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

This report explains the process of cleaning our dataset. The dataset we selected is Meta Kaggle (https://www.kaggle.com/datasets/kaggle/meta-kaggle). The report briefly describes itemset mining and association rules. We also evaluate the model's suitability (relational or document-oriented).

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" in the root folder.

Data Cleaning

The dataset mostly contains valid data. To ensure that we have valid representative data in our database, we clean the subset of the dataset we selected to be included in this project (from Phase I). In some cases, we do not have sufficient data to ensure the data validity, hence, we assume that missing data (e.g., nulls) is accepted in the database schema. We perform validation on the data to ensure that it follows the required format. We also ensure that only valid data is inserted into the database by checking foreign key constraints. A detailed description of the validity of each field is provided in the following tables. In the following table, we give a brief overview of selected tables (a subset of Kaggle Meta) and contained records after inserting them into the database. Sections following the table explain the attributes analyzed for each table. After cleaning and removing invalid data, we only lose 1% of the data from the original dataset.

Table Name	**Description**	**Input Records**	**After Cleaning**
Users	Information about Kaggle users.	20485253	20485253
Tags	Tags applied to competitions and datasets (our context).	821	818
Forums	Forum posts, including title and parent relationships.	421293	421077
Organizations	Organizations on Kaggle, including creation date and descriptions.	1601	1601
UserOrganizations	Links users with their affiliated organizations on Kaggle.	2950	2864
UserFollowers	Tracks who follows whom among Kaggle users.	1552414	1525039
Datasets	Dataset information, including total downloads, views, and votes.	397793	388702
DatasetTags	Associates datasets with tags for easier categorization and search.	366505	358258
Competitions	Competitions, including deadlines, rewards, and evaluation methods.	5695	5695
CompetitionTags	Links competitions to relevant tags for categorization purposes.	1046	1046
Teams	Teams for competitions, such as membership and medals won.	7675480	7445894
Submissions	Submissions to competitions, including scores and submission dates.	14801034	14128610
UserAchievements	Achievements of Kaggle users, including rankings, points, and medal counts.	81940708	81446553
	Total	127652593	126211410

Data Cleaning Overview

To reduce the complexity of handling uncleaned values, we excluded adding some attributes to the database. The file “clean_files.py” contains the code to remove some data attributes from Phase I. We deduce data types and extract null counts by analyzing each original file using the code provided in the “analyze_data.py” file. Subsequently, we took a subset of the attributes to manage the dataset reasonably for the scope of this project. For instance, we perform basic filtering to remove some less relevant attributes using the code provided in the “filter_data.py” file. Finally, as we explain below why we consider relational database over MongoDB, we perform cleaning and insertion on the data using the code provided in “insert_data_clean.py”. The initial database schema, ensuring the correct data types, size, and referential integrity constraints is executed using the “create_schema.sql” file. While some processing is applied using the “sql_final.sql” after data insertion.

Afterward, we clean the data to ensure it is valid, accurate, and complete. We also ensure that the data does not contain missing values (or empty rows), incorrect data types, and uncontrolled lengths. We overview the dataset and keep only the relevant attributes to ensure that values for a particular attribute are represented in only one place for consistency. Likewise, this also ensures the uniformity of data types and measurement units for each attribute.

Validity

We ensure that we only consider the valid data. For this, we first ensure the referential integrity constraints (including the self-referential integrity), specify the data types for each attribute and provide the appropriate data types (and size limits) to ascertain the validity. For example, for user submissions, we ensure that all data must be referenced appropriately to respective users and teams. These constraints (along with others explained below) ensure the validity of our data.

Accuracy

We check for the accuracy of data to the best of our ability. However, the attributes are not descriptive to apply general accuracy checks. We match the correct data types and examine the length of each attribute to potentially identify the inaccuracies, for example for correct data types, maximum lengths, and the values contained for maximum lengths for all attributes.

Completeness

We ensure the utmost detail for completeness. Except for a few attributes that we do not deem compulsory (e.g., country for a user, description for a tag, etc.), all other attributes are checked for completeness. For example, we modify the user display names with user names where those records are not present. In other contexts, where we do not have the additional data to fill incomplete records, we drop the records if the proportion of data is very negligible in terms of total data to ensure that the remaining data is completed. Furthermore, we ensure that all those attributes that are cleaned are required to insert data, making it necessary for future insertion to ensure data completeness.

