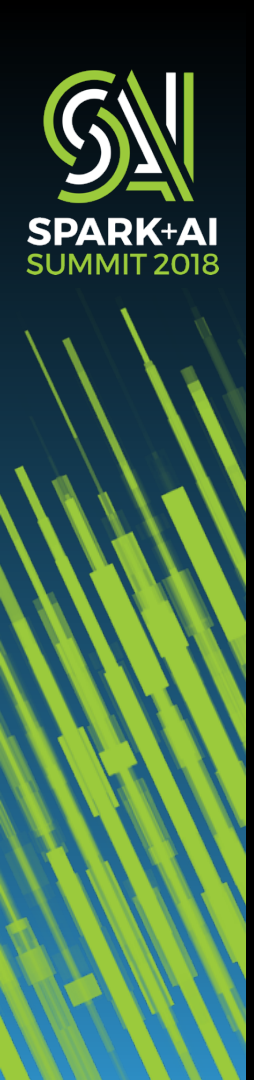
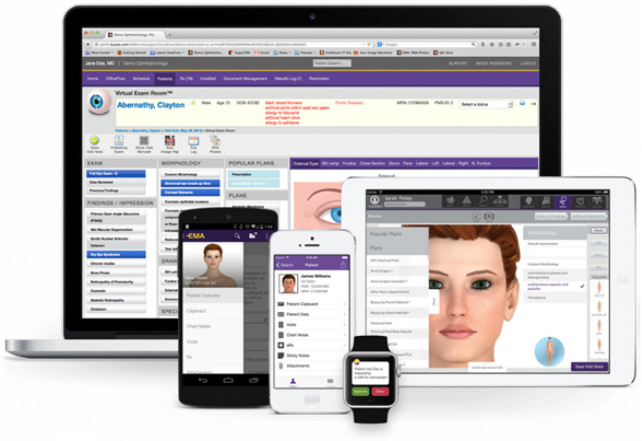


Dynamic Healthcare Dataset Generation, Curation & Quality with PySpark

Aaron Richter, Modernizing Medicine®

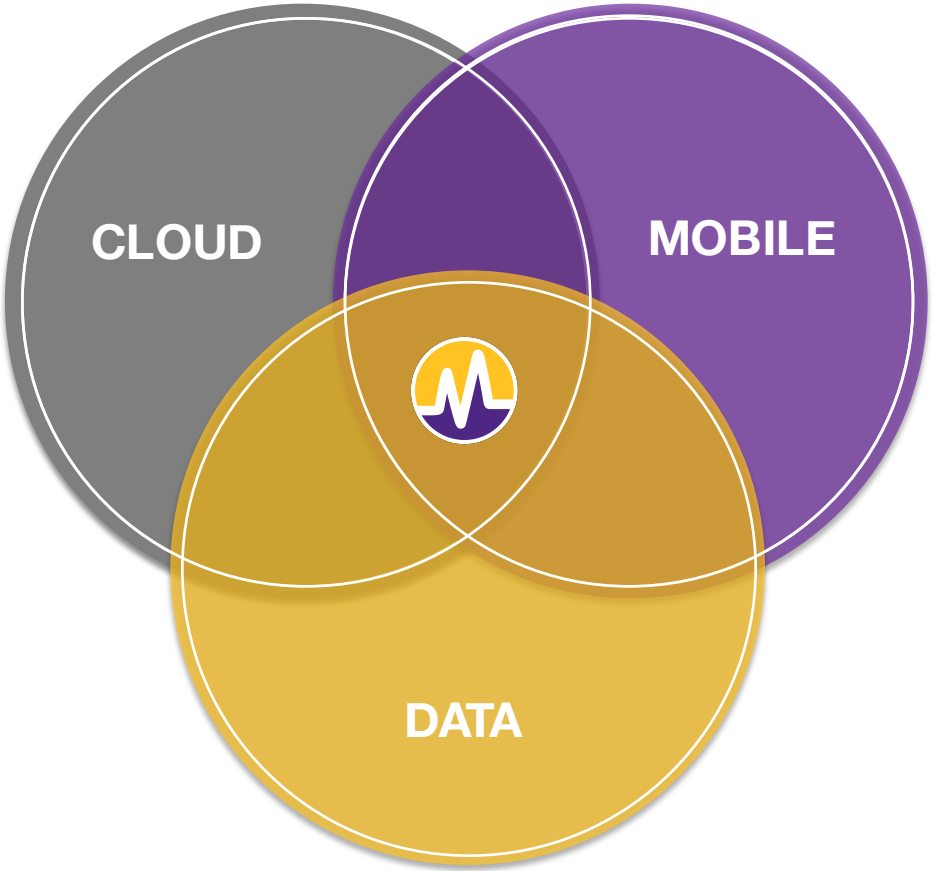
#Py3SAIS
@rikturr

MODERNIZING MEDICINE



- Electronic Health Record (EHR) system, EMA™
- Practice Management (PM)
- Revenue Cycle Management (RCM)
- Analytics
- Specialty-specific
- 14,000+ providers across the US

Transforming how healthcare information is created, consumed and utilized to increase practice efficiency and improve patient outcomes.


CLOUD
MOBILE
DATA


DATA @ MODMED
BIG DATA? BIG DATA! BIG DATA!?

DATA @ MODMED


90,000+
users
100,000,000+
patient encounters
100,000+
encounters/day

DATA @ MODMED

What we do

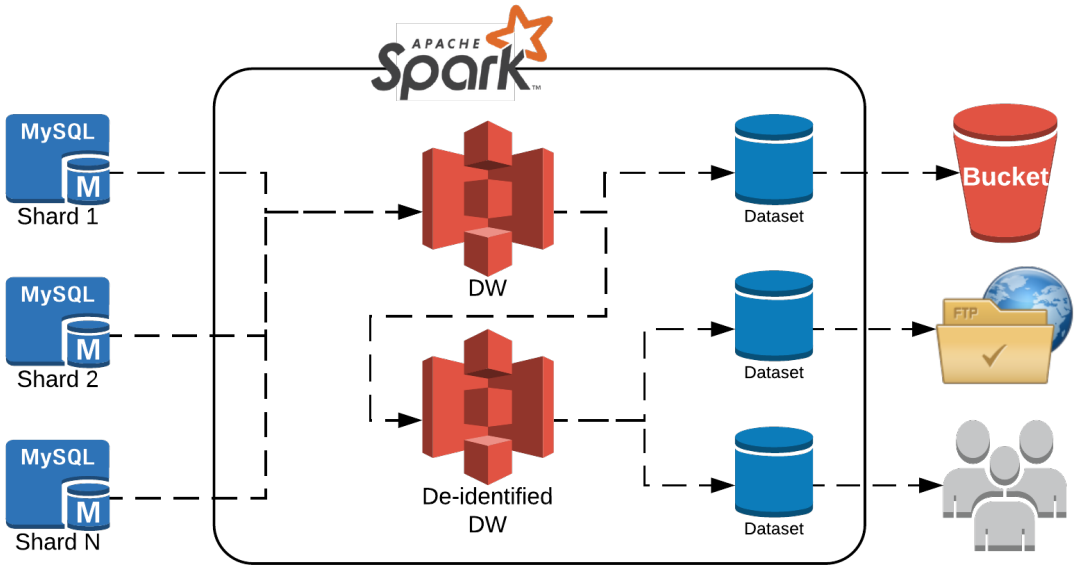
- Data Warehouse
- De-identified Data Warehouse
- Data Extracts
  - Practice data
  - Population health research
  - Business intelligence
  - Data Science

How we do it

DATA @ MODMED

Data Architecture

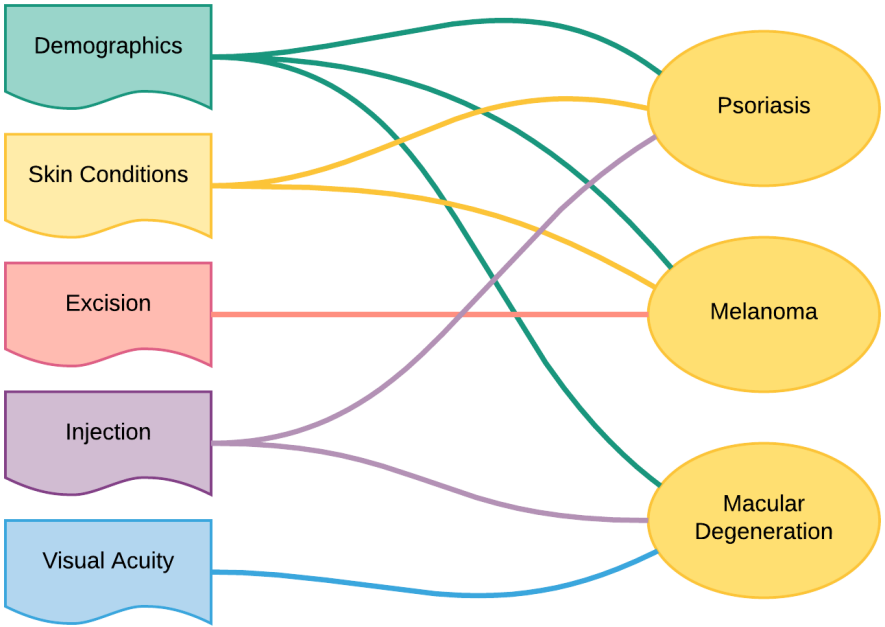
- Multi-tenant database shards
- Shard (MySQL)
  - 800+ tables
  - 600GB+
  - 2 largest tables 100GB each
- S3 data lake
  - Extract out of MySQL
  - Save to parquet



DATASET GENERATION

THE SECRET SAUCE

DYNAMIC DATASETS



WHAT DID WE DO BEFORE?

