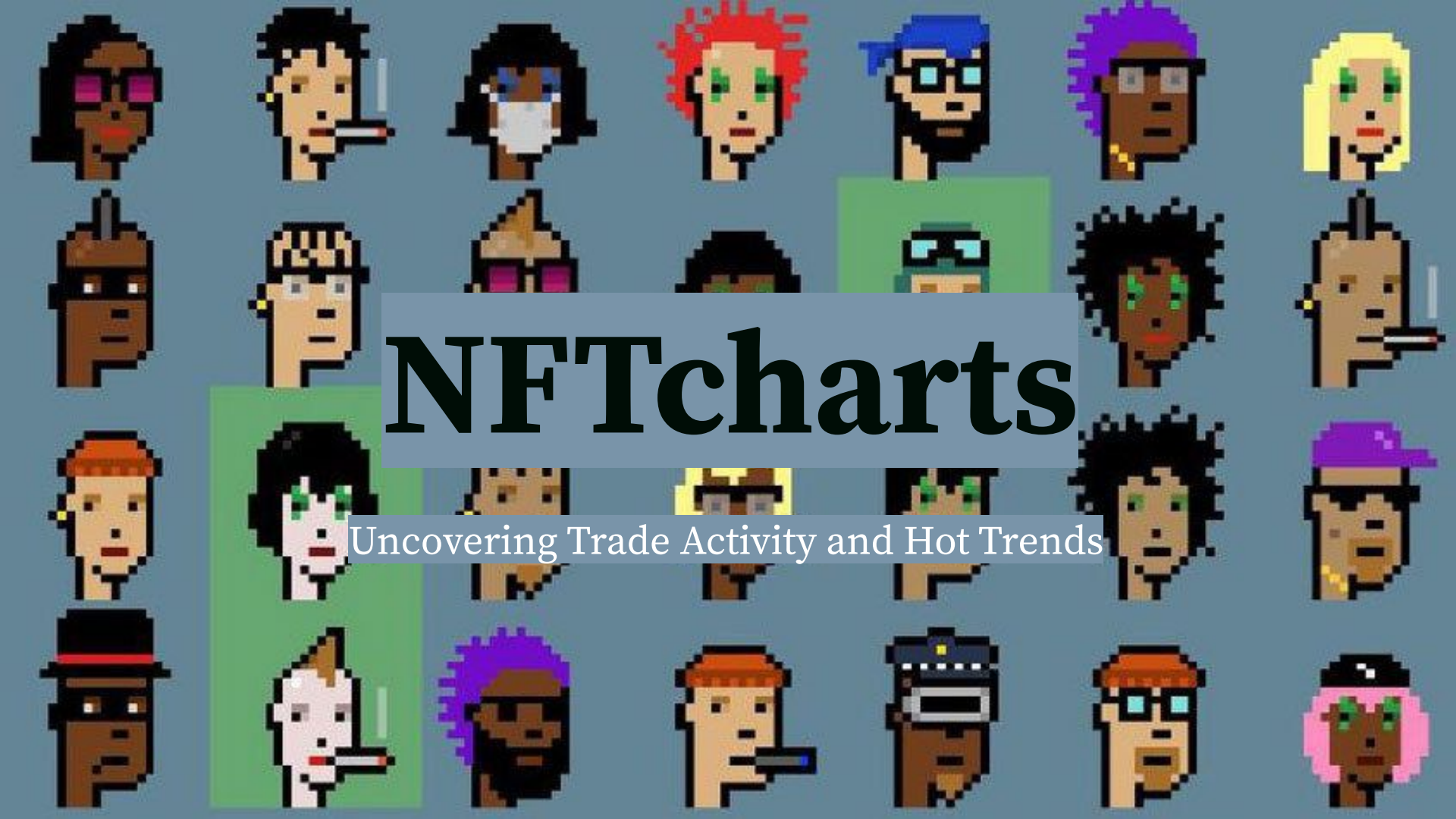

NFTcharts
Uncovering Trade Activity and Hot Trends

Table of Contents

Project Outline

What is an NFT?

