CHECKPOINT WRITEUP

Dataset Selection

The dataset selected was Reddit data from February 2023 across the top 40,000 subreddits available online for download. We selected Reddit data because of our interests in exploring machine learning data pipelining. We felt Reddit data captures more complexity in language than Yelp data and wanted to explore this further. There are TBs of free Reddit data available for training and building NLP models as well as an API that can later be used for expanding our pipeline to include real time data additions to get us close to streaming.

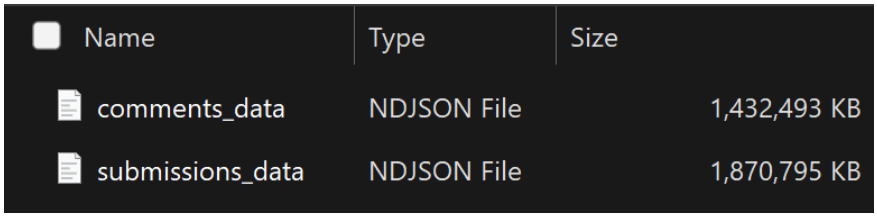
We loaded roughly 2-3GB of data by process of torrent download of compressed data and then unzipping into NDJSONs. Due to the size of the entire dataset available to us, we needed to select a subset for download and were unable to load the entire set and sample. Further, we found that some of the subreddits were NSFW, so we selected subreddits we believed were more appropriate, such as the "stocks" and "psychology" subreddits. The data comes organized into submissions and comments. Below are screenshots showing the description of the data loaded into a Spark DF. Record counts combined show at least 1M records and KB sizes combined are >1GB. The data is available at: https://academictorrents.com/details/56aa49f9653ba545f48df2e33679f014d2829c10

```
submissions_df.describe().show()
✓ 27.5s

[Stage 7:>                                                          (0 + 1) / 1]
+-------+------------------+----------+--------------------+--------------------+-------------------------+--------------------+-----------+
|summary|approved_at_utc|approved_by|              author| author_created_utc|author_flair_background_color|author_flair_css_class|author_flai
+-------+------------------+----------+--------------------+--------------------+-------------------------+--------------------+-----------+
|  count|                 0|         0|              839181|              322298|                    99337|              143862|
|   mean|              NULL|      NULL|1.357251799493590...|1.546892245241075E9|                     NULL|                NULL|
| stddev|              NULL|      NULL|2.694542489993743...|8.740386717158881E7|                     NULL|                NULL|
|    min|              NULL|      NULL|--------Username---|          1122523200|                         |                    |    0349646
|    max|              NULL|      NULL|            zzzzoooo|          1675264632|              transparent|            firefoxg|    f8d94ce
+-------+------------------+----------+--------------------+--------------------+-------------------------+--------------------+-----------+


comments_df.describe().show()
✓ 17.8s

[Stage 10:=========================================>             (3 + 1) / 4]
+-------+------------------+----------+----------+----------------+--------------------+--------------------+-------------------------+--------------------+
|summary|approved_at_utc|approved_by|associated_award|              author| author_created_utc|author_flair_background_color|author_flair_css_class|
+-------+------------------+----------+----------+----------------+--------------------+--------------------+-------------------------+--------------------+
|  count|                 0|         0|         0|         1000833|              382241|                90182|                     5302|
|   mean|              NULL|      NULL|      NULL|        Infinity|1.4801933343165333E9|                NULL|                     NULL|
| stddev|              NULL|      NULL|      NULL|             NaN|1.1858325709169938E8|                NULL|                     NULL|
|    min|              NULL|      NULL|      NULL|             ---|          1122523200|                    |                         |
|    max|              NULL|      NULL|      NULL|zzzztopportal|          1680272913|              transparent|            firefoxg|
+-------+------------------+----------+----------+----------------+--------------------+--------------------+-------------------------+--------------------+
```



Dataset Structure

Comments Table Schema:

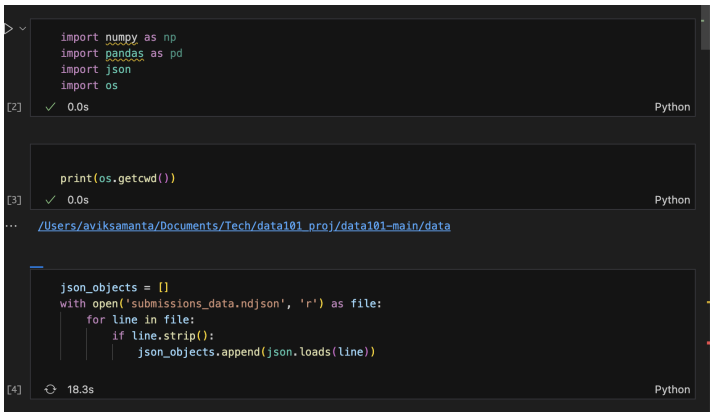
```
                        Table "public.comments"
        Column          |          Type          | Collation | Nullable | Default
------------------------+------------------------+-----------+----------+---------
 id                     | character varying(255) |           | not null |
 archived               | boolean                |           |          |
 author                 | character varying(255) |           |          |
 author_created_utc     | bigint                 |           |          |
 body                   | text                   |           |          |
 created_utc            | bigint                 |           |          |
 downs                  | integer                |           |          |
 edited                 | boolean                |           |          |
 locked                 | boolean                |           |          |
 parent_id              | character varying(255) |           |          |
 permalink              | character varying(255) |           |          |
 retrieved_on           | bigint                 |           |          |
 score                  | integer                |           |          |
 subreddit              | character varying(255) |           |          |
 subreddit_id           | character varying(255) |           |          |
 subreddit_name_prefixed | character varying(255) |           |          |
 subreddit_type         | character varying(255) |           |          |
 updated_on             | bigint                 |           |          |
 ups                    | integer                |           |          |
Indexes:
```

Submissions Table Schema:

```
                    Table "public.submissions"
        Column          |  Type   | Collation | Nullable | Default
------------------------+---------+-----------+----------+---------
 id                     | text    |           | not null |
 downs                  | integer |           |          |
 ups                    | integer |           |          |
 archived               | boolean |           |          |
 author                 | text    |           |          |
 author_created_utc     | integer |           |          |
 subreddit              | text    |           |          |
 subreddit_id           | text    |           |          |
 subreddit_subscribers  | integer |           |          |
 subreddit_type         | text    |           |          |
 title                  | text    |           |          |
 url                    | text    |           |          |
 num_comments           | integer |           |          |
 edited                 | boolean |           |          |
 permalink              | text    |           |          |
 is_self                | boolean |           |          |
 selftext               | text    |           |          |
 created_utc            | integer |           |          |
 spoiler                | boolean |           |          |
```

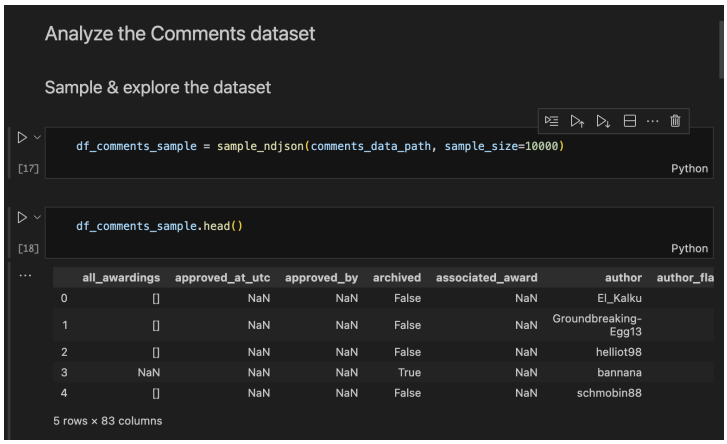
Database Setup

We first extracted all the JSON objects from both json files with python, submissions.ndjson and comments.ndjson. The objects were not originally stored in a json format so we had to process each object line by line in the file:



We then loaded each JSON object into lists for both comments and posts.

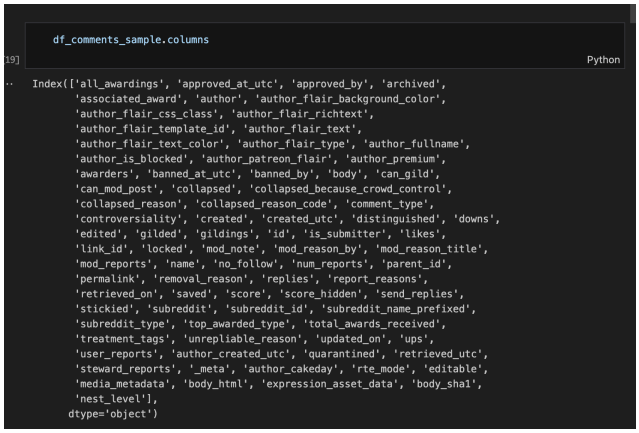
From here we load a sample of the ndJSON file into a pandas dataframe to see a small subset of the data:



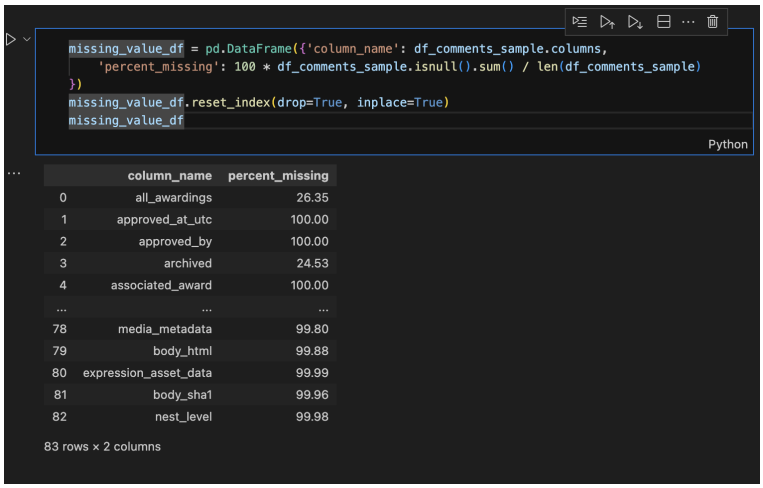
	all_awardings	approved_at_utc	approved_by	archived	associated_award	author	author_fla
0	[]	NaN	NaN	False	NaN	El_Kalku	
1	[]	NaN	NaN	False	NaN	Groundbreaking-Egg13	
2	[]	NaN	NaN	False	NaN	helliot98	
3	NaN	NaN	NaN	True	NaN	bannana	
4	[]	NaN	NaN	False	NaN	schmobin88	

5 rows × 83 columns

We then take a look at the comment columns below:



Usually in datasets pulled from online, there tends to be a lot of missing values, so we do a check to see if there are any missing values:

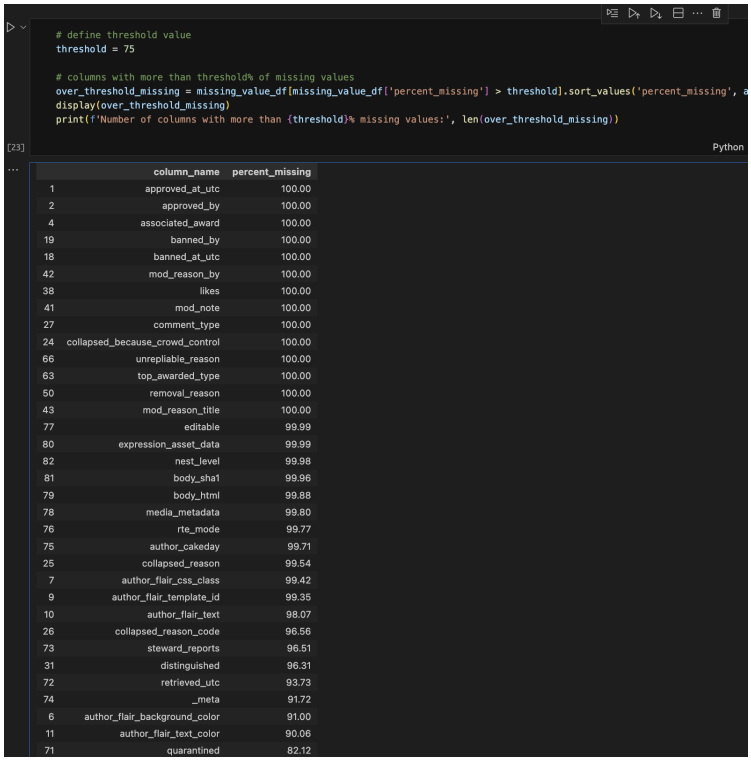


```
missing_value_df = pd.DataFrame({'column_name': df_comments_sample.columns,
    'percent_missing': 100 * df_comments_sample.isnull().sum() / len(df_comments_sample)
})
missing_value_df.reset_index(drop=True, inplace=True)
missing_value_df
```

Python

	column_name	percent_missing
0	all_awardings	26.35
1	approved_at_utc	100.00
2	approved_by	100.00
3	archived	24.53
4	associated_award	100.00
...	...	...
78	media_metadata	99.80
79	body_html	99.88
80	expression_asset_data	99.99
81	body_sha1	99.96
82	nest_level	99.98

