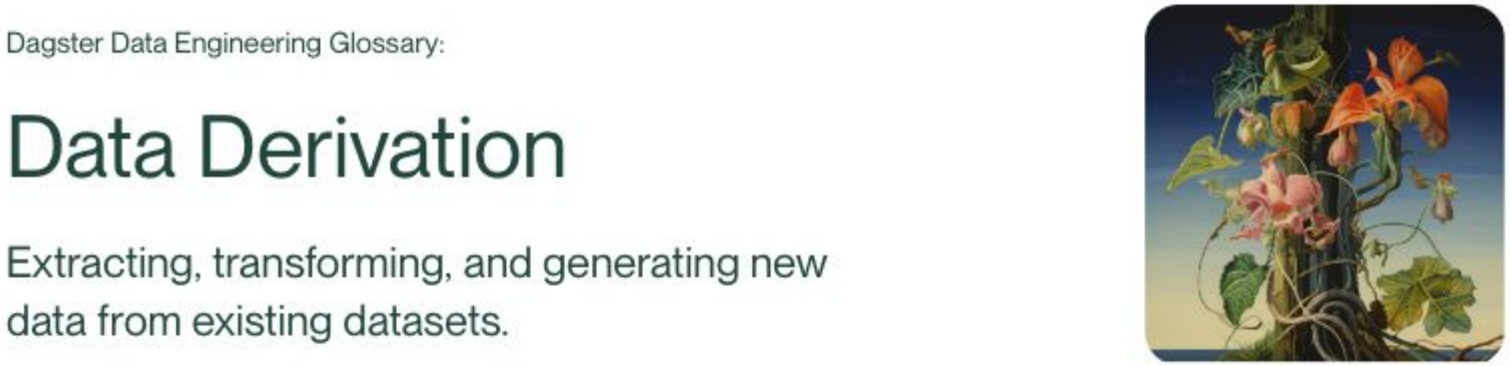
Derived Data Challenges in Single-cell Profiling

Or, “I have profiles but where did the images go?”

Outline

1. Definitions
2. Pipelines and exports
3. Future
4. What if …?

Definitions



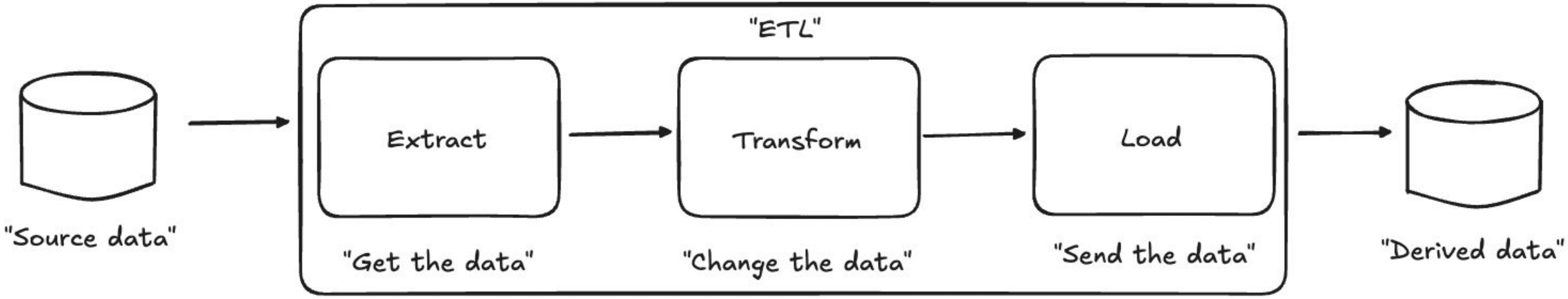
Dagster Data Engineering Glossary:

Data Derivation

Extracting, transforming, and generating new data from existing datasets.

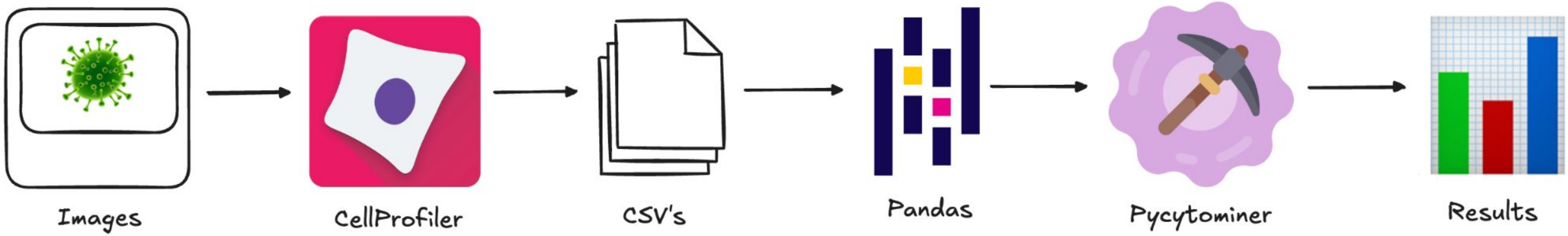
https://dagster.io/glossary/data-derivation

Definitions



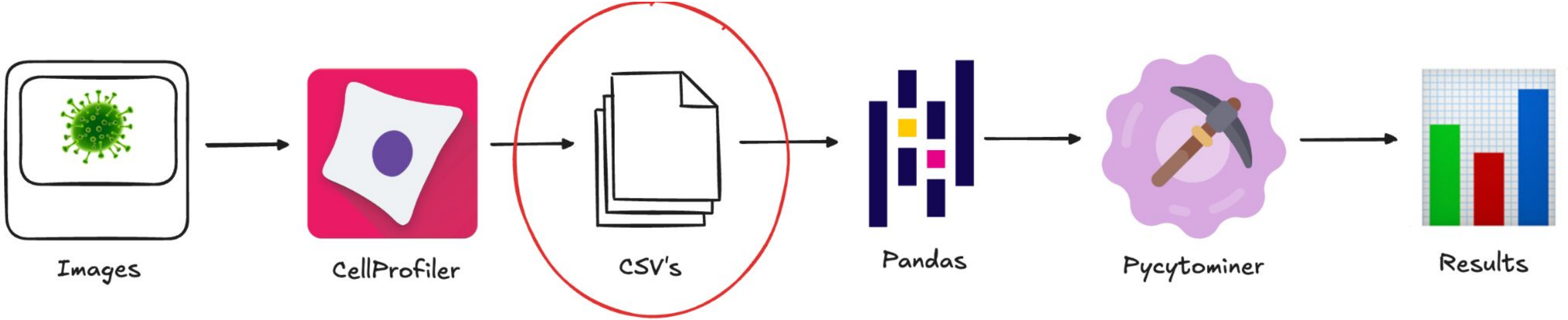
https://en.wikipedia.org/wiki/Extract,_transform,_load

Pipelines and exports



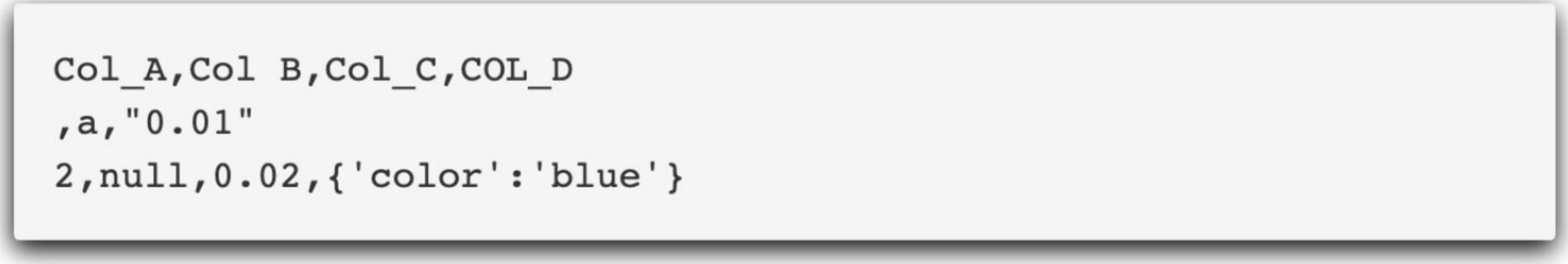
Create single-cell profiles with CellProfiler as CSV’s then read them with Pandas, process with Pcytominer, and create results.

Pipelines and exports



Challenge: CSV’s were not built for large data operations, do not include data types, and are prone to value errors (for example, null types and floating point precision). They also have no relational model.