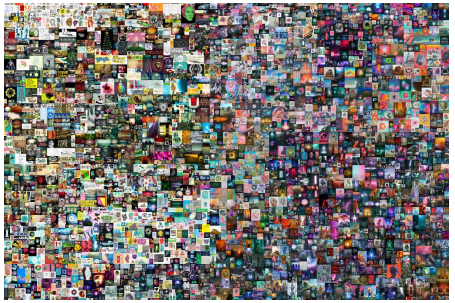
A non-fungible token is a unique and non-interchangeable unit of data stored on a blockchain, a form of digital ledger.

Can be in the form of photos, videos, and audio, etc.

Fun Facts

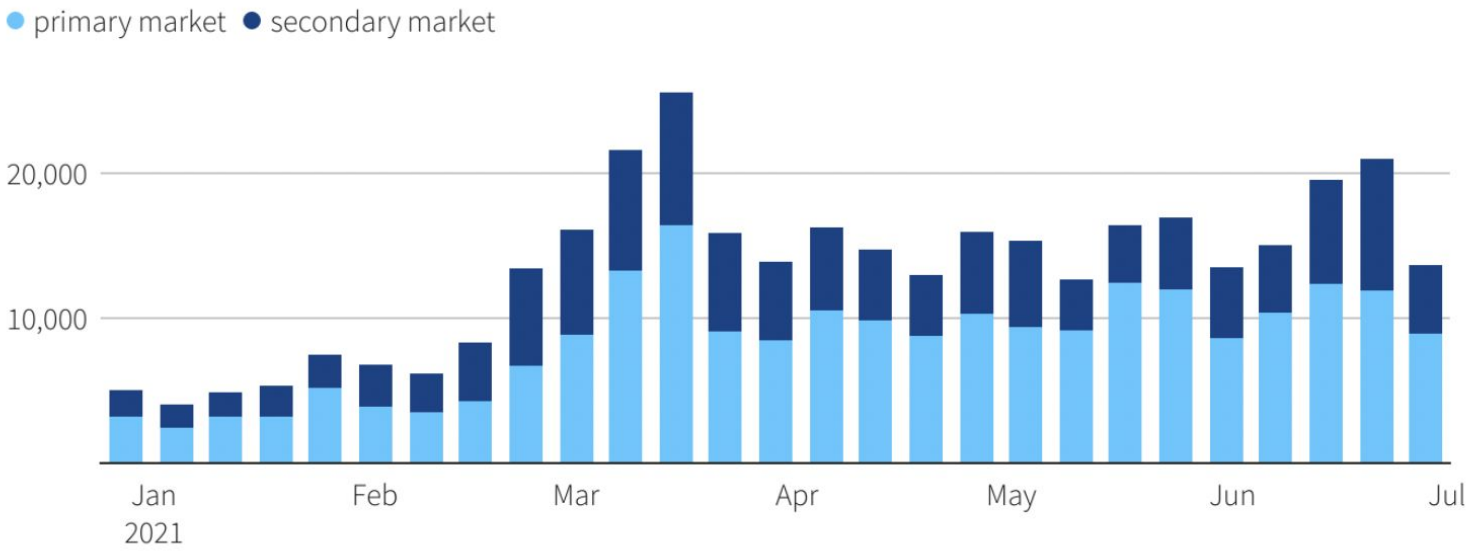
- The most expensive NFT ever sold was “The First 5000 Days” by Beeple for $69.3 million – making it the most expensive virtual item ever traded on any platform in history.

- There are $10-$20 million worth of NFT sold in the blockchain every week.



Weekly NFT buyers - NonFungible.com

Number of buyers on non-fungible tokens on the ethereum blockchain per week



Note: Data only shows sales on the ethereum blockchain, which is used for the majority of NFT sales. Data does not include sales which took place "off-chain".

Source: NonFungible.com

Project Goals and Objectives

- **Uncover** the latest and hottest NFT collections
- **Provide insights** into which NFT was recently traded and its activity history

Problem Space

- **Overwhelming** number of NFT projects in the market
- **Unable to keep track** of which collections are gaining popularity (aka “emerging”)
- **Unable to monitor** characteristics or trends of recent sales for emerging collections

My motivation

- This is 100% a passion project
- Passionate about optimising life by generating automatic pull feeds
- Want to monitor NFT activity without spending too much time on it

