Kiva Crowdfunding Database Analysis

Poojitha Katta
Department of Applied Data Science
San Jose State University
San Jose, United States
poojitha.katta@sjsu.edu

Purnima Bhukya
Department of Applied Data Science
San Jose State University
San Jose, United States
purnima.bhukya@sjsu.edu

Deepali Zutshi
Department of Applied Data Science
San Jose State University
San Jose, United States
deepali.zutshi@sjsu.edu

Yasaman Emami
Department of Applied Data Science
San Jose State University
San Jose, United States
yasaman.emami@sjsu.edu

Abstract—Crowdfunding is a platform that is gaining a lot of importance where one can seek investments from funders online. A platform where ideas can come to life. The project aims to create a database for one such crowdfunding platform-Kiva, to analyze and evaluate factors that influence various aspects of a crowdfunded project and draw conclusions about them. With understanding the perfect combination of the available database platforms, be it SQL or NoSQL, and provide a good database management system that can enhance the platform. The project aims at finding out the perfect trade off among choosing the best possible database for this crowdfunding application.

Keywords—NoSQL, SQL Crowdfunding, Database, Kiva

I. Introduction

Crowdfunding is a system that enables individuals or ventures to seek small investments, contributions, or loans from a variety of funders online. Kiva is a crowdfunding platform that offers a new financing channel for small and micro businesses as well as individuals. The aim of the project is to build a database management system for Kiva platform to analyze the factors that influence crowdfunded projects by estimating the welfare level of partners in specific regions, based on shared financial and demographic aspects. Technologies such as MongoDB would be used to create the database management system and perform analysis of the data. Using Python, a powerful programming tool, we can observe, understand, and draw conclusions on better funded categories, borrowing patterns and regional analysis from the data. The system can be deployed on cloud-based platforms such as Atlas for better accessibility and security. Apart from that, Charts on MongoDB Compass allow detailed visualization of the data and give us deeper insights. This project can help improve access to crowdfunding, assess borrower welfare levels, by analyzing the growth of previously funded projects and benefit Kiva with a better database system to enhance their platform.

II. Problem Statement

Kiva is a platform that helps with micro-financing. With growing creativity and the need to turn ideas into life, crowdfunding emerged as the best path to achieve this. Growth in the number of success stories attract a lot of people and data of such projects have been accumulating ever since then. A need to design and maintain a database system to handle such data, transform it into a form where analysis and conclusions can be drawn, that can benefit not only the platform, but as well as the potential customers of crowdfunding too. Comparing the performance on the probable operations on the platform is the means by which a conclusion will be drawn. The metrics for consideration are the transaction volumes, whether the data is structured, semi-structured or unstructured, need for pre-defined query patterns.

III. Requirements

Dataset:

A. Source

The data was obtained from Kaggle, an online community of data scientists and machine learning practitioners. It was made available by Kiva, an online crowdfunding platform, for the "Data Science for Good" Challenge and invited people to help them build a localized model to estimate various metrics in regions where Kiva has active loans.

B. Dataset Description

The dataset contains 4 csv files with 54 attributes total, which includes 30 string, 7 decimal, 4 datetime and 13 other data types. The first table consists of 20 columns, detailing the id, funded amount, loan amount, country code, country, currency, region, etc. Similarly, the second, third and fourth table outlines data snapshot and can be matched to the loan theme regions to get a loan's location and provides details for id, loan theme id, loan theme type, partner id and MPI (Multidimensional Poverty Index). Extracting several insights from the historical micro-loans over a period and correlating the regional averages by gender, sector, or borrowing behavior to estimate the welfare rate is to be followed.

C. Data Cleaning

The data required cleaning and correcting to be processed and stored into the database. Many of the tables had missing data which was replaced by the mode of the data in that column. Attributes with most of the data missing were removed from the tables altogether. Many of the tables also contained attributes which were duplicated in the same table as well as across multiple tables. These were removed and the data was pre-processed to remove redundancies.

```
In [ ]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')

In [ ]: df = pd.read_csv('kiva_loans.csv')

In [ ]: data = pandas.read_csv('kiva_loans.csv', encoding='utf-8', quotechar='"', delimiter=',')

In [ ]: df.info()

In [ ]: df.describe()

In [ ]: null  = df.isnull().sum().sort_values(ascending = False).reset_index()
        null.columns = ['Column', 'Frequency']
        null

In [ ]: ## tags column consists of 10000 null values so remove the column.
        df.drop('tags', axis=1,inplace=True)

In [ ]: ## this column null values are replaced with the mode
        df['funded_time'].mode()

In [ ]: df['borrower_genders'].mode()

In [ ]: df['funded_time'].fillna(df['funded_time'].mode(), inplace = True)

In [ ]: df['borrower_genders'].fillna(df['borrower_genders'].mode(), inplace = True)

In [ ]: df.dropna(inplace = True)

In [ ]: df.isna().sum()

In [ ]: data.to_csv('kiva_mpi_region_locations', encoding='utf-8', index = False)
```

Fig.1 Jupyter Notebook for Data Cleaning

Software Requirements:

A. MySQL

MySQL is an open source relational database management system, also managed by Oracle. It is a fast, multi-threaded, easy-to-use,multi-user and robust SQL server. MySQL uses a standard query language - SQL data language.
Features of MySQL:

- Easy to Use
- Vertical Scalable
- High flexibility
- High performance and productivity
- Memory efficiency

B. NoSQL-MongoDB-ATLAS

NoSQL stands for non relational database,where the values are stored in different categories which include document oriented,key-value pairs,graph,column oriented .
MongoDB Atlas is the most popular NoSQL database. It is an open source document-oriented database.The data is stored in BSON format similar to JSON.
MongoDB is an advanced cloud database service which also promises mobility across AWS .It takes care of time-consuming and costly administration tasks so you can get the database resources you need, when you need them.
Features of MongoDB:

- Document oriented
- High scalability
- Replication and high availability
- Indexing
- Aggregation

IV. Final Database Design

A NoSQL database system such as MongoDB requires work differently than a traditional relational database system. Where an RDBMS is incapable of handling and storing unstructured and semi-structured data, NoSQL can store, process, and visualize such data with ease. MongoDB represents the information as a JSON document instead of the row and column format adopted by the RDBMS system. MongoDB stores data in structures called ‘collections’ and the data entries are called ‘documents’. The documents contain key-value pairs of various types and can also consist of arrays, nested documents, etc. Each document can have its own structure and are self-describing in nature.

A. Document Structure

The following diagram represents the various collections created by importing the data into MongoDB. Each attribute in the collections represents the various documents keys and is of variable types (string, integer, date, timestamp, etc.).

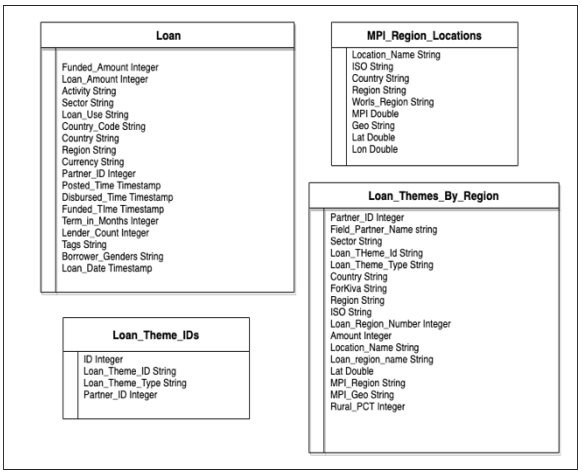


Fig. 2 Data Model

Each collection in the database contains documents with a JSON-like structure having key-value pairs for each of the entries in the collection. A total of four collections were created initially and imported into MongoDB for this project. The following are the structures of the documents in each of the collections. In the end, all of the four collections merge into one collection called the “all_kiva” collection.

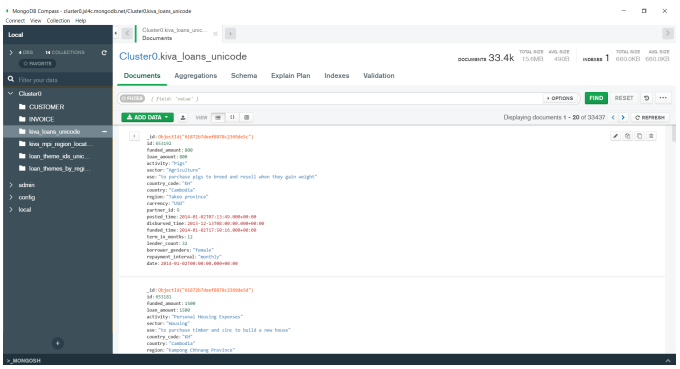


Fig. 3 Kiva_loans

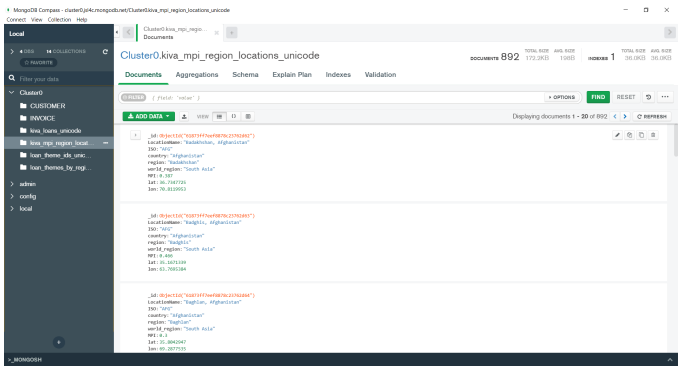


Fig. 4 Kiva_mpi_region

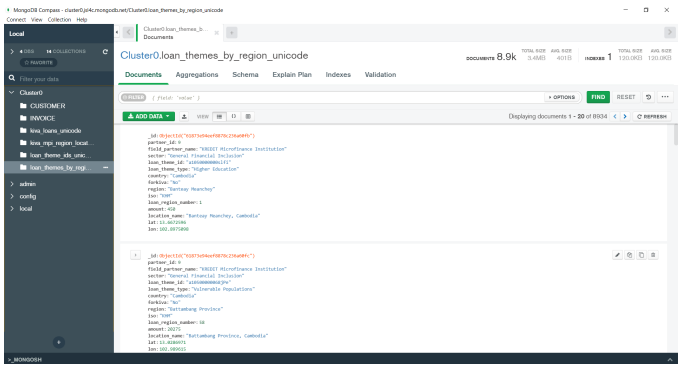


Fig. 5 Kiva_loan_themes_by_region

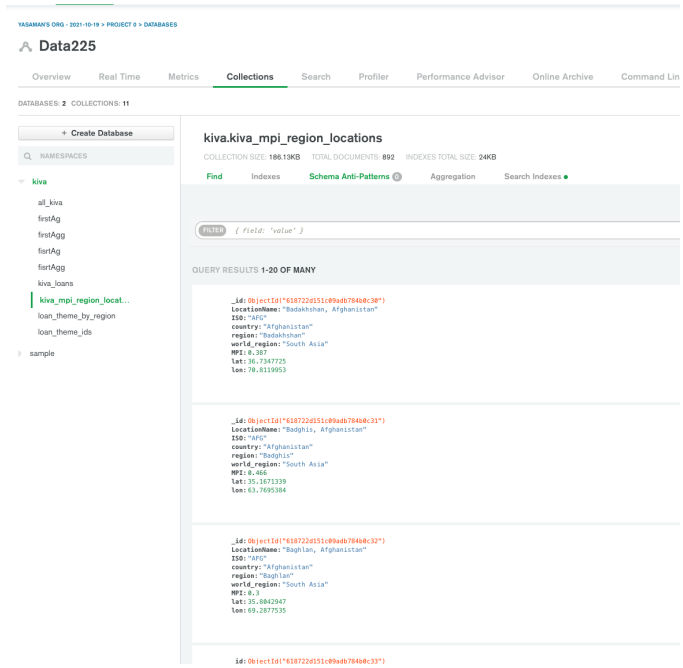


Fig.6 kiva_mpi_region_locations

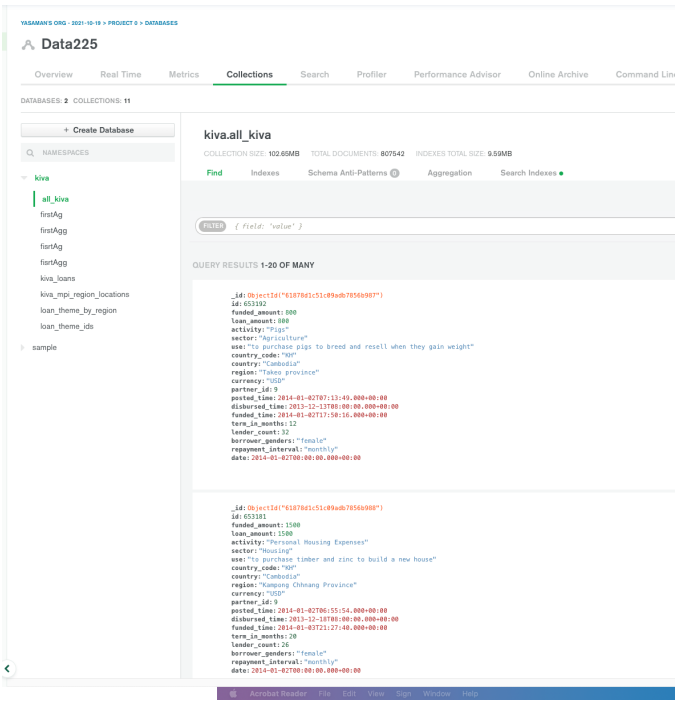


Fig. 7 all_kiva

The figure below shows the records as in the collection of the denormalized data. MongoDB removes the fields where the key has a null value and displays only the pairs with valid data as opposed to the relational database which displays the missing values as 'Null'.

```
[Atlas atlas-5a11e7-shard-0 [primary] kiva> db.all_kiva.find()
[
  {
    _id: ObjectId("61878d1c51c09adb7856b987"),
    id: 653192,
    funded_amount: 800,
    loan_amount: 800,
    activity: 'Pigs',
    sector: 'Agriculture',
    use: 'to purchase pigs to breed and resell when they gain weight',
    country_code: 'KH',
    country: 'Cambodia',
    region: 'Takeo province',
    currency: 'USD',
    partner_id: 9,
    posted_time: ISODate("2014-01-02T07:13:49.000Z"),
    disbursed_time: ISODate("2013-12-13T08:00:00.000Z"),
    funded_time: ISODate("2014-01-02T17:50:16.000Z"),
    term_in_months: 12,
    lender_count: 32,
    borrower_genders: 'female',
    repayment_interval: 'monthly',
    date: ISODate("2014-01-02T00:00:00.000Z")
  },
  {
    _id: ObjectId("61878d1c51c09adb7856b988"),
    id: 653181,
    funded_amount: 1500,
    loan_amount: 1500,
    activity: 'Personal Housing Expenses',
    sector: 'Housing',
    use: 'to purchase timber and zinc to build a new house',
    country_code: 'KH',
    country: 'Cambodia',
    region: 'Kampong Chhnang Province',
    currency: 'USD',
    partner_id: 9,
    posted_time: ISODate("2014-01-02T06:55:54.000Z"),
    disbursed_time: ISODate("2013-12-18T08:00:00.000Z"),
    funded_time: ISODate("2014-01-03T21:27:40.000Z"),
    term_in_months: 20,
    lender_count: 26,
    borrower_genders: 'female',
    repayment_interval: 'monthly',
    date: ISODate("2014-01-02T00:00:00.000Z")
  },
  {
    _id: ObjectId("61878d1c51c09adb7856b989"),
    id: 653193,
    funded_amount: 1500,
    loan_amount: 1500,
    activity: 'Motorcycle Transport',
    sector: 'Transportation',
    use: "To buy a motorcycle for his son's commute to school.    ",
    country_code: 'KH',
    country: 'Cambodia',
    region: 'Khsach Kandal district',
    currency: 'USD',
    partner_id: 61,
    posted_time: ISODate("2014-01-02T07:17:22.000Z"),
    disbursed_time: ISODate("2013-12-16T08:00:00.000Z"),
    funded_time: ISODate("2014-01-23T16:28:01.000Z"),
```