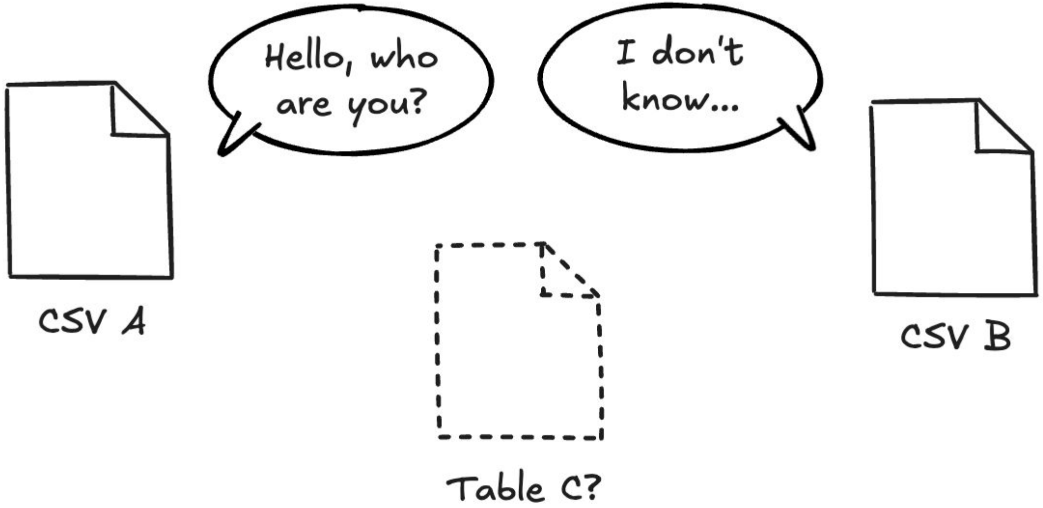
Pipelines and exports



```
Col_A,Col B,Col_C,COL_D
,a,"0.01"
2,null,0.02,{'color':'blue'}
```

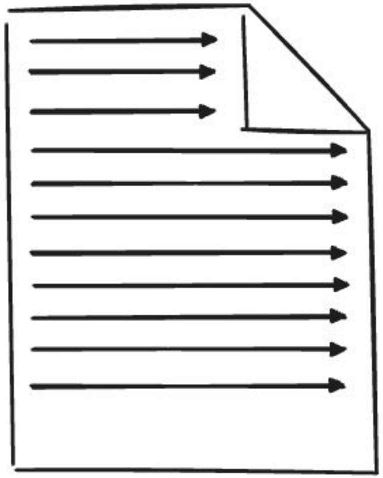
Consider the above example: multiple data types in single columns, multiple null type values, inconsistent naming, and complex data.

Pipelines and exports



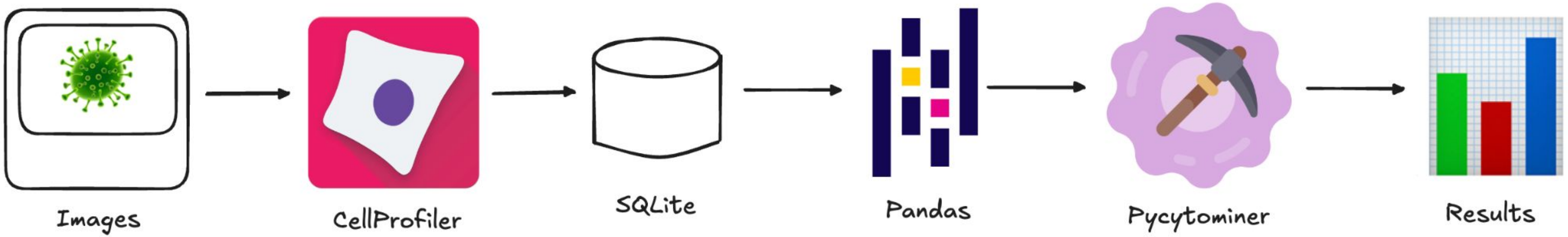
CSV’s have no relational model. They’re all standalone files who don’t know about one another, meaning we have to infer or reinvent for any join.

Pipelines and exports



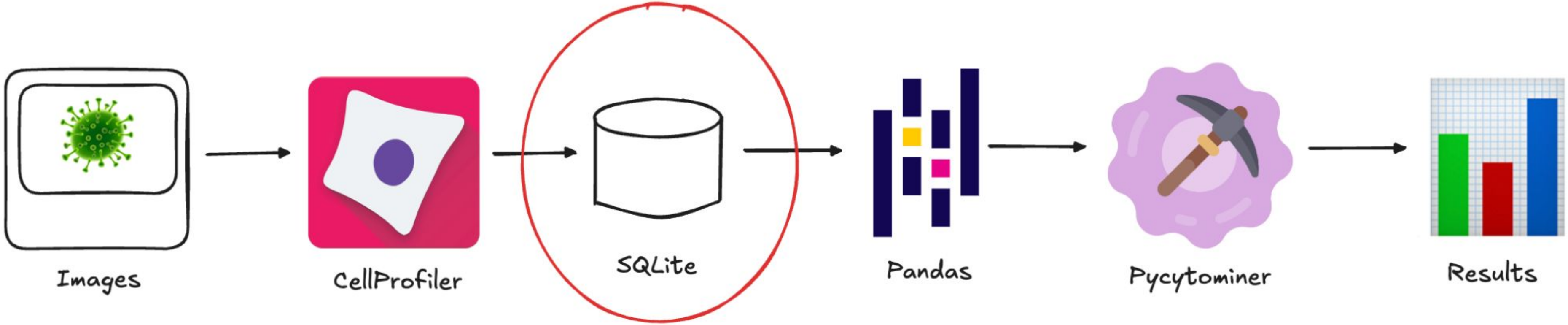
CSV’s must be read sequentially when extracting data (we have to read all columns and each row when gathering things). Data types are implied and converted through the reader. This takes a long time!

Pipelines and exports



We can solve some of these problems with SQLite, an embeddable database. CSV files become **binary tables** that include **affinity types** within a **relational model**.

Pipelines and exports

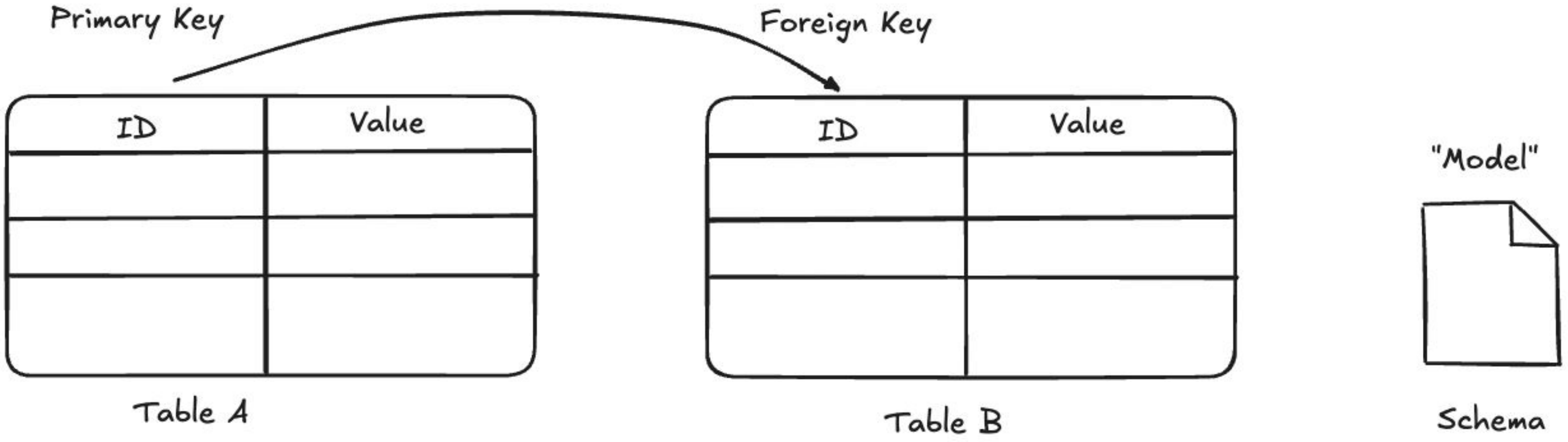


Challenges: affinity types entail implied but not required, the model must also be maintained in order to be useful, and SQLite is not optimized for large data operations.

Pipelines and exports

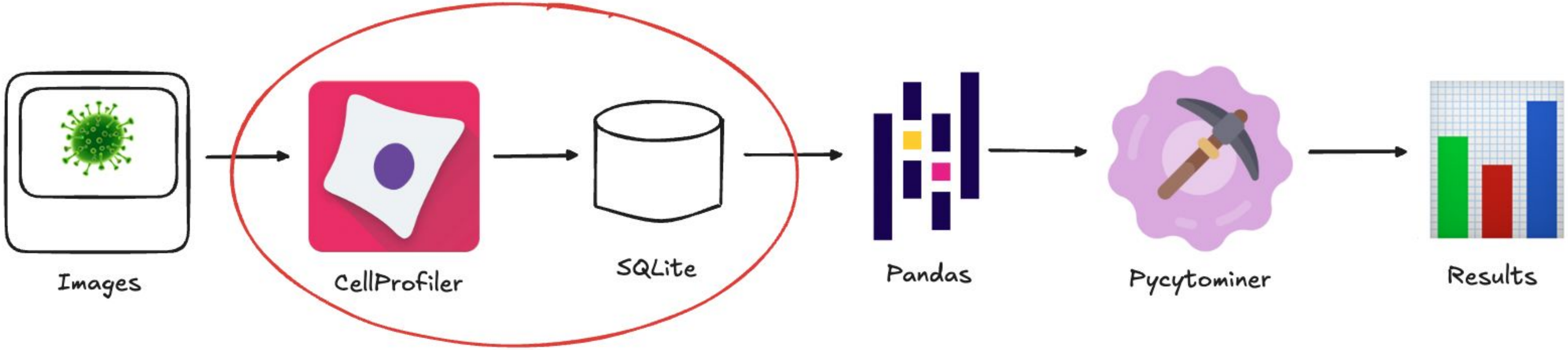
```
-- Affinity types example
-- Create the table
CREATE TABLE research_data (
standard_column INTEGER,
made_up_column FLUXCAPACITOR
);

-- Insert mixed values
INSERT INTO research_data (standard_column, made_up_column)
VALUES (1, 'text_value');
INSERT INTO research_data (standard_column, made_up_column)
VALUES (2, 3.14159);

SELECT * FROM research_data;

>
1|text_value
2|3.14159
```