83 rows × 2 columns



```
# define threshold value
threshold = 75

# columns with more than threshold% of missing values
over_threshold_missing = missing_value_df[missing_value_df['percent_missing'] > threshold].sort_values('percent_missing', a
display(over_threshold_missing)
print(f'Number of columns with more than {threshold}% missing values:', len(over_threshold_missing))
```

[23] Python

	column_name	percent_missing
1	approved_at_utc	100.00
2	approved_by	100.00
4	associated_award	100.00
19	banned_by	100.00
18	banned_at_utc	100.00
42	mod_reason_by	100.00
38	likes	100.00
41	mod_note	100.00
27	comment_type	100.00
24	collapsed_because_crowd_control	100.00
66	unrepliable_reason	100.00
63	top_awarded_type	100.00
50	removal_reason	100.00
43	mod_reason_title	100.00
77	editable	99.99
80	expression_asset_data	99.99
82	nest_level	99.98
81	body_sha1	99.96
79	body_html	99.88
78	media_metadata	99.80
76	rte_mode	99.77
75	author_cakeday	99.71
25	collapsed_reason	99.54
7	author_flair_css_class	99.42
9	author_flair_template_id	99.35
10	author_flair_text	98.07
26	collapsed_reason_code	96.56
73	steward_reports	96.51
31	distinguished	96.31
72	retrieved_utc	93.73
74	_meta	91.72
6	author_flair_background_color	91.00
11	author_flair_text_color	90.06
71	quarantined	82.12

We only upload columns that have fewer than 75% missing values:

```
# only include columns that have fewer than threshold% missing values
df_comments_sample = df_comments_sample.iloc[:, missing_value_df[missing_value_df['percent_missing'] <= threshold].index]
df_comments_sample
```

Python

	all_awardings	archived	author	author_flair_richtext	author_flair_type	author_fullname	author_is_blocked	author_pat
0	[]	False	El_Kalku	[]	text	t2_6gk5ry84	False	
1	[]	False	Groundbreaking-Egg13	[]	text	t2_5vm4hgbg	NaN	
2	[]	False	helliot98	[]	text	t2_hff1s	NaN	
3	NaN	True	bannana	NaN	NaN	NaN	NaN	
4	[]	False	schmobin88	[]	text	t2_4909hth9	False	
...	...	...	...	...	...	...	...	
9995	[]	NaN	CJP_UX	[]	text	t2_16taiq	NaN	
9996	NaN	False	One_Giant_Nostril	[]	text	t2_4adwv	NaN	
9997	[]	False	No-Bridge-7124	[]	text	t2_tj6apeoj	False	
9998	[]	NaN	mbmuenster	[]	text	t2_111r8rk	NaN	
9999	[]	False	AutoModerator	[]	text	t2_6l4z3	NaN	

10000 rows × 49 columns

Finally, above are the columns we choose to select for the comments

Fields Select

From the remaining columns, tentatively choose 20 most relevant columns to save to database. We can add or remove more down the line if necessary.

```
relevent_comment_columns = [
    'author',
    'author_created_utc',
    'body',
    'created_utc',
    'edited',
    'id',
    'locked',
    'parent_id',
    'permalink',
    'retrieved_on',
    'score',
    'subreddit',
    'subreddit_id',
    'subreddit_name_prefixed',
    'subreddit_type',
    'archived',
    'downs',
    'ups'
]

print('Number of relevant columns:', len(relevent_comment_columns))
```

The final comments table:

Construct new dataframe keeping only relevant columns

```
comments = df_comments_sample[relevent_comment_columns]
comments
```

