td>
</tr>
<tr>
<td>Maximum</td>
<td>80</td>
</tr>
<tr>
<td>Range</td>
<td>79.58</td>
</tr>
<tr>
<td>IQR</td>
<td>17.875</td>
</tr>
</tbody>
</table>
Current EDA Solutions in Python

Solution 2: Pandas-profiling

🤔 Slow
🤔 Hard to Customize

```python
profile = ProfileReport(df, title="Pandas Profiling Report")
```
# DataPrep.EDA Design Goals

<table>
<thead>
<tr>
<th>EDA Solutions</th>
<th>Easy to Use</th>
<th>Interactive Speed</th>
<th>Easy to Customize</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Pandas + Matplotlib</td>
<td>😞</td>
<td>😊</td>
<td>😊</td>
</tr>
<tr>
<td>2. Pandas-profiling</td>
<td>😊</td>
<td>😞</td>
<td>😞</td>
</tr>
<tr>
<td>3. DataPrep.EDA</td>
<td>😊</td>
<td>😊</td>
<td>😊</td>
</tr>
</tbody>
</table>
Key Idea

Task-Centric API Design

- Declarative
- Support both coarse-grained and fine-grained EDA tasks

Example

- `plot(df)`: “I want to see an overview of the dataset”
- `plot_missing(df)`: “I want to understand the missing values of the dataset”
- `plot(df, x)`: “I want to understand the column x”
- `plot(df, x, y)`: “I want to understand the relationship between x and y”
- …
DataPrep.EDA Demo

Last Checkpoint: a minute ago  (unsaved changes)

File  Edit  View  Insert  Cell  Kernel  Widgets  Help

In [2]: from dataprep.eda import plot, plot_missing, plot_correlation, create_report

In [ ]: import pandas as pd

In [ ]: df = pd.read_csv("titanic.csv")

I want an overview of the dataset

In [ ]: plot(df)

Understand Missing Value

In [ ]: plot_missing(df)

Understand Correlation

In [ ]: plot_correlation(df)

Understand Numerical Column

In [ ]: plot(df, "Age")

Understand Text Column

In [ ]: plot(df, "Name")

Understand Column Relationship
Under the Hood

Data Processing Pipeline

Mapping Rules

plot(df, "price", bins = 50)
# Mapping Rules

\[ N = \text{Numerical}, \ C = \text{Categorical} \]

<table>
<thead>
<tr>
<th>Task-Centric API Design</th>
<th>Corresponding Stats/Plots</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot(df)</td>
<td>Dataset statistics, histogram or bar chart for each column</td>
</tr>
</tbody>
</table>
| plot(df, col₁)          | (1) \(\text{col₁} = N \rightarrow\) Column statistics, histogram, kde plot, qq-normal plot, box plot  
                          (2) \(\text{col₁} = C \rightarrow\) Column statistics, bar chart, pie chart, word cloud, word frequencies |
| plot(df, col₁, col₂)    |                                                                         |
| plot_correlation(df)    |                                                                         |
| plot_correlation(df, col₁) |                                                                         |
| plot_correlation(df, col₁, col₂) |                                                              |
| plot_missing(df)        |                                                                         |
| plot_missing(df, col₁)  |                                                                         |
| plot_missing(df, col₁, col₂) | Histogram, pdf, cdf, and box plot that show the impact of the missing values from \(\text{col₁}\) on \(\text{col₂}\) |

[1] [https://www.data-to-viz.com/](https://www.data-to-viz.com/)  
[2] Exploratory data analysis with R  

...
Data Processing Pipeline

1. **Config Manager**
   - Config
     ```json
     {
     hist.bins: 50,
     kde.bins: 50,
     hist.width: 90,
     ...
     }
     ...
     ```

2. **Compute Module**
   - Intermediates
     ```json
     {  missing_count: 0,
        hist.counts: [2, 10, ...],
        hist.endpoints: [334, 421, ...],
        ...
     }
     ```