Pipelines and exports



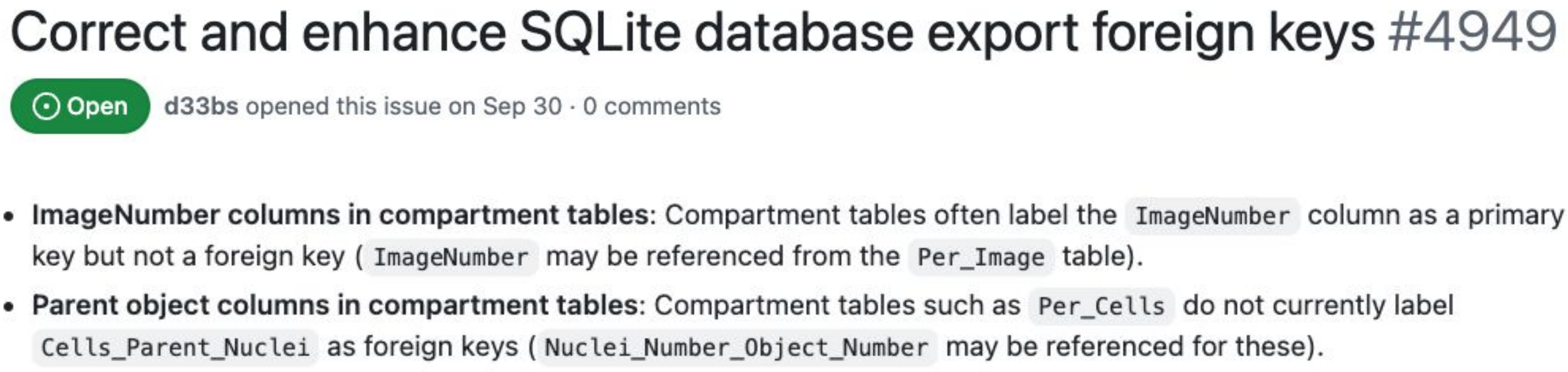
We can store the data relationships as a model within the SQLite schema.

Pipelines and exports



We have to modify the model upstream (it’s defined by CellProfiler exports).

Pipelines and exports



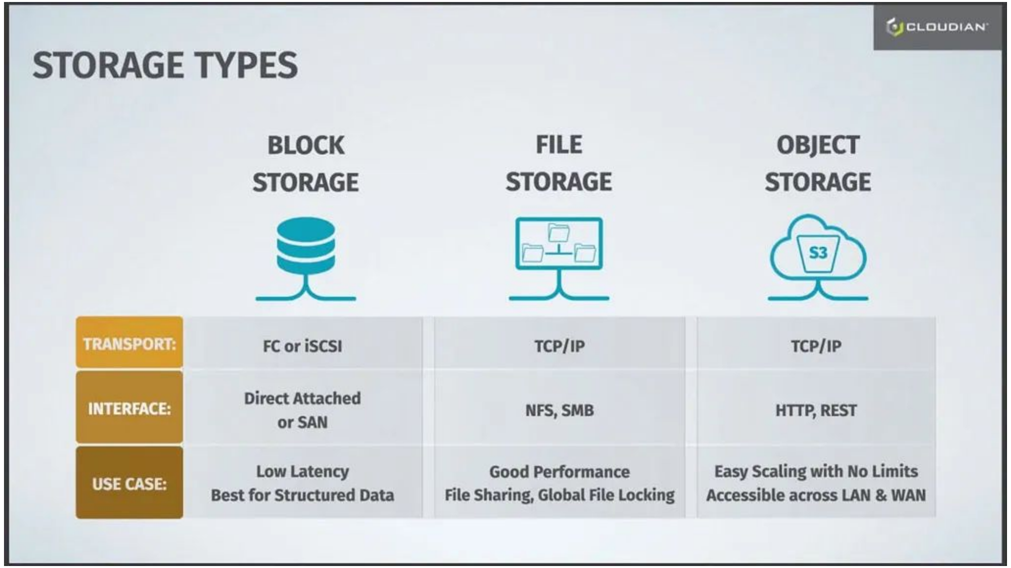
Correct and enhance SQLite database export foreign keys #4949

d33bs opened this issue on Sep 30 · 0 comments

- **ImageNumber columns in compartment tables:** Compartment tables often label the `ImageNumber` column as a primary key but not a foreign key (`ImageNumber` may be referenced from the `Per_Image` table).
- **Parent object columns in compartment tables:** Compartment tables such as `Per_Cells` do not currently label `Cells_Parent_Nuclei` as foreign keys (`Nuclei_Number_Object_Number` may be referenced for these).

https://github.com/CellProfiler/CellProfiler/issues/4949

Pipelines and exports



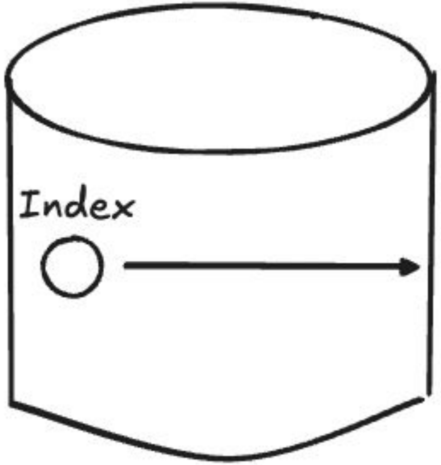
CLOUDIAN

STORAGE TYPES

	BLOCK STORAGE	FILE STORAGE	OBJECT STORAGE
TRANSPORT:	FC or iSCSI	TCP/IP	TCP/IP
INTERFACE:	Direct Attached or SAN	NFS, SMB	HTTP, REST
USE CASE:	Low Latency Best for Structured Data	Good Performance File Sharing, Global File Locking	Easy Scaling with No Limits Accessible across LAN & WAN

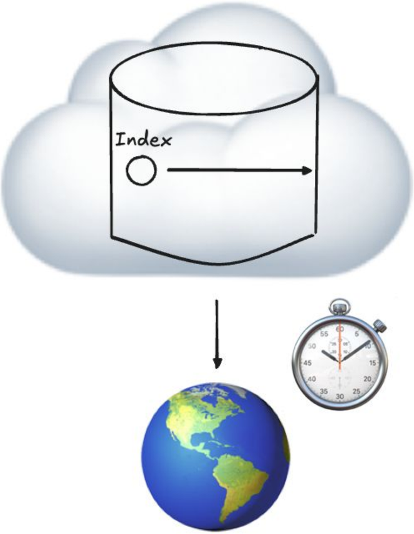
SQLite uses block storage through a filesystem.
https://cloudian.com/blog/object-storage-care/

Pipelines and exports



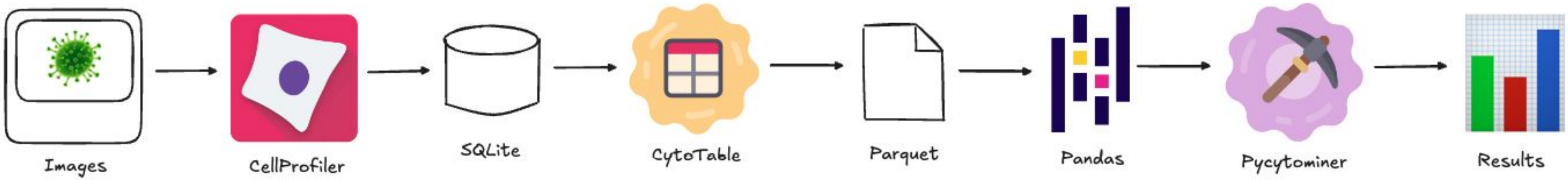
SQLite enables quicker data extraction through indexes, meaning we don’t need to read sequentially for many cases. Much faster than CSV’s.

Pipelines and exports



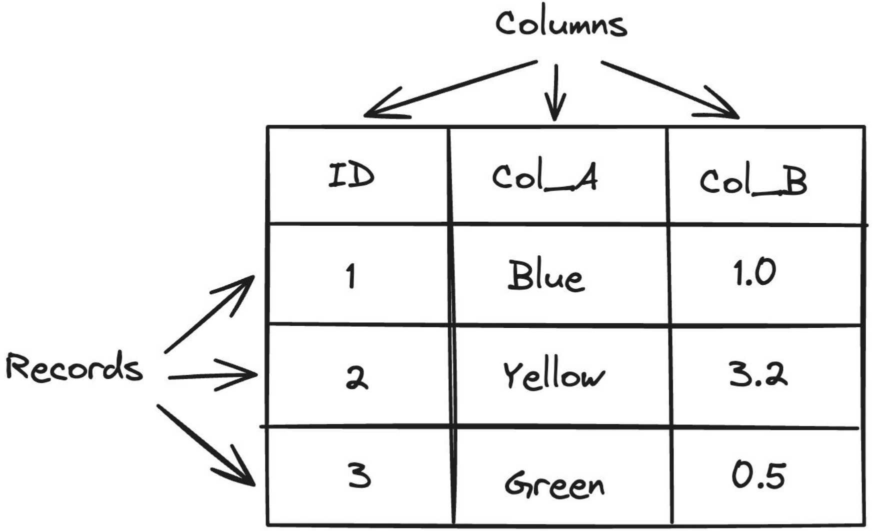
We can only gain that benefit through the local filesystem (we have to download the entire file from cloud storage).

Pipelines and exports



We can use CytoTable to create data which is purpose built for large data operations (incl. cloud), has strict data types, and use single-cell records as a model.