- Impala SQL statements
  - Impala dies -> try Hive
- XML definitions of datasets
- SQL file for each table
- String replacements to inject variables
- Java program

WHY DO WE USE PYTHON?

Team Considerations

- Everyone already knows Python
- Too stubborn to learn Scala
- Not fans of JVM, JARs, Maven, etc.
- Infrastructure consistency (i.e. Airflow)

Python ecosystem

- PyCharm
- Pandas, numpy, etc.
- Read/write JSON
- Write Excel

PYTHON, FTW!

- Define helper methods for each type of source
- File presence, saving, loading
- Unique methods for sources
  - SQL indexes
  - Text file delimiters, separator, quote char
- Dot/dictionary access to tables

file_abstractions.py — UNREGISTERED

file_abstractions.py | reuse.py | acne.py

```python
import os


class DataDirectory(object):
    def __init__(self, spark, path):
        self.spark = spark
        self.path = path

    def __getattr__(self, item):
        return self.__getitem__(item)

    def __getitem__(self, item):
        return self.load_table(item)

    def load_table(self, name):
        table_path = os.path.join(self.path, name)
        return self.spark.read.parquet(table_path)


class HiveDatabase(DataDirectory):
    def load_table(self, name):
        db_table = '{}.{}'.format(self.db, name)
        return self.spark.read.table(db_table)


class TextFiles(DataDirectory):
    def load_table(self, name):
        table_path = os.path.join(self.path, name)
        return self.spark.read.csv(table_path, sep=self.sep)
```

Line 1, Column 1 Spaces: 4 Python

PYTHON, FTW!

- DataFrame/Column functions
- Python UDFs
- OOP
- Runtime arguments

reuse.py — UNREGISTERED

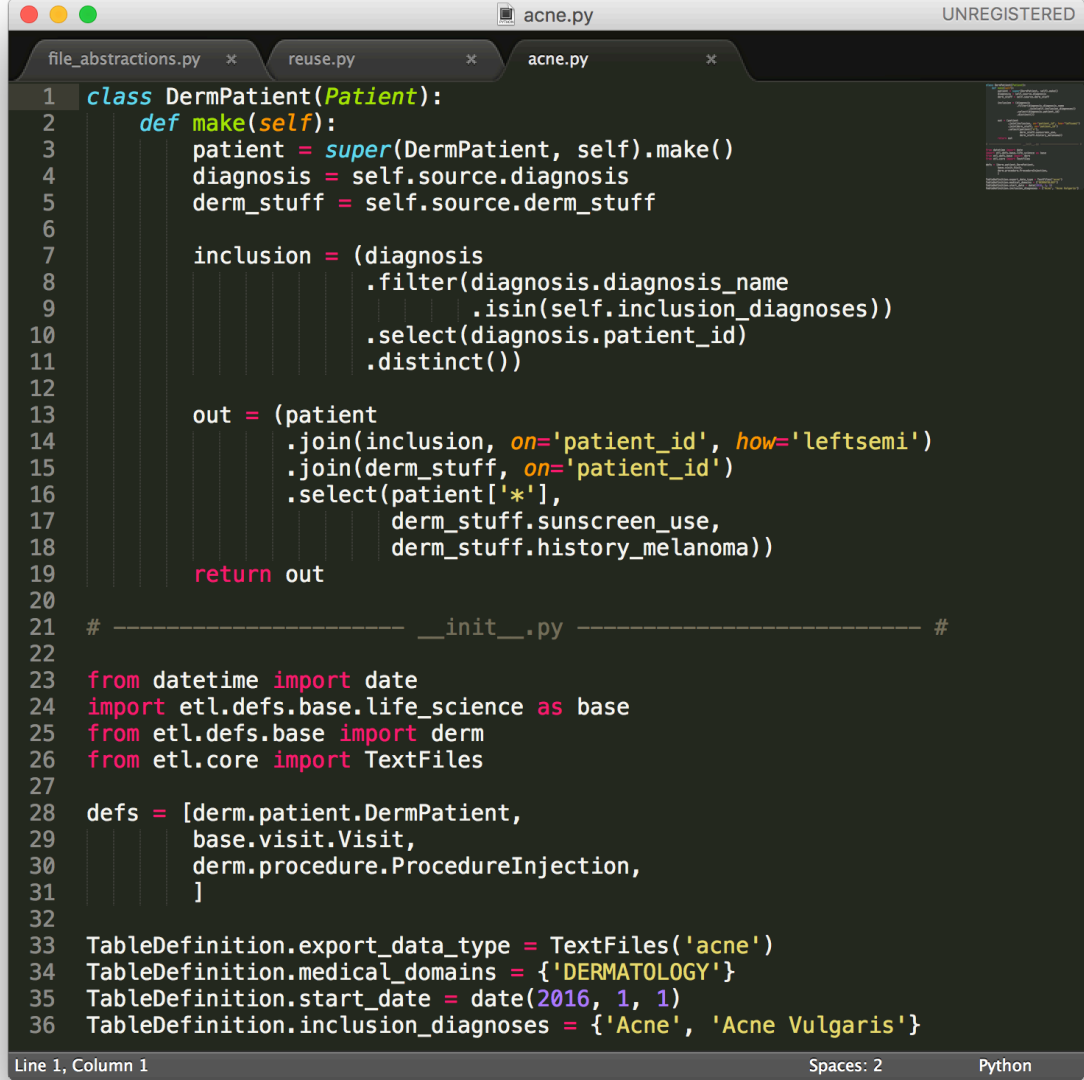
file_abstractions.py | reuse.py | acne.py

```python
class TableDefinition(object):
    name = ''
    persist = False

    @staticmethod
    def group_concat(col, sep=','):
        collect = collect_list(col.cast(StringType()))
        return concat_ws(sep, collect)

    def __init__(self, source, dest):
        self.source = source
        self.dest = dest

    def make(self):
        raise NotImplementedError("Please implement me.")

    def run(self):
        out = self.make()

        if self.persist:
            out.persist()

        return out

class MyNewTable(TableDefinition):
    name = 'my_new_table'

    def make(self):
        src_table = self.source.src_table

        return (src_table
                .groupby(src_table.group_col)
                .agg(self.group_concat(src_table.concat_col)))

```

Line 1, Column 1 | Spaces: 4 | Python

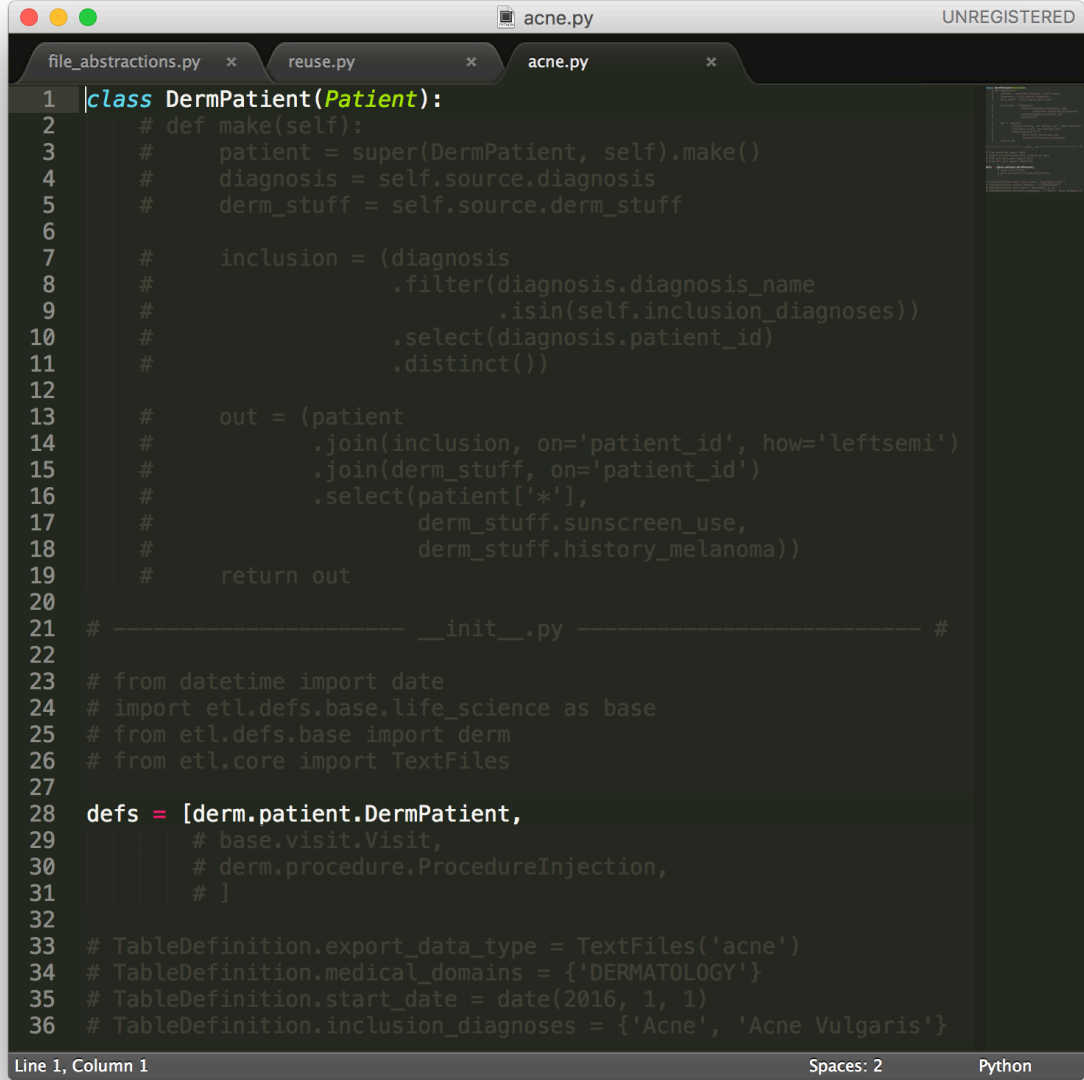
PYTHON, FTW!