Fig. 8 all_kiva key-value pairs

V. TRADE OFFS

The selection of either a SQL or a NoSQL database depends entirely on the type of data one has and how they aim to use it. There is no one-size-fits-all policy that applies here and both the schemas come with their own advantages and disadvantages. The selection of NoSQL for this project comes with a compromise on some of the features MySQL provides that MongoDB cannot.

A. Complex Queries

Although NoSQL outperforms SQL at faster execution for queries, it is SQL that is able to execute complex queries easily and much faster than NoSQL. The availability of joins in MySQL allows data to be combined from multiple tables and compared easily which is not allowed in NoSQL.

B. An Established Platform

Relational databases are well established and mature technology. SQL standards are well defined and commonly accepted whereas NoSQL is still developing and still has a long way to go.

C. Consistency

Unlike SQL databases, NoSQL databases do not follow ACID properties and hence have a latency period before consistency is achieved across all nodes in the database but it is not guaranteed when this will happen. It instead follows BASE properties which describes consistency as a developer's problem, one that should not be handled by the database.

VI. IMPLEMENTATION

IMPLEMENTATION OF NOSQL:

In this section all the SQL queries performed on the data in relational database model have been applied to denormalized form of data stored in no sql format in MongoDB. The queries run on MongoShell connected to the cloud cluster of MongoDB. The queries considered to perform are similar so the comparison between NoSQL data format and RDBMS can be performed to get a better understanding of performance metrics and execution time as well as the syntax. Figure below shows the all key-value pairs from kiva where the load_amount is between 500$ and 1500$. Also Fig. 10 is displaying the count of each loan.

Fig. 9 Retrieve all the data in all_kiva collection on MongoDB

Fig. 10 loan_amount between 500$,1500$

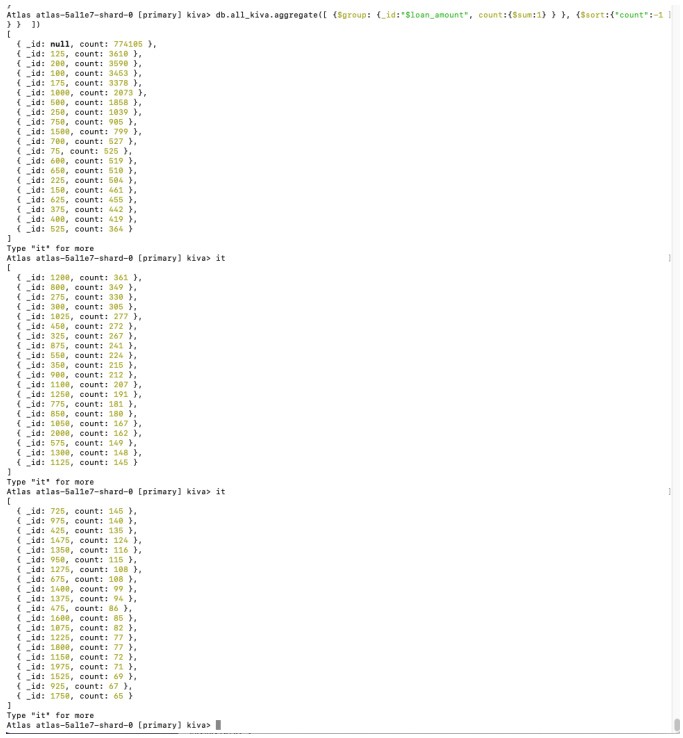


Fig.11 Count of Each Loan

In Fig.12 below is a query to display information on members from specific regions.

Fig.12. Members in Specific Region

Fig. 13 is showing the applicants from countries that their name contains “am” and their attribute of ‘forkiva’ is set to “yes”.

Fig.13 Countries Like ‘/.*am.*/’

Fig. 14 below is displaying the query and the returned result for the applicants that are active in the food industry and they have specialty and their loan_theme_type is not “General” .

Fig.14 Show the loans in food industry-specific type (exclude general types)

Last Query from Report 1 is the complex query that joins and groups by having clauses in Relational data format to return a count of activity along with other attributes for where the term in months is bigger than 10 and the count of activity is also bigger than 10 and it's ordered by count.

Fig. 15 depicts the query in MongoDB shell for NoSQLl kiva database.

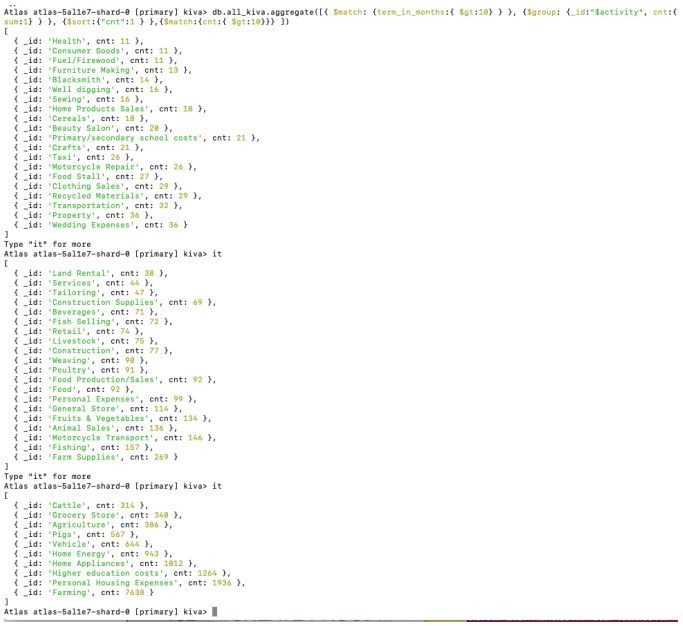


Fig. 15 Display number of times each activity gets funded with terms of bigger10 months and if they are funded more than 10 times

Figures 16 to 18 below demonstrate Delete and Insert commands in all_kiva collection in MongoDB using Mongo shell update commands performed in NoSQL.

Fig.16 Update documents set a value on condition

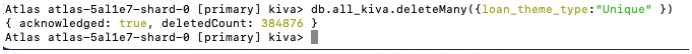


Fig.17 Delete documents based on conditions

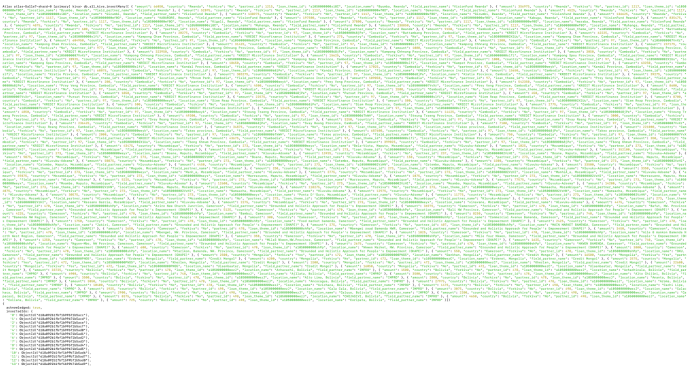


Fig.18 Insert 133 documents to all_kiva collection

VII. PERFORMANCE MEASUREMENT

In this section all the operations including create, read, update and insert (CRUD) were performed on both a SQL-based database and a NoSQL database and thoroughly investigated. Each operation has 3 related figures in this section. The first figure for each operation shows the SQL query and the time taken when the query or operation command is performed on MySQL workbench. In the same figure, below the results of the query, the execution time is visible. The second figure for the related operation displays the command on MongoDB and the third figure shows the statistics of that operation on MongoDB.

A table that shows the command and its performance is placed right after the figures for each operation separately.

A. Read Operations

In this section, 7 read operations have been performed on both RDBMS (MySQL) and NoSQL (MongoDB). The queries and the execution time for operations have been monitored and displayed in the table below screenshots of the queries and their performances.

a) Read operation - 1

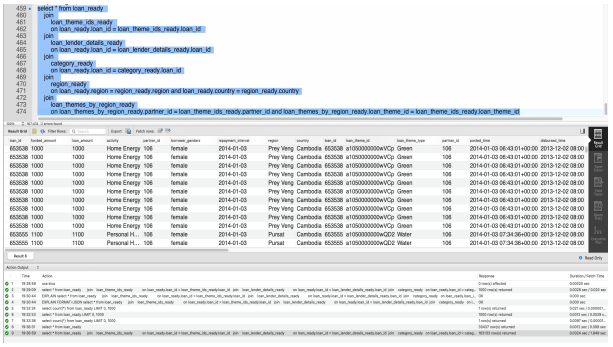


Fig. 19. Retrieve all data from joining all tables in MySQL

Fig. 20 Retrieve all the data in all_kiva collection on MongoDB

```
Atlas atlas-5a11e7-shard-0 [primary] kiva> db.all_kiva.find().explain('executionStats')
{
  queryPlanner: {
    plannerVersion: 1,
    namespace: 'kiva.all_kiva',
    indexFilterSet: false,
    parsedQuery: {},
    winningPlan: { stage: 'COLLSCAN', direction: 'forward' },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 422799,
    executionTimeMillis: 105,
    totalKeysExamined: 0,
    totalDocsExamined: 422799,
    executionStages: {
      stage: 'COLLSCAN',
      nReturned: 422799,
      executionTimeMillisEstimate: 13,
      works: 422801,
      advanced: 422799,
      needTime: 1,
      needYield: 0,
      saveState: 422,
      restoreState: 422,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 422799
    }
  },
  serverInfo: {
    host: 'data225-shard-00-02.5q6p4.mongodb.net',
    port: 27017,
    version: '4.4.10',
    gitVersion: '58971da1ef93435a9f62bf4708a81713def6e88c'
  },
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1638765534, i: 3 }),
    signature: {
      hash: Binary(Buffer.from("10a50eb942be2f11c8284d5c87e7fc56efcd290c", "hex"), 0),
      keyId: Long("6994586891324489738")
    }
  },
  operationTime: Timestamp({ t: 1638765534, i: 3 })
}
Atlas atlas-5a11e7-shard-0 [primary] kiva>
```

Fig. 21 Performance of retrieving all data on MongoDB

Read - Operation 1		
DB	Command	Execution Time(ms)
SQL	select * from loan_ready join loan_theme_ids_ready on loan_ready.loan_id = loan_theme_ids_ready.loan_id join loan_lender_details_ready on loan_ready.loan_id = loan_lender_details_ready.loan_id join category_ready on loan_ready.loan_id = category_ready.loan_id join region_ready on loan_ready.region = region_ready.region and loan_ready.country = region_ready.country join loan_themes_by_region_ready on loan_themes_by_region_ready.partner_id = loan_theme_ids_ready.partner_id and loan_themes_by_region_ready.loan_theme_id = loan_theme_ids_ready.loan_theme_id;	1837
NoSQL	db.all_kiva.find()	105

Table 1. Performance and Commands for the first read operation

b) Read operation - 2

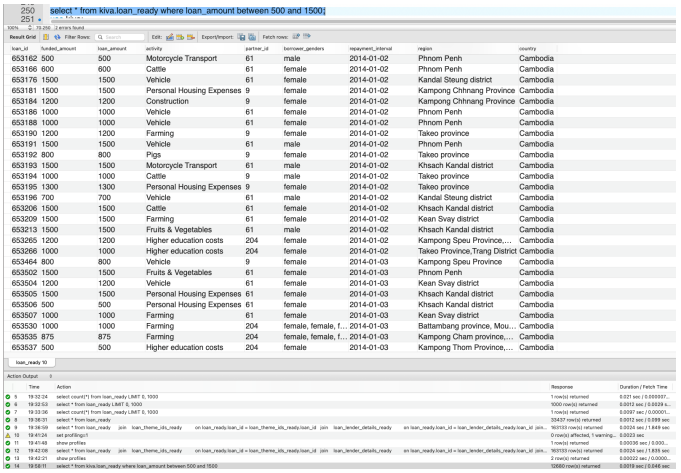


Fig. 22 loan_amount between 500$,1500$ on MySQL

Fig. 23 loan_amount between 500$,1500$ on MongoDB

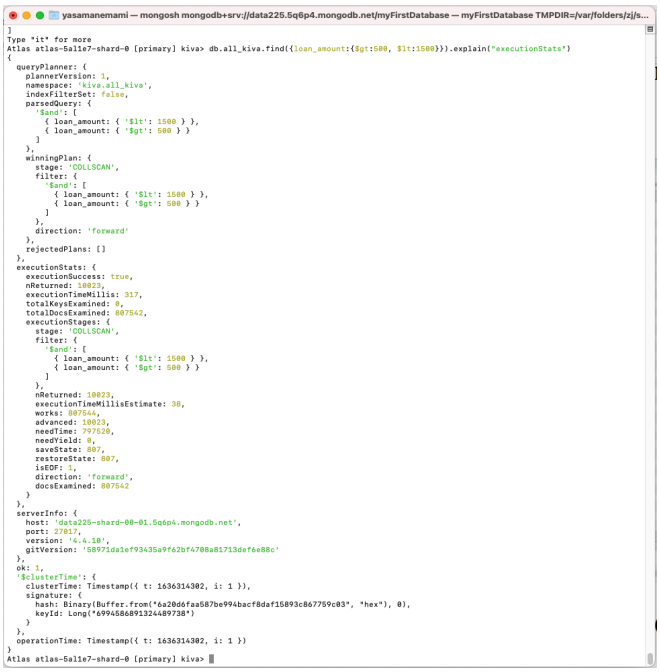


Fig. 24 Performance of loan_amount between 500$,1500$ on MongoDB

Read - Operation 2		
DB	Command	Execution Time(ms)
SQL	select * from kiva.loan_ready where loan_amount between 500 and 1500;	47
NoSQL	db.all_kiva.find({loan_amount:{$gt:500, $lt:1500}})	317