3. **Render Module**

Plot:
```python
plot(df, "price", bins = 50)
```
Interactive Speed

Ubuntu 16.04 Linux server with 64 GB memory and 8 Intel E7-4830 cores

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>#Rows</th>
<th>#Cols (N/C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>heart</td>
<td>11KB</td>
<td>303</td>
<td>14 (14/0)</td>
</tr>
<tr>
<td>diabetes</td>
<td>23KB</td>
<td>768</td>
<td>9 (9/0)</td>
</tr>
<tr>
<td>automobile</td>
<td>26KB</td>
<td>205</td>
<td>26 (10/16)</td>
</tr>
<tr>
<td>titanic</td>
<td>64KB</td>
<td>891</td>
<td>12 (7/5)</td>
</tr>
<tr>
<td>women</td>
<td>500KB</td>
<td>8553</td>
<td>10 (5/5)</td>
</tr>
<tr>
<td>credit</td>
<td>2.7MB</td>
<td>30K</td>
<td>25 (25/0)</td>
</tr>
<tr>
<td>solar</td>
<td>2.8MB</td>
<td>33K</td>
<td>11 (7/4)</td>
</tr>
<tr>
<td>suicide</td>
<td>2.8MB</td>
<td>28K</td>
<td>12 (6/6)</td>
</tr>
<tr>
<td>diamonds</td>
<td>3MB</td>
<td>54K</td>
<td>11 (8/3)</td>
</tr>
<tr>
<td>chess</td>
<td>7.3MB</td>
<td>20K</td>
<td>16 (6/10)</td>
</tr>
<tr>
<td>adult</td>
<td>5.7MB</td>
<td>49K</td>
<td>15 (6/9)</td>
</tr>
<tr>
<td>basketball</td>
<td>9.2MB</td>
<td>53K</td>
<td>31 (21/10)</td>
</tr>
<tr>
<td>conflicts</td>
<td>13MB</td>
<td>34K</td>
<td>25 (10/15)</td>
</tr>
<tr>
<td>rain</td>
<td>13.5MB</td>
<td>142K</td>
<td>24 (17/7)</td>
</tr>
<tr>
<td>hotel</td>
<td>16MB</td>
<td>119K</td>
<td>32 (20/12)</td>
</tr>
</tbody>
</table>
## Efficiency Comparison

### DataPrep.EDA vs Pandas-Profiling

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>#Rows</th>
<th>#Cols (N/C)</th>
<th>PP</th>
<th>EDA&lt;sup&gt;x&lt;/sup&gt;</th>
<th>Faster</th>
</tr>
</thead>
<tbody>
<tr>
<td>heart</td>
<td>11KB</td>
<td>303</td>
<td>14 (14/0)</td>
<td>17.7s</td>
<td>2.0s</td>
<td>8.6×</td>
</tr>
<tr>
<td>diabetes</td>
<td>23KB</td>
<td>768</td>
<td>9 (9/0)</td>
<td>28.3s</td>
<td>1.6s</td>
<td>17.7×</td>
</tr>
<tr>
<td>automobile</td>
<td>26KB</td>
<td>205</td>
<td>26 (10/16)</td>
<td>38.2s</td>
<td>3.9s</td>
<td>9.8×</td>
</tr>
<tr>
<td>titanic</td>
<td>64KB</td>
<td>891</td>
<td>12 (7/5)</td>
<td>17.8s</td>
<td>2.1s</td>
<td>8.5×</td>
</tr>
<tr>
<td>women</td>
<td>500KB</td>
<td>8553</td>
<td>10 (5/5)</td>
<td>19.8s</td>
<td>2.3s</td>
<td>8.6×</td>
</tr>
<tr>
<td>credit</td>
<td>2.7MB</td>
<td>30K</td>
<td>25 (25/0)</td>
<td>127.0s</td>
<td>6.1s</td>
<td><strong>20.8×</strong></td>
</tr>
<tr>
<td>solar</td>
<td>2.8MB</td>
<td>33K</td>
<td>11 (7/4)</td>
<td>25.1s</td>
<td>2.7s</td>
<td>9.3×</td>
</tr>
<tr>
<td>suicide</td>
<td>2.8MB</td>
<td>28K</td>
<td>12 (6/6)</td>
<td>20.6s</td>
<td>2.8s</td>
<td>7.4×</td>
</tr>
<tr>
<td>diamonds</td>
<td>3MB</td>
<td>54K</td>
<td>11 (8/3)</td>
<td>28.2s</td>
<td>3.1s</td>
<td>9×</td>
</tr>
<tr>
<td>chess</td>
<td>7.3MB</td>
<td>20K</td>
<td>16 (6/10)</td>
<td>23.6s</td>
<td>4.3s</td>
<td>5.5×</td>
</tr>
<tr>
<td>adult</td>
<td>5.7MB</td>
<td>49K</td>
<td>15 (6/9)</td>
<td>23.2s</td>
<td>4.0s</td>
<td>5.8×</td>
</tr>
<tr>
<td>basketball</td>
<td>9.2MB</td>
<td>53K</td>
<td>31 (21/10)</td>
<td>126.2s</td>
<td>9.9s</td>
<td>12.7×</td>
</tr>
<tr>
<td>conflicts</td>
<td>13MB</td>
<td>34K</td>
<td>25 (10/15)</td>
<td>34.9s</td>
<td>8.6s</td>
<td>4×</td>
</tr>
<tr>
<td>rain</td>
<td>13.5MB</td>
<td>142K</td>
<td>24 (17/7)</td>
<td>100.1s</td>
<td>11.6s</td>
<td>8.6×</td>
</tr>
<tr>
<td>hotel</td>
<td>16MB</td>
<td>119K</td>
<td>32 (20/12)</td>
<td>83.2s</td>
<td>13s</td>
<td>6.4×</td>
</tr>
</tbody>
</table>
# Easy to Customize