- Module per dataset
- Enumerate tables
- Parameters/inclusion criteria



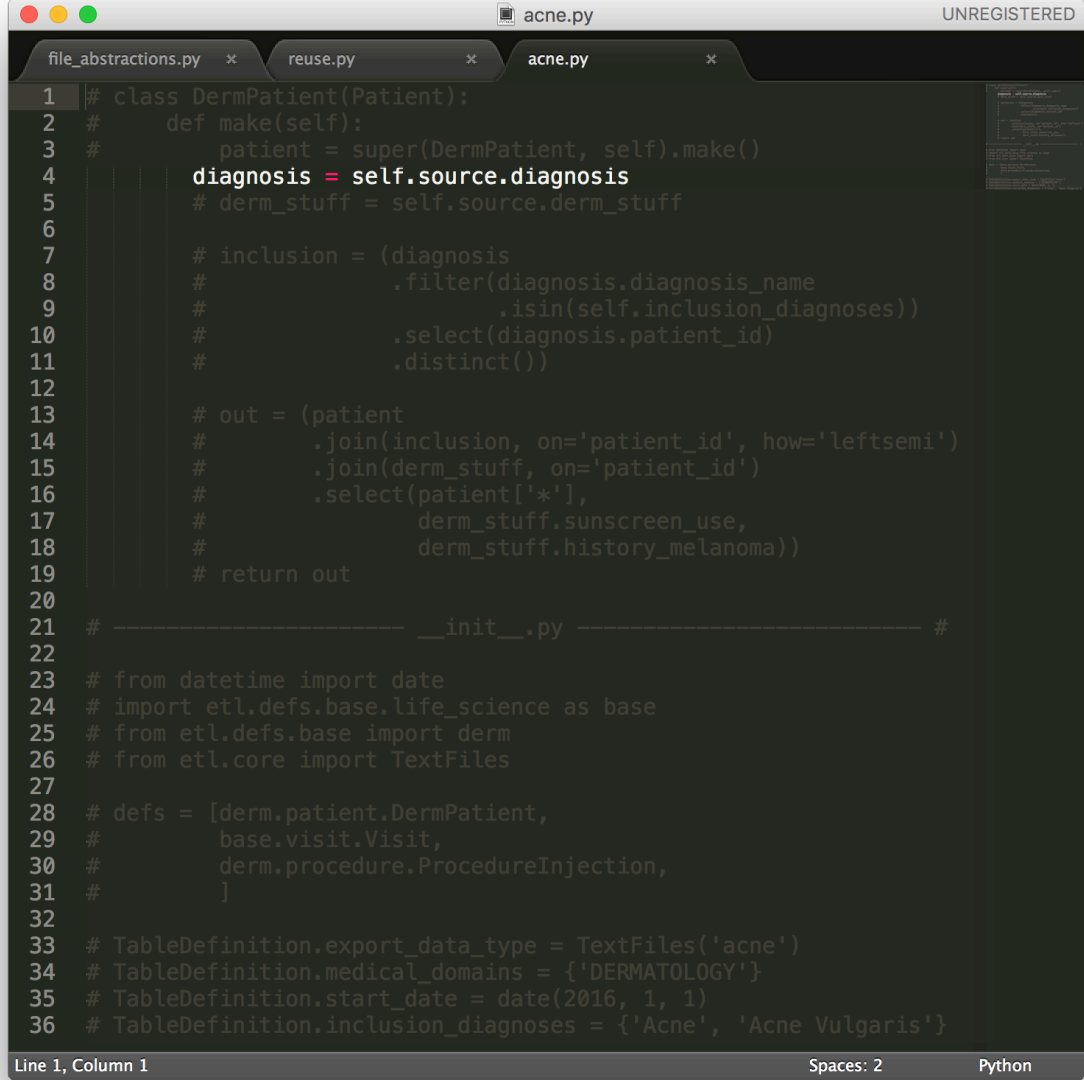
PYTHON, FTW!

- Module per dataset
- Enumerate tables
- Parameters/inclusion criteria



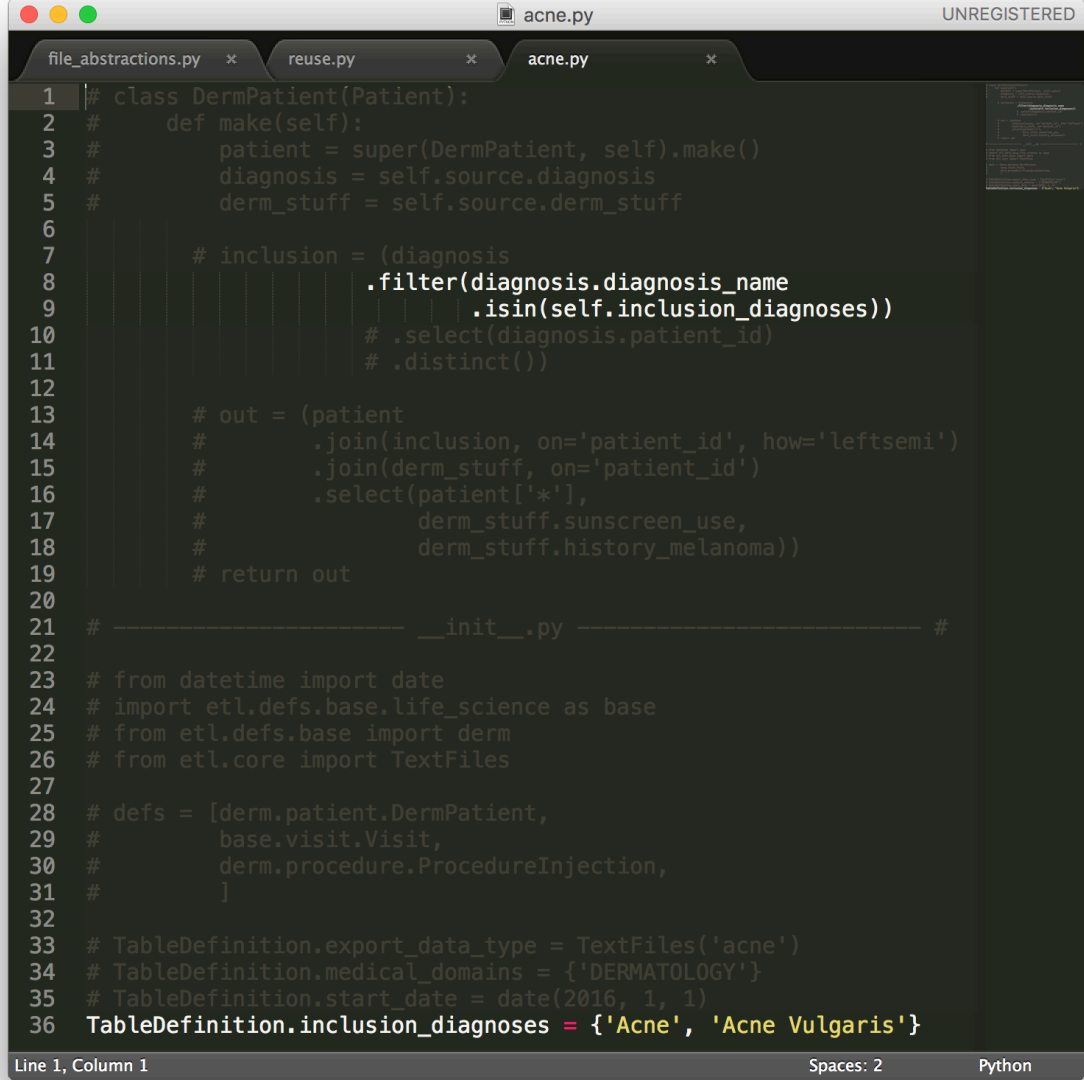
PYTHON, FTW!

