Aakriti Lnu (al1745), Gokula Ranga Naveen Chapala (gc3522),
Muhammad Raees (mr2714), Prajjwal Mehta (pm8607)

This report explains the process of creating a document-based database in Phase II of the project. The dataset we selected is Meta Kaggle (https://www.kaggle.com/datasets/kaggle/meta-kaggle). We use the same database sub-set used in Phase I of the project. The report briefly describes the structure of the document model, how each collection is laid out and mapped with data attributes, and its comparison with the relational model. This report also explores and explains the results and performance of some interesting queries on the relational database (before and after creating indexes). In addition, the report describes the identification of functional dependencies and their evaluation.

Note: The program code is provided in the zipped folder. The program can be executed by following the "README.md" instructions. The global settings are provided in "globals.py" inside the root folder. The program execution, timing, and other constraints are provided in the "README.md" and across the report while answering specific questions.

Proposed Document Model

From Phase I, we used the data files reported in Table 1 to propose and populate the document database. The complete description of each file is provided in Phase I. We propose the document model using the same data file and attribute sub-set.

Table 1. List of files used to propose and populate the database.

Table Name	**Description**
Users	Contains information about Kaggle users, including their performance tiers and registration data.
Tags	Contains information about tags applied to competitions, datasets, kernels, and forums.
Forums	Captures information about forum posts, including title and parent relationships.
Organizations	Stores details of organizations on Kaggle, including creation date and descriptions.
UserOrganizations	Links users with their affiliated organizations on Kaggle.
UserFollowers	Tracks who follows whom among Kaggle users.
Datasets	Stores dataset information, including total downloads, views, and votes.
DatasetTags	Associates datasets with tags for easier categorization and search.
Competitions	Contains details about competitions, including deadlines, rewards, and evaluation methods.
CompetitionTags	Links competitions to relevant tags for categorization purposes.
Teams	Contains team information for competitions, such as membership and medals won.
Submissions	Tracks submissions to competitions, including scores and submission dates.
UserAchievements	Records achievements of Kaggle users, including rankings, points, and medal counts.

The proposed document model, a brief description of decisions made, and a comparison with the relational model are explained in Table 2 below. We try to structure documents with related data (except having independent organizations, forums, and tags, that can be removed after insertion if attributes are mapped instead of references). This mapping balances a clear separation of entities (e.g., users, organizations, datasets) and the need for references and joins. The document structure nesting directly within the document (achievements, followers, submission, etc), makes it easier to access data. However, in comparison to relational databases, maintaining integrity constraints is difficult. However, data scaling is much easier in such a structure, which becomes difficult when schemas are highly normalized.

Table 2. Proposed document model and decisions made about defining collections.

Document Structure	**Explanation**

Organizations: {
_id: _, Id: _, Name: _,
Slug: _, Description: _,
CreationDate: _,
}
	This maps to organizations (relation) and contains attributes to represent those. Organizations can be linked to users. However, in the prospect of having independent organizations, we decided to make it an independent collection. If the purpose is to ensure only data insertion, organizations can be removed as a collection afterward, and where referenced, organization attributes can be used. When compared with relational models, it uses a similar strategy to represent the organizations.

Users:
{
_id: _, Id: _, UserName: _, DisplayName: _,
RegisterDate: _, PerformanceTier: _,
Country: _,
Organizations: [{ OrganizationId: _, JoinDate: _ }],
Followers: [{ FollowingUserId: _, CreationDate: _, }],
Achievements: [{ AchievementType: _,
Tier: _, TierAchievementDate: _,
Points: _, CurrentRanking: _,
HighestRanking: _, TotalGold: _,
TotalSilver: _, TotalBronze: _ }]
}
	User collection maps to the users' relations and contains the attributes that represent users. In addition, as users belong to different organizations, we embedded the organizations as part of the user collection to represent and map the organizations they are part of and when they join the organizations. Users are also followed by other users. Therefore, we include user followers inside the user collection with the following creation date. Users also have achievements, which we feel would be appropriate to represent with users. While comparing with a relational model this combines the four files (and relations) into one user collection. It removes the need to separately keep records of user organizations, user followers, and user achievements as a separate relation.

Forums:
{
_id : _, Id: _, Title: _, ParentForumId: _,
}
	This maps to forums (relation) and contains attributes to represent forums. Forums are associated with datasets and competitions, therefore, this is an independent collection. If the purpose is to ensure only data insertion, forums can be removed as collection afterward, and where referenced, forum titles can be used. While comparing with relational models. It uses a similar strategy to represent the forums.

Tags:
{
_id : _, Name: _, Slug: _,
FullPath: _, Description: _,
DatasetCount: _, CompetitionCount: _,
KernelCount: _, ParentTagId: _,
}
	This maps to tags (relation) and contains attributes to represent those. Tags are associated with competitions and datasets, therefore, are created as independent collections. If the purpose is to ensure only data insertion, tags can be removed as collection afterward, and where referenced, tag attributes can be used. While comparing with relational models. It uses a similar strategy to represent the tags.