Table 2. Performance and Commands for the first read operation

c) Read operation - 3

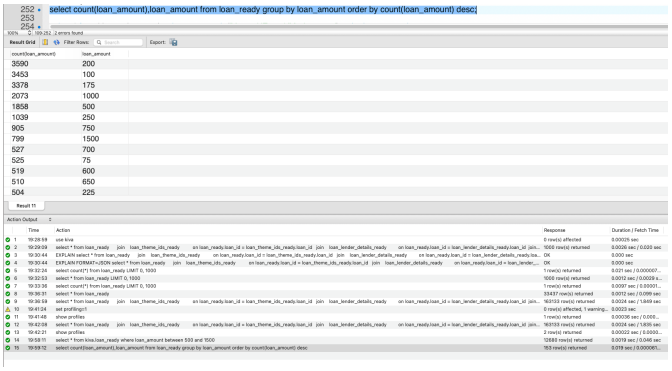


Fig.25 Count of each loan

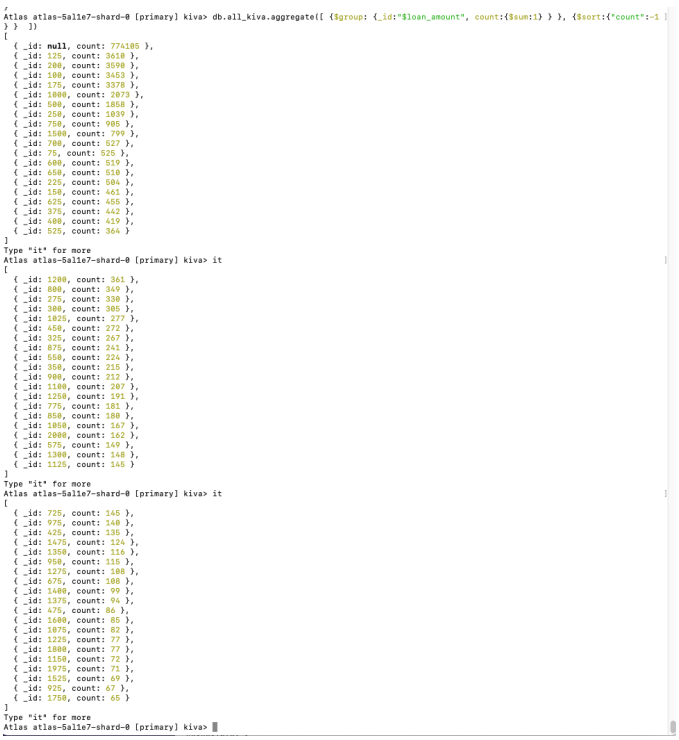


Fig. 26 Count of Each Loan

Fig. 27 Count of Each Loan NoSQL Performance

Read - Operation 3		
DB	Command	Execution Time(ms)
SQL	select count(loan_amount),loan_amount from loan_ready group by loan_amount order by count(loan_amount) desc;	18
NoSQL	db.all_kiva.aggregate([{ $group: { _id:"$loan_amount", count:{$sum:1}}}, {$sort:{"count":-1}}])	133

Table 3. Performance and Commands for the second read operation

d) Read operation - 4

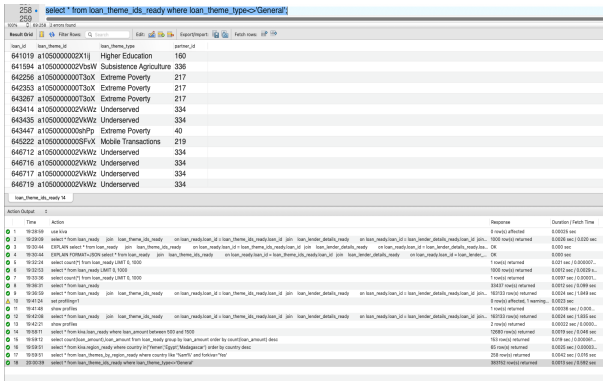


Fig. 28 Members in Specific Region on MySQL

Fig. 29 Members in Specific Region on MongoDB

Fig. 30 Performance of Members in Specific Region in MongoDB

Read - Operation 4		
DB	Command	Execution Time(ms)
SQL	select * from kiva.region_ready where country in('Yemen','Egypt','Madagascar') order by country desc;	1.3
NoSQL	db.all_kiva.find({ country:{ $in:["Yeman", "Egypt", "Madagascar"]}}).sort({country:-1})	54

Table 4. Performance and Commands for the third read operation

e) Read operation - 5

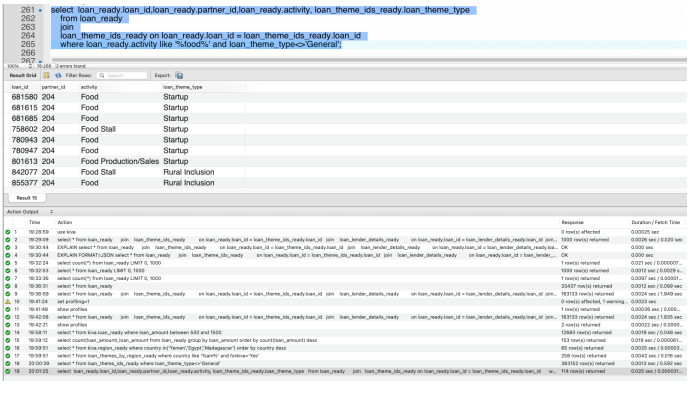


Fig. 31 Countries Like '_am%' on MySQL

Fig. 32 Countries Like '/.*am.*/'

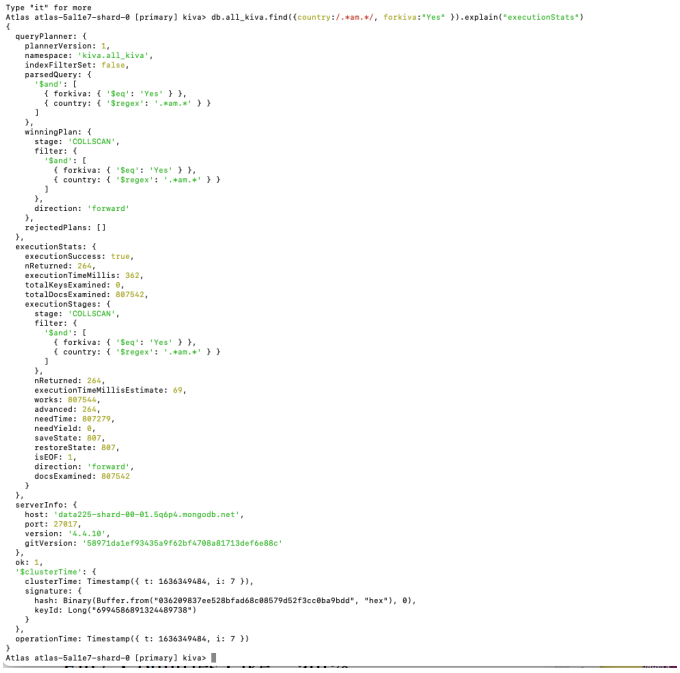


Fig. 33. Countries Like '/.*am.*/' Performance on MongoDB

Read - Operation 5		
DB	Command	Execution Time(ms)
SQL	select * from loan_themes_by_region_ready where country like '%am%' and forkiva='Yes';	20
NoSQL	db.all_kiva.find({country:/.*am.*/, forkiva:"Yes" })	362

Table 5. Performance and Commands for the fourth read operation

f) Read Operation - 6

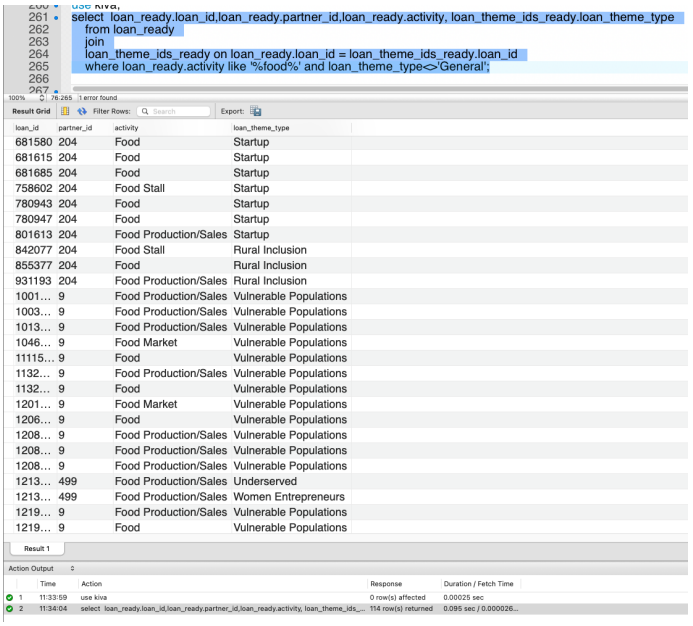


Fig.34 Records from join of two tables to show the loans in food industry specific type (exclude general types) on MySQL

Fig. 35 Show the loans in food industry-specific type (exclude general types)

Fig.36 Performance the loans in food industry-specific type (exclude general types) on MongoDB

Read - Operation 6		
DB	Command	Execution Time(ms)
SQL	select loan_ready.loan_id,loan_ready.partner_id, loan_ready.activity, loan_theme_ids_ready.loan_theme_type from loan_ready join loan_theme_ids_ready on loan_ready.loan_id = loan_theme_ids_ready.loan_id where loan_ready.activity like '%food%' and loan_theme_type<>'General';	24
NoSQL	db.all_kiva.find({activity:/.*Food.*/, loan_theme_type:{$ne:"General"} })	305

Table 6. Performance and Commands for the fifth read operation

g) Read Operation - 7

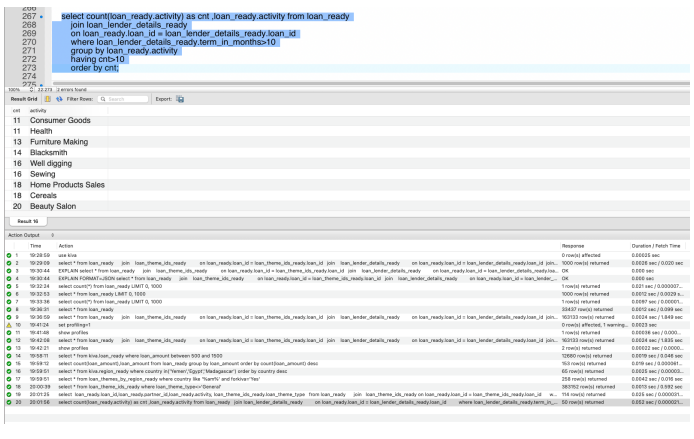


Fig.37 # of times each activity gets funded with terms of bigger 10 months and if they are funded more than 10 times

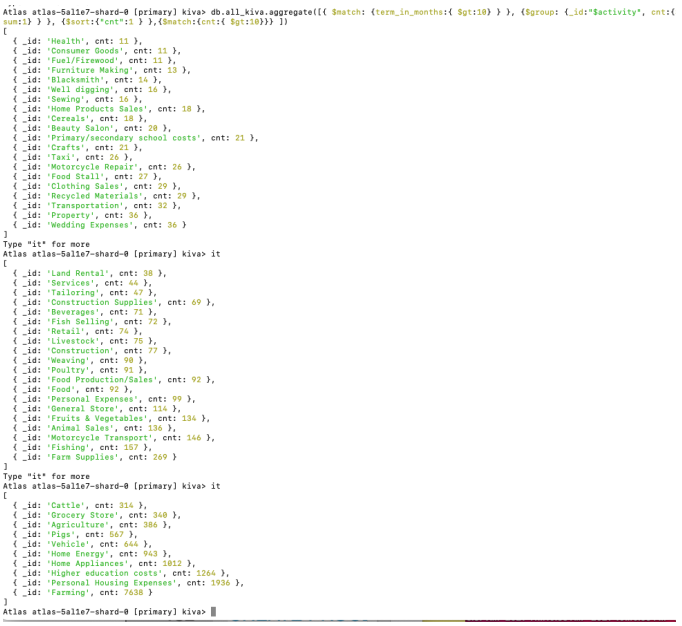


Fig. 38 Display number of times each activity gets funded with terms of bigger 10 months and if they are funded more than 10 times

Fig. 39 Performance of # of times each activity gets funded with terms of bigger10 months and if they are funded more than 10 times

Read - Operation 7		
DB	Command	Execution Time(millisec)
SQL	select count(loan_ready.activity) as cnt ,loan_ready.activity from loan_ready join loan_lender_details_ready on loan_ready.loan_id = loan_lender_details_ready.loan_id where loan_lender_details_ready.term_in_months >10 group by loan_ready.activity having cnt>10 order by cnt;	52
NoSQL	db.all_kiva.aggregate([{ $match: { term_in_months:{ $gt:10 } } }, {$group: { _id: "$activity", cnt: {$sum:1} } }, { $sort:{ "cnt":1 } },{ $match:{cnt:{ $gt:10}}}])	305

Table 7. Performance and Commands for the sixth read operation

From all of the above experiments, the read execution time for MySQL is faster than MongoDB if we are using where clause and filtering the data. However, retrieving all the data is faster in MongoDB almost 18 times compared with MySQL.