- Module per dataset
- Enumerate tables
- Parameters/inclusion criteria



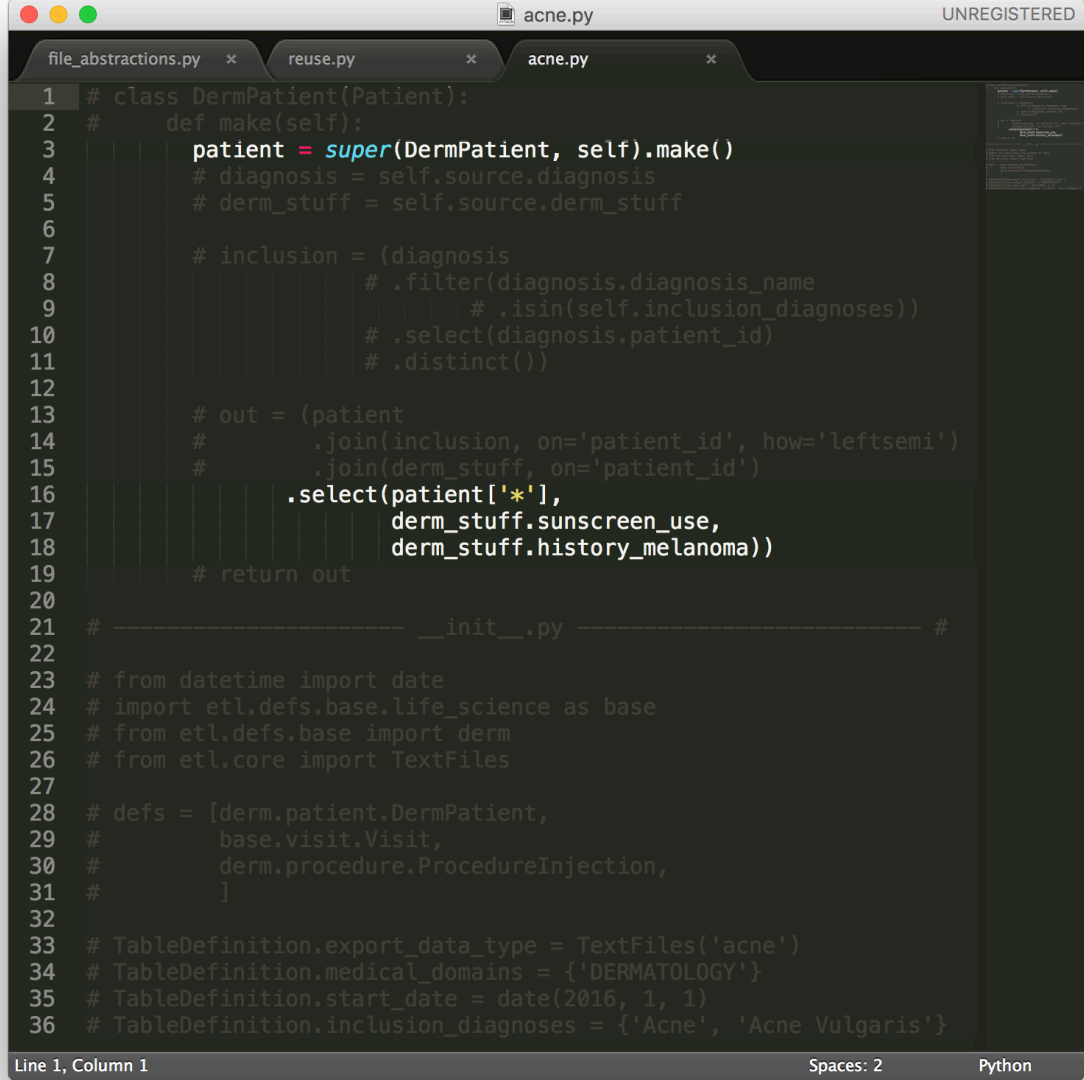
PYTHON, FTW!

- Module per dataset
- Enumerate tables
- Parameters/inclusion criteria



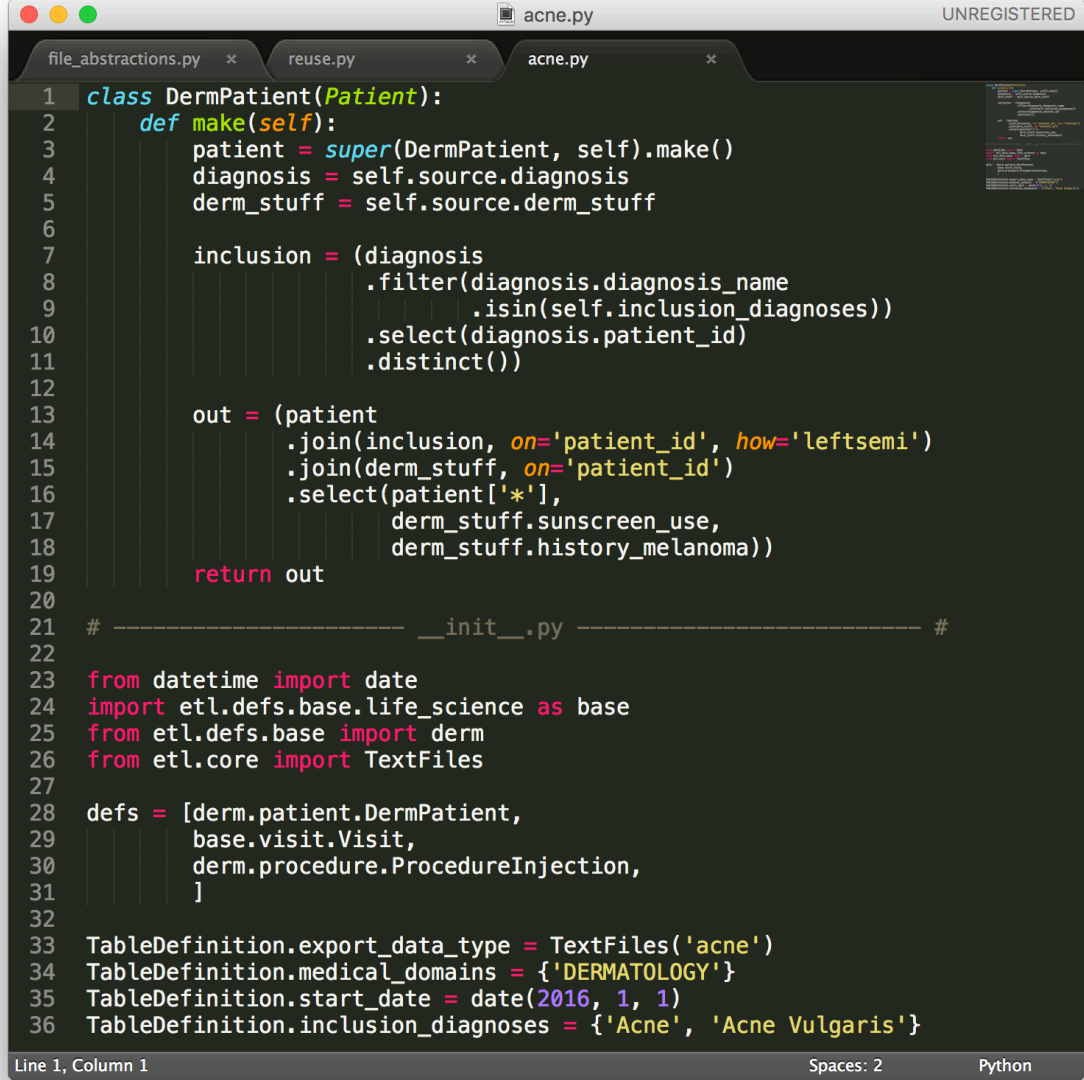
PYTHON, FTW!