Datasets:
{
_id : _, Id: _,
CreatorUserId: _, ForumId: _, CreationDate: _,
LastActivityDate: _, TotalViews: _, TotalDownloads: _,
TotalVotes: _, TotalKernels: _,
DatasetTags: [{ TagId: _,}, …],
}
	Dataset collection maps to the dataset relation and contains the attributes to represent datasets. In addition, as datasets are marked with tags, therefore, we include tag references to dataset tags inside the dataset collection. It references users and forums to correctly include valid records. While comparing with a relational model this combines the two files (and relations) into one dataset collection. It removes the need to separately keep records of dataset tags as a separate relation.

Competitions:
{
_id : _, Id: _,
Slug: _, Title: _, ForumId: _, EnabledDate: _,
DeadlineDate: _, EvaluationAlgorithmName: _,
MaxTeamSize: _, NumPrizes: _, TotalTeams: _,
TotalCompetitors: _, TotalSubmissions: _,
CompetitionTags: [{ TagId: _, }, …],
}
	Competition collection maps the competition relation and contains the attributes representing competitions. In addition, as competitions are marked with tags, therefore, we include tag references to competition tags inside the competition collection. It also maps to forums to correctly include valid records. While comparing with a relational model this combines the two files (and relations) into one competition collection. It removes the need to separately keep records of competition tags as a separate relation.

Teams:
{
_id : _, Id: _, TeamLeaderId: _, TeamName: _,
CompetitionId: _,
Submissions: [{ _id : _, SubmittedUserId: _,
SubmissionDate: _, IsAfterDeadline: _,
PublicScoreLeaderboardDisplay: _,
PrivateScoreLeaderboardDisplay: _, }, …],
}
	Team collection maps the teams' relation and contains the attributes to represent teams. In addition, teams contain submissions (which are submitted by a user). We embedded the submissions as part of the team collection to represent and map the submissions submitted for each team. This combines the two files (and relations) into one team's collection. While comparing with a relational model this combines the two files (and relations) into one team collection. It removes the need to separately keep records of submissions as a separate relation.

Dataset and Document Model Map

Similar to phase I, the dataset maps to each self-explanatory entity or set of entities to a collection. Table 3 provided below explains the dataset mapping to the document model.

Table 3. Description of how the data in the dataset maps to the proposed model.

Field Name	Description	Document/Attribute in Model
Collection: Organizations		
Dataset File: Organizations		
Id	Unique identifier for the organization. (can be removed after mapping)	Organizations.Id
CreationDate	The date when the organization was added.	Organizations.CreationDate
Description	Detailed description of the organization.	Organizations.Description
Name	Name of the organization.	Organizations.Name
Slug	URL-friendly version of the organization name.	Organizations.Slug
Collection: Users		
Dataset File: Users		
Id	Unique identifier for each user. (can be removed after mapping)	Users.Id
Country	The country where the user is located.	Users.Country
DisplayName	The display name of the user.	Users.DisplayName
PerformanceTier	User's performance tier on Kaggle.	Users.PerformanceTier
RegisterDate	The date the user registered on Kaggle.	Users.RegisterDate
UserName	Username chosen by the user.	Users.UserName
Dataset File: User Organizations (mapped with Users on UserId)		
OrganizationId	ID of the organization the user is affiliated with. **(references to Organizations)**	Users.Orgnaiztions.OrganizationId
JoinDate	The date when the user joined the organization.	Users.Orgnaiztions.JoinDate
Dataset File: User Followers (mapped with Users on UserId)		
CreationDate	The date when the follower relationship was created.	Users.Followers.CreationDate
FollowingUserId	The ID of the user following another user. **(references to Users)**	Users.Followers.FollowingUserId
Dataset File: User Achievements (mapped with Users on UserId)		
AchievementType	Type of achievement (e.g., Competitions, Datasets).	Users.Achievements.AchievementType
CurrentRanking	The current global rank of the user on Kaggle is based on the total points they have earned.	Users.Achievements.CurrentRanking
HighestRanking	The highest global rank the user has ever achieved.	Users.Achievements.HighestRanking
Points	Total points earned by the user in Kaggle competitions, kernels, datasets, or discussions.	Users.Achievements.Points

Tier	User's performance level or rank category (e.g., 0 - Novice, 1 - Expert) within Kaggle	Users.Achievements.Tier
TierAchievementDate	Date when the user achieved the tier.	Users.Achievements.TierAchievementDate
TotalBronze	Total number of bronze medals earned by the user.	Users.Achievements.TotalBronze
TotalGold	Total number of gold medals earned by the user.	Users.Achievements.TotalGold
TotalSilver	Total number of silver medals earned by the user.	Users.Achievements.TotalSilver
Collection: Forums		
Dataset File: Forums		
Id	Unique identifier for the forum post. (can be removed after mapping)	Forums.Id
Title	Title of the forum post.	Forums.Title
ParenetForumId	Parent of the forum post.	Forums.ParentForumId
Collection: Tags		
Dataset File: Tags		
Id	Unique identifier for the tag. (can be removed after mapping)	Tags.Id
ParentTagId	Parent id for the tag.	Tags.ParentTagId
CompetitionCount	Number of competitions associated with this tag.	Tags.CompetitionCount
DatasetCount	Number of datasets associated with this tag.	Tags.DatasetCount
Description	Textual field providing a detailed explanation or summary of the tag's meaning or purpose.	Tags.Description
FullPath	This field represents the full hierarchical path to the tag, starting from the top-level parent tag and going down through any sub-tags.	Tags.FullPath
KernelCount	Number of kernels (notebooks) associated with the tag.	Tags.KernelCount
Name	Name of the tag (e.g., Python, Machine Learning).	Tags.Name
Slug	URL-friendly version of the tag name. Often used in web applications for clean URLs.	Tags.Slug
Collection: Datasets		
Dataset File: Datasets		
Id	Unique identifier for the dataset. (can be removed after mapping)	Datasets.Id
CreationDate	The date the dataset was created.	Datasets.CreationDate
CreatorUserId	ID of the user who created the dataset. **(references users)**	Datasets.CreatorUserId
ForumId	ID of the forum associated with the dataset. **(references forums)**	Datasets.ForumId
LastActivityDate	Date of the last activity related to the dataset.	Datasets.LastActivityDate
TotalDownloads	Total number of times the dataset has been downloaded.	Datasets.TotalDownloads
TotalKernels	Total number of kernels (notebooks) created using the dataset.	Datasets.TotalKernels