B. *Update Operation*

In this section, update operation on MySQL and MongoDB is investigated.

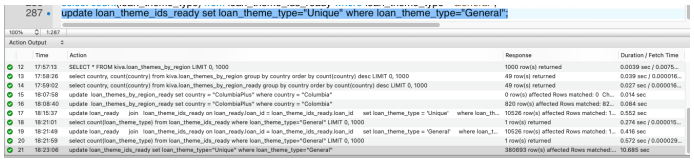


Fig. 40 Update on Mysql

```
Atlas atlas-5a11e7-shard-0 [primary] kiva> db.all_kiva.updateMany({loan_theme_type:"General" },{$set:{loan_theme_type:"Unique" }} )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 384876,
  modifiedCount: 384876,
  upsertedCount: 0
}
```

Fig. 41 Update documents set a value on condition

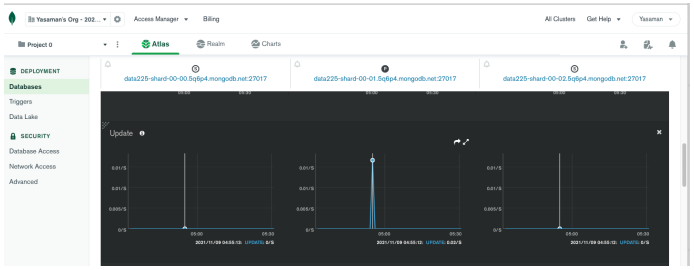


Fig.42 Update Performance on MongoDB

Update - Operation		
DB	Command	Execution Time(ms)
SQL	update loan_ready join loan_theme_ids_ready on loan_ready.loan_id = loan_theme_ids_ready.loan_id set loan_theme_type = 'General' where loan_theme_type = 'Unique';	10685
NoSQL	db.all_kiva.updateMany({loan_theme_type: "General"}, {$set:{loan_theme_type: "Unique" }})	20

Table 8. Performance and Commands for update operation

The results show that for Kiva database **update** operation on MongoDB is faster than MySQL.

C. *Insert Operations*

This section focuses on insert operation and performances.

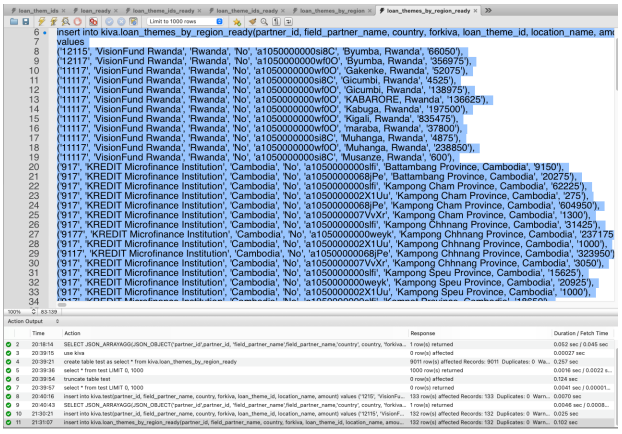


Fig.43 Insert 133 records to MySQL

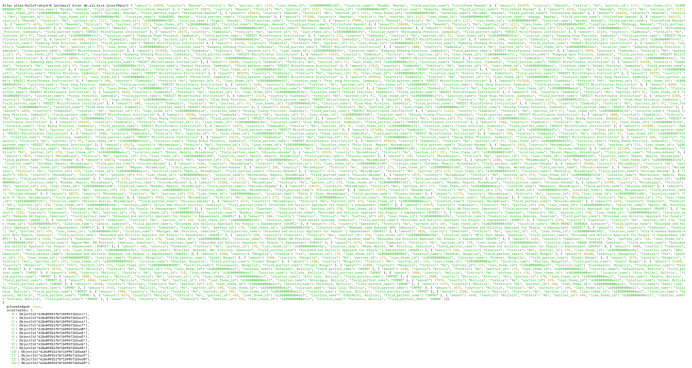


Fig. 44 Insert 133 documents to all_kiva collection

Fig. 45 Performance of Insert on MongoDB

Fig.48 Performance of delete documents on MongoDB

Insert - Operation		
DB	Command	Execution Time(ms)
SQL	insert into loan_ready(loan_id, funded_amount, loan_amount, activity, partner_id, borrower_genders, repayment_interval, region, country) values (1653162, 50, 500, 'Motorcycle Transport', 61, 'male', '2014-01-02', 'Phnom Penh', 'Cambodia'), 132 other records	102
NoSQL	db.all_kiva.insertMany([{ "amount": 66050, "country": "Rwanda", "forkiva": "No", "partner_id": 1215, "loan_theme_id": "a1050000000si8C", "location_name": "Byumba, Rwanda", "field_partner_name": "VisionFund Rwanda" }, 132 more records])	20

Table 9. Performance and Commands for insert operation

Results of insert operation show that Kiva database insert operation on MongoDB is faster than MySQL.

D. Delete Operations

This section shows experiments on delete operation on MySQL and MongoDB.

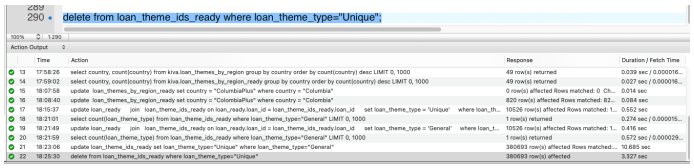


Fig. 46 Delete records based on condition on MySQL

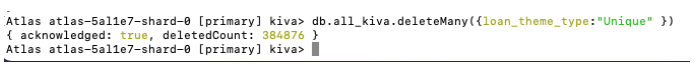


Fig. 47 Delete documents based on conditions on MongoDB

Delete - Operation		
DB	Command	Execution Time(ms)
SQL	delete from loan_theme_ids_ready where loan_theme_type="Unique";	3300
NoSQL	db.all_kiva.deleteMany({loan_theme_type:"Unique" })	3

Table 10. Performance and Commands for delete operation

The results of this section show that the delete operation on MongoDB is faster than MySQL.

VIII. Outcomes / Lessons Learned

Following are the points learnt from the project:

- We learnt to implement the operations discussed in class which include basic CRUD operations, triggers, stored procedures , joins using SQL queries.
- We understood how to overcome the issues faced while loading the data using the import wizard in MySQL. It was much easier to combine the data and import it using MongoDB.
- We learnt the connection between AWS and Python.
- We made use of the visualization tools to showcase our output for the sql queries and we used Atlas for NoSQL .
- We understood the differences between MySQL and NoSQL with respect to performance evaluation.

IX. Conclusion

This project has analyzed two different database designs for the Kiva crowdfunding project, namely SQL (MySQL) and NoSQL (MongoDB). After performing a comparative analysis of the two databases for basic CRUD operations, considering the transaction volumes, the type of data and the need for pre-defined query patterns, it is observed that NoSQL performs well in create, update, delete and fetching record operations whereas MySQL is faster in searching data with filtering. The selected database is not tested for complex operations and the comparison results hold good only for the operations being performed. It is to be noted that in real time applications where software is included , databases undergo various changes which might improve or degrade its performance.

Every business has a unique requirement for how they intend to use a database, and these requirements are determined by what the business expects from its database. Both the SQL and NoSQL models have their own set of advantages and disadvantages that each business must weigh before deciding which is best practice for them. There is no absolute single solution for choosing a database for all the businesses.

It is evident that the crowdfunding platform will see growth , that too exponentially and the data will grow at the same pace as well. There is a need to automatically recover the data and at a fast pace. The schema needs to be flexible in order to accommodate the unforeseen aspects that may come in the way of the future of crowdfunding platforms. For such reasons NoSQL is well suited. The trade off will be compromising on the transaction rates offered by MySQL. Thus, this project concludes that NoSQL is the best fit for crowdfunding platforms like Kiva.

Acknowledgment

We would like to express our sincere gratitude to Professor Simon Shim for his guidance that aided in the completion of this project and helped us understand database concepts using practical examples. We would also like to thank Shiva Abhishek Varma Penmetsa and Rushikesh Jagtap for their continued help and support throughout the development of the project.

References

[1] S. Yu, Crowdfunding and regional entrepreneurial investment: an application of the CrowdBerkeley database, Research Policy, Volume 46, Issue 10, 2017, Pages 1723-1737, ISSN 0048-7333

[2] Burtch, G., Ghose, A., Wattal, S., 2014. Cultural Differences and geography as determinants of online prosocial lending. MIS Q. 38 (3), 773–794.

[3] Mollick, E., Nanda, R., 2015. Wisdom or Madness? Comparing Crowds with Expert Evaluation in Funding the Arts. Manage. Sci. 62 (6), 1533–1553.Mollick, E.R., 2014. The Dynamics of Crowdfunding: An Exploratory Study. J. Bus.Venturing 29 (January (1)), 1–16.

[4] Smith, Tim. “Crowdfunding.” Investopedia, Investopedia, 13Sept.2021,www.investopedia.com/terms/c/crowdfunding

[5] Shah, Mihir. “MongoDB vs MySQL: A Comparative Study on Databases.” Simform, 23Nov.2017,www.simform.com/blog/mongodb-vs-mysql-databases

[6] Tzoof Avny Broosh. “ How to Choose the Right Database”,Sept. 2019, https://towardsdatascience.com/how-to-choose-the-right-database-afcf95541741.

[7] Jatin Raisinghani. “Should You Use MongoDB or SQL Databases for Analytics?”, Aug. 2018, https://www.holistics.io/blog/should-you-use-mongodb-or-sql-databases-for-analytics/.

[8] Nance, Cory; Losser, Travis; Iype, Reenu; and Harmon, Gary, "NOSQL VS RDBMS - WHY THERE IS ROOM FOR BOTH" (2013). SAIS 2013Proceedings. 27.

[9] Venkatraman, Sitalakshmi, et al. "SQL versus NoSQL movement with big data analytics." International Journal of Information Technology and Computer Science 8.12 (2016): 59-66.

PROJECT - 3 REPORT

Kiva Crowdfunding Dataset Analysis

Poojitha Katta
Department of Applied Data Science
San Jose State University
San Jose, United States
poojitha.katta@sjsu.edu

Purnima Bhukya
Department of Applied Data Science
San Jose State University
San Jose, United States
purnima.bhukya@sjsu.edu

Deepali Zutshi
Department of Applied Data Science
San Jose State University
San Jose, United States
deepali.zutshi@sjsu.edu

Yasaman Emami
Department of Applied Data Science
San Jose State University
San Jose, United States
yasaman.emami@sjsu.edu

Abstract—The main aim of this project is to design a database for the crowdfunding platform-Kiva, suggest the best solution based on the trade-offs between SQL and NoSQL schemas for Kiva as a crowdfunding database. The final solution would be decided based on the performance of each approach and the requirements of the project.

Keywords—SQL, NoSQL, Crowdfunding, Database, Kiva

I. Introduction

Crowdfunding is a system that enables individuals or ventures to seek small investments, contributions, or loans from a variety of funders online. Kiva is a crowdfunding platform that offers a new financing channel for small and micro-businesses as well as individuals. The aim of the project is to offer the best solution as a database management system for Kiva platform to analyze the factors that influence crowdfunded projects by estimating the welfare level of partners in specific regions, based on shared financial and demographic aspects. In this study, the comparison between MySQL as a Relational Database Management System (RDBMS) model and MongoDB as NoSQL approach was investigated and various conclusions were drawn.

II. Dataset

A. Source

The data was obtained from Kaggle, an online community of data scientists and machine learning practitioners. It was made available by Kiva, an online crowdfunding platform, for the "Data Science for Good" Challenge and invited people to help them build a localized model to estimate various metrics in regions where Kiva has active loans.

B. Dataset Description

The dataset contains 4 csv files with 54 attributes total, which includes 30 string, 7 decimal, 4 datetime, and 13 other data types. The first table consists of 20 columns, detailing the id, funded amount, loan amount, country code, country, currency, region, etc. Similarly, the second, third and fourth table outlines data snapshots and can be matched to the loan theme regions to get a loan's location and provides details for id, loan theme id, loan theme type, partner id, and MPI (Multidimensional Poverty Index). Extracting several insights from the historical micro-loans over a period of time and correlating the regional averages by gender, sector, or borrowing behavior to estimate the welfare rate is to be followed.

C. Data Cleaning

The data needed to be cleaned and corrected before it could be processed and stored in the database. Many of the tables had missing data which was replaced by the mode of the data in that column. Attributes with most of the data missing were removed from the tables altogether. Many of the tables also contained attributes that were duplicated in the same table as well as across multiple tables. These were removed and the data was pre-processed to remove redundancies.

```
In [ ]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')

In [ ]: df = pd.read_csv('kiva_loans.csv')

In [ ]: data = pandas.read_csv('kiva_loans.csv', encoding='utf-8', quotechar='"', delimiter=',')

In [ ]: df.info()

In [ ]: df.describe()

In [ ]: null  = df.isnull().sum().sort_values(ascending = False).reset_index()
        null.columns = ['Column', 'Frequency']
        null

In [ ]: ## tags column consists of 10000 null values so remove the column.
        df.drop('tags', axis=1,inplace=True)

In [ ]: ## this column null values are replaced with the mode
        df['funded_time'].mode()

In [ ]: df['borrower_genders'].mode()

In [ ]: df['funded_time'].fillna(df['funded_time'].mode(), inplace = True)

In [ ]: df['borrower_genders'].fillna(df['borrower_genders'].mode(), inplace = True)

In [ ]: df.dropna(inplace = True)

In [ ]: df.isna().sum()

In [ ]: data.to_csv('kiva_mpi_region_locations', encoding='utf-8', index = False)
```

Fig. 1 Jupyter Notebook for Data Cleaning

III. Data Model

A NoSQL database system such as MongoDB performs differently compared to a traditional relational database system. Where an RDBMS is incapable of handling and storing unstructured and semi-structured data, NoSQL can store, process, and visualize such data with ease. MongoDB represents the information as a JSON document instead of the row and column format adopted by an RDBMS system. MongoDB stores data in structures called 'collections' and the data entries are called 'documents'. The documents contain key-value pairs of various types and can also consist of arrays,

nested documents, etc. Each document can have its own structure and are self-describing in nature.