- Module per dataset
- Enumerate tables
- Parameters/inclusion criteria



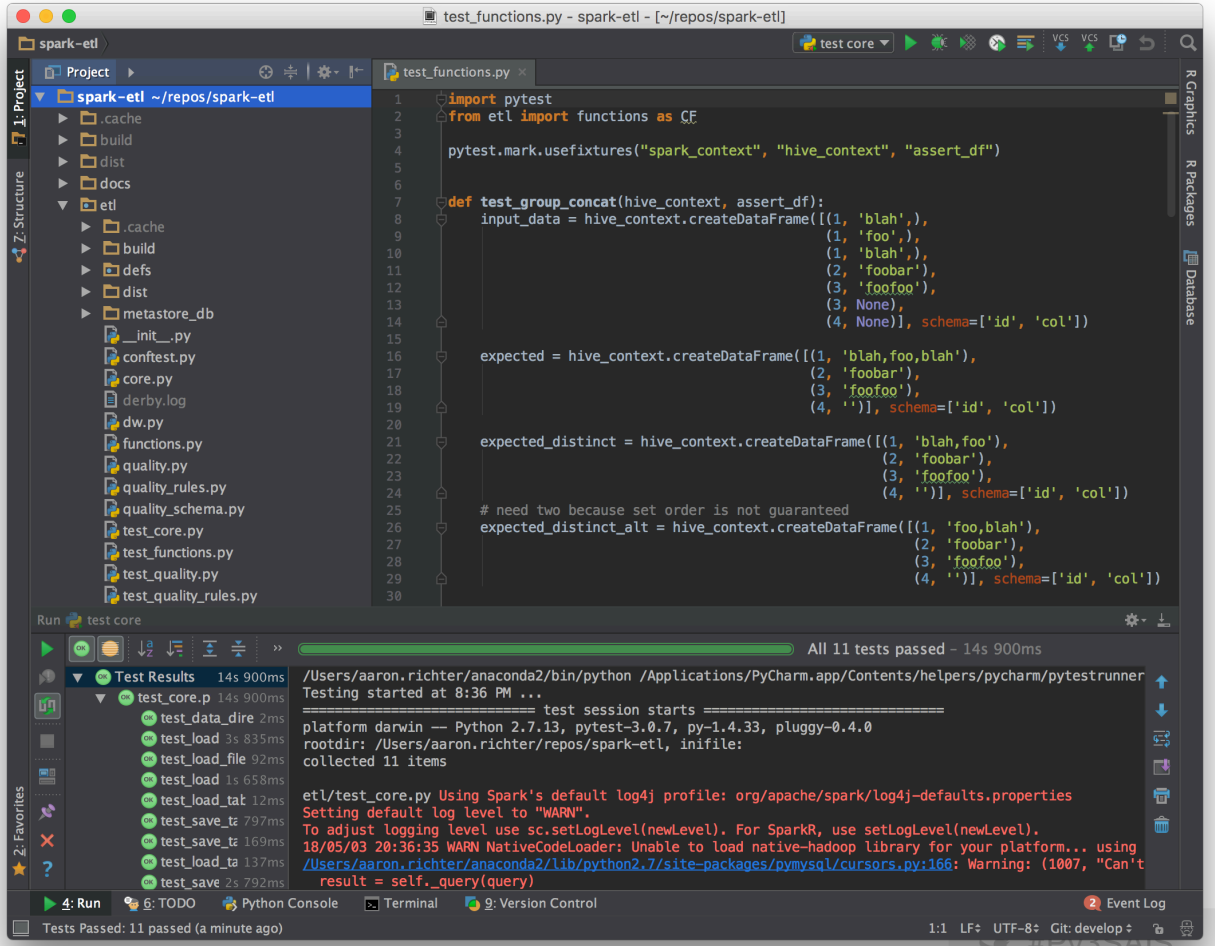
PYTHON, FTW!

- Module per dataset
- Enumerate tables
- Parameters/inclusion criteria



PYTHON, FTW!

- PyCharm
  - Autocomplete
  - Code checks
- Unit tests



CURATION & QUALITY

DATA ABOUT THE DATA

CURATION & QUALITY

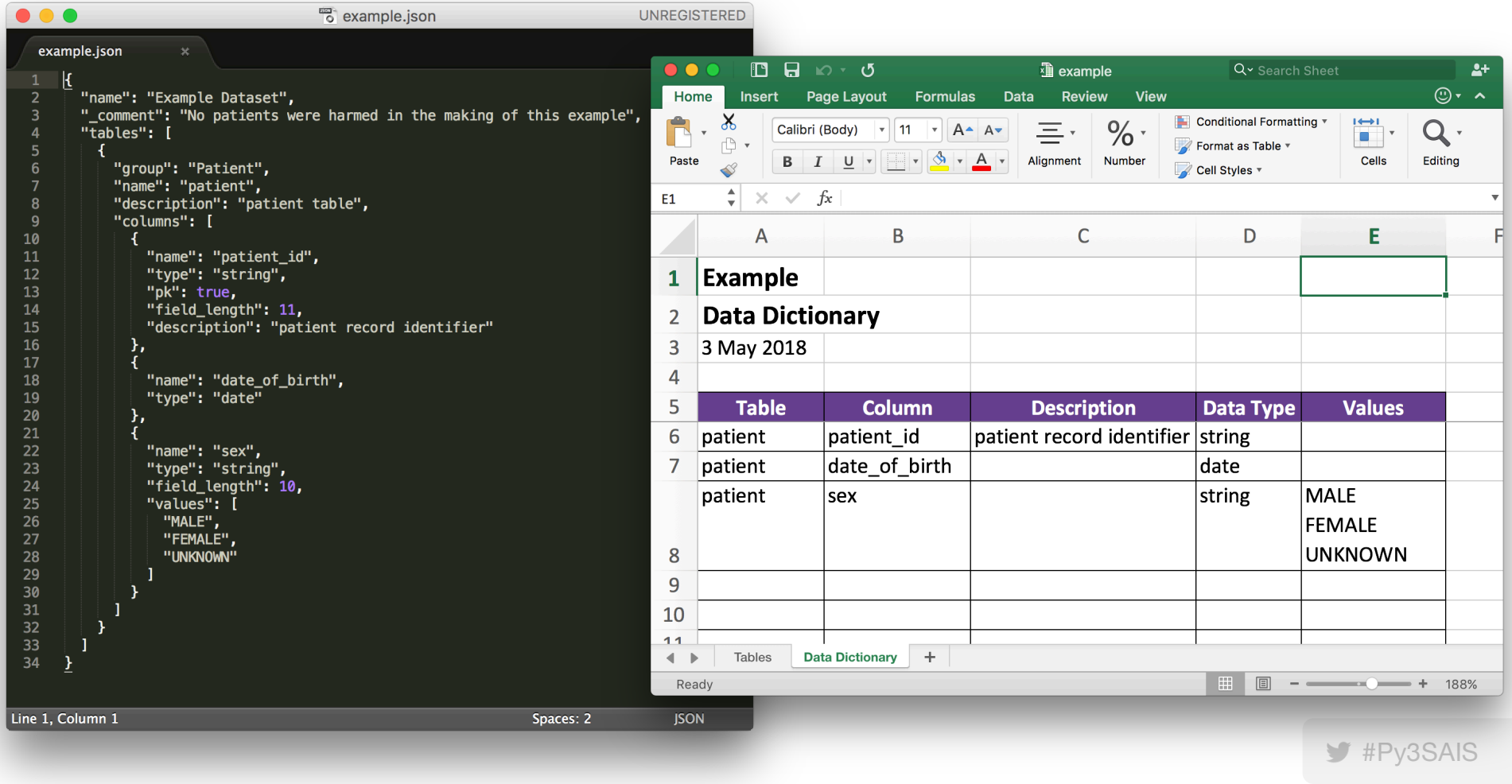
Data Dictionaries

- JSON file for each dataset's schema + metadata
- Generate Excel files from JSON definitions
  - Version control
  - Source of truth
  - pandas + xlsxwriter

Quality

- JSON file includes quality rules
  - Column names and types
  - Primary/foreign keys
  - Accepted values
- Processing in PySpark
- Future: Web-based content management system

Shoutout to *Smartsheet* for inspiration
https://pages.databricks.com/Smartsheet-operationalized-Spark.html