(Available Soon)

## How to Guide

<table>
<thead>
<tr>
<th>In [*]:</th>
<th>1 plot(df, &quot;age&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
<tr>
<td>In [ ]:</td>
<td>1</td>
</tr>
</tbody>
</table>
DataPrep.EDA Takeaways

Innovation

The **first** task-centric EDA system in Python

Achieve three design goals

- Easy to use
- Interactive speed
- Easy to customize
DataPrep.Connector

A Unified API Wrapper to Simplify Web Data Collection
Data Collection Through Restful APIs

- Social Data
- Business Data
- Event Data
- Publication Data

[public-apis](https://public-apis.io)
A collective list of free APIs for use in software and web development.

[ultimatecourses.com](https://ultimatecourses.com)
Stars: 98.4k, Forks: 12k

Index
- Animals
- Anime
- Anti-Malware
- Art & Design
- Books
- Business
- Calendar
- Cloud Storage & File Sharing
- Continuous Integration
- Cryptocurrency
- Currency Exchange
- Data Validation
- Development
- Dictionaries
- Documents & Productivity
- Environment
- Events
- Finance
Restful API Example

Request
GET https://api.yelp.com/v3/businesses/search

Parameters
These parameters should be in the query string.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>term</td>
<td>string</td>
<td>Optional. Search term, for example &quot;food&quot; or &quot;restaurant&quot;. Default to searching across businesses with these names, such as &quot;Starbucks&quot;. If term is not provided, the default is to searching across businesses with &quot;food&quot; or &quot;restaurant&quot; names.</td>
</tr>
<tr>
<td>location</td>
<td>string</td>
<td>Required if either latitude or longitude is not provided. Geographic area to be used when searching for businesses within the specified location. The search may be not strictly within the specified location.</td>
</tr>
<tr>
<td>latitude</td>
<td>decimal</td>
<td>Required if location is not provided. Latitude of the location.</td>
</tr>
<tr>
<td>longitude</td>
<td>decimal</td>
<td>Required if location is not provided. Longitude of the location.</td>
</tr>
<tr>
<td>radius</td>
<td>int</td>
<td>Optional. A suggested search radius in meters. The actual search radius may be less in dense urban areas, and higher in regions of less population density. If this value is too large, a &quot;AREA_TOO_LARGE&quot; error may occur. The default value is 40000 meters (about 25 miles).</td>
</tr>
</tbody>
</table>

Response Body
```json
{
  "total": 8228,
  "businesses": [
    {
      "rating": 4,
      "price": "$",
      "phone": "+14152520800",
      "id": "EBR3kJ3dcmgyoPMyJ0_0lg",
      "alias": "four-barrel-coffee-san-francisco",
      "is_closed": false,
      "categories": [
        {
          "alias": "coffee",
          "title": "Coffee & Tea"
        }
      ],
      "review_count": 1738,
      "name": "Four Barrel Coffee",
      "url": "https://www.yelp.com/biz/four-barrel-coffee-san-francisco",
      "coordinates": {
        "latitude": 37.7670169511878,
        "longitude": -122.42184275
      },
      "image_url": "http://s3-media2.fl.yelpcdn.com/bphoto/MgtASP3l_t4tPCL1iAsCg/"
    },
    {
      "city": "San Francisco",
      "country": "US",
      "address2": "",
      "address3": "",
      "state": "CA",
      "address1": "375 Valencia St",
      "zip_code": "94103"
    },
    {
      "distance": 1604.23,
      "transactions": "["pickup", "delivery"]"
    }
  ]
}
```
Restful API Wrapper

