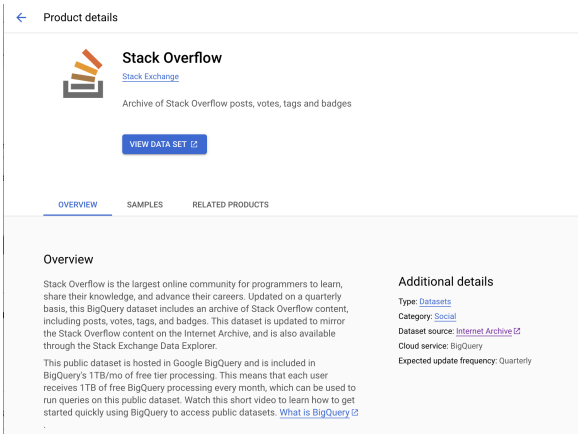
Stack Overflow User Analysis Using Machine Learning Technology

Members: Trishala Jayesh Ahalpara, Chen Lin, Daria Popova, Divya Nalam, Xiaoyi Wang

Dataset

The Stack Overflow BigQuery table is a large dataset that contains information from the Stack Overflow question and answer platform. This dataset is included in Bigquery's 1TB/mo of free tier processing. It includes user profiles, questions, answers, comments, and tags. The table enables analysis and insights into programming topics, trends, and community behavior. It offers a valuable resource for researchers, developers, and data analysts interested in exploring Stack Overflow's vast collection of programming knowledge.



Here are the key tables and their descriptions:

- users: Stack Overflow users' display names, reputation, creation date, and location are all listed in this table. Separate tables contain user-specific information such as badges and activity.
- tags: This table contains information about the tags assigned to Stack Overflow questions. It includes the tag's name, popularity (based on how many questions have the tag), and other information.
- votes: This table stores information about post votes, such as upvotes, downvotes, and favorites. It contains information like the voter's ID, the post ID, and the type of vote.
- comments: This table includes comments made on questions and answers. It contains the comment text, the creation date, and the post ID associated with it.
- posts_questions and posts_answers: These tables contain the Stack Overflow questions and answers, respectively. They contain details like the post title, body, creation date, score, and tags. To retrieve question-answer pairs, these tables can be joined using the id field.

- badges: This table contains information about the badges given to Stack Overflow users. It contains information such as the badge's name, description, and the user ID of the recipient.
- post_history: This table keeps track of the revision history of Stack Overflow posts, capturing changes to questions and answers over time. It contains details such as the post ID, revision type, and the user who made the change.
- post_links: This table contains information about the links included in Stack Overflow posts. It contains information such as the post ID, linked post ID, and link type.
- stackoverflow_posts: It includes an extensive collection of posts from the Stack Overflow platform. Each row in the table represents a single post, which could be a question or an answer.
- posts_moderator_nomination: This table contains information about Stack Overflow moderator nominations. It contains information about the nomination process, nominated users, nomination posts, timestamps, and other relevant metadata.
- posts_orphaned_tag_wiki: This table contains information about tag wikis that have been disconnected from the tags to which they belong. It contains the content of the orphaned tag wikis, as well as timestamps and other pertinent information.
- posts_privilege_wiki: This table contains information about Stack Overflow's privilege wikis. Privilege wikis detail the various privileges and levels of user reputation required to carry out specific actions on the site. The table contains the privilege wiki content, timestamps, and metadata.
- posts_tag_wiki_excerpt and posts_tag_wiki: These tables contain information about the tag wikis and Stack Overflow excerpts. Tag wikis provide in-depth information about specific tags, whereas tag excerpts provide concise summaries. The tables contain tag wiki content, excerpts, timestamps, and other pertinent information.
- posts_wiki_placeholder: This table is about Stack Overflow's wiki placeholders. These placeholders are placeholders for future wiki content that is being prepared or reviewed. The table contains metadata as well as information about the placeholder posts.

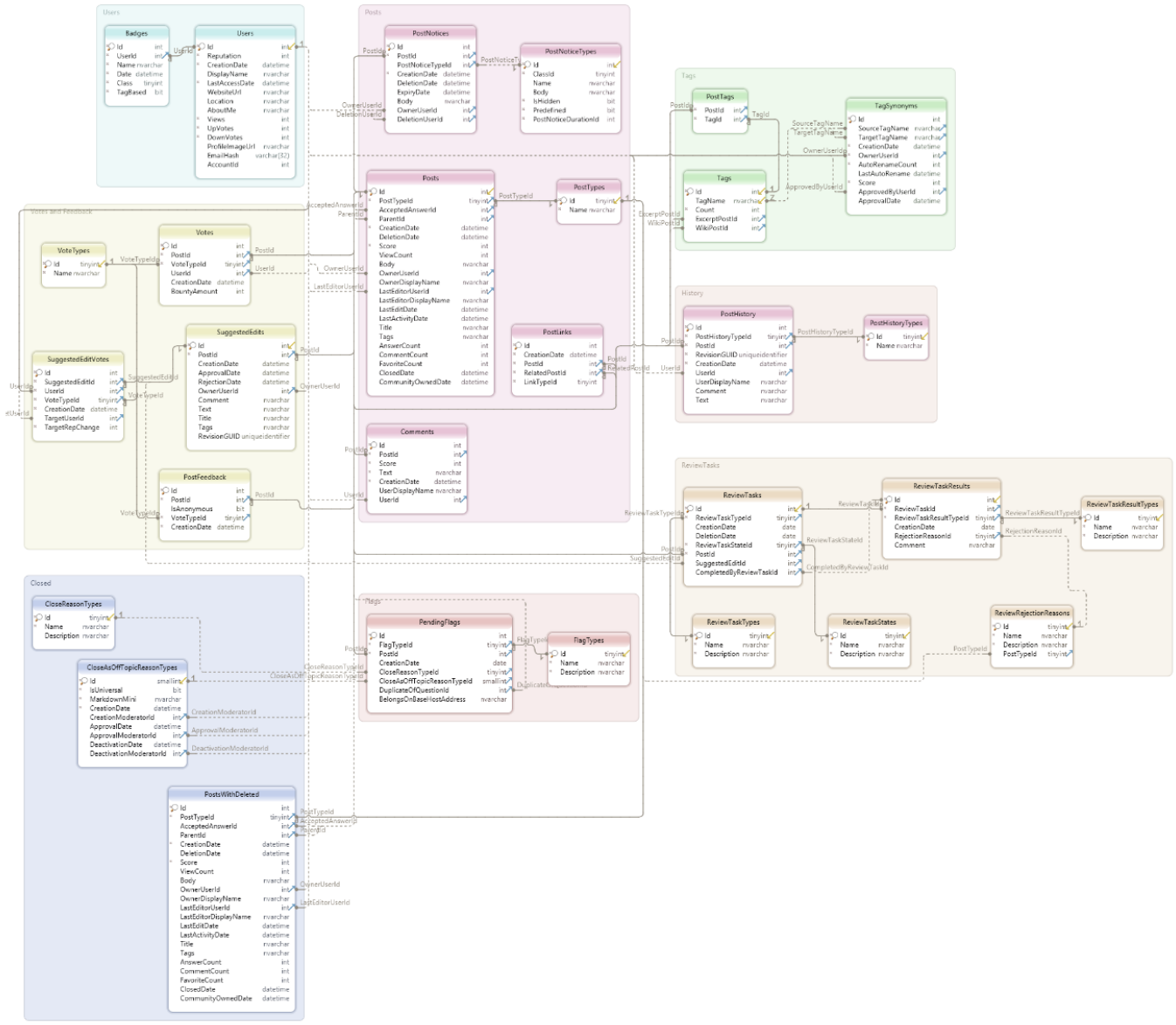
References:

https://meta.stackexchange.com/questions/2677/database-schema-documentation-for-the-public-data-dump-and-sede

https://www.kaggle.com/datasets/stackoverflow/stackoverflow

https://cloud.google.com/bigquery/public-data

Below is an Entity Relationship Diagram (ERD) based on the tables in the StackOverflow dataset on BigQuery we referred to understand and work with.



Data Sourcing

S.No.	Dataset Name	Size	Source	Link
1.	Stack Exchange Data Dump	7.9 GB	Stack Exchange, Inc.	https://archive.org/details/stackexchange
2.	Stack Overflow Developer Survey	~110 MB per year	Stack Overflow	https://survey.stackoverflow.co/2023/
3.	Stack Overflow API	Depends on the amount retrieved. The entire Stack Overflow database expands to ~430 GB.	Stack Overflow	https://stackoverflow.blog/api/
4.	Stack Overflow Tags from Stack Exchange Data Explorer (SEDE)	<10 megabytes in size	Stack Exchange, Inc.	https://data.stackexchange.com/

Stack Overflow also provides other valuable datasets that provide insights into various aspects of the platform and its community, in addition to the Stack Overflow BigQuery dataset. The Stack Exchange Data Dump contains anonymized information about questions, answers, comments, and user profiles in XML format, which can be used for research and analysis. The Stack Overflow Developer Survey dataset, available as CSV files, provides industry trends, programming language preferences, and career-related information from developers worldwide. The Stack Overflow API allows developers to programmatically access real-time data, enabling custom analyses and application development. The Stack Overflow Tags dataset focuses on tag popularity, relationships, and metadata, providing information about programming languages, technologies, and trends.

Data Wrangling

We utilized the query below for display all the related columns.

```
SELECT table_name, column_name, data_type
FROM `bigquery-public-data`.stackoverflow.INFORMATION_SCHEMA.COLUMNS
ORDER BY table_name, ordinal_position
```

- badges table:

table_name	column_name	data_type
badges	id	INT64
badges	name	STRING
badges	date	TIMESTAMP
badges	user_id	INT64
badges	class	INT64
badges	tag_based	BOOL

- Comments table:

table_name	column_name	data_type
comments	id	INT64
comments	text	STRING
comments	creation_date	TIMESTAMP
comments	post_id	INT64
comments	user_id	INT64
comments	user_display_name	STRING
comments	score	INT64

- post_history table:

table_name	column_name	data_type
post_history	id	INT64
post_history	creation_date	TIMESTAMP
post_history	post_id	INT64
post_history	post_history_type_id	INT64
post_history	revision_guid	STRING
post_history	user_id	INT64
post_history	text	STRING
post_history	comment	STRING

- Post_links table:

table_name	column_name	data_type
post_links	id	INT64
post_links	creation_date	TIMESTAMP
post_links	link_type_id	INT64
post_links	post_id	INT64
post_links	related_post_id	INT64

- Post_answers table:

table_name	column_name	data_type
posts_answers	id	INT64
posts_answers	title	STRING
posts_answers	body	STRING
posts_answers	accepted_answer_id	STRING
posts_answers	answer_count	STRING
posts_answers	comment_count	INT64
posts_answers	community_owned_date	TIMESTAMP
posts_answers	creation_date	TIMESTAMP
posts_answers	favorite_count	STRING
posts_answers	last_activity_date	TIMESTAMP
posts_answers	last_edit_date	TIMESTAMP
posts_answers	last_editor_display_name	STRING
posts_answers	last_editor_user_id	INT64
posts_answers	owner_display_name	STRING
posts_answers	owner_user_id	INT64
posts_answers	parent_id	INT64

table_name	column_name	data_type
posts_answers	post_type_id	INT64
posts_answers	score	INT64
posts_answers	tags	STRING
posts_answers	view_count	STRING