Missing Values and Duplicates

Missing values are handled carefully and we decide about each attribute individually. More details are explained in the following table. We ensure any necessary attribute is a required attribute in the database to reduce the possibility of missing values for insertion. Thus our data does not, by any means contain empty rows. Each entry is marked with a primary key, which we expected to reduce the probability of data duplication. We separately checked for data duplication on the filtered files and did not find any duplicate records among all selected dataset files.

Incorrect Data Types

We observe incorrect data types for various fields (user references, team references, and dates). For instance, we observed the user references and team references were represented in FLOAT values, however, the primary tables contain the integer values. Therefore, to correct the data types and ensure the referential integrity constraint, we converted the data types from float to integers. In some cases, the referential integrity used the self-references (parent tag), we only convert the valid reference values and ignore the empty values (as some tags can be without parents). Likewise, we observed that some dates were missing (Tier Achievement Date) and the data type could not be matched before cleaning. Hence, we perform the date check to ensure that valid dates are inserted.

The following tables provide more details on individual attributes and the decisions taken to clean those. Valid attributes (complete, correct data type, no missing values, no inconsistent values) are cross-checked with the initial analysis, while other attributes are analyzed additionally. The decision for each attribute is explained after additional analysis.

Users - This table contains information about the users of Kaggle. After data processing and updates, the users' table contains the complete data except the country.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
UserName	Text	1	We checked that there is only one user with a missing user name. We anticipate that users are unique and would be needed to use across the Kaggle. There is only one user with the value of 'NaN' as the user name, which we assume is still a valid user as it has an associated user display name (some Arabic name) with the creation date (2014-01-27) belonging from Iran. Hence, we decide to keep this user.
DisplayName	Text	314	We checked that only a small number of users with a missing display name, hence we updated it with its user name to ensure completeness. That is the closest attribute that can be used alternatively for only very few records (314 in 20M).
RegisterDate	Date/Time	0	It's a valid attribute.
PerformanceTier	Integer	0	It's a valid attribute.
Country	Text	19034320	We assume that users can exist without a country. Hence, it is kept as is.

Tags - The table contains information about the tags used. After processing tags, the table contains the accurate data and self-reference to parent tags. (except optional description).

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
ParentTagId	Integer	10	We see that the source data has this attribute as float, however, it is a self-reference to the parent tag ID. Hence, we process the tags for converting the data to an integer. We also see that this attribute is a self-reference, and we find that only three tags do not have references to ID. As the data size is small, we can manually remove those records to ensure further dataset is cleaned and reference the correct PK.
Name	Text	0	It's a valid attribute.
Slug	Text	0	It's a valid attribute.
FullPath	Text	0	It's a valid attribute.
Description	Text	675	We assume that tags can exist without a description. Hence, it is kept as is.
DatasetCount	Integer	0	It's a valid attribute.
CompetitionCount	Integer	0	It's a valid attribute.
KernelCount	Integer	0	It's a valid attribute.

Forums - This table holds information about forum posts made on Kaggle.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
ParentForumId	Integer	27	We see that the source data has this attribute as float, however, it is an integer. Hence, we process the forums for converting the data to an integer. We can assume that there is a self-reference, however, we analyzed the data and found substantial values that do not have valid parents. So, we assume that it is not a self-reference and hence ignore to enforce it, otherwise, this will result in significant data not being inserted.
Title	Text	189	We assume that forums can exist without a title. However, we find that only a few (189 in 420K) rows do not have a title, hence, we clean for empty titles before inserting data.

Organizations - This table contains information about organizations associated with Kaggle. This table has almost complete information.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
Name	Text	0	It's a valid attribute.
Slug	Text	0	It's a valid attribute.
CreationDate	Date/Time	0	It's a valid attribute.
Description	Text	1188	We assume that organizations can exist without a description. Hence, it is kept as is.

UserOrganizations - This table links users with their organizations.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
UserId (FK)	Integer	0	We validate and remove the user organization records that do not have a valid user reference.
OrganizationId (FK)	Integer	0	We validate and remove the user organization records that do not have a valid organization reference.
JoinDate	DateTime	0	It's a valid attribute.

UserFollowers - This table provides data about relationships between users, specifically, who is following whom.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
UserId (FK)	Integer	0	We validate and remove the user followers' records that do not have a valid user reference.
FollowingUserId (FK)	Integer	0	We validate and remove the user followers' records that do not have a valid user reference.
CreationDate	DateTime	0	It's a valid attribute.