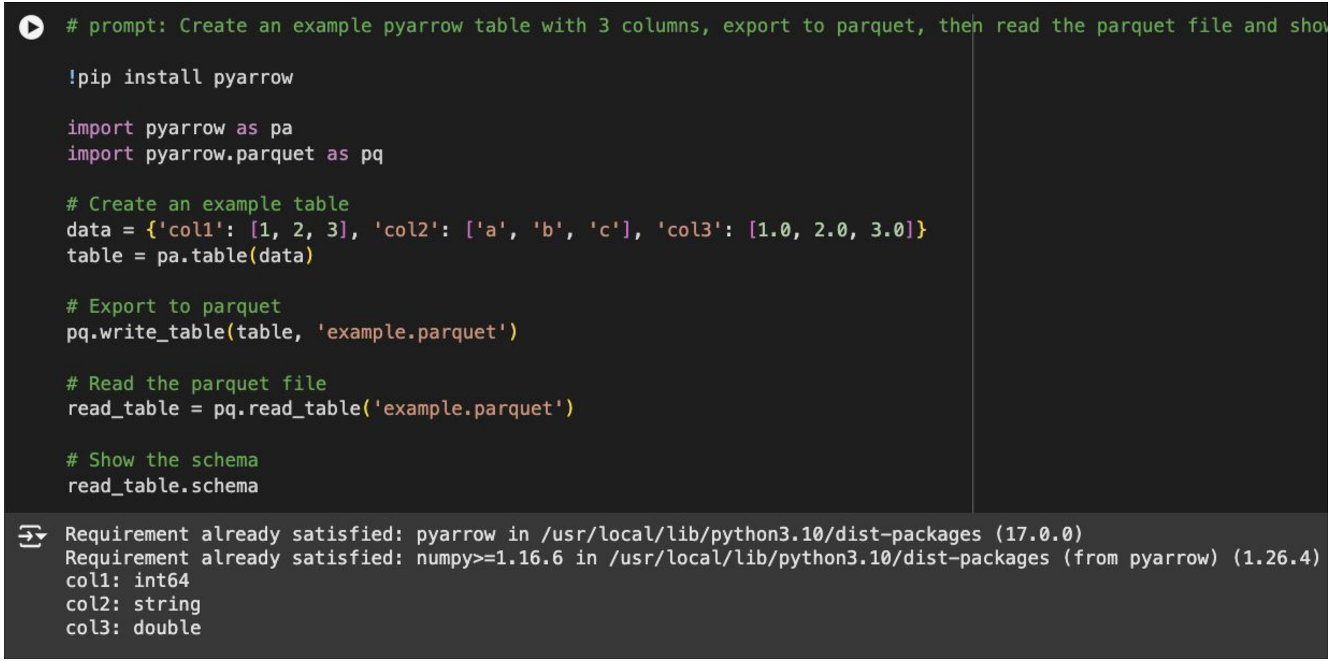
Pipelines and exports



ID	Col_A	Col_B
1	Blue	1.0
2	Yellow	3.2
3	Green	0.5

We use one table to imply the relationship among all columns through rows (records) of single-cell objects. This reduces complexity without losing a model.

Pipelines and exports

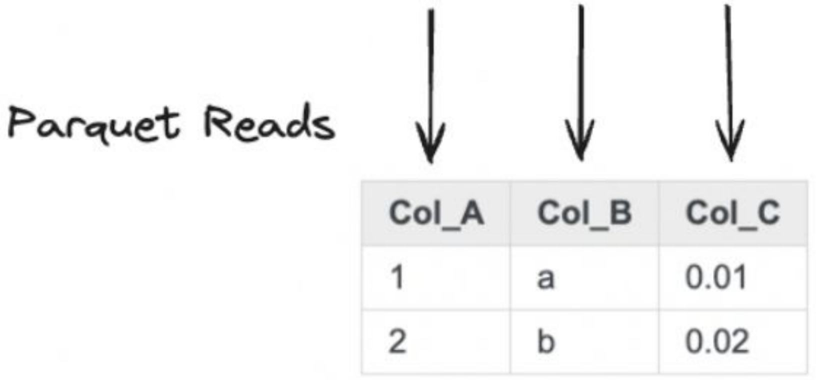


```
# prompt: Create an example pyarrow table with 3 columns, export to parquet, then read the parquet file and sho

!pip install pyarrow

import pyarrow as pa
import pyarrow.parquet as pq

# Create an example table
data = {'col1': [1, 2, 3], 'col2': ['a', 'b', 'c'], 'col3': [1.0, 2.0, 3.0]}
table = pa.table(data)

# Export to parquet
pq.write_table(table, 'example.parquet')

# Read the parquet file
read_table = pq.read_table('example.parquet')

# Show the schema
read_table.schema
```

```
Requirement already satisfied: pyarrow in /usr/local/lib/python3.10/dist-packages (17.0.0)
Requirement already satisfied: numpy>=1.16.6 in /usr/local/lib/python3.10/dist-packages (from pyarrow) (1.26.4)
col1: int64
col2: string
col3: double
```

Parquet columns and values are strict: column and value types must exist and must match. https://colab.research.google.com/drive/1pkitiLmbp_JlNSp1UIISafY8NjNvtrkc?authuser=0#scrollTo=vJ1kjYjIcUDR

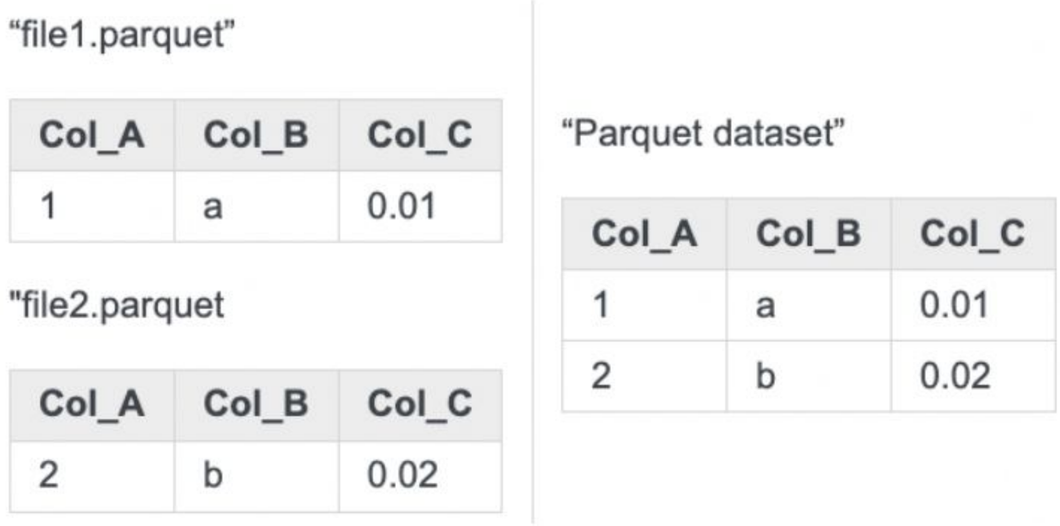
Pipelines and exports



Col_A	Col_B	Col_C
1	a	0.01
2	b	0.02

Parquet reads are “columnar” meaning we don’t need to scan each row and we gain efficiencies.

Pipelines and exports



“file1.parquet”

Col_A	Col_B	Col_C
1	a	0.01

"file2.parquet

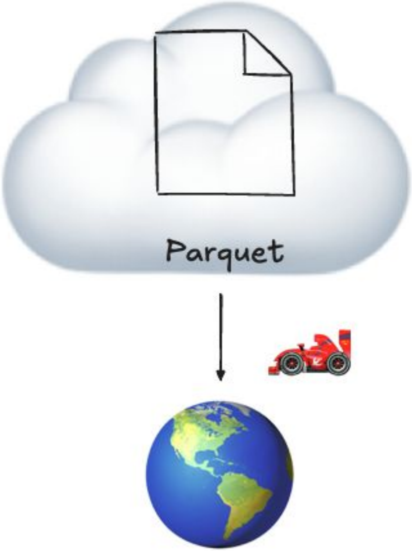
Col_A	Col_B	Col_C
2	b	0.02

“Parquet dataset”

Col_A	Col_B	Col_C
1	a	0.01
2	b	0.02

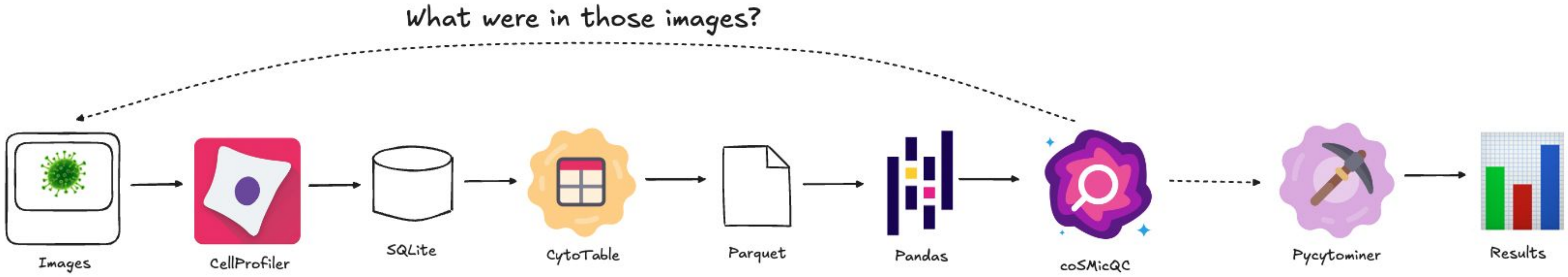
Parquet tables can be composed of one or many individual files (we aren’t constrained to a single file). Multi-file Parquet tables are called “datasets”.