Implementation

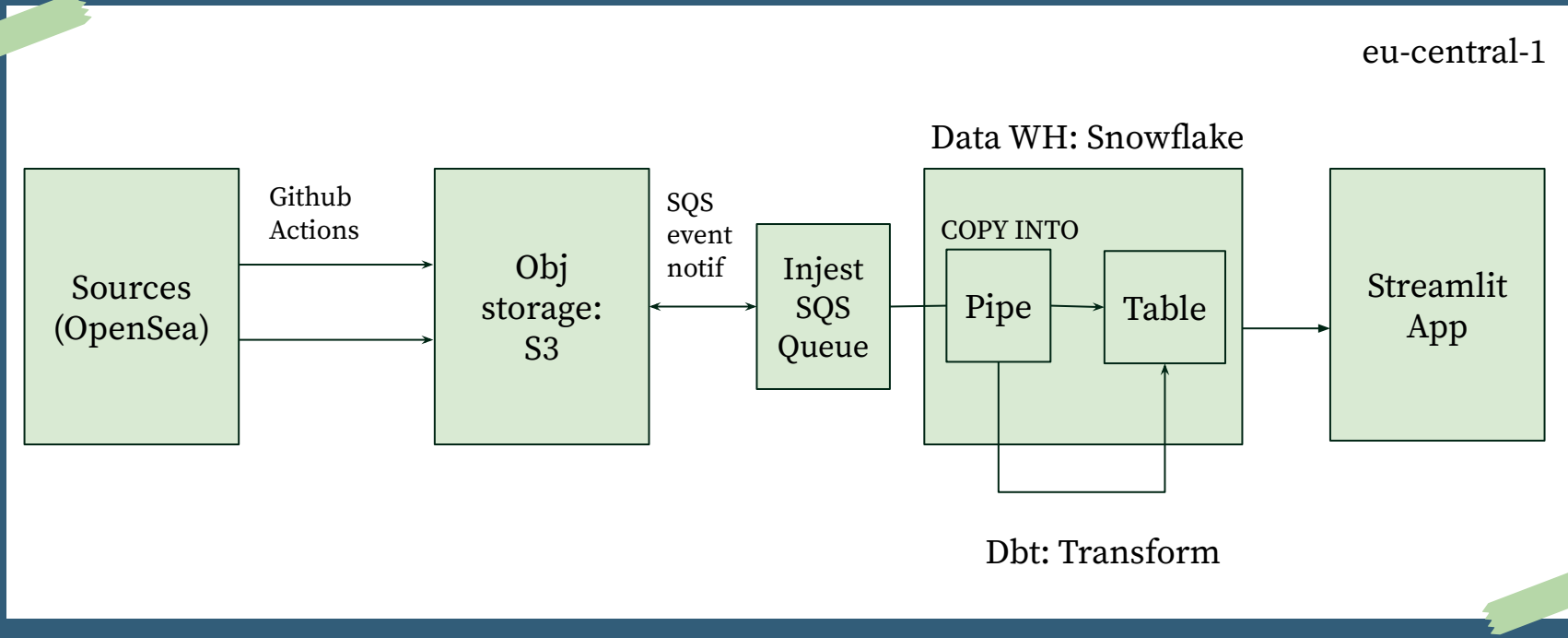
Personal Tech Objectives

- Try as many tools as possible 🤗
- Determined to try **snowflake** + **ELT**
- Wanted some implementation of automated asynchronous queues
- Hoped for a **stable, visualisation** (grafana … mode … Streamlit!)
- Wanted to work with **json files** (not csv)
- Python #1 😵