- Posts_moderator_nomination table:

table_name	column_name	data_type
posts_moderator_nomination	id	INT64
posts_moderator_nomination	title	STRING
posts_moderator_nomination	body	STRING
posts_moderator_nomination	accepted_answer_id	STRING
posts_moderator_nomination	answer_count	STRING
posts_moderator_nomination	comment_count	INT64
posts_moderator_nomination	community_owned_date	TIMESTAMP
posts_moderator_nomination	creation_date	TIMESTAMP
posts_moderator_nomination	favorite_count	STRING
posts_moderator_nomination	last_activity_date	TIMESTAMP
posts_moderator_nomination	last_edit_date	TIMESTAMP
posts_moderator_nomination	last_editor_display_name	STRING
posts_moderator_nomination	last_editor_user_id	INT64
posts_moderator_nomination	owner_display_name	STRING
posts_moderator_nomination	owner_user_id	INT64
posts_moderator_nomination	parent_id	STRING
posts_moderator_nomination	post_type_id	INT64
posts_moderator_nomination	score	INT64
posts_moderator_nomination	tags	STRING
posts_moderator_nomination	view_count	STRING

- Post_orphaned_tag_wiki table

table_name	column_name	data_type
posts_orphaned_tag_wiki	id	INT64
posts_orphaned_tag_wiki	title	STRING
posts_orphaned_tag_wiki	body	STRING
posts_orphaned_tag_wiki	accepted_answer_id	STRING
posts_orphaned_tag_wiki	answer_count	STRING
posts_orphaned_tag_wiki	comment_count	INT64
posts_orphaned_tag_wiki	community_owned_date	TIMESTAMP
posts_orphaned_tag_wiki	creation_date	TIMESTAMP
posts_orphaned_tag_wiki	favorite_count	STRING
posts_orphaned_tag_wiki	last_activity_date	TIMESTAMP
posts_orphaned_tag_wiki	last_edit_date	TIMESTAMP
posts_orphaned_tag_wiki	last_editor_display_name	STRING
posts_orphaned_tag_wiki	last_editor_user_id	INT64
posts_orphaned_tag_wiki	owner_display_name	STRING
posts_orphaned_tag_wiki	owner_user_id	INT64
posts_orphaned_tag_wiki	parent_id	STRING
posts_orphaned_tag_wiki	post_type_id	INT64
posts_orphaned_tag_wiki	score	INT64
posts_orphaned_tag_wiki	tags	STRING
posts_orphaned_tag_wiki	view_count	STRING

- Posts_privilege_wiki table

posts_privilege_wiki	id	INT64
posts_privilege_wiki	title	STRING
posts_privilege_wiki	body	STRING
posts_privilege_wiki	accepted_answer_id	STRING
posts_privilege_wiki	answer_count	STRING
posts_privilege_wiki	comment_count	INT64
posts_privilege_wiki	community_owned_date	STRING
posts_privilege_wiki	creation_date	TIMESTAMP
posts_privilege_wiki	favorite_count	STRING
posts_privilege_wiki	last_activity_date	TIMESTAMP
posts_privilege_wiki	last_edit_date	TIMESTAMP
posts_privilege_wiki	last_editor_display_name	STRING
posts_privilege_wiki	last_editor_user_id	INT64
posts_privilege_wiki	owner_display_name	STRING

The size of the datasets is given below:

Row	table_name	size_gb
1	post_history	113.18
2	posts_questions	37.17
3	stackoverflow_posts	29.36
4	posts_answers	28.62
5	comments	16.03
6	votes	7.05
7	users	3.14
8	badges	1.96
9	post_links	0.31
10	posts_tag_wiki	0.04
11	posts_tag_wiki_excerpt	0.01
12	posts_moderator_nomination	0.0
13	posts_orphaned_tag_wiki	0.0
14	posts_privilege_wiki	0.0
15	posts_wiki_placeholder	0.0
16	tags	0.0

The number of columns for each table is mentioned below.

Row	table_name	num_columns
1	badges	6
2	comments	7
3	post_history	8
4	post_links	5
5	posts_answers	20
6	posts_moderator_nomination	20
7	posts_orphaned_tag_wiki	20
8	posts_privilege_wiki	20
9	posts_questions	20
10	posts_tag_wiki	20
11	posts_tag_wiki_excerpt	20
12	posts_wiki_placeholder	20
13	stackoverflow_posts	20
14	tags	5
15	users	13
16	votes	4

The number of rows for each table is mentioned below.

Row	table_name	num_rows
1	badges	46135386
2	comments	86754111
3	post_history	152435941
4	post_links	8395210
5	posts_answers	34024119
6	posts_moderator_nomination	342
7	posts_orphaned_tag_wiki	167
8	posts_privilege_wiki	2
9	posts_questions	23020127
10	posts_tag_wiki	55113
11	posts_tag_wiki_excerpt	55115
12	posts_wiki_placeholder	5
13	stackoverflow_posts	31017889
14	tags	63653
15	users	18712212
16	votes	236452885

The data wrangling consists of all the years in the dataset. In the next sections, we would be

working within a certain time period (5 years duration) only.

Exploratory Data Analysis (EDA)

We have performed EDA on individual tables and segments of the stack overflow data. Below are the different subsets of our EDA Analysis.

1. Big Query on Highest Scoring Question and Answers for individual years

Big Query SQL Code for Data Creation:

The query starts with a common table expression (CTE) named ranked_posts. It selects various fields from the users, posts_questions, posts_answers, and badges tables, joined on specific conditions. The ROW_NUMBER() function is used to assign a ranking to questions and answers based on their scores within each user group.

The main query selects specific fields from the users table and the ranked_posts CTE. It joins the users table with the CTE on the id column.

The WHERE clause filters the results to include only users who accessed the site in 2017 and have the highest-scoring question and answer (based on their scores and rankings).

Finally, the LIMIT clause limits the output to a maximum of 3000 rows.