TotalViews	Total number of views the dataset has received.	Datasets.TotalViews
TotalVotes	Total number of votes (upvotes) the dataset has received.	Datasets.TotalVotes
Dataset File: Dataset Tags (mapped with Datasets on DatasetId)		
TagId	The ID of the tag associated with the dataset. **(references Tags)**	Datasets.DatasetTags.TagId
Collection: Competitions		
Dataset File: Competitions		
Id	Unique identifier for the competition. (can be removed after mapping)	Competitions.Id
Slug	URL-friendly version of the competition title.	Competitions.Slug
Title	Name of the competition.	Competitions.Title
ForumId	ID of the forum associated with the competition. **(references forums)**	Competitions.ForumId
EnabledDate	The date when the competition was opened.	Competitions.EnabledDate
DeadlineDate	The final date for submissions.	Competitions.DeadlineDate
EvaluationAlgorithmName	Full name of the evaluation algorithm used.	Competitions.EvaluationAlgorithmName
MaxTeamSize	Maximum team size allowed in the competition.	Competitions.MaxTeamSize
TotalTeams	Total number of teams participating.	Competitions.TotalTeams
TotalCompetitors	Total number of individual participants.	Competitions.TotalCompetitors
TotalSubmissions	Total number of submissions made.	Competitions.TotalSubmissions
Dataset File: Competition Tags (mapped with Competitions on CompetitionId)		
TagId	The ID of the tag associated with the Competition. **(references Tags)**	Competitions.CompetitionTags.TagId
Collection: Teams		
Dataset File: Teams		
Id	Unique identifier for the team. (can be removed after mapping)	Teams.Id
CompetitionId	The ID of the competition the team participated in. **(references Competitions)**	Teams.CompetitionId
TeamLeaderId	ID of the team leader.	Teams.TeamLeaderId
TeamName	The name of the team.	Teams.TeamName
Dataset File: Submissions (mapped with Teams on TeamId)		
SubmittedUserId	ID of the user who submitted the entry.	Teams.Submissions.SubmittedUserId
SubmissionDate	The date the submission was made.	Teams.Submissions.SubmissionDate
IsAfterDeadline	Boolean indicating if the submission was made after the deadline.	Teams.Submissions.IsAfterDeadline
PrivateScoreLeaderboardDisplay	Private score as displayed on the leaderboard.	Teams.Submissions.IsAfterDeadline
PublicScoreLeaderboardDisplay	Public score as displayed on the leaderboard.	Teams.Submissions.IsAfterDeadline

Examples of Loaded Document Model

Users:


```
_id: ObjectId('672d28147810bd3e39c68d9d')
Id : 368
UserName : "antgoldbloom"
DisplayName : "Anthony Goldbloom"
RegisterDate : 2010-01-20T00:00:00.000+00:00
PerformanceTier : 2
Country : "United States"
▾ Organizations : Array (1)
  ▾ 0: Object
      UserId : ObjectId('672d28147810bd3e39c68d9d')
      OrganizationId : ObjectId('672d27f17810bd3e39c0027a')
      JoinDate : 2020-03-15T00:00:00.000+00:00
▾ Followers : Array (6)
  ▾ 0: Object
      UserId : ObjectId('672d28147810bd3e39c68d9d')
      FollowingUserId : ObjectId('672d28147810bd3e39c77bfb')
      CreationDate : 2018-08-07T00:00:00.000+00:00
  ▸ 1: Object
  ▸ 2: Object
  ▸ 3: Object
  ▸ 4: Object
  ▸ 5: Object
▾ Achievements : Array (4)
  ▾ 0: Object
      UserId : ObjectId('672d28147810bd3e39c68d9d')
      AchievementType : "Competitions"
      Tier : 1
      TierAchievementDate : "07/15/2016"
      Points : 43
      CurrentRanking : NaN
      HighestRanking : 75
      TotalGold : 0
      TotalSilver : 0
      TotalBronze : 0
  ▸ 1: Object
  ▸ 2: Object
  ▸ 3: Object
```