Under Time Pressure: Use whatever works, cry later

Don’t do this

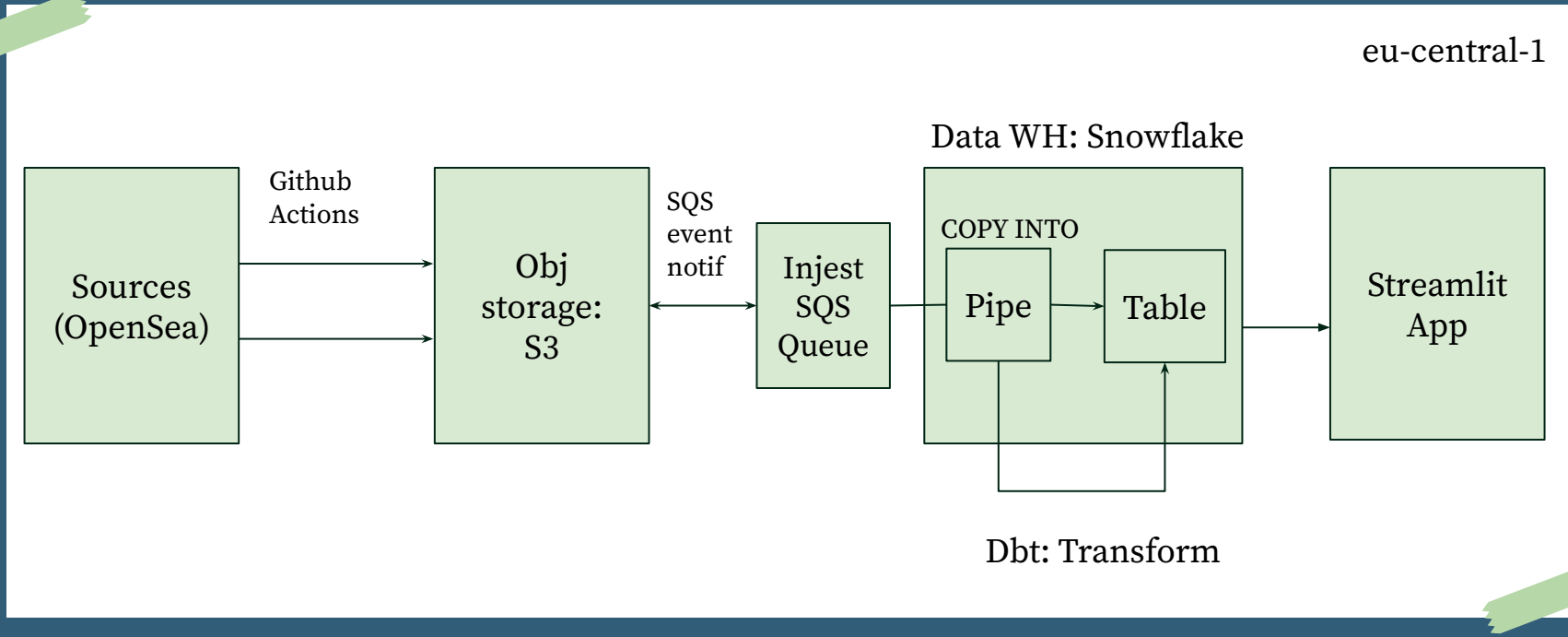
Architecture Diagram



Requirements.txt

```
beautifulsoup4==4.10.0
boto3==1.20.23
botocore==1.23.23
cloudscraper==1.2.58
pandas==1.3.3
Pillow==8.4.0
plotly==5.5.0
requests==2.26.0
snowflake==0.0.3
snowflake_connector_python==2.6.2
streamlit==1.0.0
```

My Approach: Sequence of Implementation



https://share.streamlit.io/sni-c/final-project :(

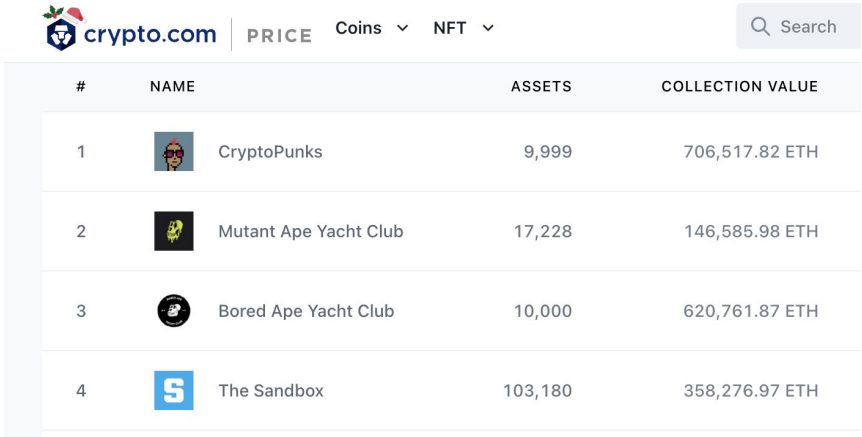
Your app is in the oven

Currently down because... virtualenv python 3.9 mac something not compatible with snowflake-connector something something