In summary, the query retrieves information about users from the Stack Overflow dataset who accessed the site in 2017 and had the highest-scoring question and answer during that year. The query includes additional user details, such as display name, reputation, location, and badges.

```
WITH ranked_posts AS (
  SELECT
    user.id,
    post.body AS ques_text,
    post.score AS ques_score,
    post.creation_date,
    post.last_activity_date,
    answer.body AS ans_text,
    answer.score AS ans_score,
    answer.creation_date,
    answer.last_activity_date,
    badge.name  as badge_name,
```

```
   badge.tag_based as badge_custm,
   badge.date as badge_date,

   ROW_NUMBER() OVER (PARTITION BY user.id ORDER BY post.score DESC) AS
ques_rn,
   ROW_NUMBER() OVER (PARTITION BY user.id ORDER BY answer.score DESC) AS
ans_rn,

  FROM `bigquery-public-data.stackoverflow.users` AS user
  JOIN `bigquery-public-data.stackoverflow.posts_questions` AS post
    ON user.id = post.owner_user_id AND EXTRACT(YEAR FROM post.creation_date) =
2017 AND EXTRACT(YEAR FROM post.last_activity_date) = 2017 AND post.score>0

  JOIN `bigquery-public-data.stackoverflow.posts_answers` AS answer
    ON user.id = answer.owner_user_id AND EXTRACT(YEAR FROM
answer.creation_date) = 2017 AND EXTRACT(YEAR FROM answer.last_activity_date) =
2017 AND answer.score>0


  JOIN `bigquery-public-data.stackoverflow.badges` AS badge
  ON user.id=badge.user_id AND EXTRACT(YEAR FROM badge.date) = 2017

  WHERE EXTRACT(YEAR FROM user.last_access_date) = 2017
)
SELECT
  user.id,
  user.about_me,
  user.display_name,
  user.age,
  user.creation_date,
  user.last_access_date,
  user.location,
  user.reputation,
  user.up_votes AS user_upv,
  user.down_votes AS user_downv,
  user.views AS user_views,
  ranked_posts.ques_text AS highest_scoring_question,
  ranked_posts.ques_score,
  ranked_posts.ans_text AS highsest_scoring_answer,
  ranked_posts.ans_score,
  badge_name,
  badge_custm,
  badge_date
```

```
FROM ranked_posts
JOIN `bigquery-public-data.stackoverflow.users` AS user
  ON ranked_posts.id = user.id
WHERE ranked_posts.ques_rn = 1 AND ranked_posts.ans_rn= 1
LIMIT 3000;
```

Step 1: Data Loading and Finding Duplicate Users

```
import pandas as pd

data_2022=pd.read_csv("/content/User_Y_2022.csv")

data_2022.columns
```

```
Index(['id', 'about_me', 'display_name', 'age', 'creation_date',
       'last_access_date', 'location', 'reputation', 'user_upv', 'user_downv',
       'user_views', 'highest_scoring_question', 'ques_score',
       'highsest_scoring_answer', 'ans_score', 'badge_name', 'badge_custm',
       'badge_date'],
      dtype='object')
```

```
data_2022.head()
```

	id	about_me	display_name	age	creation_date	last_access_date	location	reputation	user_upv	user_downv	user_views	
0	576857	<p>public static void main...\nAfter 17 years ...	Al Kepp	NaN	2011-01-15 16:31:37.590000 UTC	2022-08-30 20:16:49.837000 UTC	Vienna, Rakousko	5725	695	185	821	<p>I've
1	7040601	<p>Computer Vision, Android and Machine learni...	FutureJJ	NaN	2016-10-19 06:48:40.530000 UTC	2022-09-24 16:03:13.437000 UTC	India	1936	754	13	173	<code>ERR
2	6285054	<p>11+ Years of experience	Avinash Dalvi	NaN	2016-05-03 09:21:40.237000	2022-09-24 17:19:48.970000	Bangalore, Karnataka,	7810	3205	15	1061	

```
data_2021=pd.read_csv("/content/User_Y_2021.csv")
data_2020=pd.read_csv("/content/User_Y_2020.csv")
data_2019=pd.read_csv("/content/User_Y_2019.csv")
data_2018=pd.read_csv("/content/User_Y_2018.csv")
data_2017=pd.read_csv("/content/User_Y_2017.csv")
```

```
# Merge all the tables together based on the 'id' column
merged_data = pd.concat([data_2022, data_2021, data_2020, data_2019, data_2018, data_2017])
```

```
merged_data.to_csv('final_merged_data.csv', index=False)
```

```
#Finding duplicates

duplicates = merged_data.duplicated(subset='id')

duplicate_count = duplicates.sum()

print("Number of duplicates:", duplicate_count)
```

```
Number of duplicates: 0
```

Step 2: Checking for datatype, unique values, and null values

```
#Checking datatype of each attribute
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18098 entries, 0 to 2999
Data columns (total 18 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   id                        18098 non-null  int64
 1   about_me                  2933 non-null   object
 2   display_name              18096 non-null  object
 3   age                       0 non-null      float64
 4   creation_date             18098 non-null  object
 5   last_access_date          18098 non-null  object
 6   location                  6499 non-null   object
 7   reputation                18098 non-null  int64
 8   user_upv                  18098 non-null  int64
 9   user_downv                18098 non-null  int64
 10  user_views                18098 non-null  int64
 11  highest_scoring_question  18098 non-null  object
 12  ques_score                18098 non-null  int64
 13  highsest_scoring_answer   18098 non-null  object
 14  ans_score                 18098 non-null  int64
 15  badge_name                18098 non-null  object
 16  badge_custm               18098 non-null  bool
 17  badge_date                18098 non-null  object
dtypes: bool(1), float64(1), int64(7), object(9)
memory usage: 2.5+ MB
```

```
#Checking unique values in each column

merged_data.nunique()
```