Teams:


```
_id: ObjectId('6723ef14beb18ca7a4628a01')
Id : 499
CompetitionId : ObjectId('6723e59ebeb18ca7a423f402')
TeamLeaderId : 663
TeamName : "Bwaas"
▾ Submissions : Array (1)
  ▾ 0: Object
      SubmittedUserId : 663
      TeamId : ObjectId('6723ef14beb18ca7a4628a01')
      SubmissionDate : "05/01/2010"
      IsAfterDeadline : false
      PublicScoreLeaderboardDisplay : 47.11539
      PrivateScoreLeaderboardDisplay : 50
```


Datasets:


```
_id: ObjectId('672d35267810bd3e39ff223d')
Id : 10
CreatorUserId : ObjectId('672d28147810bd3e39c68fd4')
ForumId : ObjectId('672d27f17810bd3e39c00737')
CreationDate : 2015-09-11T01:56:00.000+00:00
LastActivityDate : 2018-02-06T00:00:00.000+00:00
TotalViews : 207984
TotalDownloads : 19261
TotalVotes : 301
TotalKernels : 341
▾ DatasetTags : Array (3)
  ▾ 0: Object
      DatasetId : ObjectId('672d35267810bd3e39ff223d')
      TagId : ObjectId('672d28107810bd3e39c67484')
  ▸ 1: Object
  ▸ 2: Object
```

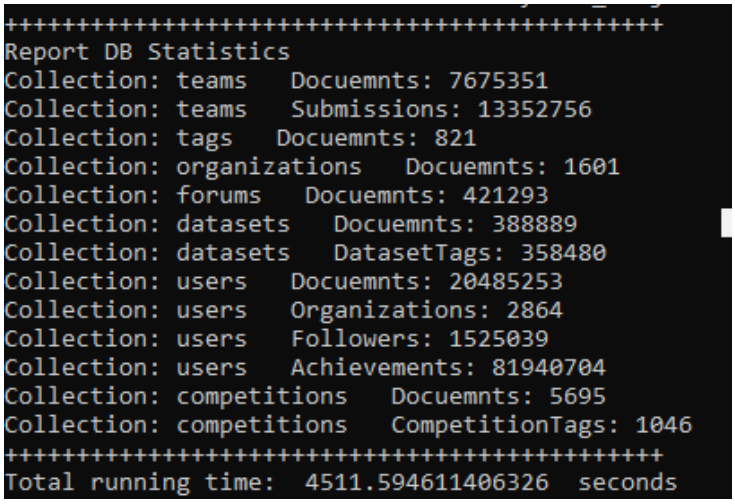

Loading Data into Document Model

The code to load data is provided in “mongo_app.py” in the root directory, and the instructions for code execution are provided in “README.md.” The dataset requires some filtering before insertion. This can be achieved by running “clean_files.py.”

After the successful execution, load data into the document model. The inserted document records are shown in the following screenshot. The records for all collections and sub-documents in respective collections match with the inserted records in Phase I. There is a document size limit error while inserting two chunks of data in inserting submission inside the team's collection. We believe some documents do

not meet the MongoDB document size limit. However, those records were correctly entered in the relational database. We can explore more on the validity of those documents in Phase III.

In general, the code takes longer to load data in MongoDB (around 215 minutes) as compared to the relational model (around 90 minutes). We feel that the checks to ensure that correct records are inserted are a bit slower in MongoDB than in relational databases. In addition, there is an additional step to map document IDs read from the file to map those to MongoDB-assigned document IDs. In case it is needed to remove those additional attributes such as IDs (primary) and referenced keys to be removed from the database, the program provided in "mongo_rem_keys.py" can be executed.



Querying Relational Model

We explain five created queries to report competition, achievements, and engagement statistics from the inserted data from Phase I. We use complex SQL queries and multiple tables related to user activities, competitions, dataset creation, and achievements. These queries show interesting insights into the created database. The code for running queries is provided in "execute_queries.py" in the root folder, while queries are provided in "queries.py" (named as all_queries), inside the "sql" subfolder.

Then, we created indexes to improve the devised queries. We report execution time for queries before and after creating indexes. Each query showed an improved execution based on the columns involved in joins, filters, aggregations, and sorting. We briefly explain the result of each query and the effect of indexes. The instruction to restart the database server to observe the effect of indexing is provided in the "README.md" file. The code for creating indexes is provided in "execute_indexes.py" in the root folder, while index queries are provided in "queries.py" (named index_queries), inside the "sql" subfolder.

For repeated trials, indexes can also be deleted using the code provided in "execute_drop_index.py" in the root folder, while index drop queries are provided in "queries.py" (named as drop_queries), inside the "sql" subfolder.

Query 1. Top Competition Tags by User Medals

This query analyzes the most popular competition tags based on user engagement and achievement metrics. Specifically, it calculates the number of active users, as well as the total count of gold, silver, and bronze medals awarded for each tag. The purpose of this query is to identify which competition topics have the highest levels of user participation and achievement, thereby revealing popular areas of engagement.