Tech Dive

ELT: Scraping + Github Actions

1. Scrape Top 10 Collections: Crypto.com
2. Pull detailed collection data from OpenSea

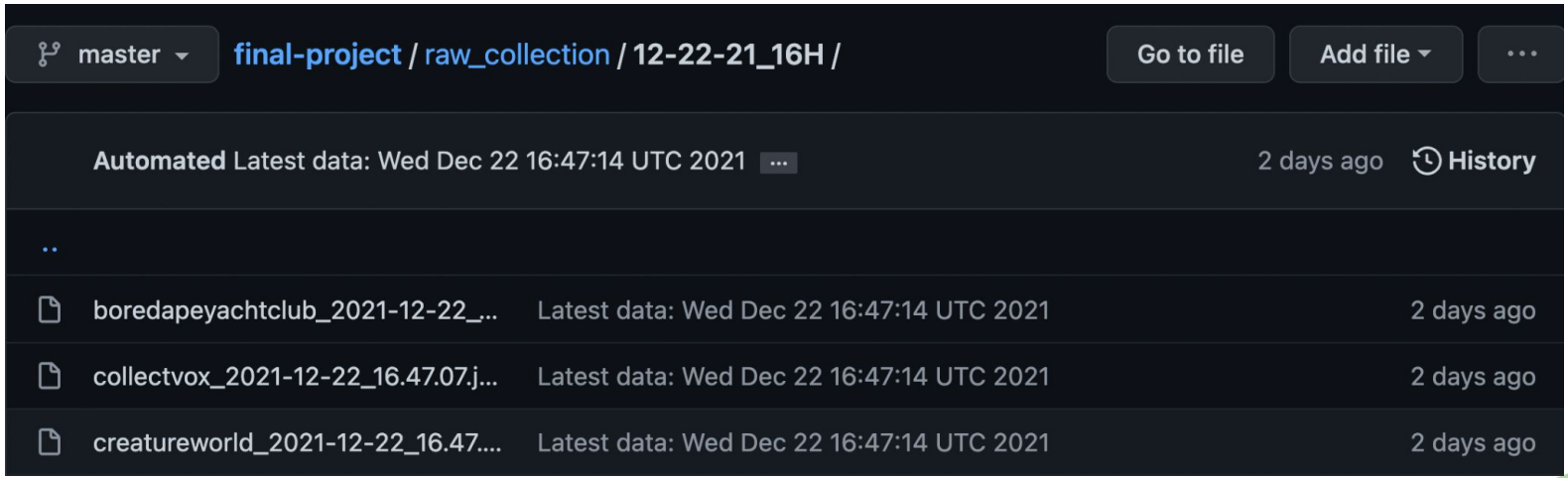


Collection: 1x per 24 hour cron job

Sales: 1x per 15 min cron job

ELT: Scraping + Github Actions

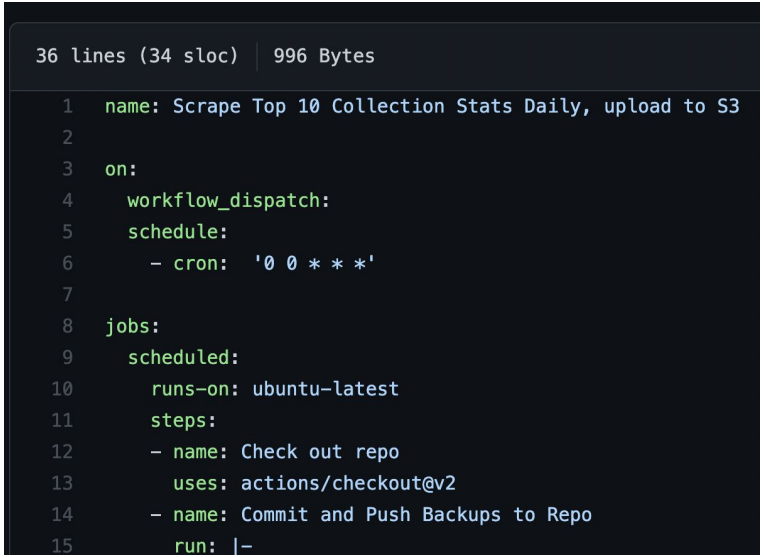
Decisions made Folder hierarchy: /json → /date and hour → /json file



ELT: Scraping + Github Actions

Decisions made

- Upload files to s3 first > then upload to Github as backup
- `git pull` before `git add`