Pipelines and exports



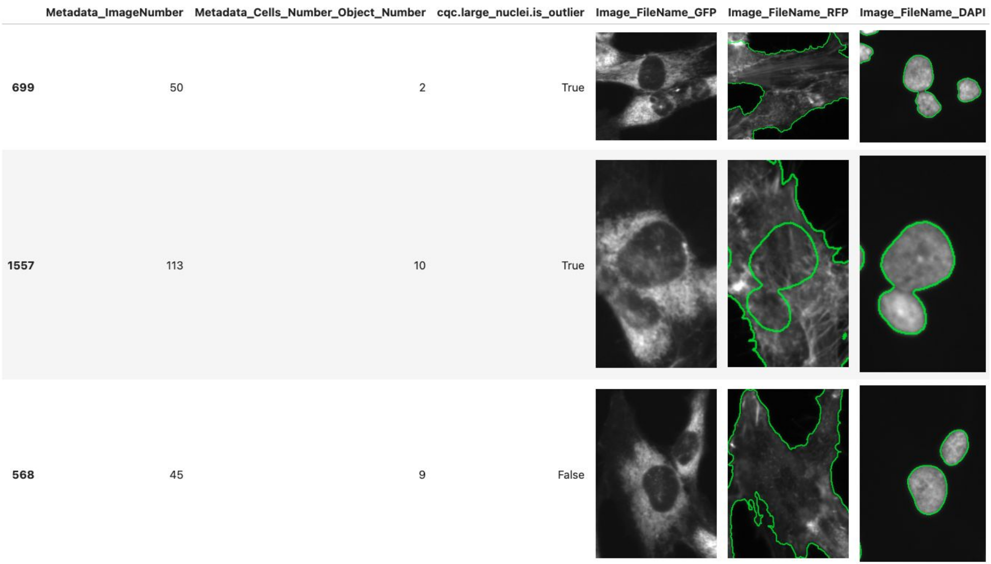
Parquet is optimized for cloud storage, meaning we have greater flexibility when it comes to storage and distribution.

Pipelines and exports



What if we need to go back to where the data was derived from (the images)?
https://github.com/WayScience/coSMicQC

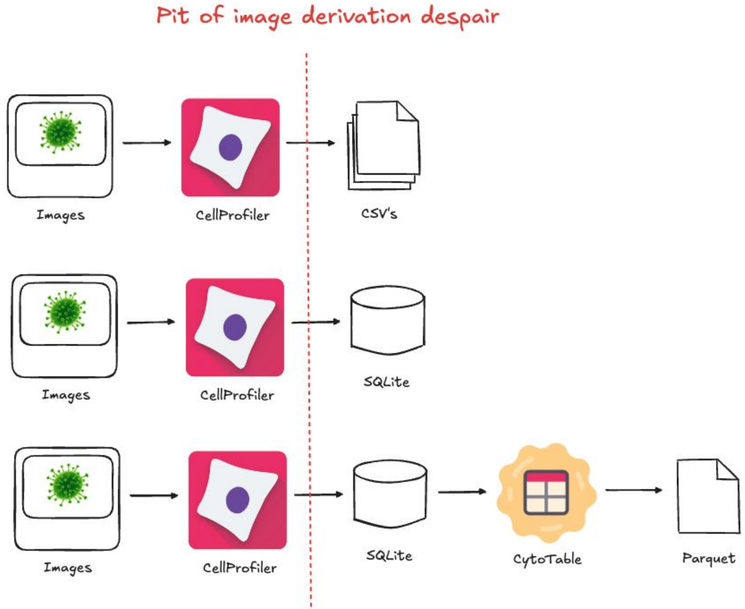
Pipelines and exports



	Metadata_ImageNumber	Metadata_Cells_Number_Object_Number	cqc.large_nuclei.is_outlier	Image_FileName_GFP	Image_FileName_RFP	Image_FileName_DAPI
699	50	2	True			
1557	113	10	True			
568	45	9	False			

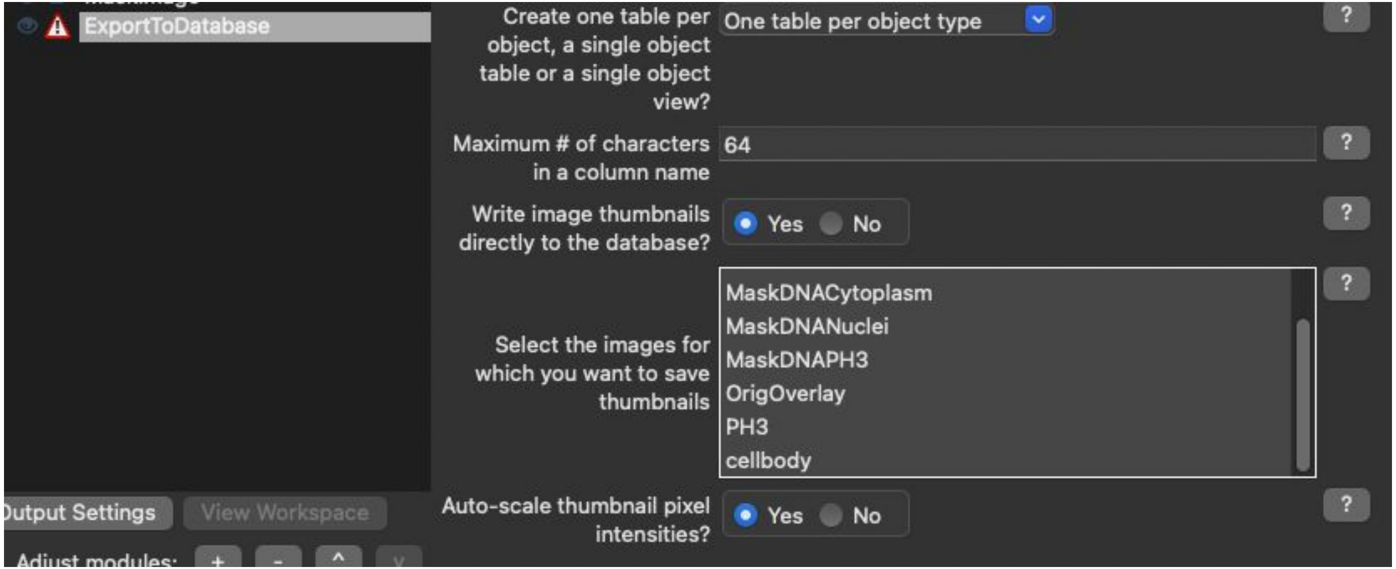
We can view image data alongside the profiles if the image data are available.
https://github.com/WayScience/CytoDataFrame

Pipelines and exports



Pit of image derivation despair: we're leaving behind the images with potentially no provenance back.

Pipelines and exports



Create one table per object, a single object table or a single object view? One table per object type

Maximum # of characters in a column name 64

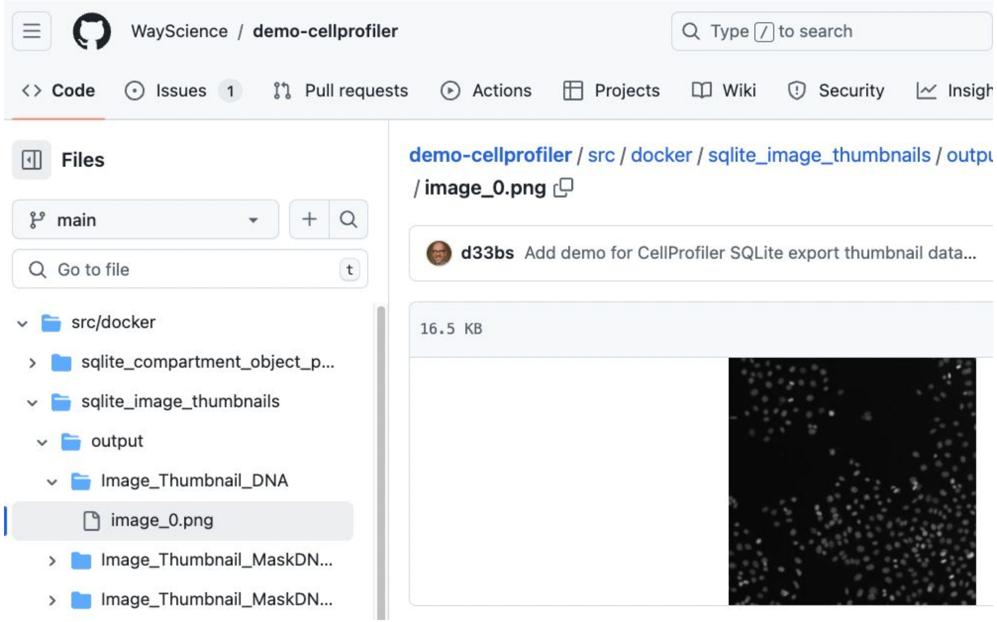
Write image thumbnails directly to the database? ● Yes ○ No

Select the images for which you want to save thumbnails
MaskDNACytoplasm
MaskDNANuclei
MaskDNAPH3
OrigOverlay
PH3
cellbody

Auto-scale thumbnail pixel intensities? ● Yes ○ No

Can we export images through CellProfiler module settings?

Pipelines and exports



- src/docker
  - sqlite_compartment_object_p...
  - sqlite_image_thumbnails
    - output
      - Image_Thumbnail_DNA
        - image_0.png
      - Image_Thumbnail_MaskDN...
      - Image_Thumbnail_MaskDN...