A. Data Model in MongoDB

The following diagram represents the various collections created by importing the data into MongoDB. Each attribute in the collections represents the various documents keys and is of variable types (string, integer, date, timestamp, etc.).

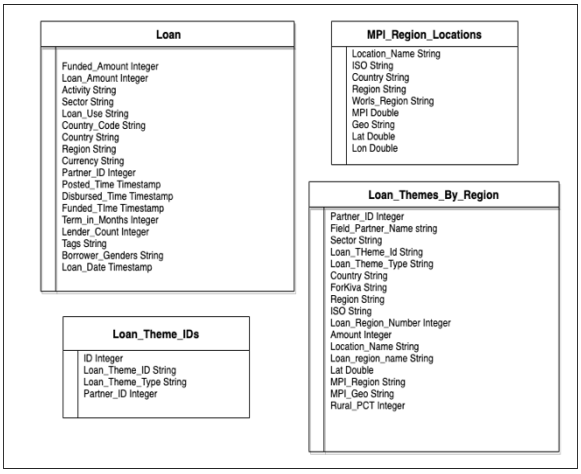


Fig. 2 Data Model - NoSQL

Each collection in the database contains documents with a JSON-like structure having key-value pairs for each of the entries in the collection. A total of four collections were created initially and imported into MongoDB for this project. These were then aggregated to create a single collection named ‘all_kiva’.

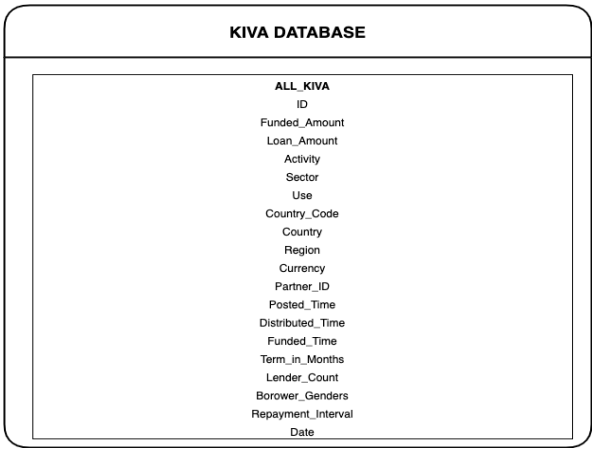


Fig. 3 Document Structure in MongoDB

B. Data Model in MySQL

An RDBMS model such as MySQL represents the data as a collection of relations. Each relation is represented by a table structure with rows and columns of records. In a relation thought of as a table, every row indicates a collection of data values related to each other. The name of the table and the columns help users interpret the meaning of the values stored in the rows.

Data in a relational model is normalized, which is the process of structuring the data in order to reduce redundancy as well as improve the integrity of the database.

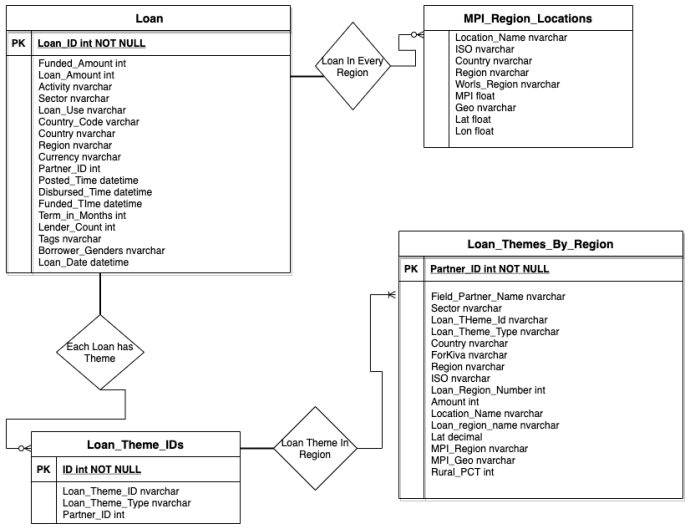


Fig. 4 Denormalized Data in SQL

The figure above represents the raw data available to us in a denormalized manner. It can be seen that various columns have been repeated in several tables which introduces redundancy in the database.

Normalization of this data generated 7 tables from the existing 4 and improved the overall quality of the data. For example, attributes such as 'ISO ', ‘country’, ‘latitude’, ‘longitude’, ‘world_region’, ’location’ were common in many of the entities. These were extracted to create a separate entity table called ‘Region’ which was then linked to the table ‘Loan’ to define the loan details for each region. The following schema was created after the process of normalization was performed.

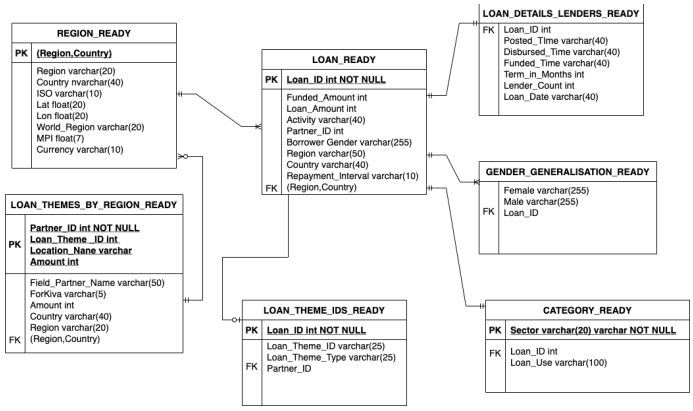


Fig. 5 Normalized Data Model in SQL

IV. Queries and Performance

In this section all the operations include create, read, update and insert (CRUD) were performed on both a SQL-based database and a NoSQL database and thoroughly investigated. Each operation has 3 related figures in this section. The first figure for each operation shows the SQL query and the time taken when the query or operation command is performed on MySQL workbench. In the same figure, below the results of the query, the execution time is visible. The second figure for the related operation displays the command on MongoDB and the third figure shows the statistics of that operation on MongoDB. A table that shows the command and its performance is placed right after the figures for each operation separately.

A. Read Operations

In this section, 7 read operations have been performed on both RDBMS (MySQL) and NoSQL (MongoDB). The queries and the execution time for operations have been monitored and displayed in the table below screenshots of the queries and their performances.

a) Read operation - 1

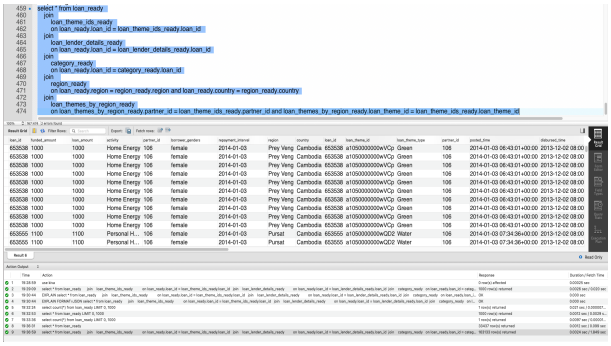


Fig. 6.a. Retrieve all data from joining all tables in MySQL

Fig. 6.b. Retrieve all the data in all_kiva collection on MongoDB

Fig. 6.c. Performance of retrieving all data on MongoDB

Read - Operation 1		
DB	Command	Execution Time(ms)
SQL	select * from loan_ready join loan_theme_ids_ready on loan_ready.loan_id = loan_theme_ids_ready.loan_id join loan_lender_details_ready on loan_ready.loan_id = loan_lender_details_ready.loan_id join category_ready on loan_ready.loan_id = category_ready.loan_id join region_ready on loan_ready.region = region_ready.region and loan_ready.country = region_ready.country join loan_themes_by_region_ready on loan_themes_by_region_ready.partner_id = loan_theme_ids_ready.partner_id and loan_themes_by_region_ready.loan_theme_id = loan_theme_ids_ready.loan_theme_id;	1837
NoSQL	db.all_kiva.find()	105

Table 1. Performance and Commands for the first read operation

Fig. 7.b loan_amount between 500$,1500$ on MongoDB

b) Read operation - 2

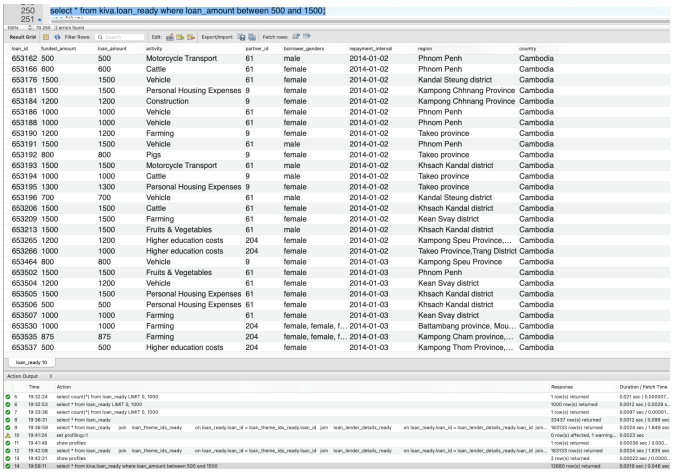


Fig. 7.a loan_amount between 500$,1500$ on MySQL

Fig. 7.c Performance of loan_amount between 500$,1500$ on MongoDB

Read - Operation 2		
DB	Command	Execution Time(ms)
SQL	select * from kiva.loan_ready where loan_amount between 500 and 1500;	47
NoSQL	db.all_kiva.find({loan_amount:{$gt:500, $lt:1500}})	317

Table 2. Performance and Commands for the first read operation

c) Read operation - 3

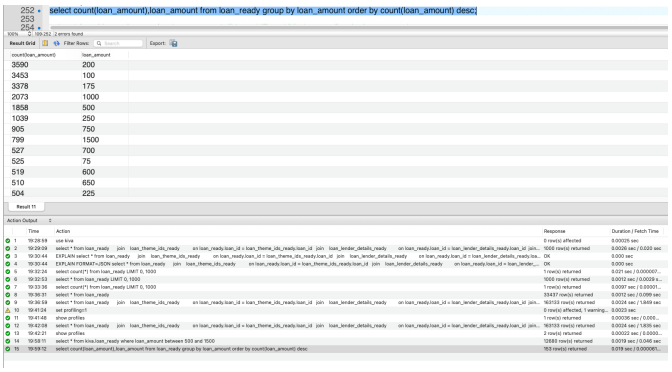


Fig. 8.a. Count of each loan

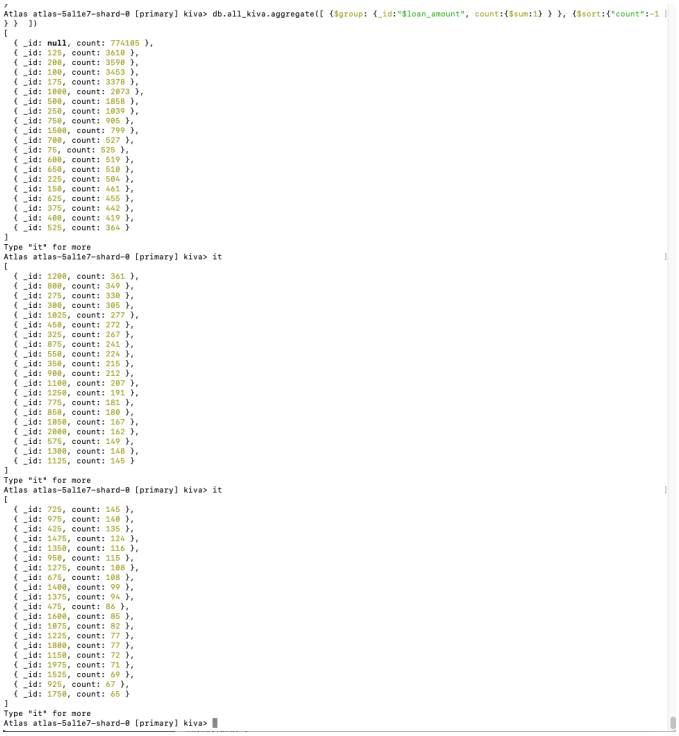


Fig. 8.b Count of Each Loan

Fig. 8.c Count of Each Loan NoSQL Performance

Read - Operation 3		
DB	Command	Execution Time(ms)
SQL	select count(loan_amount),loan_amount from loan_ready group by loan_amount order by count(loan_amount) desc;	18
NoSQL	db.all_kiva.aggregate([{ $group: { _id:"$loan_amount", count:{$sum:1}}}, {$sort:{"count":-1}}])	133

Table 3. Performance and Commands for the second read operation

a) Read operation - 4

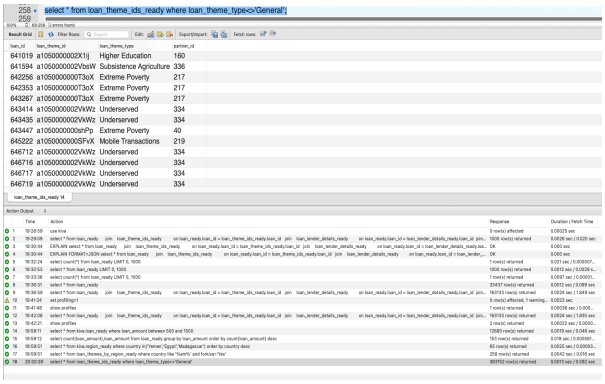


Fig. 9.a Members in Specific Region on MySQL

Fig. 9.b Members in Specific Region on MongoDB

Fig. 9.c Performance of Members in Specific Region in MongoDB

Read - Operation 4		
DB	Command	Execution Time(ms)
SQL	select * from kiva.region_ready where country in('Yemen','Egypt','Madagascar') order by country desc;	1.3
NoSQL	db.all_kiva.find({ country:{ $in:["Yeman", "Egypt", "Madagascar"]}}).sort({country:-1})	54