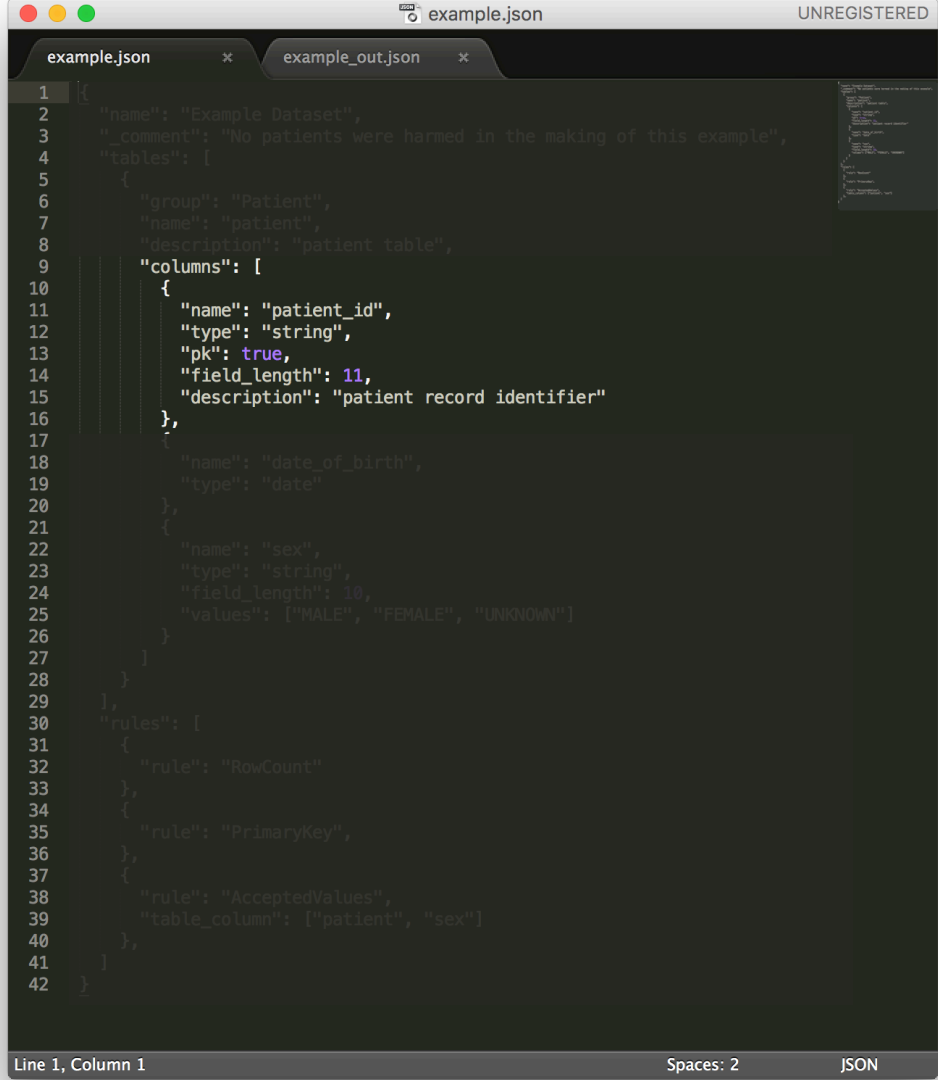

example.json
UNREGISTERED
example.json
{
"name": "Example Dataset",
"_comment": "No patients were harmed in the making of this example",
"tables": [
{
"group": "Patient",
"name": "patient",
"description": "patient table",
"columns": [
{
"name": "patient_id",
"type": "string",
"pk": true,
"field_length": 11,
"description": "patient record identifier"
},
{
"name": "date_of_birth",
"type": "date"
},
{
"name": "sex",
"type": "string",
"field_length": 10,
"values": [
"MALE",
"FEMALE",
"UNKNOWN"
]
}
]
}
]
}
Line 1, Column 1
Spaces: 2
JSON
example
Search Sheet
Home
Insert
Page Layout
Formulas
Data
Review
View
Paste
Calibri (Body)
11
Alignment
Number
Conditional Formatting
Format as Table
Cell Styles
Cells
Editing
E1
Example
Data Dictionary
3 May 2018
Table
Column
Description
Data Type
Values
patient
patient_id
patient record identifier
string
patient
date_of_birth
date
patient
sex
string
MALE
FEMALE
UNKNOWN
Tables
Data Dictionary
Ready
188%

QUALITY

- Column properties trigger rules
- Specify table/column to execute rule for

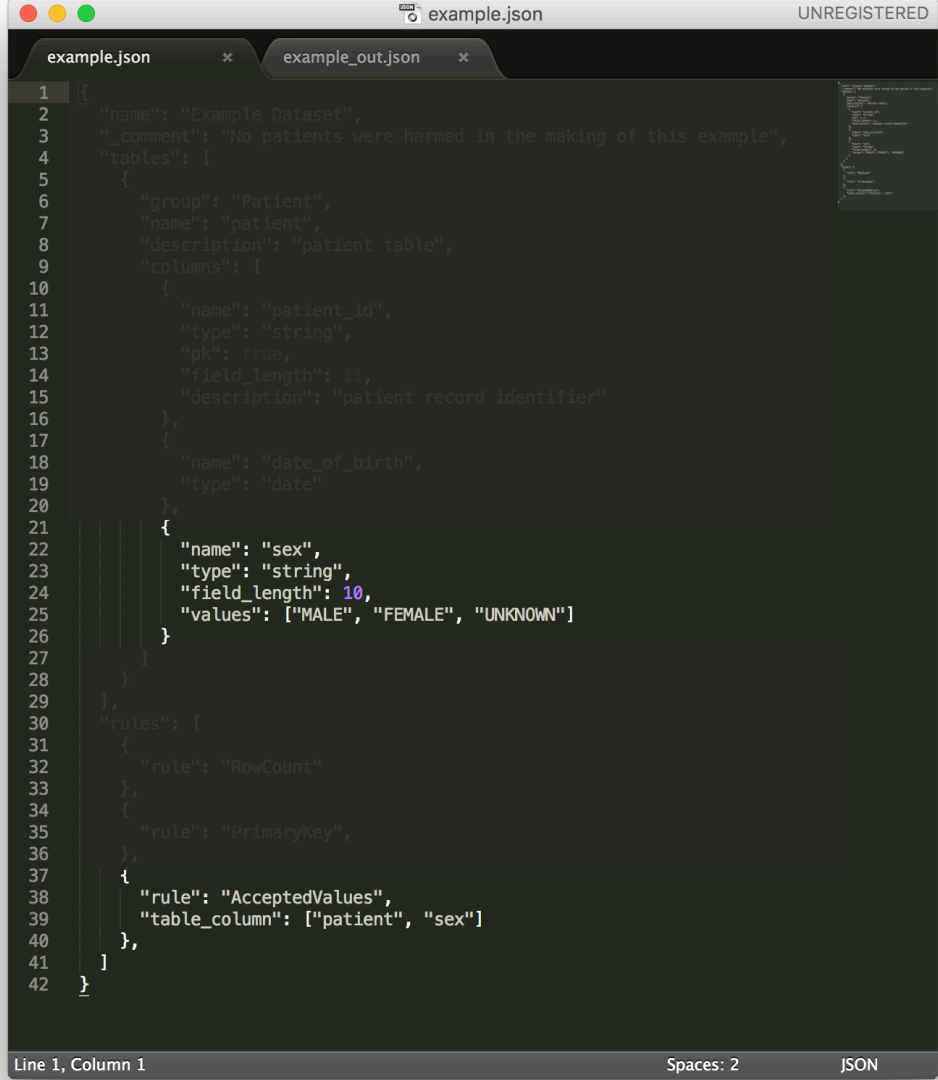
QUALITY

- Column properties trigger rules
- Specify table/column to execute rule for



QUALITY

- Column properties trigger rules
- Specify table/column to execute rule for



QUALITY

- Execution date
- Pass/fail for each rule
- Messages

example_out.json

UNREGISTERED

example.json

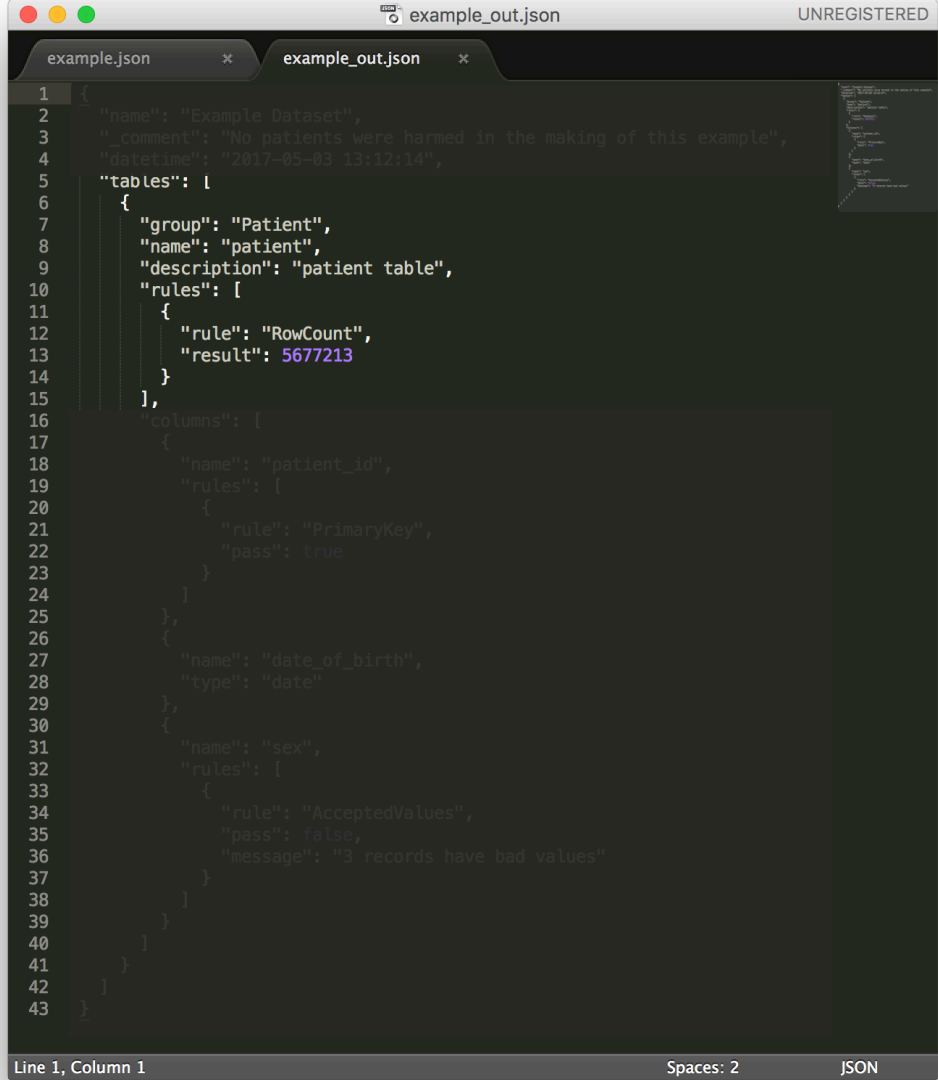
example_out.json

```
{
  "name": "Example Dataset",
  "_comment": "No patients were harmed in the making of this example",
  "datetime": "2017-05-03 13:12:14",
  "tables": [
    {
      "group": "Patient",
      "name": "patient",
      "description": "patient table",
      "rules": [
        {
          "rule": "RowCount",
          "result": 5677213
        }
      ],
      "columns": [
        {
          "name": "patient_id",
          "rules": [
            {
              "rule": "PrimaryKey",
              "pass": true
            }
          ]
        },
        {
          "name": "date_of_birth",
          "type": "date"
        },
        {
          "name": "sex",
          "rules": [
            {
              "rule": "AcceptedValues",
              "pass": false,
              "message": "3 records have bad values"
            }
          ]
        }
      ]
    }
  ]
}
```