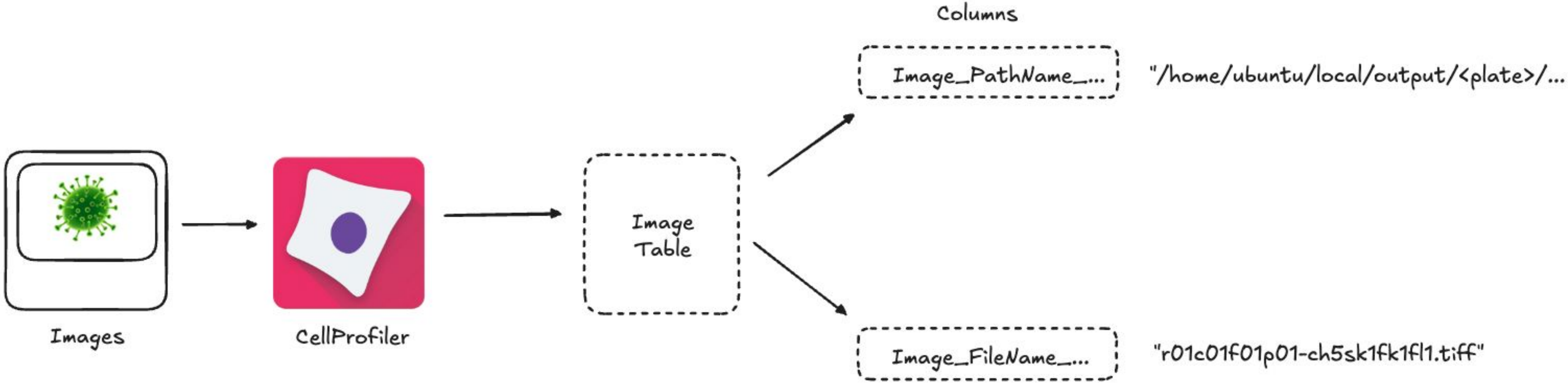
demo-cellprofiler / src / docker / sqlite_image_thumbnails / outpu
/ image_0.png

d33bs Add demo for CellProfiler SQLite export thumbnail data...

16.5 KB

No. ☹️

Pipelines and exports



Columns

Image_PathName_... "/home/ubuntu/local/output/<plate>/...

Images

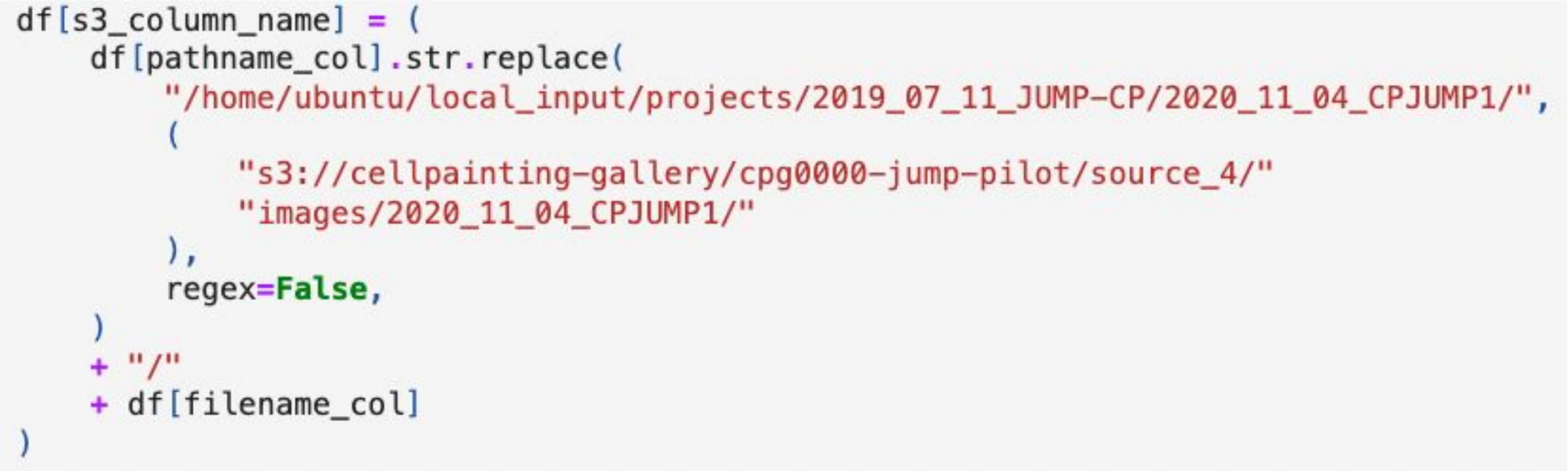
CellProfiler

Image Table

Image_FileName_... "r01c01f01p01-ch5sk1fk1fl1.tiff"

CellProfiler Image table output includes some clues.
Derivational data stored through “Image_FileName_...” column values.

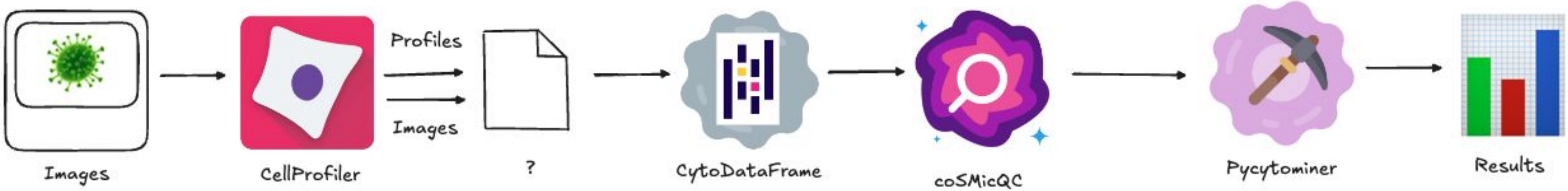
Pipelines and exports



```python
df[s3_column_name] = (
    df[pathname_col].str.replace(
        "/home/ubuntu/local_input/projects/2019_07_11_JUMP-CP/2020_11_04_CPJUMP1/",
        (
            "s3://cellpainting-gallery/cpg0000-jump-pilot/source_4/"
            "images/2020_11_04_CPJUMP1/"
        ),
        regex=False,
    )
    + "/"
    + df[filename_col]
)
```

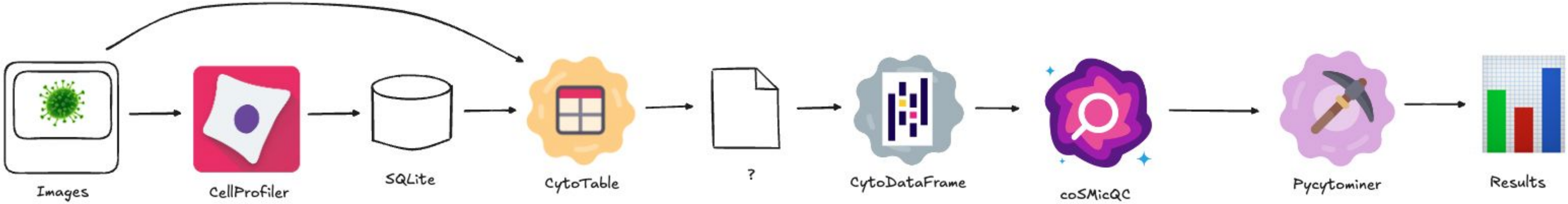
We can fix this, but it’s ugly and will result in bespoke solutions every time.

Future



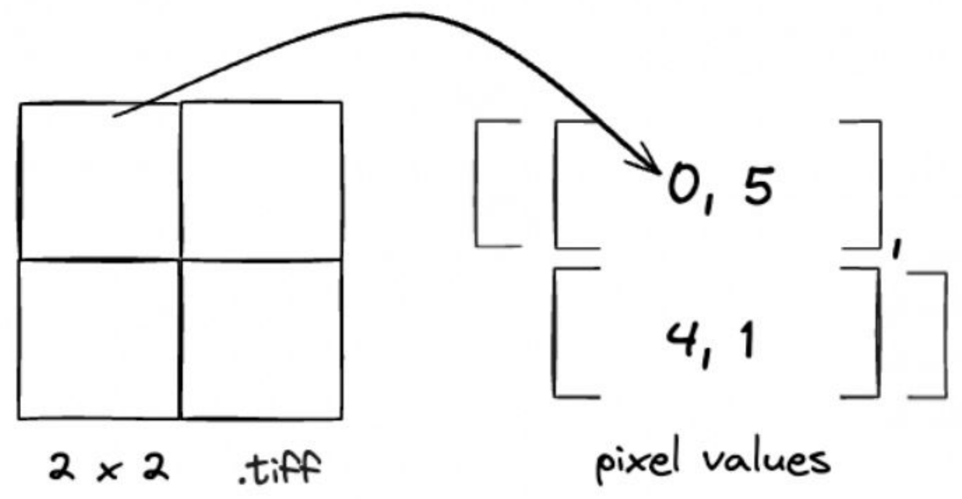
Could we build a format which retains the image data somehow? Ideally:
Strict data types, large data operable, cloud-compatible, and image data compatible.

Future



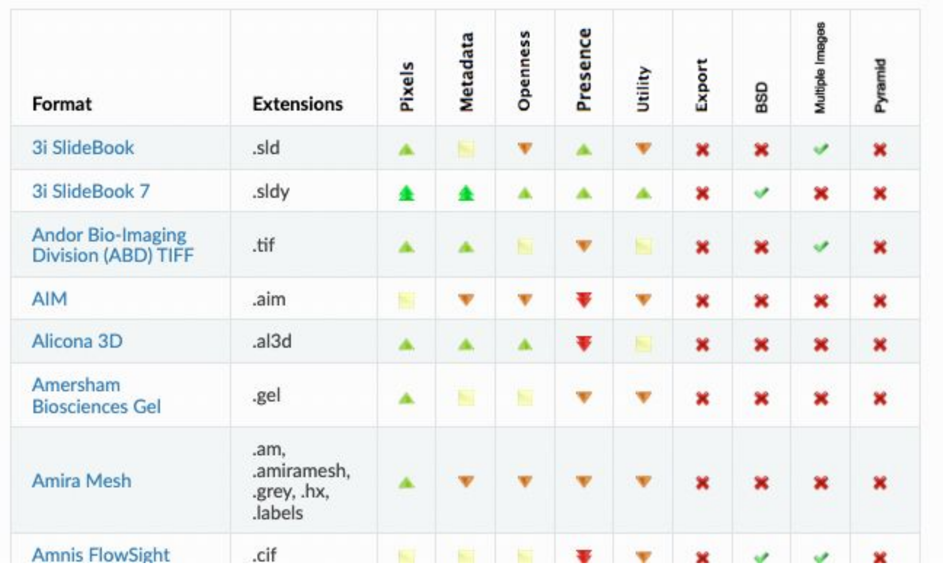
Open source science: in the meantime, we can build it!