Table 4. Performance and Commands for the third read operation

b) Read operation - 5

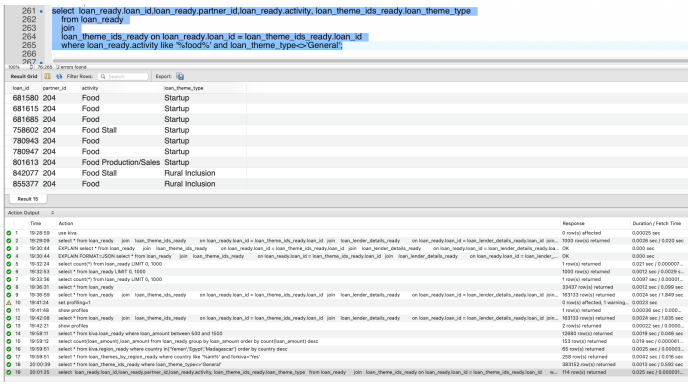


Fig. 10.a Countries Like '_am%' on MySQL

Fig. 10.b Countries Like '/.*am.*/'

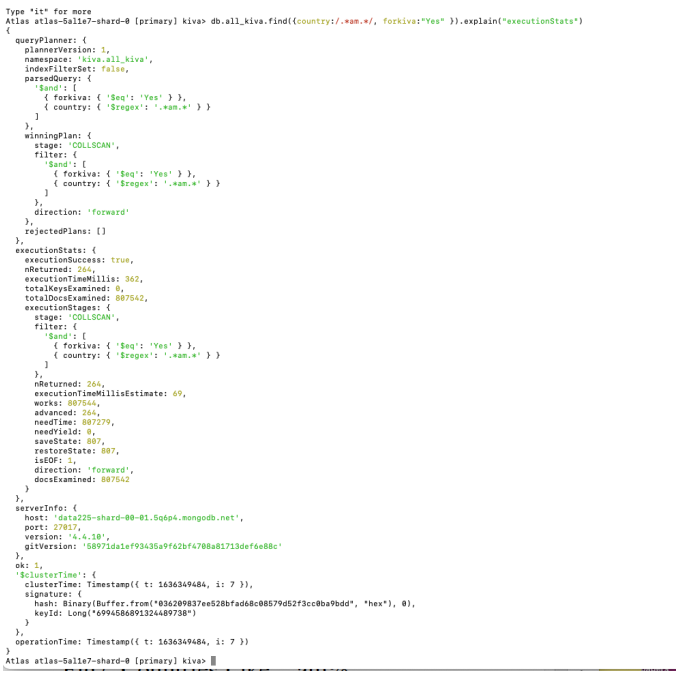
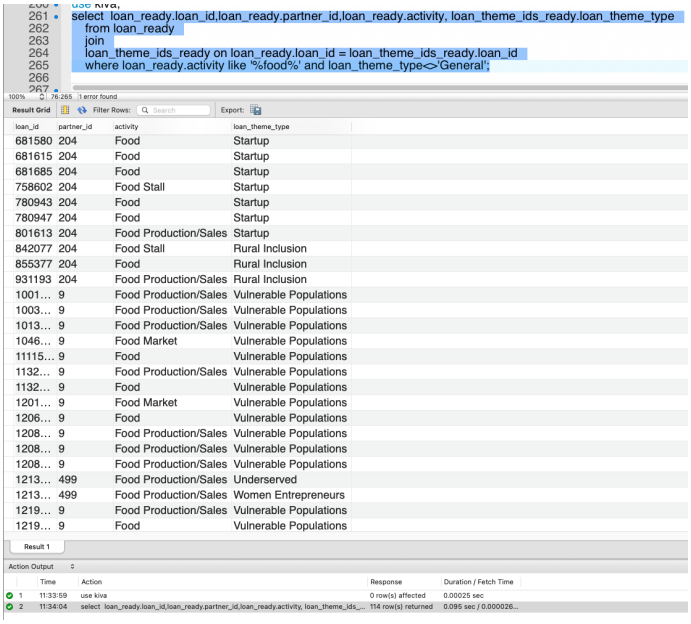


Fig. 10.c . Countries Like '/.*am.*/' Performance on MongoDB

Read - Operation 5		
DB	Command	Execution Time(ms)
SQL	select * from loan_themes_by_region_ready where country like '%am%' and forkiva='Yes';	20
NoSQL	db.all_kiva.find({country:/.*am.*/, forkiva:"Yes" })	362

Table 5. Performance and Commands for the fourth read operation

e) Read Operation - 6



loan_id	partner_id	activity	loan_theme_type
681580	204	Food	Startup
681615	204	Food	Startup
681685	204	Food	Startup
758602	204	Food Stall	Startup
780943	204	Food	Startup
780947	204	Food	Startup
801613	204	Food Production/Sales	Startup
842077	204	Food Stall	Rural Inclusion
855377	204	Food	Rural Inclusion
931193	204	Food Production/Sales	Rural Inclusion
1001…	9	Food Production/Sales	Vulnerable Populations
1003…	9	Food Production/Sales	Vulnerable Populations
1013…	9	Food Production/Sales	Vulnerable Populations
1046…	9	Food Market	Vulnerable Populations
11115…	9	Food	Vulnerable Populations
1132…	9	Food Production/Sales	Vulnerable Populations
1132…	9	Food	Vulnerable Populations
1201…	9	Food Market	Vulnerable Populations
1206…	9	Food	Vulnerable Populations
1208…	9	Food Production/Sales	Vulnerable Populations
1208…	9	Food Production/Sales	Vulnerable Populations
1208…	9	Food Production/Sales	Vulnerable Populations
1213…	499	Food Production/Sales	Underserved
1213…	499	Food Production/Sales	Women Entrepreneurs
1219…	9	Food Production/Sales	Vulnerable Populations
1219…	9	Food	Vulnerable Populations

Fig.11.a Records from join of two tables to show the loans in food industry specific type (exclude general types) on MySQL

Fig. 11.b Show the loans in food industry-specific type (exclude general types)

Fig.11.c Performance the loans in food industry-specific type (exclude general types) on MongoDB

Read - Operation 6		
DB	Command	Execution Time(ms)
SQL	select loan_ready.loan_id,loan_ready.partner_id, loan_ready.activity, loan_theme_ids_ready.loan_theme_type from loan_ready join loan_theme_ids_ready on loan_ready.loan_id = loan_theme_ids_ready.loan_id where loan_ready.activity like '%food%' and loan_theme_type<>'General';	24
NoSQL	db.all_kiva.find({activity:/.*Food.*/, loan_theme_type:{$ne:"General"} })	305

Table 6. Performance and Commands for the fifth read operation

f) Read Operation - 7

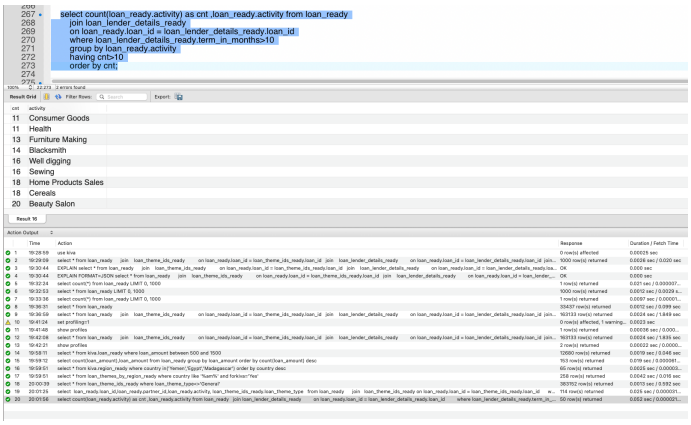


Fig.12.a # of times each activity gets funded with terms of bigger 10 months and if they are funded more than 10 times

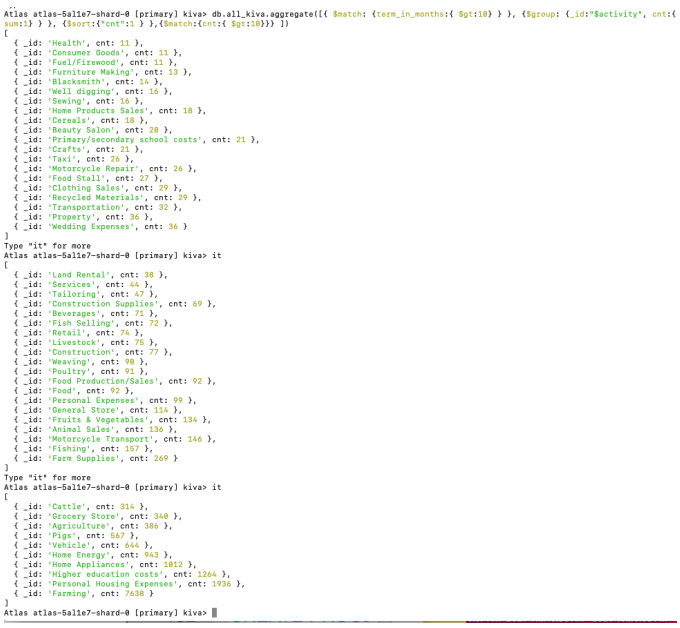


Fig. 12.b Display number of times each activity gets funded with terms of bigger 10 months and if they are funded more than 10 times

Fig. 12.c Performance of # of times each activity gets funded with terms of bigger10 months and if they are funded more than 10 times

Read - Operation 7		
DB	Command	Execution Time(millisec)
SQL	select count(loan_ready.activity) as cnt ,loan_ready.activity from loan_ready join loan_lender_details_ready on loan_ready.loan_id = loan_lender_details_ready.loan_id where loan_lender_details_ready.term_in_months >10 group by loan_ready.activity having cnt>10 order by cnt;	52
NoSQL	db.all_kiva.aggregate([{ $match: { term_in_months:{ $gt:10 } } }, {$group: { _id: "$activity", cnt: {$sum:1} } }, { $sort:{ "cnt":1 } },{ $match:{cnt:{ $gt:10}}}])	305

Table 7. Performance and Commands for the sixth read operation

From all of the above experiments, the read execution time for MySQL is faster than MongoDB if we are using where clause and filtering the data. However, retrieving all the data is faster in MongoDB almost 18 times compared with MySQL.

B. *Update Operation*

In this section, update operation on MySQL and MongoDB is investigated.

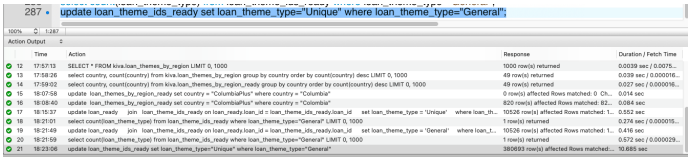


Fig. 13.a Update on Mysql

Fig. 13.b Update documents set a value on condition

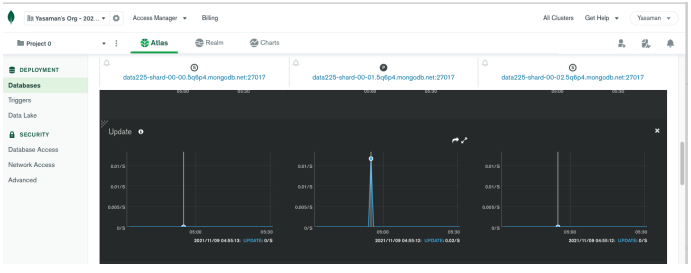


Fig. 13.c Update Performance on MongoDB

Update - Operation		
DB	Command	Execution Time(ms)
SQL	update loan_ready join loan_theme_ids_ready on loan_ready.loan_id = loan_theme_ids_ready.loan_id set loan_theme_type = 'General' where loan_theme_type = 'Unique';	10685
NoSQL	db.all_kiva.updateMany({loan_theme_type: "General"}, {$set:{loan_theme_type: "Unique" }})	20

Table 8. Performance and Commands for update operation

The results show that for Kiva database **update** operation on MongoDB is faster than MySQL.

C. Insert Operations

This section focuses on insert operation and performances.

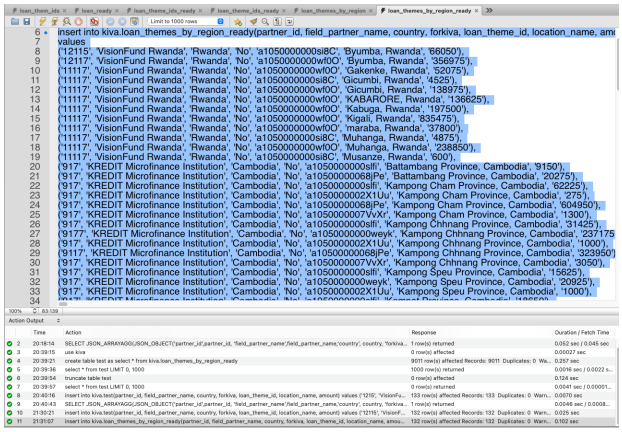


Fig. 14.a Insert 133 records to MySQL

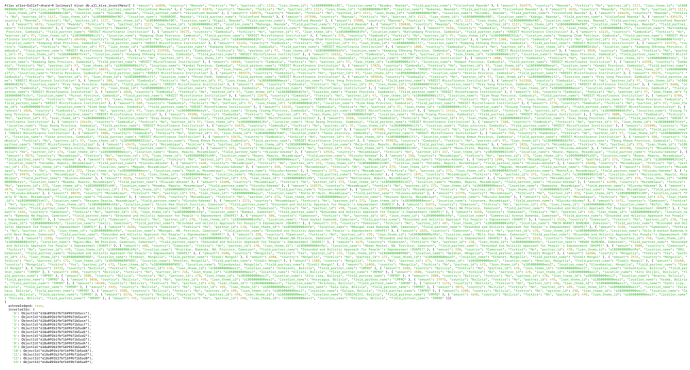


Fig. 14.b Insert 133 documents to all_kiva collection

Fig. 14.c Performance of Insert on MongoDB

Insert - Operation		
DB	Command	Execution Time(ms)
SQL	insert into loan_ready(loan_id, funded_amount, loan_amount, activity, partner_id, borrower_genders, repayment_interval, region, country) values (1653162, 50, 500, 'Motorcycle Transport', 61, 'male', '2014-01-02', 'Phnom Penh', 'Cambodia'), 132 other records	102
NoSQL	db.all_kiva.insertMany([{ "amount": 66050, "country": "Rwanda", "forkiva": "No", "partner_id": 1215, "loan_theme_id": "a1050000000si8C", "location_name": "Byumba, Rwanda", "field_partner_name": "VisionFund Rwanda" }, 132 more records])	20

Table 9. Performance and Commands for insert operation

Results of insert operation show that Kiva database insert operation on MongoDB is faster than MySQL.