Wrap API calls into Easy-to-Use Python Functions

- **bear / python-twitter**
  A Python wrapper around the Twitter API.
  - View license
  - ★ 3k stars ★ 922 forks

- **Yelp / yelp-fusion**
  Yelp Fusion API
  - [yelp.com/developers](http://yelp.com/developers)
  - MIT License
  - ★ 331 stars ★ 305 forks

- **scholarly / dblp-python**
  A simple Python wrapper around search and author and publication.
  - ★ 61 stars ★ 34 forks

- **plamere / spotipy**
  A light weight Python library for the Spotify Web API
  - [spotipy.readthedocs.org/](http://spotipy.readthedocs.org/)
  - MIT License
  - ★ 2.6k stars ★ 568 forks

- **srcece / python-youtube-api**
  A basic Python YouTube v3 API to fetch data from YouTube Key without OAuth
  - GPL-3.0 License
  - ★ 73 stars ★ 35 forks
Build a New API Wrapper is **Tedious!**

<table>
<thead>
<tr>
<th>HTTP Connection</th>
<th>Connect to the website server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Handle authorization schemes</td>
</tr>
<tr>
<td>Pagination</td>
<td>Request data from multiple pages</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Retrieve data in parallel with less time</td>
</tr>
<tr>
<td>Result Parsing</td>
<td>Convert Json string to Pandas Dataframe</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
</tbody>
</table>
If we don’t unify API wrappers, then ...
If we don’t unify API wrappers, then ...

- **Bad for developers** (repetitive building efforts)
- **Bad for users** (burden to learn many API wrappers)
DataPrep.Connector
A Unified API Wrapper

Reusable Components

HTTP Connection
Authorization
Pagination
Concurrency
Result Parsing

.....

Configuration Files

Yelp Config File
Spotify Config File
Youtube Config File
Twitter Config File
DBLP Config File
Facebook Config File
Reddit Config File

.....

Good for developers (No repetitive building efforts)
The Unified API

1. Connect

```python
conn = connect(website_name, _auth, _concurrency)
```

2. Understand (Optional)

```python
conn.table_names
conn.show_schema(table_name)
```

3. Query

```python
conn.query(table_name, query_parameter_list, _count)
```

Good for users (No burden to learn many API wrappers)
A Unified API Design

DBLP

In [ ]: conn_dblp = Connector("dblp")

In [ ]: conn_dblp.table_names

In [ ]: conn_dblp.show_schema("publication")

In [ ]: df = await conn_dblp.query("publication", q = "machine learning")

In [ ]: df.head(5)

Youtube

In [ ]: conn_youtube = Connector('youtube', _auth={"access_token":youtube_auth_token})

In [ ]: conn_youtube.table_names

In [ ]: conn_youtube.show_schema("videos")

In [ ]: df = await conn_youtube.query("videos", q = "data science", part = "snippet", type = "videos")
DataPrep.Connector Takeaways

Innovation
The first unified API Wrapper in Python

Good For Developers
Speed up wrapper development process

Good For Users
Speed up data collection from Web APIs
Talk Outline

1. DataPrep Overview
2. Dive into DataPrep
   - DataPrep.EDA
   - DataPrep.Connector
3. Future Direction
Future Direction

DataPrep.EDA

• Make plots look **attractive**
• Understand **multiple** dataframes (plot_diff, plot_db, ...)

DataPrep.Connector

• Speed up `read_sql()` with arrow and parallel connection

DataPrep.Clean

• **Goal:** Implement 100+ `clean_{type}(df, x)` functions
• **Example:** clean_email, clean_date, clean_phone, clean_country, etc.
• **Application:** Data Validation, Data Standardization, Semantic Type Detection
The easiest way to prepare data in Python

pip install -U dataprep

Thank you!

http://dataprep.ai