Datasets - This table contains information about datasets uploaded to Kaggle.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
CreatorUserId (FK)	Integer	0	We validate and remove the dataset records that do not have a valid user reference.
ForumId (FK)	Integer	0	We validate and remove the dataset records that do not have a valid forum reference.
CreationDate	DateTime	0	It's a valid attribute.
LastActivityDate	Date/Time	0	It's a valid attribute.
TotalViews	Integer	0	It's a valid attribute.
TotalDownloads	Integer	0	It's a valid attribute.
TotalVotes	Integer	0	It's a valid attribute.
TotalKernels	Integer	0	It's a valid attribute.

DatasetTags - This table links datasets to their associated tags.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
DatasetId (FK)	Integer	0	We validate and remove the dataset tag records that do not have a valid dataset reference.
TagId (FK)	Integer	0	We validate and remove the dataset tag records that do not have a valid tag reference.

Competitions - The table provides essential information about the setup, rules, and logistics of Kaggle competitions.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
Slug	Text	0	It's a valid attribute.
Title	Text	0	It's a valid attribute.
ForumId (FK)	Integer	0	We validate and remove the competition records that do not have a valid forum reference.
EnabledDate	Date/Time	0	It's a valid attribute.
DeadlineDate	Date/Time	0	It's a valid attribute.
EvaluationAlgorithmName	Text	1	We remove the evaluation algorithm name as there is only 1 missing value. There is no relevant column to replace this with.
MaxTeamSize	Integer	0	It's a valid attribute.
NumPrizes	Integer	0	It's a valid attribute.
TotalTeams	Integer	0	It's a valid attribute.
TotalCompetitors	Integer	0	It's a valid attribute.
TotalSubmissions	Integer	0	It's a valid attribute.

CompetitionTags - This table links competitions to their associated tags, categorizing competitions into relevant topics.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
CompetitionId (FK)	Integer	0	We validate and remove the competition tag records that do not have a valid competition reference.
TagId (FK)	Integer	0	We validate and remove the competition tag records that do not have a valid tag reference.

Teams - This table contains information about teams participating in Kaggle competitions.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
CompetitionId (FK)	Integer	0	We validate and remove the team records that do not have a valid competition reference.
TeamLeaderId (FK)	Integer	26306	We see that the source data has this attribute as float, however, it is an integer. Hence, we process the teams for converting the data to an integer. We can assume that there is a reference to users as team leaders, and we see only a few missing values (25K of 7.6M). Hence, we enforce this reference constraint and validate data before insertion to avoid any issues.
TeamName	Text	845	We assume that teams can exist without a team name. However, we find that only a few (845 in 76M) rows do not have a team name, hence, we clean for empty team names before inserting data.

Submissions - This table tracks submissions to Kaggle competitions.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
SubmittedUserId (FK)	Integer	1346	We see that the source data has this attribute as float, however, it is an integer. Hence, we process the submissions for converting the data to an integer. We can assume that there is a reference to users as submitters, and we see only a few missing values (1346 of 14M). Hence, we enforce this reference constraint and validate data before insertion to avoid any issues.
Team (FK)	Integer	0	We validate and remove the submission records that do not have a valid team reference.
SubmissionDate	Date/Time	0	It's a valid attribute.
IsAfterDeadline	Boolean	0	It's a valid attribute.
PublicScoreLeaderboardDisplay	Float	579285	As almost half of the data is not present. We assume that it is a default behavior. Also, there is no relevant field to extract this data.
PrivateScoreLeaderboardDisplay	Float	579285	As almost half of the data is not present. We assume that it is a default behavior. Also, there is no relevant field to extract this data.

UserAchievements - This table captures the achievements earned by users on the platform.

Field Name	Data Type	Null Count	Description
Id (PK)	Integer	0	It's a valid attribute.
UserId (FK)	Integer	1346	We validate and remove the achievement records that do not have a valid user reference.
AchievementType	Text	0	It's a valid attribute.
Tier	Integer	0	It's a valid attribute.
TierAchievementDate	Date/Time	494184	We assume that teams can exist without valid achievement data. However, we find that only a few (494K in 81M) rows do not have achievement data, hence, we clean for empty team names before inserting data.
Points	Integer	0	It's a valid attribute.

CurrentRanking	Integer	81417741	As almost all of the data is not present. We assume that it is a default behavior. Also, there is no relevant field to extract this data.
HighestRanking	Integer	81417236	As almost all of the data is not present. We assume that it is a default behavior. Also, there is no relevant field to extract this data.
TotalBronze	Integer	0	It's a valid attribute.
TotalGold	Integer	0	It's a valid attribute.
TotalSilver	Integer	0	It's a valid attribute.

Optimal Model for Our Dataset

In each of the previous phases, we saw the differences in inserting the same data set into relational and document databases. In this section, we provide a brief comparison of both and reasons for choosing the right database for our case.