The results show that the image-data tag has the highest level of engagement, with 103 active users and a combined total of 79 medals across all levels. Other notable tags include animals, with a strong showing in bronze medals, and tabular data, with a substantial active user count of 149. Tags like text data and internet also appear, though with lower medal counts, indicating varied levels of engagement across topics. The query was executed in 4.87 seconds without indexing. This baseline performance will serve as a reference for evaluating the impact of subsequent indexing strategies.

```
Executing query: Top Competition Tags by User Medals

(TagName, ActiveUsers, GoldMedals, BronzeMedals, TotalMedals)
('image-data', 103, 8, 13, 58, 79)
('animals', 14, 1, 4, 44, 49)
('automobiles', 5, 6, 6, 9, 21)
('tabular-data', 148, 3, 7, 8, 18)
('text-data', 42, 0, 8, 5, 13)
('internet', 15, 0, 6, 3, 9)
('binary-classification', 48, 1, 2, 4, 7)
('nlp', 19, 0, 2, 4, 6)
('biology', 13, 1, 0, 3, 4)
('audio-data', 8, 2, 2, 0, 4)
Top Competition Tags by User Medals completed in 4.87 seconds.
```

Query 2. Top Users by Followers and Achievements

This query identifies the most followed users on the platform and examines their achievements in competitions. Specifically, it retrieves each user's follower count and aggregates their medal counts (gold, silver, bronze) to assess their level of achievement. This analysis provides insights into high-profile users who not only have a large following but have also demonstrated significant success in competitions.

The results indicate that Santiago Mota is the most followed user with 3,880 followers and a total of 86 medals, showcasing a well-balanced achievement profile. Interestingly, ARPAN CHOUDHURY 98 has a high follower count (899) but no recorded medals, indicating a potentially influential user with a non-competitive focus. The query was executed in 11.91 seconds without indexing. This execution time will be compared to the post-indexed timing to evaluate the effectiveness of the chosen indexing strategy.

The indexing approach for this query will focus on columns involved in joins and aggregations, particularly in the user followers and user-achievements tables, to improve efficiency.

```
Executing query: Top Users by Followers and Achievements

(UserName, FollowerCount, SilverMedals, BronzeMedals, TotalMedals)
('Santiago Mota', 3880, 2, 12, 72, 86)
('Yasir Hussein Shakir', 3304, 1, 13, 281, 295)
('Márcio Santos', 3162, 0, 1, 38, 39)
('PAVAN KUMAR D', 1766, 60, 42, 692, 794)
('asaniczka', 1723, 37, 26, 1082, 1145)
('Vitaliy Lyalin', 1210, 0, 0, 1, 1)
('Firat Gonen', 908, 76, 42, 711, 829)
('OH SEOK KIM', 899, 23, 20, 1300, 1343)
('ARPAN CHOUDHURY 98', 899, 0, 0, 0, 0)
('Carl McBride Ellis', 839, 197, 240, 1880, 2317)
Top Users by Followers and Achievements completed in 11.91 seconds.
```

Query 3. User Achievements and Dataset Creation Patterns by Competition Topic.

This query explores the relationship between user achievements in competitions and their dataset creation patterns across different topics. It identifies users who have participated in competitions, counts the datasets they have created, and analyzes the engagement metrics (average views, downloads, and votes) for each dataset. The results are segmented by competition topics, revealing which topics attract high-achieving users who also contribute datasets.

The results show varied engagement across different topics. Konrad Banachewicz, for example, has created 166 datasets in the biology category with an impressive total medal count of 4101120. His datasets have an average of 2631 views, indicating significant community interest. Similarly, Eu Jin Lok has achieved high average views (82,383) for datasets in the time-series-analysis category, though with fewer datasets. Psi, active in audio-data, shows moderate dataset creation but consistent engagement metrics, suggesting topic specialization. The query took 22.58 seconds to execute without indexing. This time reflects the complexity of joining multiple tables and aggregating data for each user by competition topic. The indexing strategy for this query will target columns in user achievements, datasets, and submissions, which are heavily involved in joins and aggregations, aiming to reduce the execution time in subsequent tests.

```
Executing query: User Achievements and Dataset Creation Patterns by Competition Topic

(UserName, CompetitionsParticipated, DatasetsCreated, CompetitionTopic, TotalMedals, AvgDatasetViews, AvgDatasetDownloads, AvgDatasetVotes)
('Konrad Banachewicz', 1, 160, 'biology', 4101120, Decimal('2631.6187500000000000'), Decimal('258.2812500000000000'), Decimal('14.5562500000000000')
('Psi', 1, 11, 'audio-data', 39094, Decimal('965.0909090909090909'), Decimal('19.2727272727272727'), Decimal('9.1818181818181818'))
('Eu Jin Lok', 1, 3, 'time-series-analysis', 11169, Decimal('80383.333333333333'), Decimal('19529.666666666667'), Decimal('121.6666666666666667'))
User Achievements and Dataset Creation Patterns by Competition Topic completed in 22.58 seconds.
```

Query 4. Competition Tags with Highest Engagement by Submissions.

This query identifies competition tags with the highest engagement, measured by the total number of submissions and average scores (public and private) associated with each tag. By examining the total submissions and average scores, this query highlights the competition topics that attract substantial user participation and engagement, as indicated by frequent submissions and scoring metrics.