36 lines (34 sloc) | 996 Bytes

```yaml
name: Scrape Top 10 Collection Stats Daily, upload to S3

on:
  workflow_dispatch:
  schedule:
    - cron:  '0 0 * * *'

jobs:
  scheduled:
    runs-on: ubuntu-latest
    steps:
    - name: Check out repo
      uses: actions/checkout@v2
    - name: Commit and Push Backups to Repo
      run: |-
```

ELT: Snowflake + Snowpipe

Load raw data with duplicates 🤢

Results | Data Preview | Open Histor

Query ID | SQL | 903ms | 9,600 rows

Filter result... | Copy | Columns

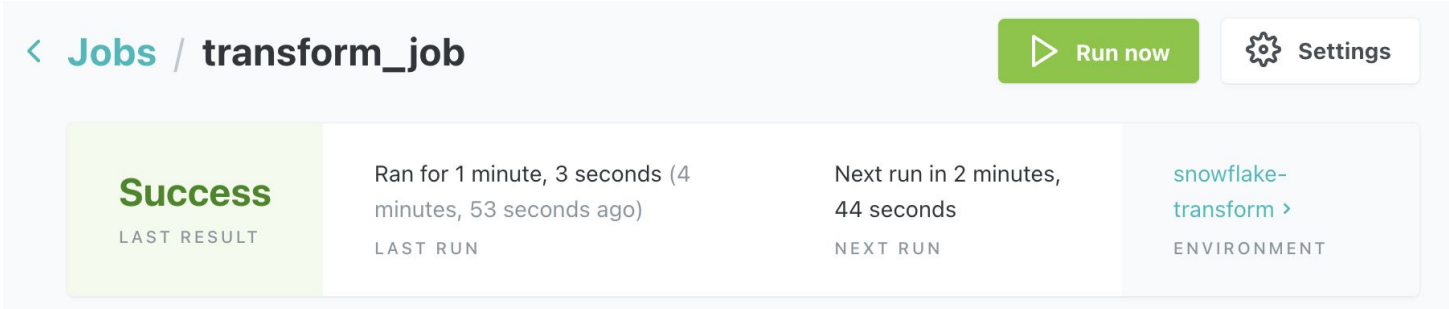
Row	RECENTSALE	CURRENT_TIMESTAMP
1	{ "collectionslug": "collectvox", "create_time": "2021-12-22_17....	2021-12-22 17:40:40.868 +0000
2	{ "collectionslug": "collectvox", "create_time": "2021-12-22_17....	2021-12-22 17:40:40.868 +0000

ELT: Snowflake + Snowpipe

1. Create Snowflake external stage that references your S3 bucket.
2. **An S3 event notification** informs Snowpipe via an SQS queue that new files have been created and ready to load. Snowpipe copies files into a queue.
3. **A Snowflake-provided virtual warehouse** loads data from the queued files into the target table based on parameters defined in the **pipe.**

https://docs.snowflake.com/en/user-guide/data-load-snowpipe-auto-s3.html

ELT: Materialised views with DBT



1 x 10 min cron job

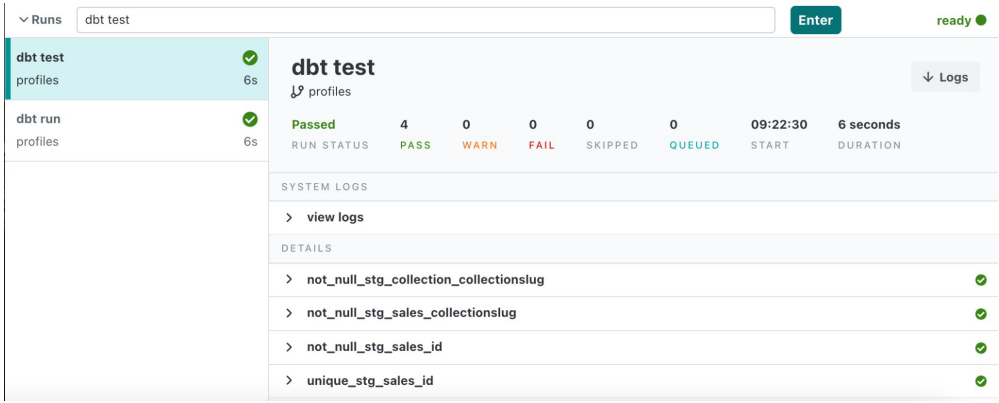
https://www.startdataengineering.com/post/build-a-simple-data-engineering-platform/

ELT: Materialised views with DBT

```yaml
version: 2
models:
  - name: stg_collection
    columns:
      - name: collectionslug
        tests:
          - not_null

  - name: stg_sales
    columns:
      - name: collectionslug
        tests:
          - not_null
      - name: id
        tests:
          - unique
          - not_null
```

dbt .yml file “tests” are helpful



ELT: Materialised views with DBT

Materialised view!