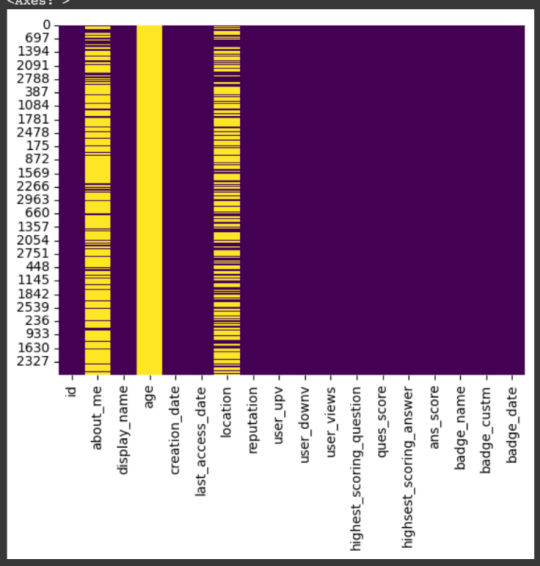
```
id                          18098
about_me                     2894
display_name                17171
age                             0
creation_date               18098
last_access_date            18098
location                     2447
reputation                   1560
user_upv                      592
user_downv                    144
user_views                    558
highest_scoring_question    18098
ques_score                     40
highsest_scoring_answer     18097
ans_score                      53
badge_name                     76
badge_custm                     2
badge_date                  17523
dtype: int64
```

```
#Checking for null values

merged_data.isnull().sum()
```

```
id                          0
about_me                15165
display_name                2
age                     18098
creation_date               0
last_access_date            0
location                11599
reputation                  0
user_upv                    0
user_downv                  0
user_views                  0
highest_scoring_question    0
ques_score                  0
highsest_scoring_answer     0
ans_score                   0
badge_name                  0
badge_custm                 0
badge_date                  0
dtype: int64
```

```
import seaborn as sns
#Visualizing Missing values
sns.heatmap(merged_data.isnull(), cbar=False, cmap='viridis')

#We can see about me, age and location has a lot of missing values It is best to drop those
#Values when dealing with the ml part.
```

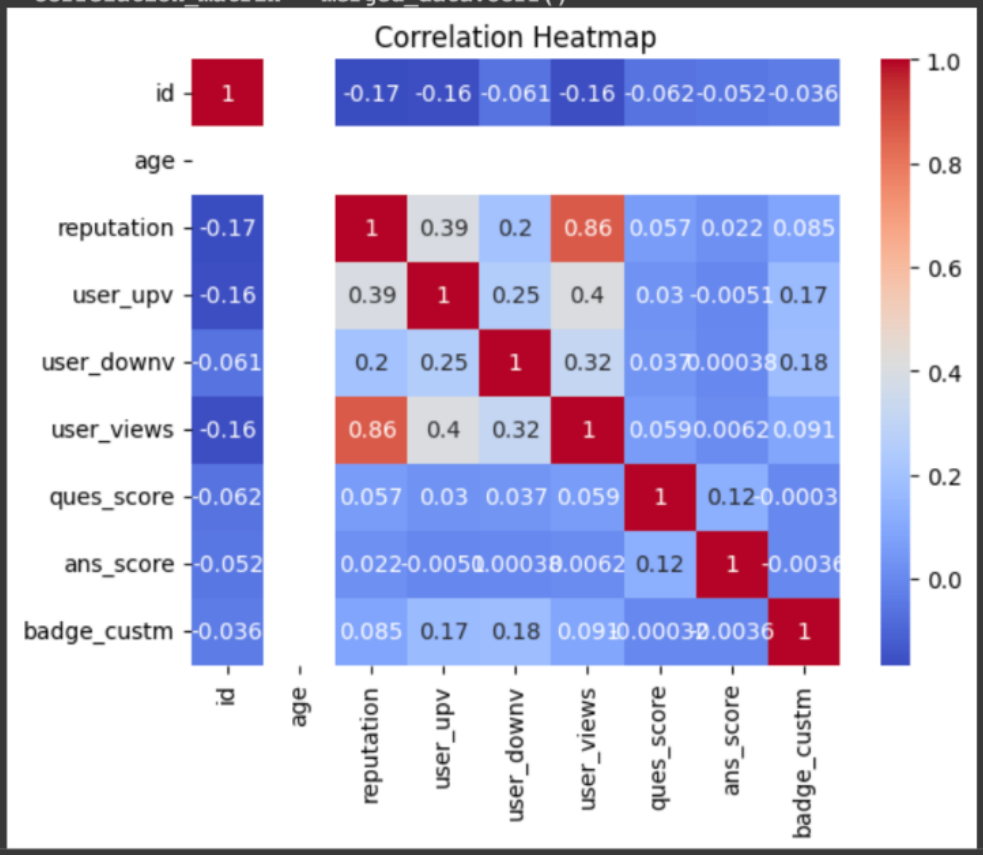
<Axes: >



Step 3: Correlation Heatmap for python seaborn library

```
import matplotlib.pyplot as plt

# Assuming your DataFrame is called 'data'
correlation_matrix = merged_data.corr()

# Generate the correlation heatmap
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")

# Set the plot title
plt.title("Correlation Heatmap")

# Display the heatmap
plt.show()
```

```
<ipython-input-10-7af0fd280768>:7: FutureWarning: The default value of numer
  correlation_matrix = merged_data.corr()
```



Step 4: Topic Modeling for ‘About Me’ section using LDA

Topic Modeling: Apply topic modeling algorithms such as Latent Dirichlet Allocation (LDA) or Non-Negative Matrix Factorization (NMF) to discover underlying topics or themes in the 'about_me' texts. These algorithms group similar words together and assign them to specific topics. By analyzing the topics, you can identify common patterns or interests among users.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation

# Preprocess the 'about_me' text data
preprocessed_data = merged_data['about_me'].fillna('').str.lower()  # Convert to lowercase and handle missing values