The results indicate that tabular data is the most engaged competition tag, with 997 submissions across 75 competitions and a high average private score of 10,805.4. Other notable tags include multiclass-classification and image-data, both with significant submission counts and moderate average scores, suggesting sustained interest in these topics. Some tags, such as internet and binary-classification, show lower public and private scores despite a fair number of submissions, which may indicate differing competition structures or scoring challenges. This query executed efficiently in 1.28 seconds without indexing, likely due to the relatively straightforward structure of joins and aggregations in comparison to previous queries. Although the baseline performance is already optimal, indexing strategies may still enhance responsiveness for larger datasets, especially by indexing competition tags and submissions columns involved in joins and aggregations. This initial timing provides a strong reference for evaluating any marginal benefits of indexing in a high-engagement dataset context.

```
Executing query: Competition Tags with Highest Engagement by Submissions

(TagName, NumberOfCompetitions, TotalSubmissions, AvgPublicScore, AvgPrivateScore)
('tabular-data', 75, 997, Decimal('278.21'), Decimal('10805.38'))
('multiclass-classification', 33, 643, Decimal('93.65'), Decimal('468.53'))
('image-data', 57, 488, Decimal('122.22'), Decimal('616.14'))
('internet', 11, 255, Decimal('0.86'), Decimal('0.89'))
('binary-classification', 22, 200, Decimal('0.61'), Decimal('50000.54'))
('geography', 1, 162, Decimal('0.47'), Decimal('0.48'))
('text-data', 18, 162, Decimal('1.54'), Decimal('1.61'))
('time-series-analysis', 7, 139, Decimal('0.76'), Decimal('0.78'))
('card-games', 1, 132, Decimal('0.89'), Decimal('0.89'))
('marketing', 3, 114, Decimal('0.47'), Decimal('0.48'))
Competition Tags with Highest Engagement by Submissions completed in 1.28 seconds.
```

Query 5. Dataset Tag Engagement by High-Achieving Users.

This query examines the engagement levels of datasets created under specific tags by users who are considered high achievers in competitions. It filters for users with notable achievements (e.g., those with at least five medals) and analyzes their dataset contributions by tag. The metrics calculated include the average views, downloads, and votes for datasets under each tag, which provide insight into the most popular and impactful topics among high-achieving users.

The results highlight that the tidy verse tag has high engagement, with an average of 64,185 views per dataset. Other popular tags include linguistics, simulation-games, and websites, each showing substantial average views and download counts, indicating strong community interest. The tag healthcare, with the highest total datasets (840) and high average views, shows that it's a broadly impactful area, attracting

both high-achieving contributors and active engagement. The query was executed in 6.34 seconds without indexing. The execution time reflects the complexity of aggregating engagement metrics for many datasets under specific tags and filtering by user achievement levels. Indexing will focus on optimizing the joins and filtering in tables like user achievements and dataset tags to potentially improve execution time by reducing the scan range for high-achievement filters and tag-based aggregations.

```
Executing query: Dataset Tag Engagement by High-Achieving Users

(DatasetTag, TotalDatasets, HighAchievingUsers, AvgViews, AvgDownloads, AvgVotes)
('tidyverse', 33, 14, Decimal('64185.569767441860'), Decimal('9431.3023255813953488'), Decimal('145.6046511627906977'))
('simulation-games', 51, 40, Decimal('54848.064220183486'), Decimal('3354.1009174311926606'), Decimal('1352.6330275229357798'))
('linguistics', 273, 123, Decimal('44236.463375796178'), Decimal('5734.9984076433121019'), Decimal('95.2659235668789809'))
('web-sites', 178, 113, Decimal('36770.337320574163'), Decimal('5684.3062200956937799'), Decimal('108.9736842105263158'))
('aviation', 114, 91, Decimal('31790.535156250000'), Decimal('4871.3203125000000000'), Decimal('66.2109375000000000'))
('popular-culture', 223, 93, Decimal('31471.786713286713'), Decimal('4469.7604895104895105'), Decimal('77.6101398601398601'))
('diabetes', 207, 186, Decimal('29972.600441501104'), Decimal('5726.5474613686534216'), Decimal('66.1015452538631347'))
('healthcare', 840, 428, Decimal('28859.798130841121'), Decimal('3902.6981308411214953'), Decimal('64.7336448598130841'))
('multiclass-classification', 560, 361, Decimal('28160.559190031153'), Decimal('3776.2788161993769470'), Decimal('62.9859813084112150'
('bigquery', 118, 35, Decimal('28018.825000000000'), Decimal('195.2750000000000000'), Decimal('81.7187500000000000'))
('binary-classification', 580, 404, Decimal('27608.801843317972'), Decimal('4321.0337941628264209'), Decimal('70.5384024577572965'))
Dataset Tag Engagement by High-Achieving Users completed in 6.34 seconds.
```

Indexing Strategy Summary

Indexes marked with, **Tested, but not created,** mean that those indexes were created but did not add significant value to query execution. Hence, those are later omitted.

Query 1: Top Competition Tags by User Medals

- Index on competitiontags (TagId, CompetitionId): This index optimizes joins between the competitiontags, tags, and competition tables. By indexing on both TagId and CompetitionId, this index helps reduce the time required to filter and join these tables.
- Index on user achievements (UserId, TotalGold, TotalSilver, TotalBronze): This composite index optimizes joins and aggregations in the user achievements table, where medal counts are calculated. This enables efficient retrieval of medal counts for each user across the different competition tags.

```
Executing query: Top Competition Tags by User Medals

(TagName, ActiveUsers, GoldMedals, BronzeMedals, TotalMedals)
('image-data', 103, 8, 13, 58, 79)
('animals', 14, 1, 4, 44, 49)
('automobiles', 5, 6, 6, 9, 21)
('tabular-data', 148, 3, 7, 8, 18)
('text-data', 42, 0, 8, 5, 13)
('internet', 15, 0, 6, 3, 9)
('binary-classification', 48, 1, 2, 4, 7)
('nlp', 19, 0, 2, 4, 6)
('biology', 13, 1, 0, 3, 4)
('audio-data', 8, 2, 2, 0, 4)
Top Competition Tags by User Medals completed in 0.09 seconds.
```