Future



What do you mean image data compatible? Image data as arrays.

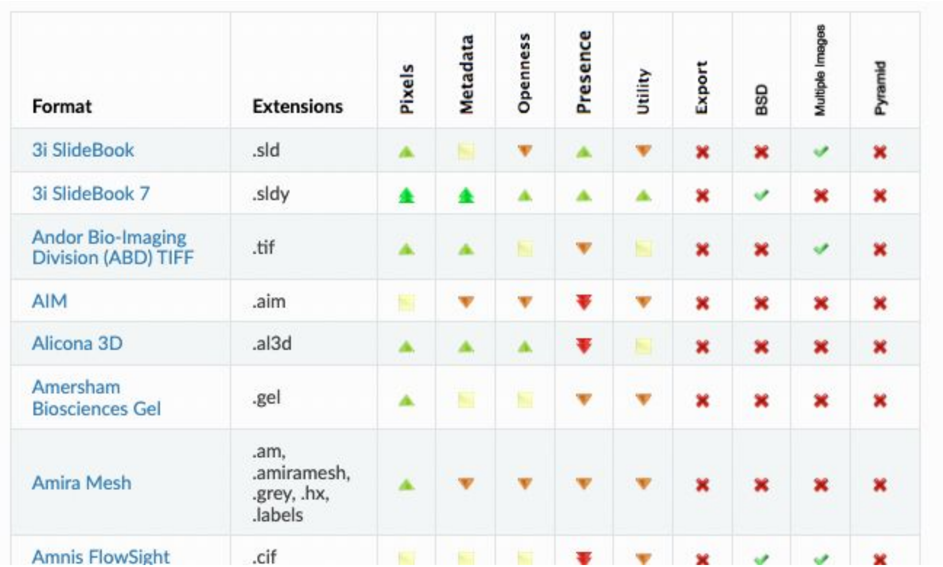
Future



Format	Extensions	Pixels	Metadata	Openness	Presence	Utility	Export	BSD	Multiple Images	Pyramid
3i SlideBook	.sld	▲	🟨	▼	▲	▼	✖	✖	✓	✖
3i SlideBook 7	.sldy	⏫	⏫	▲	▲	▲	✖	✓	✖	✖
Andor Bio-Imaging Division (ABD) TIFF	.tif	▲	▲	🟨	▼	🟨	✖	✖	✓	✖
AIM	.aim	🟨	▼	▼	⏬	▼	✖	✖	✖	✖
Alicona 3D	.al3d	▲	▲	▲	⏬	🟨	✖	✖	✖	✖
Amersham Biosciences Gel	.gel	▲	🟨	🟨	▼	▼	✖	✖	✖	✖
Amira Mesh	.am, .amiramesh, .grey, .hx, .labels	▲	▼	▼	▼	▼	✖	✖	✖	✖
Amnis FlowSight	.cif	🟨	🟨	🟨	⏬	▼	✖	✓	✓	✖

Why not binary large objects (BLOBs)? Standardizing data readers for binary formats would be chaotic and voids benefits of data typing.
https://bio-formats.readthedocs.io/en/v8.0.0/supported-formats.html

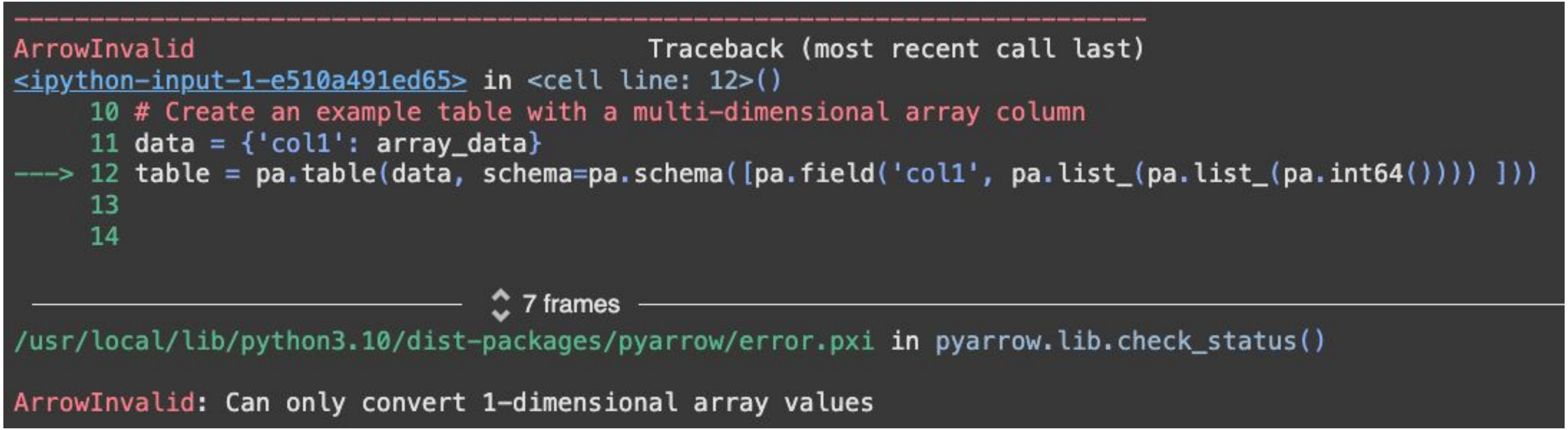
Future



Format	Extensions	Pixels	Metadata	Openness	Presence	Utility	Export	BSD	Multiple Images	Pyramid
3i SlideBook	.sld	▲	■	▼	▲	▼	✖	✖	✓	✖
3i SlideBook 7	.sldy	⏫	⏫	▲	▲	▲	✖	✓	✖	✖
Andor Bio-Imaging Division (ABD) TIFF	.tif	▲	▲	■	▼	■	✖	✖	✓	✖
AIM	.aim	■	▼	▼	⏬	▼	✖	✖	✖	✖
Alicona 3D	.al3d	▲	▲	▲	⏬	■	✖	✖	✖	✖
Amersham Biosciences Gel	.gel	▲	■	■	▼	▼	✖	✖	✖	✖
Amira Mesh	.am, .amiramesh, .grey, .hx, .labels	▲	▼	▼	▼	▼	✖	✖	✖	✖
Amnis FlowSight	.cif	■	■	■	⏬	▼	✖	✓	✓	✖

Why not binary large objects (BLOBs)? Standardizing data readers for binary formats would be chaotic and voids benefits of data typing.
https://bio-formats.readthedocs.io/en/v8.0.0/supported-formats.html

Future



```
---------------------------------------------------------------------------
ArrowInvalid                              Traceback (most recent call last)
<ipython-input-1-e510a491ed65> in <cell line: 12>()
     10 # Create an example table with a multi-dimensional array column
     11 data = {'col1': array_data}
---> 12 table = pa.table(data, schema=pa.schema([pa.field('col1', pa.list_(pa.list_(pa.int64()))) ]))
     13
     14

                              ⌃⌄ 7 frames
/usr/local/lib/python3.10/dist-packages/pyarrow/error.pxi in pyarrow.lib.check_status()

ArrowInvalid: Can only convert 1-dimensional array values
```

Parquet has issues with array values.

https://colab.research.google.com/drive/1DJ3z5zhKc5wO8rs5GYdT0h9y6cgMCWXy

Future

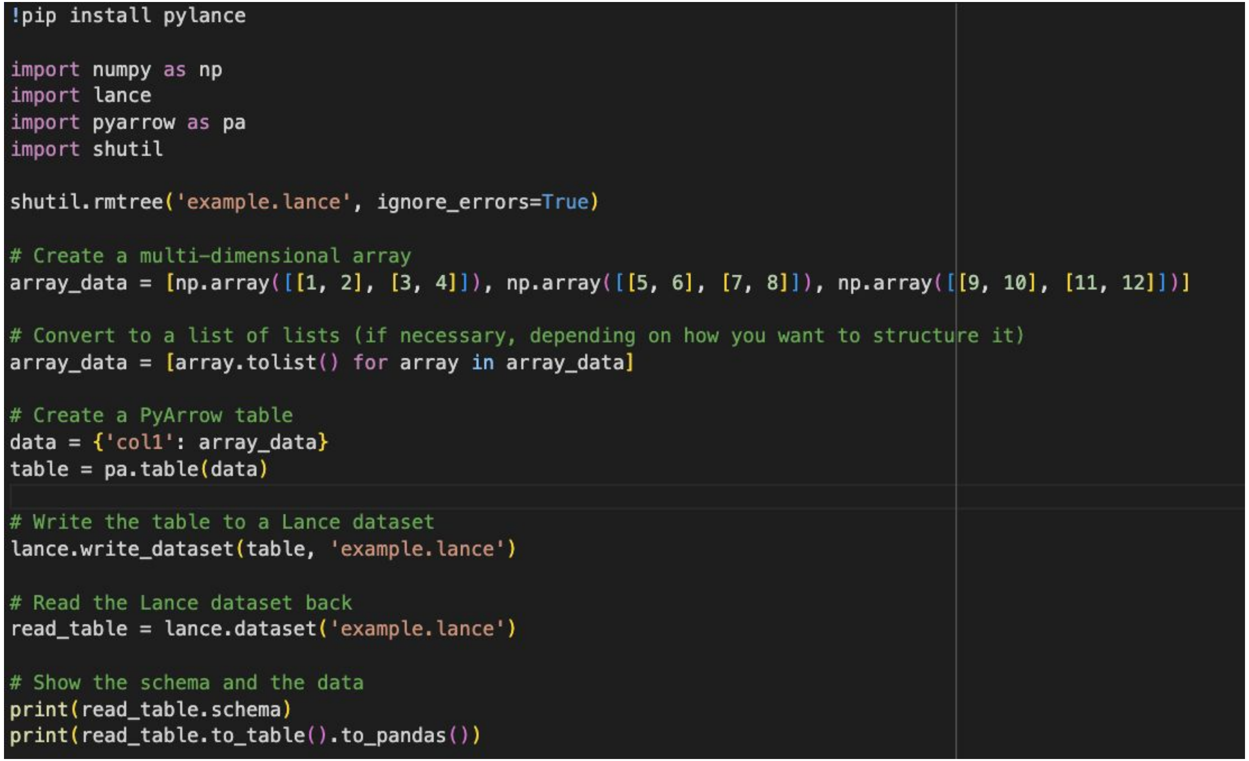
Lance

Lance includes:

- Strict data typing
- Large data operation compatible
- Cloud compatibilities
- Array value handling

https://github.com/lancedb/lance

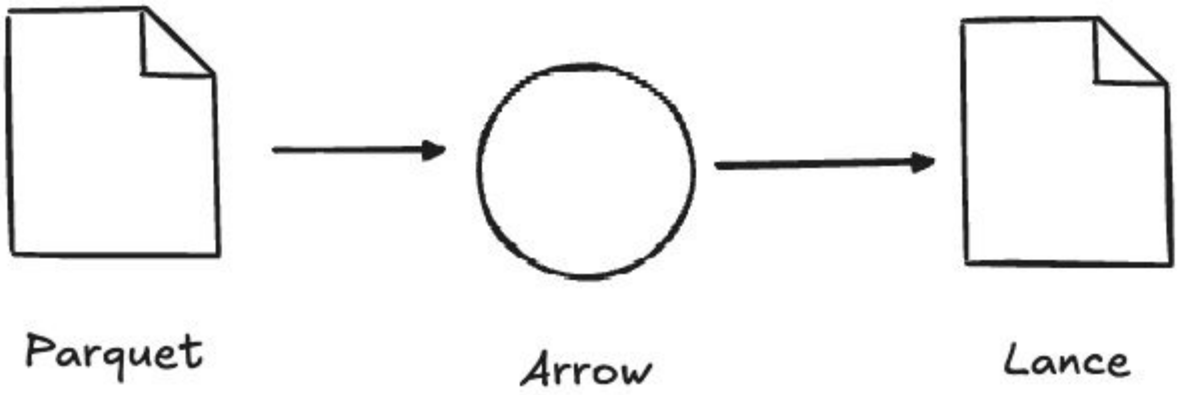
Future



```
!pip install pylance

import numpy as np
import lance
import pyarrow as pa
import shutil

shutil.rmtree('example.lance', ignore_errors=True)

# Create a multi-dimensional array
array_data = [np.array([[1, 2], [3, 4]]), np.array([[5, 6], [7, 8]]), np.array([[9, 10], [11, 12]])]

# Convert to a list of lists (if necessary, depending on how you want to structure it)
array_data = [array.tolist() for array in array_data]

# Create a PyArrow table
data = {'col1': array_data}
table = pa.table(data)

# Write the table to a Lance dataset
lance.write_dataset(table, 'example.lance')

# Read the Lance dataset back
read_table = lance.dataset('example.lance')

# Show the schema and the data
print(read_table.schema)
print(read_table.to_table().to_pandas())
```

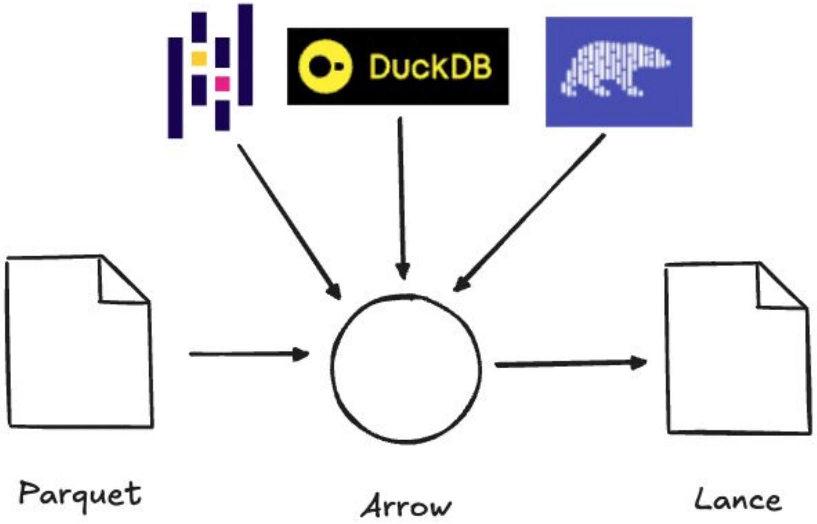
https://colab.research.google.com/drive/10dEpmtC1_pGm2qyc771DsqgVeNXnFQ85

Future



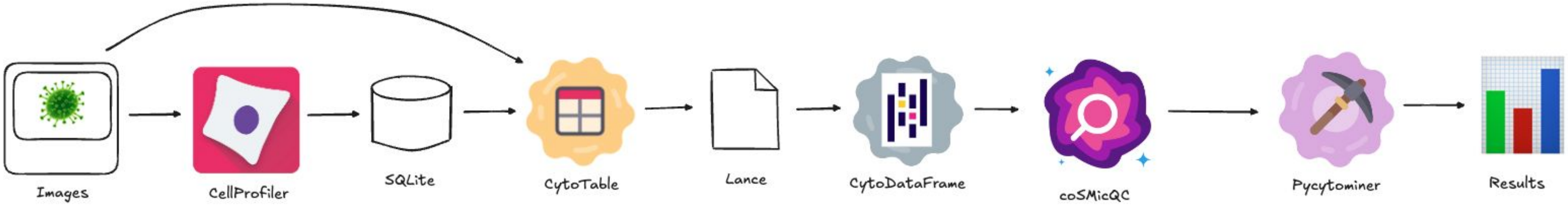
Lance uses Apache Arrow, meaning existing Parquet are backwards compatible.

Future

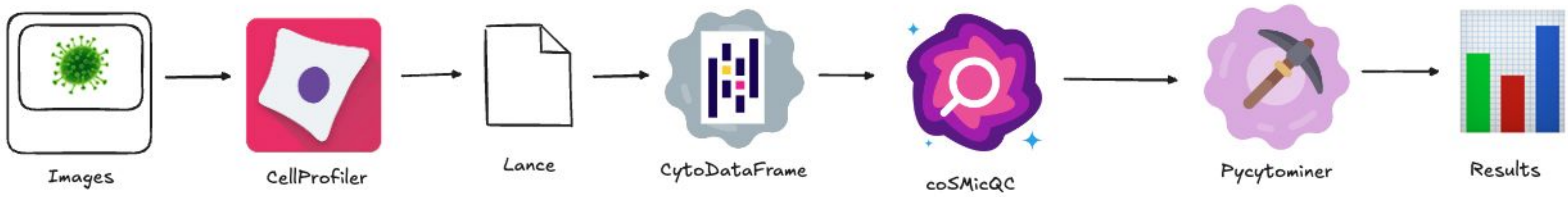


This also means we can use our favorite data analytics engines (so long as they're compatible with Apache Arrow).

What if … ?

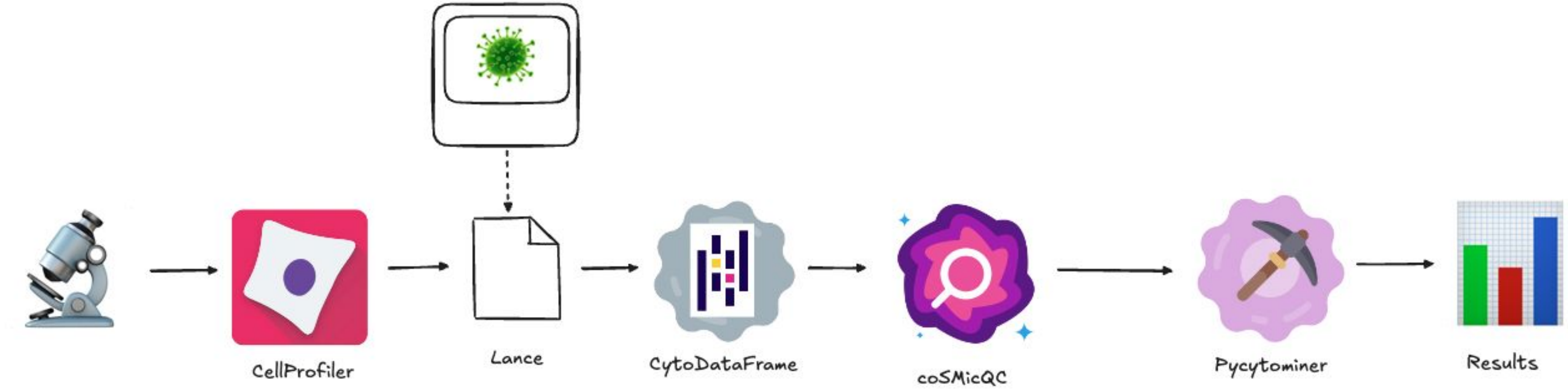

Images
CellProfiler
SQLite
CytoTable
Lance
CytoDataFrame
coSMicQC
Pycytominer
Results

What if ... ?



This work could be a stopgap to further complexity reduction, increasing research velocity.

What if ... ?



What if ...?

Thanks! Questions / comments?