D. Delete Operations

This section shows experiments on delete operation on MySQL and MongoDB.

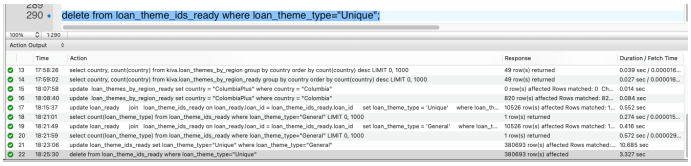


Fig. 15.a Delete records based on condition on MySQL

Fig. 15.b Delete documents based on conditions on MongoDB

Fig. 15.c Performance of delete documents on MongoDB

Delete - Operation		
DB	Command	Execution Time(ms)
SQL	delete from loan_theme_ids_ready where loan_theme_type="Unique";	3300
NoSQL	db.all_kiva.deleteMany({loan_theme_type:"Unique" })	3

Table 10. Performance and Commands for delete operation

The results of this section show that the delete operation on MongoDB is faster than MySQL.

V. Design Choice

The choice of database design is a critical decision and there are important factors such as data size, relationships, structure, and the necessity to enforce a schema and ensure consistency that needs to be addressed. Both the relational database model and the NoSQL database model are appropriate for certain applications. Depending on the problem that the organization is attempting to solve, it will determine whether a NoSQL database model or a relational database model should be used. In addition, some organizations may opt for a hybrid mix of NoSQL and relational databases.

Developers are particularly attracted to the document data model. Objects must be transformed back and forth between the database and objects in the programming language in the relational model used by RDBMS. This can slow down development and add complexity to the program. In a document database, the document can almost directly map to the class structure of the programming language, saving the programmer time[7].

As scale takes precedence, it becomes increasingly expensive to maintain and manage a relational database system and we start looking at non-relational models that don't require a strong schema. Also, the aim of the Kiva platform is to gather the data and perform data analytics on the data to extract information that enhances businesses.

The most frequent operations of this project are inserting new records of data. Update and delete records also might happen from time to time and the data analysis would be required to derive decisions. For this project, after creating, storing, querying both the relational and non-relational models and comparing each of their performances against each other, we conclude that a non-relational schema is the one that works best.

A. Data Updation

A NoSQL database is extremely flexible and allows changes to be made to the database easily as required. With our project, updating records regarding new projects is an operation that is prioritized over others and a comparison of the performance of these operations with MySQL reveals that they are exponentially slower than NoSQL. For example, an update operation performed in MySQL took 10685 ms whereas the same operation, when performed in NoSQL, took only 20 ms. This shows NoSQL is approximately 500x faster than MySQL.

B. Schema Enforcement

NoSQL databases tend to have an implicit schema and are not entirely schemaless. With changing requirements and new data coming frequently it becomes increasingly difficult to modify and enforce a new schema every time. It would take enormous amounts of time to keep rewriting enforcement rules and updating them daily.

C. Record Insertion

Ease of access to the internet has brought inventors and investors closer than ever before. With the means to make available their idea online and gather funding for their projects, entrepreneurs use crowdfunding requests to platforms such as Kiva that are in large demand. This means the creation and insertion of the details of these new projects in the database are much more frequent. A comparison of these operations in MySQL and NoSQL demonstrates that it is much easier and faster to do this in NoSQL with execution times being extremely lower. For instance, the insertion of a record in MongoDB took 20 ms, on the other hand, MySQL took more than 5 times that to insert the same record.

D. Data Analytics

The purpose of creating a database for Kiva Crowdfunding is to analyze this data and draw conclusions about the factors that influence crowdfunding, how financial and demographic aspects affect the creators and their general welfare. The analysis of data requires powerful visualization BI tools. With time MongoDB has built BI capabilities directly into the NoSQL database and allowed users to take a deeper dive into their data. The use of visualization tools available in MongoDB Atlas has allowed us to map the distribution of crowdfunded projects across the world and helped pose meaningful questions such as why a certain part of the world sees a greater proportion of such projects than the rest.

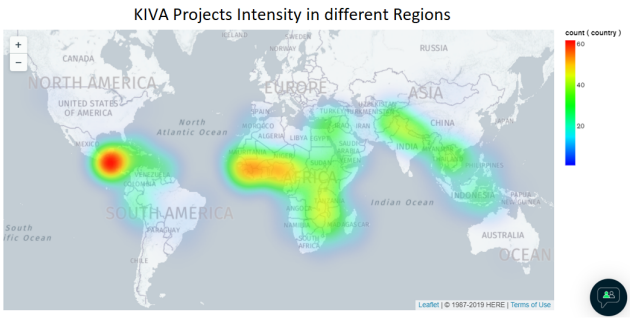


Fig. 16 Heat Map of Distribution of Crowdfunded Projects

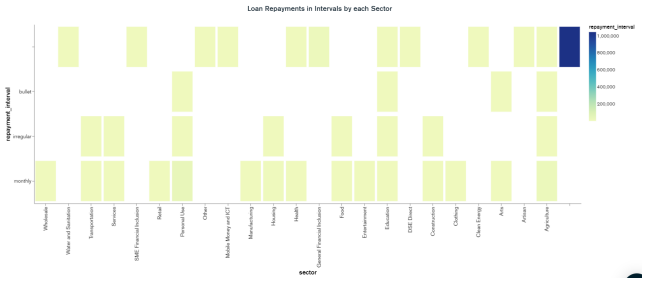


Fig. 17 Distribution of loan repayment interval over various sectors

This type of analysis serves as an answer as to which sector is gaining the most out of such crowdfunding campaigns and thus help improve upon ways in which those who can benefit from it.

Moreover, NoSQL's flexible data modeling is well suited to support dynamic scalability and improved performance for Big Data analytics[8], and it could be leveraged as a new category of data architecture coexisting with traditional SQL databases.

E. Use of Semi- and Unstructured Data

A NoSQL database such as MongoDB is very well suited to handle semi-structured and unstructured data alike. Businesses all the time are changing to best serve partners and customers. To satisfy this requirement, a schema-free database would be a significant factor to align to updated requirements. A database such as one for Kiva requires the storage of unstructured data like log files, spreadsheets, and images. The inability of a traditional relational database model to handle such data does not make it suitable for this database.

The following figure shows the creation of a non-relational database schema for Kiva and the records stored in the collections.

VI. TRADE-OFFS

The selection of either a SQL or a NoSQL database depends entirely on the type of data one has and how they aim to use it. There is no one-size-fits-all policy that applies here and both the schemas come with their own advantages and disadvantages. The selection of NoSQL for this project comes with a compromise on some of the features MySQL provides that MongoDB cannot.

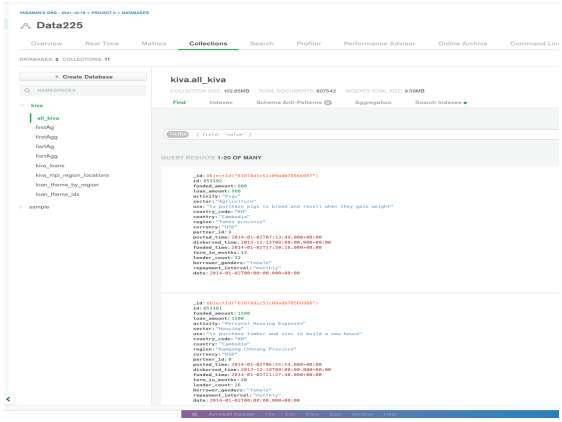


Fig. 18 Collection in MongoDb - 'all_kiva'

A. Complex Queries

Although NoSQL outperforms SQL at faster execution for queries, it is SQL that is able to execute complex queries easily and much faster than NoSQL. The availability of joins in MySQL allows data to be combined from multiple tables and compared easily which is not allowed in NoSQL.

B. Consistency

Unlike SQL databases, NoSQL databases do not follow ACID properties and hence have a latency period before consistency is achieved across all nodes in the database but it is not guaranteed when this will happen. It instead follows BASE properties which describes consistency as a developer's problem, one that should not be handled by the database.

C. An Established Platform

Relational databases are well established and mature technology. SQL standards are well defined and commonly accepted whereas NoSQL is still developing and still has a long way to go.

VII. CONCLUSION

It is not an option to use only SQL or NoSQL for all business practices. Every business has a unique requirement for how they intend to use a database, and these requirements are determined by what the business expects from its database. Both the SQL and NoSQL models have their own set of advantages and disadvantages that each business must weigh before deciding which is best for them. There is no absolute single solution for choosing a database for all the businesses. The database design should be based on the purpose of the business and the cost-performance factor. In this study, we investigated and designed a relational model as well as NoSQL

document-based model for Kiva crowdfunding platform. Based on the performance of each approach and the requirement of the business for big data analytics on the data, we decided to choose NoSQL design over the relational data model for querying and storing the data and performing analytics.

Acknowledgement

We would like to express our sincere gratitude to Professor Simon Shim for his guidance that aided in the completion of this project and helped us understand database concepts using practical examples. We would also like to thank Shiva Abhishek Varma Penmetsa and Rushikesh Jagtap for their continued help and support throughout the development of the project.

References

[1] S. Yu, Crowdfunding and regional entrepreneurial investment: an application of the CrowdBerkeley database, Research Policy, Volume 46, Issue 10, 2017, Pages 1723-1737, ISSN 0048-7333

[2] Burtch, G., Ghose, A., Wattal, S., 2014. Cultural Differences and geography as determinants of online prosocial lending. MIS Q. 38 (3), 773–794.

[3] Mollick, E., Nanda, R., 2015. Wisdom or Madness? Comparing Crowds with Expert Evaluation in Funding the Arts. Manage. Sci. 62 (6), 1533–1553.Mollick, E.R., 2014. The Dynamics of Crowdfunding: An Exploratory Study. J. Bus.Venturing 29 (January (1)), 1–16.

[4] Smith, Tim. “Crowdfunding.” Investopedia, Investopedia, 13Sept.2021,www.investopedia.com/terms/c/crowdfunding

[5] Tzoof Avny Broosh. “ How to Choose the Right Database”,Sept. 2019, https://towardsdatascience.com/how-to-choose-the-right-database-afcf95541741.

[6] Jatin Raisinghani. “Should You Use MongoDB or SQL Databases for Analytics?”, Aug. 2018, https://www.holistics.io/blog/should-you-use-mongodb-or-sql-databases-for-analytics/.

[7] Nance, Cory; Losser, Travis; Iype, Reenu; and Harmon, Gary, "NOSQL VS RDBMS - WHY THERE IS ROOM FOR BOTH" (2013). SAIS 2013Proceedings. 27.

[8] Venkatraman, Sitalakshmi, et al. "SQL versus NoSQL movement with big data analytics." International Journal of Information Technology and Computer Science 8.12 (2016): 59-66.

PROJECT - 2 REPORT

Kiva Crowdfunding Dataset Analysis

Poojitha Katta
Department of Applied Data Science
San Jose State University
San Jose, United States
poojitha.katta@sjsu.edu

Purnima Bhukya
Department of Applied Data Science
San Jose State University
San Jose, United States
purnima.bhukya@sjsu.edu

Deepali Zutshi
Department of Applied Data Science
San Jose State University
San Jose, United States
deepali.zutshi@sjsu.edu

Yasaman Emami
Department of Applied Data Science
San Jose State University
San Jose, United States
yasaman.emami@sjsu.ed

Abstract—The main aim of this project is to create a database for the crowdfunding platform-Kiva, to analyze and evaluate factors which influence various aspects of a crowdfunded project and draw conclusions about them.

Keywords—NoSQL, Crowdfunding, Database, Kiva

I. Introduction

Crowdfunding is a system that enables individuals or ventures to seek small investments, contributions, or loans from a variety of funders online. Kiva is a crowdfunding platform that offers a new financing channel for small and micro businesses as well as individuals. The aim of the project is to build a database management system for Kiva platform to analyze the factors that influence crowdfunded projects by estimating the welfare level of partners in specific regions, based on shared financial and demographic aspects. Technologies such as MongoDB would be used to create the database management system and perform analysis of the data. Using Python, a powerful programming tool, we can observe, understand, and draw conclusions on better funded categories, borrowing patterns and regional analysis from the data. The system can be deployed on cloud-based platforms such as Atlas for better accessibility and security. Apart from that, Charts on MongoDB Compass allow detailed visualization of the data and give us deeper insights. This project can help improve access to crowdfunding, assess borrower welfare levels, by analyzing the growth of previously funded projects and benefit Kiva with a better database system to enhance their platform.

II. Dataset

A. Source

The data was obtained from Kaggle, an online community of data scientists and machine learning practitioners. It was made available by Kiva, an online crowdfunding platform, for the “Data Science for Good” Challenge and invited people to help them build a localized model to estimate various metrics in regions where Kiva has active loans.

B. Dataset Description

The dataset contains 4 csv files with 54 attributes total, which includes 30 string, 7 decimal, 4 datetime and 13 other data types. The first table consists of 20 columns, detailing the id, funded amount, loan amount, country code, country, currency, region, etc. Similarly, the second, third and fourth table outlines data snapshot and can be matched to the loan theme regions to get a loan's location and provides details for id, loan theme id, loan theme type, partner id and MPI (Multidimensional Poverty Index). Extracting several insights from the historical micro-loans over a period and correlating the regional averages by gender, sector, or borrowing behavior to estimate the welfare rate is to be followed.