# Create a document-term matrix
vectorizer = CountVectorizer(stop_words='english')
dtm = vectorizer.fit_transform(preprocessed_data)

# Get the feature names
feature_names = vectorizer.get_feature_names_out()

# Apply LDA
num_topics = 5  # Set the number of topics
lda = LatentDirichletAllocation(n_components=num_topics, random_state=42)
lda.fit(dtm)

# Get the most important words for each topic
num_top_words = 10  # Number of top words to display for each topic
for topic_idx, topic in enumerate(lda.components_):
    print(f"Topic #{topic_idx + 1}:")
    top_word_indices = topic.argsort()[:-num_top_words - 1:-1]
    top_words = [feature_names[i] for i in top_word_indices]
    print(", ".join(top_words))
    print()

# Assign topics to each 'about_me' text
topic_assignments = lda.transform(dtm)
merged_data['topic'] = topic_assignments.argmax(axis=1)

# Display the topics assigned to each user
print(merged_data[['about_me', 'topic']])
```

```
# Display the topics assigned to each user
print(merged_data[['about_me', 'topic']])
```

```
Topic #1:
strong, developer, python, blockquote, programming, design, br, amp, code, work

Topic #2:
com, href, https, rel, nofollow, noreferrer, http, li, www, github

Topic #3:
developer, web, software, code, development, learning, programming, working, like, python

Topic #4:
li, ul, data, strong, day, student, working, python, learning, linux

Topic #5:
em, li, questions, kbd, quot, tagged, 39, just, tag, learn

                                              about_me  topic
0     <p>public static void main...\nAfter 17 years ...      2
1     <p>Computer Vision, Android and Machine learni...      1
2     <p>11+ Years of experience as Full Stack Devel...      2
3     <p>I love functional programming and building ...      2
4     <p>Some of my Stack Overflow contributions:</p...      3
...                                                 ...    ...
2995                                                NaN      0
2996                                                NaN      0
2997                                                NaN      0
2998                                                NaN      0
2999                                                NaN      0

[18098 rows x 2 columns]
```

[18098 rows x 2 columns]

merged_data

	id	about_me	display_name	age	creation_date	last_access_date	location	reputation	user_upv	user_downv	user_views	highest_scoring_question	ques_s
0	576857	public static void main...\nAfter 17 years ...	Al Kepp	NaN	2011-01-15 16:31:37.590000 UTC	2022-08-30 20:16:49.837000 UTC	Vienna, Rakousko	5725	695	185	821	I've got the infamous error message in C++ ...	
1	7040601	Computer Vision, Android and Machine learni...	FutureJJ	NaN	2016-10-19 06:48:40.530000 UTC	2022-09-24 16:03:13.437000 UTC	India	1936	754	13	173	I am getting <code>ERR_UNKNOWN_URL_SCHEME</...	
2	6285054	11+ Years of experience as Full Stack Devel...	Avinash Dalvi	NaN	2016-05-03 09:21:40.237000 UTC	2022-09-24 17:19:48.970000 UTC	Bangalore, Karnataka, India	7810	3205	15	1061	While installing <code>npm install</code> a...	
3	8033562	I love functional programming and building ...	Geoff Langenderfer	NaN	2017-05-18 21:29:41.027000 UTC	2022-09-23 09:32:24.743000 UTC	Austin, TX, USA	652	1105	4	241	not sure what to do \n and having a to...	
4	3787051	Some of my Stack Overflow contributions: </p...	Alex Harvey	NaN	2014-06-29 03:24:53.323000 UTC	2022-09-22 05:55:56.023000 UTC	Sydney, Australia	13470	1401	22	1308	I am trying to build a pip package from sou...	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2995	2544888	NaN	noetic	NaN	2013-07-03 03:20:18.837000 UTC	2017-11-11 04:59:32.977000 UTC	NaN	23	0	0	9	Good day! \n\n<p	
2996	7730604	NaN	CSHORT1964	NaN	2017-03-18 06:55:43.947000 UTC	2017-03-23 11:31:12.990000 UTC	NaN	31	0	0	28	I have been working diligently to get Dynam...	

Step 5: Word Cloud to visualize textual data and highlight the keywords

Word Cloud: Visualize the word frequencies using a word cloud. A word cloud represents the most frequently occurring words, with the size of each word indicating its frequency. This visualization can give you a quick overview of the common keywords or themes in user descriptions.

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Concatenate all 'about_me' texts into a single string
text = ' '.join(merged_data['about_me'].fillna(''))

# Generate the word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)

# Display the word cloud using matplotlib
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Reference colab/jupyter notebook:

https://colab.research.google.com/drive/1VwTRpt227qnHYD93vNgsn-2njbaH3-NX?usp=sharing

https://console.cloud.google.com/bigquery?sq=177690437362:1c0721c495f540f3a9f1cd57492a7b7b&project=user-segmentation-391501&ws=!1m5!1m4!1m3!1suser-segmentation-391501!2sbquxjob_18b2980f_189282513d6!3sUS

2. RFM Segmentation Analysis:

RFM segmentation is a technique for customer segmentation widely used in marketing and customer analytics. It is an acronym for Recency, Frequency, and Monetary Value. While RFM segmentation is most commonly used in the e-commerce or retail industries, it can also be used to analyze user behavior on Stack Overflow. Recency: Recency refers to the time elapsed since a user's last activity. Here, it is the number of days that passed from the user's last answer.

Recency: Recency refers to the time elapsed since a user's last activity. Here, it is the number of days that passed from the user's last answer.