Relational Model (PostgreSQL) advantages for this case:

- Efficient processing to handle complex joins between multiple related entities (Users, Organizations, Tags)
- Easier and intuitive enforcement of referential integrity through foreign keys
- Efficient querying performance with aggregation (we observed much faster query processing in the relational database than in the document database).
- Transaction consistency is important for maintaining relationships (however, as the data is not changing at the moment. With the size of the dataset, there could be a possibility that this could be an issue with document databases).

Document Model (MongoDB) advantages:

- Flexible schema for varying structures (we used effective document model handling to represent each entity with its associated data, however, querying the data takes more time).
- Better for storing hierarchical data with less collection to maintain as compared to relations, making it easier to understand the overall data hierarchy.
- It is easier to scale data horizontally. For example, by adding new attributes, or documents.
- No need for complex joins when accessing single-entity data as we can store all data related to a single entity in the same collection.

For this specific implementation, the **Relational Model** is a better fit because:

1. The data has well-defined relationships, for example;
    - Users belong to organizations
    - Competitions and datasets have tags and are created by users
    - Submissions are created by teams that compete in competitions and have leaders or individuals (who are users) for any entry
    - Clear foreign key relationships exist among most entities. The obvious relationships are easier to handle in the relational model.

2. The queries require joins to find relationships, and SQL is inherently better for this user case.
3. Data integrity is crucial. We anticipate ACID properties to hold, for instance,
    - Tag relationships must be maintained
    - User-organization relationships need referential integrity
    - Competition and dataset attributes must remain consistent
4. The analysis requires aggregations and grouping operations, which are SQL's strengths.

While MongoDB could handle this data, the relational nature of the associations and the need for complex joins makes PostgreSQL a more suitable choice for this specific implementation. In addition, over the previous two phases, we observed that handling data insertion, adding referential constraints, and querying is much faster in PostgreSQL than MongoDB in our case. Collectively, these observations lead us to choose the relational model for this task. Hence, the code provided translates the dataset into the relational model and performs the item and association set mining using that model.

Itemset Mining

In this implementation, transactions are defined differently for three types of analyses. The code can be executed using “itemset_mining.py”.

1. Competition Tags Analysis:
    - Each competition is treated as a transaction
    - Items are the tags associated with that competition
    - Parameters used: min_support=0.01, min_confidence=0.5
    - This captures which tags frequently co-occur in competitions
2. Dataset Tags Analysis:
    - Each dataset is treated as a transaction
    - Items are the tags associated with that dataset
    - Parameters used: min_support=0.005, min_confidence=0.4
    - Lower thresholds suggest more diverse tag combinations in datasets
3. User Organizations Analysis:
    - Each user is treated as a transaction
    - Items are the organizations they belong to
    - Parameters used: min_support=0.01, min_confidence=0.4

The create_transactions method converts the comma-separated items into sets. The implementation uses the Apriori algorithm to discover:

1. Frequent Itemsets:
    - Found using a level-wise approach starting with 1-itemsets
    - Support is calculated as count/total_transactions
    - Only itemsets meeting the minimum support threshold are kept
2. Association Rules:
    - Generated from frequent itemsets
    - Metrics calculated: support, confidence, and lift
    - Rules are filtered based on minimum confidence

Result Discussion

1. Competition Tags Analysis:
    - 440 competitions were analyzed
    - The most frequent individual tag is "tabular-data" (40% of competitions)
    - Strong rules discovered include:
        i. Beginner competitions are strongly associated with tabular data (confidence=1.0)
        ii. Banking competitions are very likely to use tabular data (confidence=0.91)
    - High average participation: 1,312 teams and 1,528 competitors per competition
2. Dataset Tags Analysis:
    - Much larger scale: 171,990 datasets analyzed
    - Most common tags:
        i. "pre-trained-model" (14.2%)
        ii. "business" (13.1%)
        iii. "natural-and-physical-sciences" (7.7%)
    - Interesting rules found:
        i. Strong association between "data-cleaning" and "marketing-analytics" (lift=16.75)
        ii. "business" and "finance" frequently co-occur (lift=18.56)
    - Average dataset engagement: 4,201 views and 536 downloads
3. User Organizations Analysis:
    - 2,066 user-organization relationships analyzed
    - Key findings:
        i. OpenDataScience [ods.ai] is the most popular (8.4% of users)
        ii. Perfect association between IIT KHARAGPUR and SPARK4AI (confidence=1.0, lift=39.73)
        iii. More fragmented membership pattern compared to tags

Note: Instructions to run the code are provided in the README.md file in the root folder.