Python

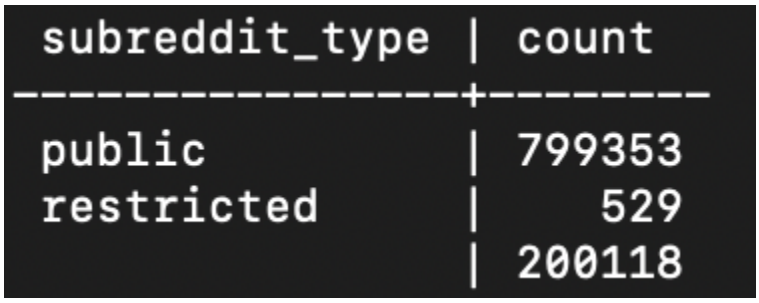
	author	author_created_utc	body	created_utc	edited	id	locked	parent_id	permalink	ret
0	El_Kalku	NaN	Thanks!!!	1686664935.0	False	jnztbiy	False	t1_jnzijlk	/r/ImaginaryTechnology/comments/14867ry/ariel_...	1.68
1	Groundbreaking-Egg13	1.596724e+09	Artbreeder? The way you speak makes me think i...	1668517861	False	iwgd97b	False	t1_iwg7cwf	/r/ArtificialInteligence/comments/yvqei5/help/...	1.6
2	helliot98	1.405513e+09	I have wanted to do this program here since I ...	1637163978	False	hkzz6z0	False	t3_qw1zj6	/r/cogsci/comments/qw1zj6/online_cognitive_sci...	1.64
3	bannana	NaN	Wow two pages of:\n\n*Well, maybe it's bad but...	1234478798	True	c07mvo3	NaN	t3_7wz2l	NaN	1.42
4	schmobin88	NaN	I looked at the conversation. I found it inte...	1680584704.0	False	jevtgoi	False	t3_12b3at0	/r/ArtificialInteligence/comments/12b3at0/i_ga...	1.68
...	...	...	...	...	...	...	...	...	...	
9995	CJP_UX	1.491404e+09	No requests for personal diagnosis	1565628037	False	ewow67c	False	t3_cpdnpm	/r/AcademicPsychology/comments/cpdnpm/how_to_g...	1.57
9996	One_Giant_Nostril	1.283044e+09	Sidebar: **Requests:** For payable requests, t...	1532277909	False	e2u6o8y	NaN	t3_90yviq	/r/ImaginaryTechnology/comments/90yviq/request...	1.53
9997	No-Bridge-7124	NaN	The spinning is a trip. Why does the mind do t...	1683855114.0	False	jjtgikh	False	t1_jhyj8b3	/r/NLP/comments/1304pqe/i_wonder_who_knows_rhe...	1.68
9998	mbmuenster	NaN	of course ;-)	1599235971	False	g40331g	False	t1_g4026ak	/r/NLP/comments/imigzf/looking_for_a_good_prim...	1.61
9999	AutoModerator	1.325741e+09	🚀🌑 -- Join our discord!! https://discord.gg/...	1660642245	False	ikhyeo1	False	t3_wppss5	/r/StocksAndTrading/comments/wppss5/interests_...	1.66

Database Upload

To handle the large .ndjson file, which exceeds 1 GB in size, we process the data in chunks of 1000 entries at a time. Each chunk is parsed into a Pandas DataFrame, allowing us to work with manageable portions of the dataset. After loading a chunk, we filter the data to retain only the relevant columns. Once the data is in the desired format, we use the DataFrame.to_sql() method to insert the entries into a PostgreSQL table. Due to the presence of duplicate id values in the dataset, likely caused by multiple retrievals of the same comment, we temporarily relax the PRIMARY KEY constraint on the comments.id column. This issue will be revisited later to implement a more robust solution.

Task Selection

SELECT COUNT(*) FROM comments GROUP BY subreddit_type



subreddit_type	count
public	799353
restricted	529
	200118

The above query aims to see the distribution of comments based on the subreddit types. The idea is to understand the submissions table better by seeing which subreddit types are more populated.

Future Plan

For the non-relational parts of the project we are interested in implementing Spark for distributed batch processing due to its ability to efficiently process large datasets in the TBs and its ease of use with other software tools used in ML pipelining, such as Kafka. Although our dataset is still a smaller dataset, Reddit has TBs of data available for use, so we see this as exploring this use case on a smaller scale. Data loaded into Spark can be processed similar to how processing is done on Pandas, through a DataFrame object so we are using that and our understanding of Spark's distributed processing structure to influence our comparison tasks.

Task comparisons include:

Data loading, preprocessing: comparing Spark with Postgres on efficiency and flexibility for loading and processing raw Reddit data

Simple queries involving lookups

Complex queries involving aggregation, joins

Setup/Scalability with dataset size

Visualizations and exploration

As a backup or extension to this, we are also considering loading MongoDB. MongoDB offers ease for storing semi-structured data, such as Reddit data straight from streaming to Kafka using the API. It also reduces the complexity needed for storage and making it easier to access key value data.

We aim to expand our SQL dataset with more tables representing additional entities across our entire reddit datapoints. These entities will include:

- **Users**
  - Users on reddit make the posts and comments that populate our dataset
- **Subreddits**
  - The individual forums housing multiple posts and columns, many users can subscribe to a single subreddit and make submissions. Subreddits can also have different types, including news, sports, games, etc.
- **Approval**
  - Each user can make a post, but for a post to be publicly available on a subreddit it must be approved by the subreddit's guidelines or moderators