```sql
SELECT
  u.display_name AS User,
  DATE_DIFF(CURRENT_DATE(), DATE(MAX(p.creation_date)), DAY) AS DaysPassed
FROM
  `bigquery-public-data.stackoverflow.users` AS u
JOIN
  `bigquery-public-data.stackoverflow.posts_answers` AS p
  ON u.id = p.owner_user_id
GROUP BY
  User
```

Press Alt+F1 for accessibility options.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	User	DaysPassed
1	Stephen Hendry	5452
2	Piotr Tyburski	5450
3	Daniel Broekman	5448
4	Conrad Halling	5448
5	Ciro Villa	5445
6	Dag Haavi Finstad	5444
7	jsmorris	5444
8	DannySmurf	5443
9	Marcio DaSilva	5443
10	Björn Waide	5441

Results per page: 50 | 1 – 50 of 2181921

Frequency: This represents how often users engage with Stack Overflow. Here, it is measured by the number of times the user answered questions in a specific time period. We considered the time period of 2021 to 2022. We are looking at expanding the period to 5 years.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	User	AnswerCount
1	Stokely	3
2	Pedro Lobito	2
3	yoAlex5	2
4	Ashutosh Ranjan	2
5	hanshenrik	2
6	Misha Zaslavsky	2
7	Gerold Broser	2
8	kalopseeia	1
9	Michu93	1
10	Dogeek	1
11	Senseful	1
12	otmar	1
13	Blogspot OSOL	1
14	SQLpro	1

Monetary: There is no direct monetary value associated with user activities on Stack Overflow. The indicator used here for monetary value is the number of times the user answered from his/her first activity.

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	User	AnsweredQuestions(
1	Jon Skeet	33135
2	Gordon Linoff	27474
3	Alex	21683
4	Darin Dimitrov	21220
5	David	18591
6	Chris	18096
7	anubhava	17802
8	CommonsWare	17312
9	BalusC	16889
10	Matt	16247
11	Hans Passant	15688
12	Martijn Pieters	15662
13	Quentin	15533
14	VonC	15066
15	Daniel	14483

RFM segmentation enables you to create meaningful user segments that can inform various strategies such as personalized content recommendations, targeted notifications, community engagement initiatives, or user retention efforts by combining these three factors. It aids in comprehending and catering to various types of users based on their levels of engagement, contribution patterns, and overall value to the Stack Overflow community.

SQL Queries:

Recency :

```sql
SELECT
  u.display_name AS User,
  DATE_DIFF(CURRENT_DATE(), DATE(MAX(p.creation_date)), DAY) AS DaysPassed
FROM
  `bigquery-public-data.stackoverflow.users` AS u
JOIN
  `bigquery-public-data.stackoverflow.posts_answers` AS p
  ON u.id = p.owner_user_id
GROUP BY
  User
ORDER BY
  DaysPassed DESC;
```

Frequency:

```sql
SELECT
  u.display_name AS User,
  COUNT(*) AS AnswerCount
FROM
  `bigquery-public-data.stackoverflow.users` AS u
JOIN
  `bigquery-public-data.stackoverflow.posts_answers` AS a
  ON u.id = a.owner_user_id
JOIN
  `bigquery-public-data.stackoverflow.posts_questions` AS q
  ON a.parent_id = q.id
JOIN
  `bigquery-public-data.stackoverflow.tags` AS t
  ON q.id = t.id
WHERE
  a.creation_date >= TIMESTAMP('2021-01-01') -- Start date of the 6-month period
  AND a.creation_date < TIMESTAMP('2022-01-01') -- End date of the 6-month period
GROUP BY
  User
ORDER BY
  AnswerCount DESC;
```

Monetary Value:

```sql
SELECT
  u.display_name AS User,
  COUNT(*) AS AnsweredQuestionsCount
FROM
  `bigquery-public-data.stackoverflow.users` AS u
JOIN
  `bigquery-public-data.stackoverflow.stackoverflow_posts` AS p
  ON u.id = p.owner_user_id
WHERE
  p.post_type_id = 2 -- 2 represents answers
GROUP BY
  User
ORDER BY
  AnsweredQuestionsCount DESC;
```

https://drive.google.com/file/d/1MXeTgRtD64aK2UWD_Hfv_o4eazzQnTwG/view?usp=drive_link

The above link contains other basic EDA.

3. Big Query for statistical features

The given query retrieves user information from the Stack Overflow dataset, including details such as user ID, display name, age, location, reputation, and various statistics related to their questions, answers, and badges. The query uses subqueries to calculate aggregated statistics and joins them with the main user table based on user IDs. The results are limited to a maximum of 30,000 rows.

```
SELECT
    user.id,
    user.about_me,
    user.display_name,
    user.age,
    user.creation_date,
    user.last_access_date,
    user.location,
    user.reputation,
    user.up_votes as user_upv,
    user.down_votes as user_downv,
    user.views as user_views,
    post.ques_cnt,
    post.ques_answer_cnt_tot,
```

```
    post.ques_comment_cnt_tot,
    post.ques_favorite_cnt_tot,
    post.ques_score_tot,
    post.ques_view_cnt_tot,
    post.ques_answer_cnt_avg,
    post.ques_comment_cnt_avg,
    post.ques_favorite_cnt_avg,
    post.ques_score_avg,
    post.ques_view_cnt_avg,
    answer.ans_cnt,
    answer.ans_comment_cnt_tot,
    answer.ans_score_tot,
    answer.ans_comment_cnt_avg,
    answer.ans_score_avg,
    badge.badge_cnt
FROM (
    SELECT *
    FROM `bigquery-public-data.stackoverflow.users`
    WHERE EXTRACT(YEAR FROM last_access_date) = 2017
    AND about_me is not null
) AS user
JOIN (
    SELECT
      owner_user_id,
      COUNT(id) AS ques_cnt,
      SUM(answer_count) AS ques_answer_cnt_tot,
      SUM(comment_count) AS ques_comment_cnt_tot,
      SUM(favorite_count) AS ques_favorite_cnt_tot,
      SUM(score) AS ques_score_tot,
      SUM(view_count) AS ques_view_cnt_tot,
      AVG(answer_count) AS ques_answer_cnt_avg,
      AVG(comment_count) AS ques_comment_cnt_avg,
      AVG(favorite_count) AS ques_favorite_cnt_avg,
      AVG(score) AS ques_score_avg,
      AVG(view_count) AS ques_view_cnt_avg
    FROM `bigquery-public-data.stackoverflow.posts_questions`
    WHERE EXTRACT(YEAR FROM last_activity_date) <= 2017
    GROUP BY 1
) AS post
  ON user.id=post.owner_user_id
JOIN (
    SELECT
      owner_user_id,
      COUNT(id) AS ans_cnt,
```