Results Data Preview

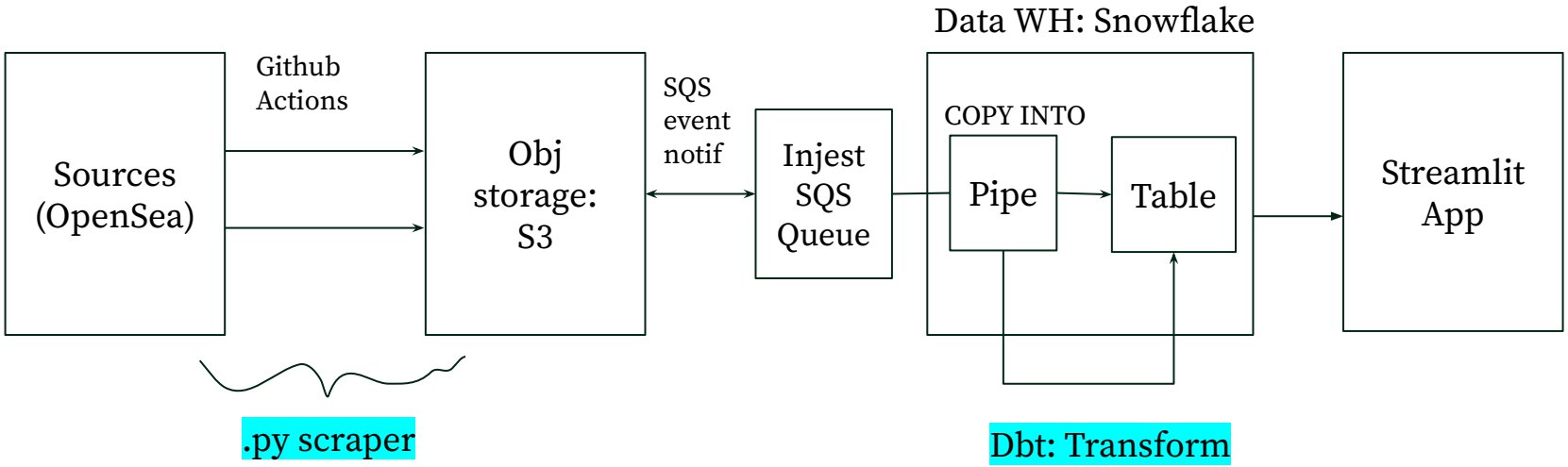
Query ID SQL 484ms 1,110 rows

Filter result... Copy

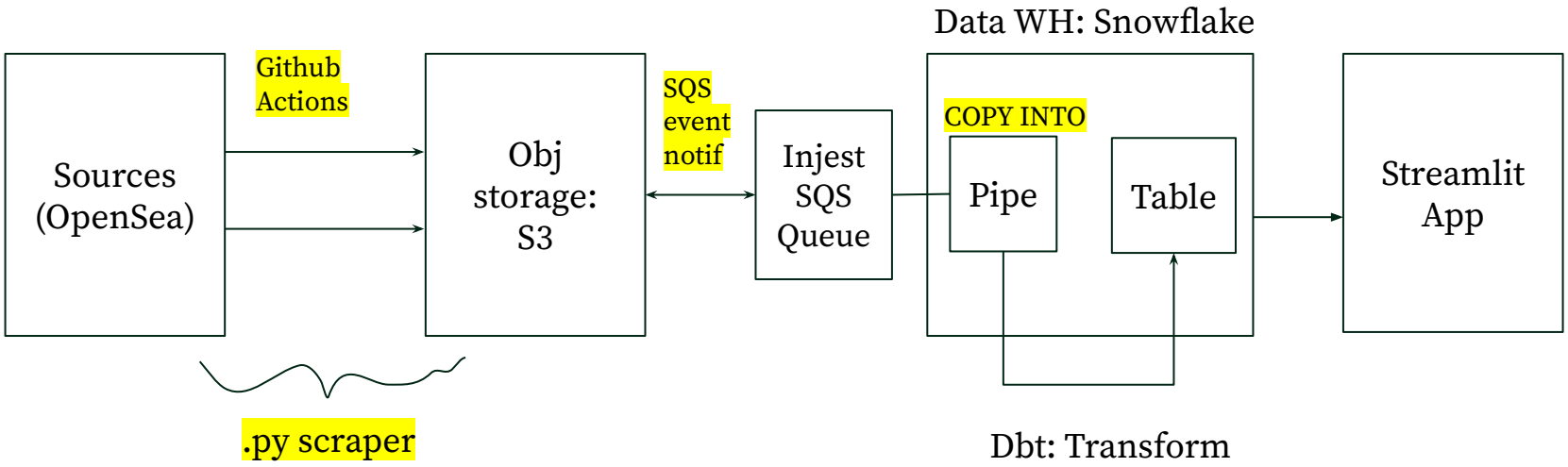
Row	CREATE_TIME	ID	URL	COLLECTIONSL	NAME	EVENT_TIMESTA	ETH_PRICE
1	2021-12-24_...	36112740	https://open...	collectvox	Rancher VO...	2021-12-23 ...	0.00012279...
2	2021-12-23_...	162554867	https://open...	punks-comic	PUNKS 2: X ...	2021-12-23 ...	0.00025302...
3	2021-12-22_...	31557874	https://open...	sandbox	LAND (43, 1...	2021-12-22 ...	0.00129351...
4	2021-12-23_...	150714021	https://open...	sandbox	LAND (-1, -1...	2021-12-23 ...	0.00129726...
5	2021-12-24	18216908	https://open	sandbox	LAND (-177	2021-12-24	0.00149647

Some thoughts that I have not found the answer to

(1) CD: Modifying code to pull the correct price field



(2) What if I wanted to pull from more sources?



Summary

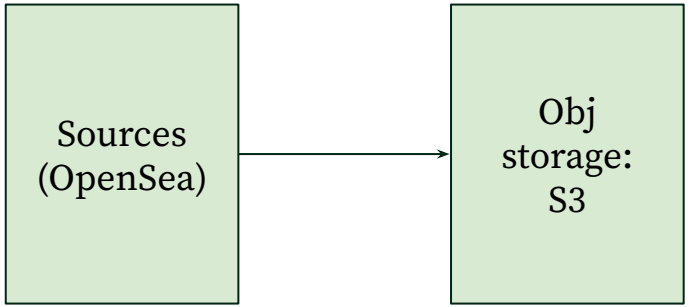
Handling edge cases

- Make sure scraper always returns a collection, len 10

```
def scraperreturn(results):
    assert(len(results) == 10)
```

Handling edge cases

- Check source/destination



Handling edge cases

- Make sure an image url exists

```
"image_url": "https://www.larvalabs.com/cryptop
"image_preview_url": "https://www.larvalabs.com
"image_thumbnail_url": null,
"image_original_url": null,
```

Limitations

- Only able to pull up to 50 recent sales per collection

  → need to get creative around free APIs

- My implementation is heavily reliant on 3rd party tools

  → how would I do it differently next time?

Limitations

- 15 min cron job x 10 json files = Already reached 85% of AWS S3 Free Tier Limit → opportunity for compression?

Dear AWS Customer,

Your AWS account 012842063955 has exceeded 85% of the usage limit for one or more AWS Free Tier-eligible services for the month of December.

Product	AWS Free Tier Usage as of 12/23/2021	Usage Limit	AWS Free Tier Usage Limit
AmazonS3	1805.0 Requests	2000.0 Requests	2,000 Put, Copy, Post or List Requests of Amazon S3

What's next?

Scrape more sources (build core engine)

Better ranking and presentation

Improve materialized view schema

Application beyond NFTs?

If I had more time, I would...

- Create a setup for collaboration

  → Implement some Dev / Prod separation

- Create an environment for production “real world” development

  → Learn CI / CD better

- Reliability: Concerns with potential data loss

  → count rows in Snowflake vs S3?

If I had more time, I would...

- Explore each tool in greater depth!

Thanks!
Any questions?