Query 2: Top Users by Followers and Achievements

- Index on user followers (UserId, FollowingUserId): This index is critical for efficiently counting followers per user and filtering users based on follower count. The FollowingUserId attribute is included to facilitate joins, enabling faster retrieval of follower counts.

```
Executing query: Top Users by Followers and Achievements

(UserName, FollowerCount, SilverMedals, BronzeMedals, TotalMedals)
('Santiago Mota', 3880, 2, 12, 72, 86)
('Yasir Hussein Shakir', 3304, 1, 13, 281, 295)
('Márcio Santos', 3162, 0, 1, 38, 39)
('PAVAN KUMAR D', 1766, 60, 42, 692, 794)
('asaniczka', 1723, 37, 26, 1082, 1145)
('Vitaliy Lyalin', 1210, 0, 0, 1, 1)
('Firat Gonen', 908, 76, 42, 711, 829)
('OH SEOK KIM', 899, 23, 20, 1300, 1343)
('ARPAN CHOUDHURY 98', 899, 0, 0, 0, 0)
('Carl McBride Ellis', 839, 197, 240, 1880, 2317)
Top Users by Followers and Achievements completed in 0.46 seconds.
```

Query 3: User Achievements and Dataset Creation Patterns by Competition Topic. (almost similar performance).

- **Tested, but not created.** Index on datasets (CreatorUserId): This index can optimize the join between datasets and users, where datasets are filtered by the creator.
- **Tested, but not created.** Index on submissions (SubmittedUserId, TeamId): This index can support joins between submissions and competitions by filtering submissions linked to each competition.

```
Executing query: User Achievements and Dataset Creation Patterns by Competition Topic

(UserName, CompetitionsParticipated, DatasetsCreated, CompetitionTopic, TotalMedals, AvgDatasetV
('Konrad Banachewicz', 1, 160, 'biology', 4101120, Decimal('2631.6187500000000000'), Decimal('25
('Psi', 1, 11, 'audio-data', 39094, Decimal('965.0909090909090909'), Decimal('19.272727272727272
('Eu Jin Lok', 1, 3, 'time-series-analysis', 11169, Decimal('80383.333333333333'), Decimal('1952
User Achievements and Dataset Creation Patterns by Competition Topic completed in 22.46 seconds.
```

Query 4: Competition Tags with Highest Engagement by Submissions

- Index on submissions (TeamId, PublicScoreLeaderboardDisplay, PrivateScoreLeaderboardDisplay): This composite index allows the query to efficiently aggregate scores by TeamId, facilitating calculations for both public and private leaderboard scores. This improves the performance of aggregating and calculating average scores for submissions.
- Index on competitiontags (TagId, CompetitionId): This index supports the joining of tags and competitions by filtering and joining on TagId and CompetitionId. It ensures efficient retrieval of relevant competition tags.

```
Executing query: Competition Tags with Highest Engagement by Submissions

(TagName, NumberOfCompetitions, TotalSubmissions, AvgPublicScore, AvgPrivateScore)
('tabular-data', 75, 997, Decimal('278.21'), Decimal('10805.38'))
('multiclass-classification', 33, 643, Decimal('93.65'), Decimal('468.53'))
('image-data', 57, 488, Decimal('122.22'), Decimal('616.14'))
('internet', 11, 255, Decimal('0.86'), Decimal('0.89'))
('binary-classification', 22, 200, Decimal('0.61'), Decimal('50000.54'))
('geography', 1, 162, Decimal('0.47'), Decimal('0.48'))
('text-data', 18, 162, Decimal('1.54'), Decimal('1.61'))
('time-series-analysis', 7, 139, Decimal('0.76'), Decimal('0.78'))
('card-games', 1, 132, Decimal('0.89'), Decimal('0.89'))
('marketing', 3, 114, Decimal('0.47'), Decimal('0.48'))
Competition Tags with Highest Engagement by Submissions completed in 0.20 seconds.
```