Line 1, Column 1 Spaces: 2 JSON

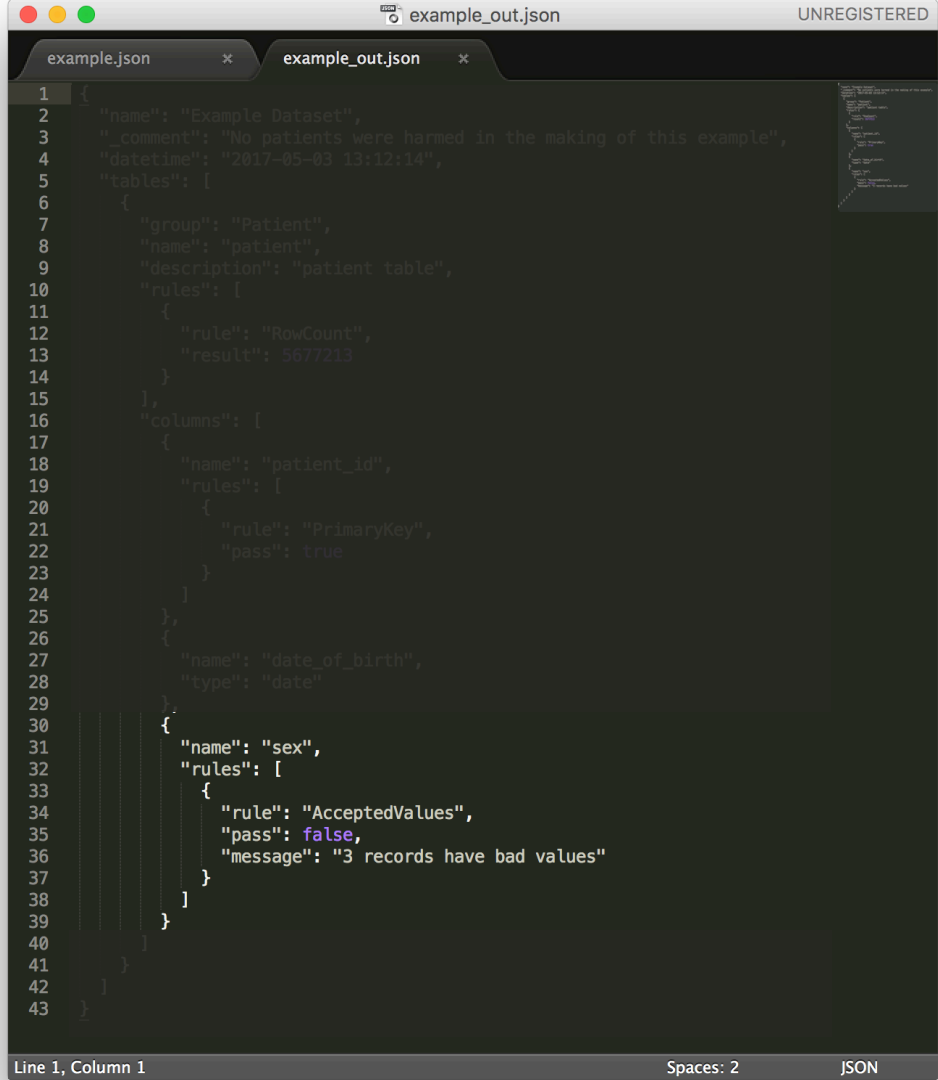
QUALITY

- Execution date
- Pass/fail for each rule
- Messages



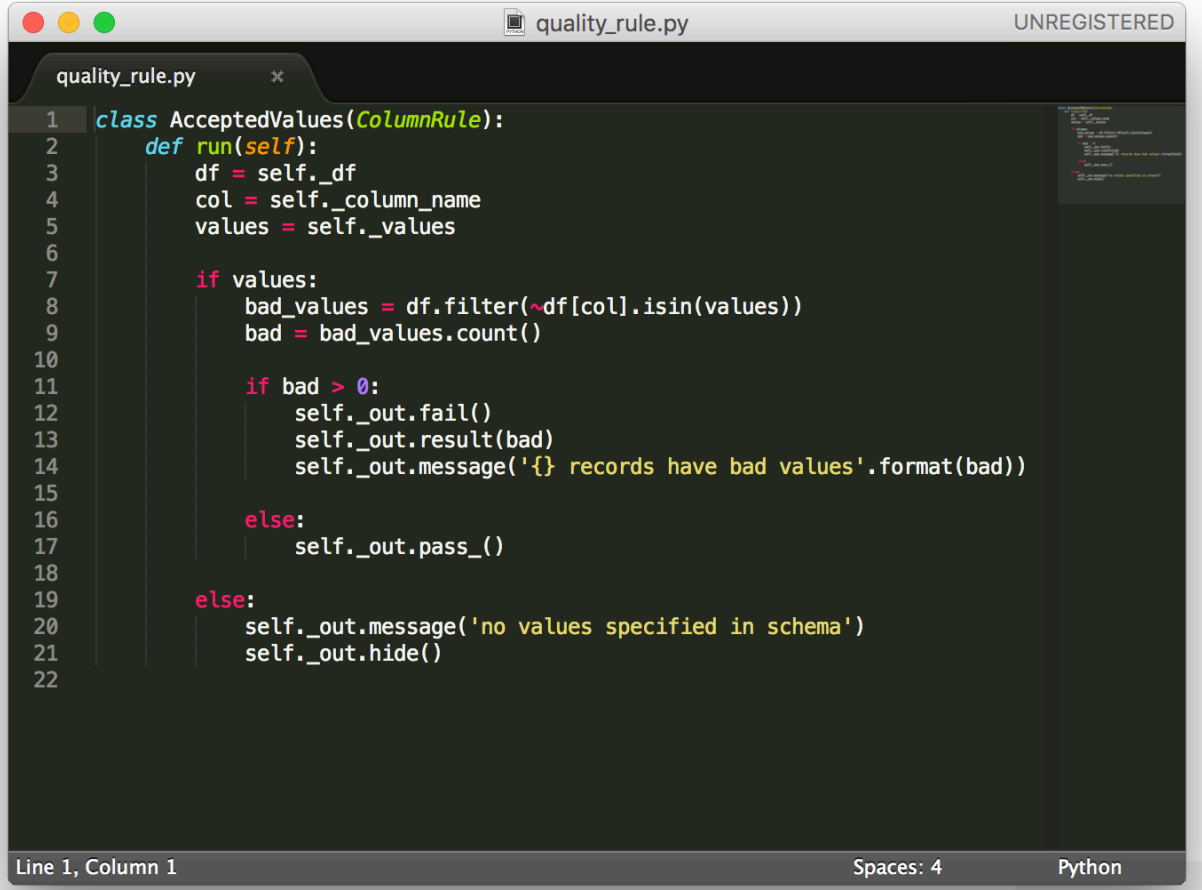
QUALITY

- Execution date
- Pass/fail for each rule
- Messages

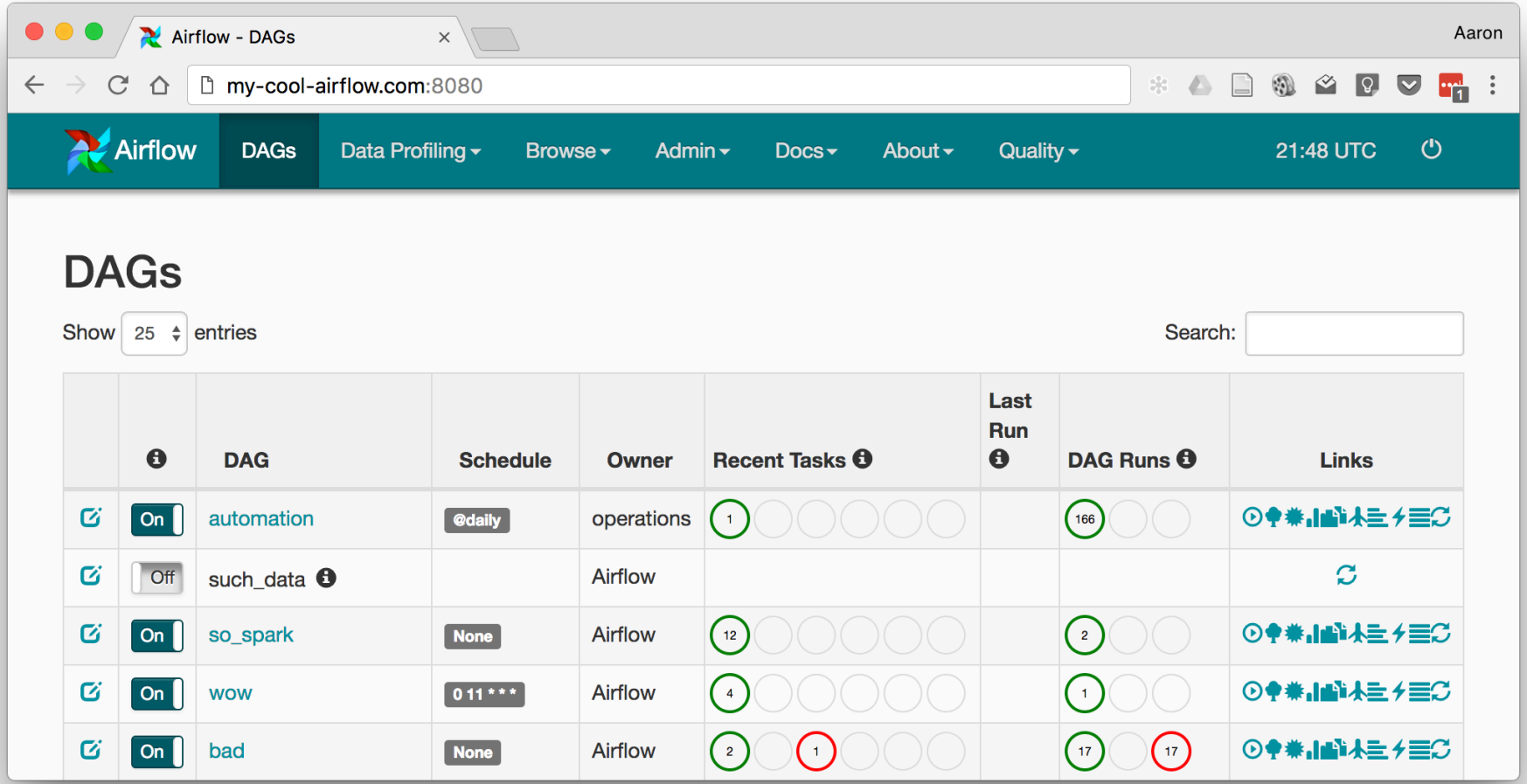


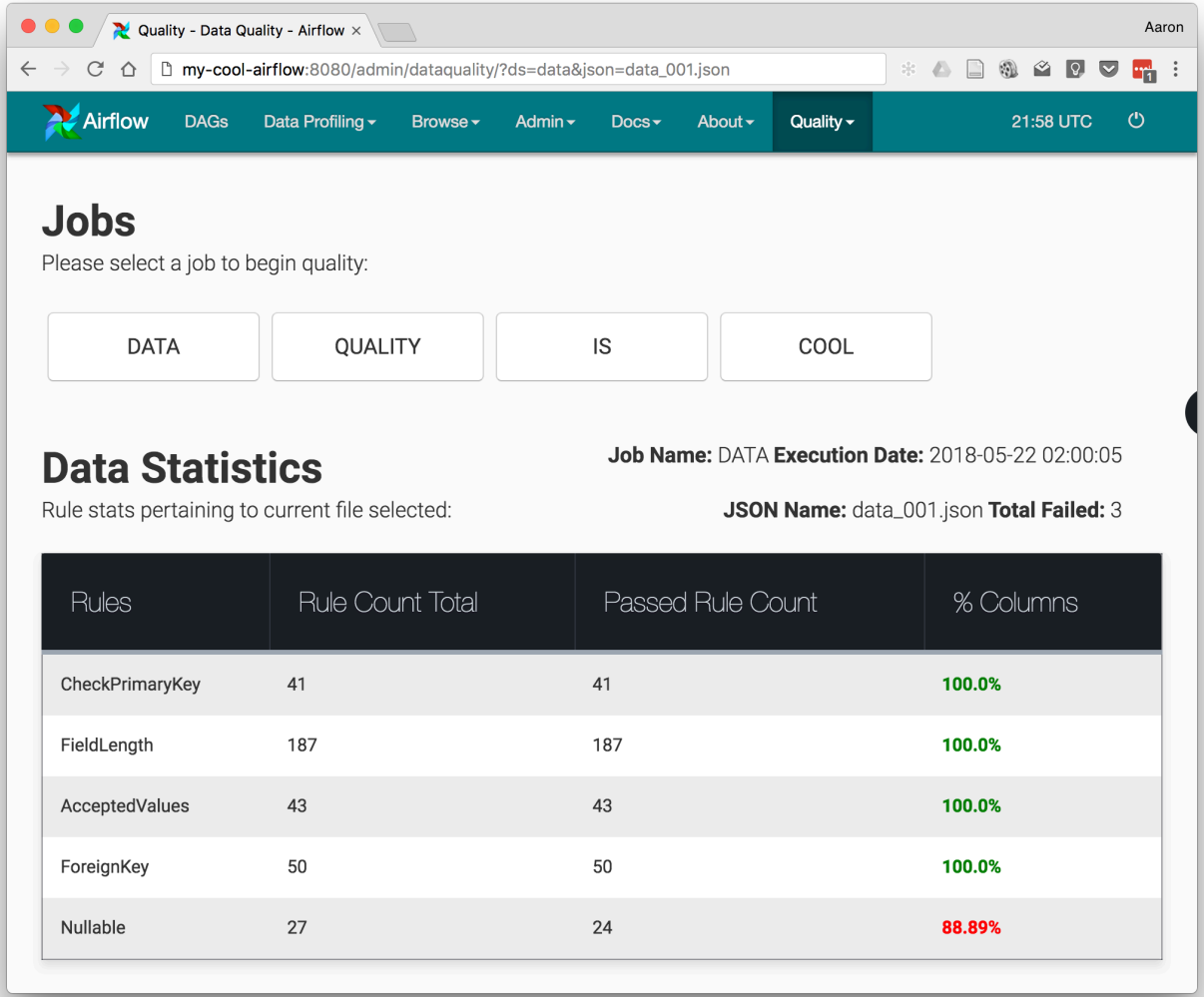
QUALITY

- Dataset rules
- Table rules
- Inter-column rules
- Column rules




PRODUCTION
LET THE AIRFLOW


Airflow - DAGs
Aaron
my-cool-airflow.com:8080
Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
Quality
21:48 UTC
DAGs
Show 25 entries
Search:
DAG
Schedule
Owner
Recent Tasks
Last Run
DAG Runs
Links
On automation @daily operations 1 166
Off such_data Airflow
On so_spark None Airflow 12 2
On wow 0 11 * * * Airflow 4 1
On bad None Airflow 2 1 17 17


Quality - Data Quality - Airflow
Aaron
my-cool-airflow:8080/admin/dataquality/?ds=data&json=data_001.json
Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
Quality
21:58 UTC
Jobs
Please select a job to begin quality:
DATA
QUALITY
IS
COOL
Data Statistics
Rule stats pertaining to current file selected:
Job Name: DATA Execution Date: 2018-05-22 02:00:05
JSON Name: data_001.json Total Failed: 3
Rules
Rule Count Total
Passed Rule Count
% Columns
CheckPrimaryKey
41
41
100.0%
FieldLength
187
187
100.0%
AcceptedValues
43
43
100.0%
ForeignKey
50
50
100.0%
Nullable
27
24
88.89%

CONTRIBUTORS

Aaron Richter
Data Scientist

Michael Crawford
Data Engineer

Eric Golinko
Data Scientist

Kevin Da Silva
Data Engineering Intern

THANK YOU!

AARON RICHTER

Data Scientist

PhD Candidate

@rikturr

MODERNIZING MEDICINE

Boca Raton, FL

We're hiring!

@modmed