```
        SUM(comment_count) AS ans_comment_cnt_tot,
        SUM(score) AS ans_score_tot,
        AVG(comment_count) AS ans_comment_cnt_avg,
        AVG(score) AS ans_score_avg
      FROM `bigquery-public-data.stackoverflow.posts_answers`
      WHERE EXTRACT(YEAR FROM last_edit_date) <= 2017
      GROUP BY 1
  ) AS answer
    ON user.id=answer.owner_user_id
  JOIN (
      SELECT
        user_id,
        COUNT(*) AS badge_cnt
      FROM `bigquery-public-data.stackoverflow.badges`
      WHERE EXTRACT(YEAR FROM date) <= 2017
      GROUP BY 1
  ) AS badge
    ON user.id=badge.user_id
  LIMIT 30000
```

References:
https://console.cloud.google.com/bigquery?sq=684030019557:f05af6ff75ce4ba384c5198ee49c1909&project=user-segmentation-391501&ws=!1m5!1m4!1m3!1suser-segmentation-391501!2sbquxjob_5045336d_189282e6c40!3sUS

4. Current Work:

Currently, we are working on merging Query 1: Highest Scoring Questions and Answers for individual years and Query 3: Big Query for statistical features, which will take merged features such as the highest scoring question posted by the user over the years and its statistical features.

The final dataset will have the temporal/time series features that can be used to analyze user retention on forums, such as stack overflow, and also help us to study the Clusters of users over the years.

Below are the references on how we plan to merge the dataset and how we are planning to work on generating different trends from the data.

The only difference between the below queries and the above queries is that we are using an external table of unique users that tracks different user IDs over the years from 2017 to 2021. Based on the reference of the table, we are able to find different user engagement over the years.

Example:

2017 year Statistical Big Query
https://console.cloud.google.com/bigquery?sq=428442525029:f49441b960174a8bab38d7c7e3af616b

2017 year Highest scoring question and answer query

https://console.cloud.google.com/bigquery?sq=428442525029:f48c0c28544344cbb4727d01a44df025

Below notebook shows how both the tables are joined to get a final merged data for the year 2017 and so on.

https://drive.google.com/file/d/17_4iJTnx1B1bv7UMKSThBjxJcmPOiXaE/view?usp=sharing

Clustering features:
#id
about_me
reputation -(harmonic mean of highest quest and answer)*reputation
user_upv - (harmonic mean of highest quest and answer)*upv
user_downv -(harmonic mean of highest quest and answer)*down
user_views -(harmonic mean of highest quest and answer)*view
ques_cnt - n of questions asked by a user in a year
ques_answer_cnt_tot - n of asnwers to the questiosn asked by a user in a year
ques_score_tot - sum of all scores user questions received in a year
ques_view_cnt_tot - sum of all views user questions received in a year
ques_score_avg - median score - mid value of all scores received to the questions asked - TA
ques_view_cnt_avg - median score - mid value of all views received by the questions asked - TA
ans_cnt - n of answers given by a user in a year
ans_score_tot - sum of all scores user answers received in a year
ans_score_avg - median score - mid value of all scores received by the answers to the questions asked - TA
#year
highest_scoring_question text
ques_score
highest_scoring_answer text
ans_score
tenure - last_access_date-creation_date - number of user tenure years

20000-< contributes multiple
6000- once morning

harmonic mean - n/(1/x1+1/x2+1/x3+1/x4+1/x5)

Year-> 1/0 BEING Active or not.

reputation/tenure - 100/5 = 20 - 1 way
1st year - 20
2d year - 40
3d year - 60
4th year - 80
5th year - 100

1st year - highest score - 10 & ans -8
2d year - highest score - 5 & ans - 6
3d year highest score - 8 & ans - 4

reputation - 70 - should we use it at all? TA - 2 way
harmonic mean ques & ans 1st year = 2/(1/10+1/8) = 8.89 - 8.89/(8.89+5.45+5.33)=0.45 * 70 ->31.5
harmonic mean ques & ans 2st year = 2/(1/5+1/6) = 5.45 - 0.28 -> 31.5+0.28*70=51.1 OR 19.5
harmonic mean ques & ans 3d year = 2/(1/8+1/4) = 5.33 - 0.27 -> 70 - OR 18.9

Check with Univariant Correlation-> Check first then consider.

Clustering features:
#id
about_me
reputation -(harmonic mean of highest quest and answer)*reputation
user_upv - (harmonic mean of highest quest and answer)*upv
user_downv -(harmonic mean of highest quest and answer)*down
user_views -(harmonic mean of highest quest and answer)*view
ques_cnt - n of questions asked by a user in a year
ques_answer_cnt_tot - n of asnwers to the questiosn asked by a user in a year
ques_score_tot - sum of all scores user questions received in a year
ques_view_cnt_tot - sum of all views user questions received in a year
ques_score_avg - median score - mid value of all scores received to the questions asked - TA
ques_view_cnt_avg - median score - mid value of all views received by the questions asked - TA
ans_cnt - n of answers given by a user in a year
ans_score_tot - sum of all scores user answers received in a year
ans_score_avg - median score - mid value of all scores received by the answers to the questions asked - TA
#year
highest_scoring_question text
ques_score
highest_scoring_answer text
ans_score
tenure - last_access_date-creation_date - number of user tenure years