Query 5: Dataset Tag Engagement by High-Achieving Users

- **Tested, but not created.** Index on user achievements (UserId, TotalGold, TotalSilver, TotalBronze): This index filters high-achieving users (those with five or more medals) and improves joins with users. It enables efficient retrieval of users who meet achievement thresholds, reducing the scanning time in high-achievement filtering.
- Index on dataset tags (DatasetId, TagId): This index supports joins between datasets and tags, optimizing the retrieval of dataset tags and allowing aggregations by tag. This improves the performance of tag-based engagement analysis for datasets created by high-achieving users.

```
Executing query: Dataset Tag Engagement by High-Achieving Users

(DatasetTag, TotalDatasets, HighAchievingUsers, AvgViews, AvgDownloads, AvgVotes)
('tidyverse', 33, 14, Decimal('64185.569767441860'), Decimal('9431.3023255813953488'), Decimal('145.6046511627906977'))
('simulation-games', 51, 40, Decimal('54848.064220183486'), Decimal('3354.1009174311926606'), Decimal('1352.6330275229357798'))
('linguistics', 273, 123, Decimal('44236.463375796178'), Decimal('5734.9984076433121019'), Decimal('95.2659235668789809'))
('web-sites', 178, 113, Decimal('36770.337320574163'), Decimal('5684.3062200956937799'), Decimal('108.9736842105263158'))
('aviation', 114, 91, Decimal('31790.535156250000'), Decimal('4871.3203125000000000'), Decimal('66.2109375000000000'))
('popular-culture', 223, 93, Decimal('31471.786713286713'), Decimal('4469.7604895104895105'), Decimal('77.6101398601398601'))
('diabetes', 207, 186, Decimal('29972.600441501104'), Decimal('5726.5474613686534216'), Decimal('66.1015452538631347'))
('healthcare', 840, 428, Decimal('28859.798130841121'), Decimal('3902.6981308411214953'), Decimal('64.7336448598130841'))
('multiclass-classification', 560, 361, Decimal('28160.559190031153'), Decimal('3776.2788161993769470'), Decimal('62.9859813084112150
('bigquery', 118, 35, Decimal('28018.825000000000'), Decimal('195.2750000000000000'), Decimal('81.7187500000000000'))
('binary-classification', 580, 404, Decimal('27608.801843317972'), Decimal('4321.0337941628264209'), Decimal('70.5384024577572965'))
Dataset Tag Engagement by High-Achieving Users completed in 6.20 seconds.
```

Sr	Without Indexes	With Indexes	Improvement %
Query 1	4.87 seconds	0.09 seconds	98%
Query 2	11.91 seconds	0.46 seconds	96%
Query 3	22.58 secondscls	No benefit seen	Same
Query 4	1.28 seconds	0.20 seconds	84%
Query 5	6.34 seconds	6.20 seconds	2%

Functional Dependencies and Normalization

1. Users Table

Key FDs:

- {id} → {username}
- {username} → {id}
- {id, displayname} → {username}

Validity:
- {id} → {username} is valid because id is the primary key and uniquely identifies the username.
- {username} → {id} is valid if the username is unique, which aligns with typical user systems.
- {id, displayname} → {username} assumes displayname is unique for each user, which may not hold without an explicit uniqueness constraint.

Normalization:
- 1NF: Atomic attributes are present.
- 2NF: All non-prime attributes depend only on the primary key ID.
- 3NF: No transitive dependencies exist unless constraints on displayname introduce dependencies.

2. Tags Table

Key FDs:
- {id} → {name, slug, fullpath, description}
- {name} → {id, slug}
- {slug} → {id}

Validity:
- {id} → {name, slug, fullpath, description} is valid due to the primary key.
- {name} → {id, slug} assumes `name` is unique, which may not hold universally.
- {slug} → {id} is valid if `slug` is unique, as expected for tag URLs.

Normalization:
- 1NF: Atomic attributes.
- 2NF: All non-prime attributes depend on the primary key ID.
- 3NF: Transitive dependencies like {name} → {slug} need to be verified against uniqueness.

3. Forums Table

Key FDs:
- {parentforumid, title} → {id}

Validity:
- {parentforumid, title} → {id} is valid if titles are unique within a parent forum.

Normalization:
- 1NF: Atomic attributes.
- 2NF: The composite key ensures full dependency.
- 3NF: No transitive dependencies if the title is unique within parentforumid.

4. Organizations Table

Key FDs:
- {id} → {name, slug, creationdate}
- {name} → {id}

Validity:
- {id} → {name, slug, creationdate} is valid due to the primary key.
- {name} → {id} assumes the name is unique, which aligns with most organization systems.

Normalization:
- Fully normalized (1NF, 2NF, and 3NF).

5. UserOrganizations Table

Key FDs:
- {userid, organizationid} → {id, joindate}

Validity:
- {userid, organizationid} → {id, joindate} is valid if userid and organizationid uniquely identify a membership.

Normalization:
- Fully normalized (1NF, 2NF, and 3NF).

6. Datasets Table

Key FDs:
- {id} → {forumid, creatoruserid, creationdate}

Validity:
- {id} → {forumid, creatoruserid, creationdate} is valid due to the primary key.

Normalization:
- Fully normalized.

7. Competitions Table

Key FDs:
- {id} → {slug, title, forumid, enableddate}
- {slug} → {id}

Validity:
- {id} → {slug, title, forumid, enableddate} is valid because id is the primary key.
- {slug} → {id} assumes slug is unique.

Normalization:
- Fully normalized.

8. Teams Table

Key FDs:
- {teamleaderid, competitionid} → {id}

Validity:
- {teamleaderid, competitionid} → {id} is valid if a leader can only manage one team per competition.

Normalization:
- Fully normalized.

9. Submissions Table

Key FDs:
- {id} → {teamid, submitteduserid, submissiondate}

Validity:
- {id} → {teamid, submitteduserid, submissiondate} is valid due to the primary key.

Normalization:
- Fully normalized.

10. UserAchievements Table

Key FDs:
- {userid, achievementtype} → {id}

Validity:
- {userid, achievementtype} → {id} is valid if achievements are unique for a user by type.

Normalization:
- Fully normalized.