C. Data Cleaning

The data required cleaning and correcting to be processed and stored into the database. Many of the tables had missing data which was replaced by the mode of the data in that column. Attributes with most of the data missing were removed from the tables altogether. Many of the tables also contained attributes which were duplicated in the same table as well as across multiple tables. These were removed and the data was pre-processed to remove redundancies.

```
In [ ]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')

In [ ]: df = pd.read_csv('kiva_loans.csv')

In [ ]: data = pandas.read_csv('kiva_loans.csv', encoding='utf-8', quotechar='"', delimiter=',')

In [ ]: df.info()

In [ ]: df.describe()

In [ ]: null  = df.isnull().sum().sort_values(ascending = False).reset_index()
        null.columns = ['Column', 'Frequency']
        null

In [ ]: ## tags column consists of 10000 null values so remove the column.
        df.drop('tags', axis=1,inplace=True)

In [ ]: ## this column null values are replaced with the mode
        df['funded_time'].mode()

In [ ]: df['borrower_genders'].mode()

In [ ]: df['funded_time'].fillna(df['funded_time'].mode(), inplace = True)

In [ ]: df['borrower_genders'].fillna(df['borrower_genders'].mode(), inplace = True)

In [ ]: df.dropna(inplace = True)

In [ ]: df.isna().sum()

In [ ]: data.to_csv('kiva_mpi_region_locations', encoding='utf-8', index = False)
```

Fig. 1 Jupyter Notebook for Data Cleaning

III. DATA MODEL

A NoSQL database system such as MongoDB requires work differently than a traditional relational database system. Where an RDBMS is incapable of handling and storing unstructured and semi-structured data, NoSQL can store, process, and visualize such data with ease. MongoDB represents the information as a JSON document instead of the row and column format adopted by the RDBMS system. MongoDB stores data in structures called 'collections' and the data entries are called 'documents'. The documents contain key-value pairs of various types and can also consist of arrays, nested documents, etc. Each document can have its own structure and are self-describing in nature.

A. Document Structure

The following diagram represents the various collections created by importing the data into MongoDB. Each attribute in the collections represents the various documents keys and is of variable types (string, integer, date, timestamp, etc.).

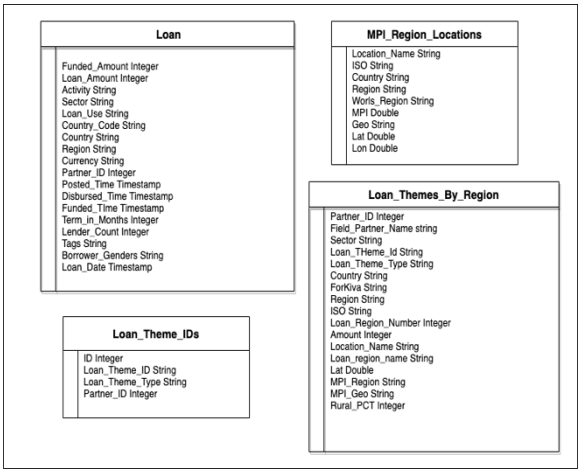


Fig. 2 Data Model

Each collection in the database contains documents with a JSON-like structure having key-value pairs for each of the entries in the collection. A total of four collections were created initially and imported into MongoDB for this project. The following are the structures of the documents in each of the collections. In the end, all of the four collections merge into one collection called the "all_kiva" collection.

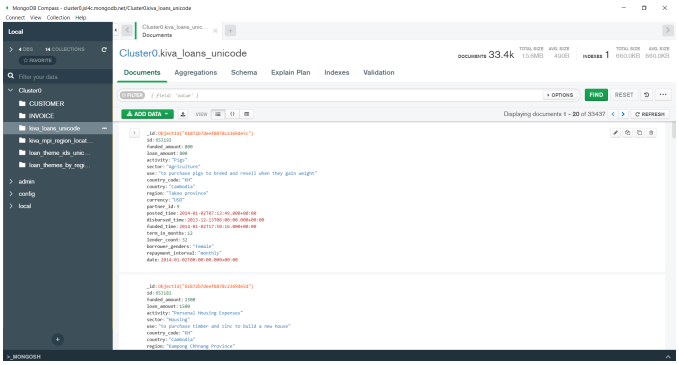


Fig. 3 Kiva_loans

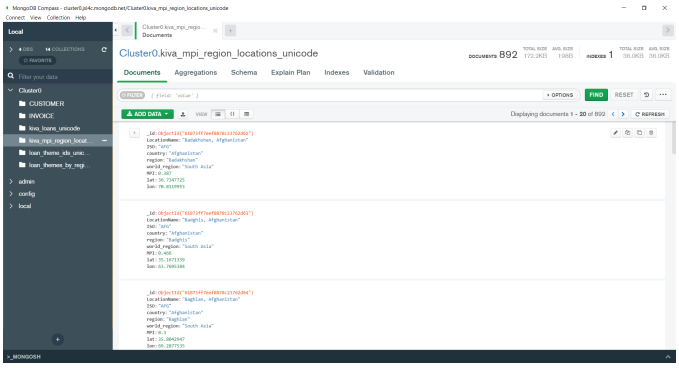


Fig. 4 Kiva_mpi_region

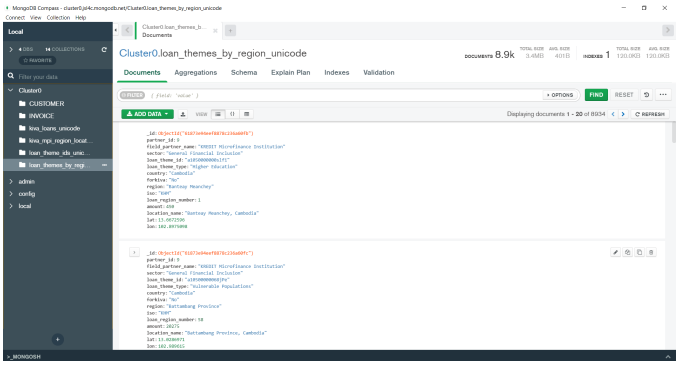


Fig. 5 Kiva_loan_themes_by_region

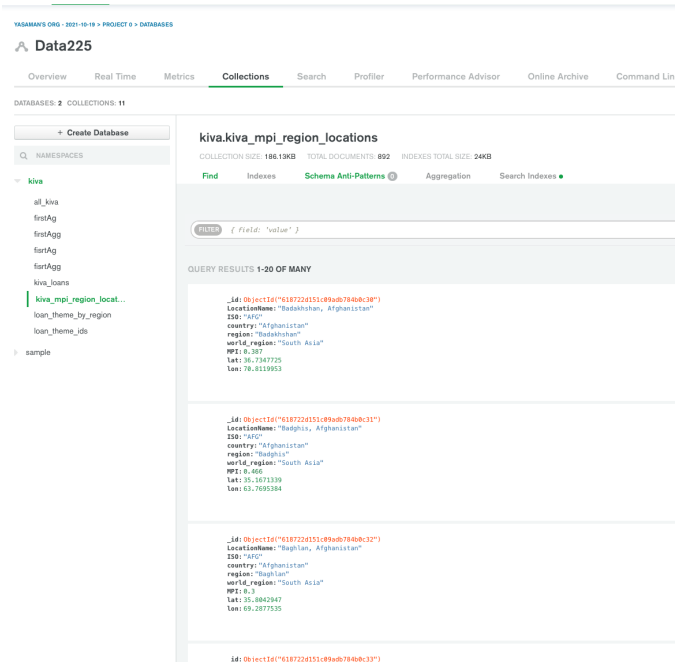


Fig.6 kiva_mpi_region_locations

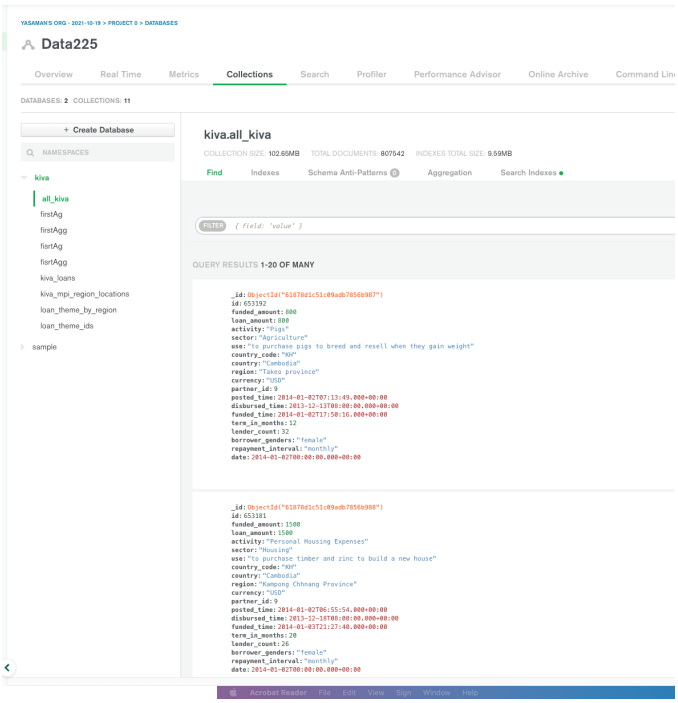


Fig. 7 all_kiva

B. Denormalization

Denormalization is the process of improving the read performance of a database at the expense of some of the write performance by either making redundant copies of the data or by grouping the data. For this project, the previously normalized data from Project-1 was denormalized to improve database functionalities.

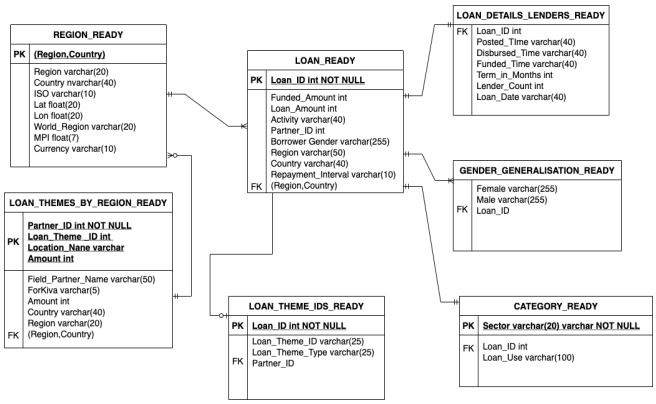


Fig. 8 Normalized Data (Project-1)

The diagram below represents the Kiva database and the collection 'All_Kiva' which is denormalized with the various keys to store the data.

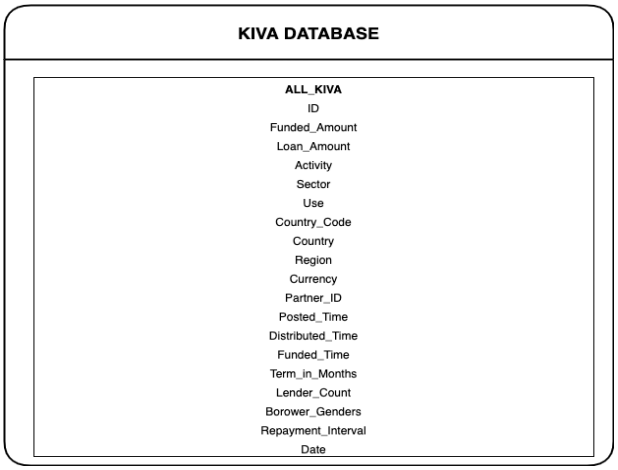


Fig. 9 Document Structure of Kiva- MongoDB

The figure below shows the records as in the collection of the denormalized data. MongoDB removes the fields where the key has a null value and displays only the pairs with valid data as opposed to the relational database which displays the missing values as 'Null'.

Fig. 10 all_kiva key-value pairs

IV. NoSQL Queries

In this section all the SQL queries performed on the data in relational database model have been applied to denormalized form of data stored in no sql format in MongoDB. The queries run on MongoShell connected to the cloud cluster of MongoDB. The queries considered to perform are similar so the comparison between NoSQL data format and RDBMS can be performed to get a better understanding of performance metrics and execution time as well as the syntax. Figure below shows the all key-value pairs from kiva where the load_amount is between 500$ and 1500$. Also Fig. 9 is displaying the count of each loan.

Fig. 11 loan_amount between 500$,1500$

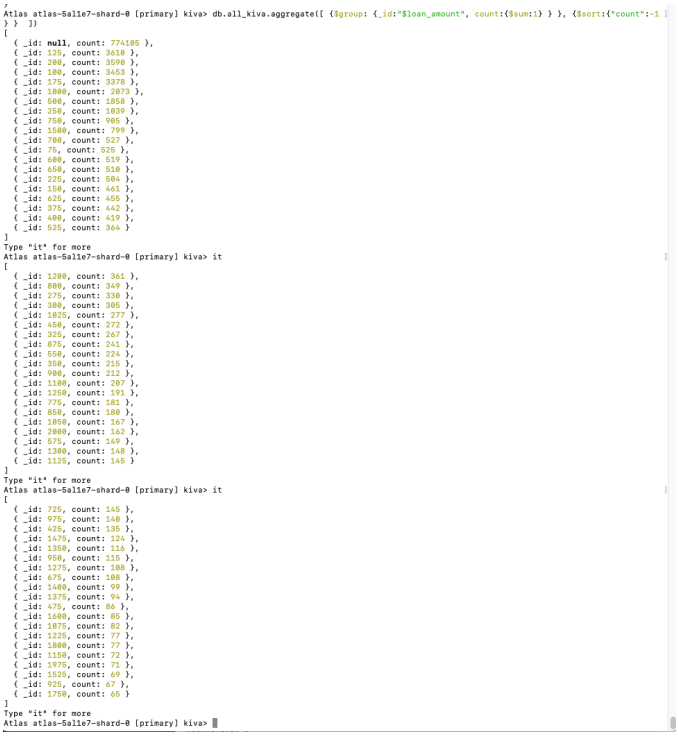


Fig. 12 Count of Each Loan

In Fig.13 below is a query to display information on members from specific regions.

Fig. 13. Members in Specific Region

Fig. 14 is showing the applicants from countries that their name contains “am” and their attribute of ‘forkiva’ is set to “yes”.

Fig. 14 Countries Like ‘/.*am.*/’

Fig. 15 below is displaying the query and the returned result for the applicants that are active in the food industry and they have specialty and their loan_theme_type is not “General” .

Fig. 15 Show the loans in food industry-specific type (exclude general types)

Last Query from Report 1 is the complex query that joins and groups by having clauses in Relational data format to return a count of activity along with other attributes for where the term in months is bigger than 10 and the count of activity is also bigger than 10 and it's ordered by count.

Fig. 16 depicts the query in MongoDB shell for NoSQLl kiva database.

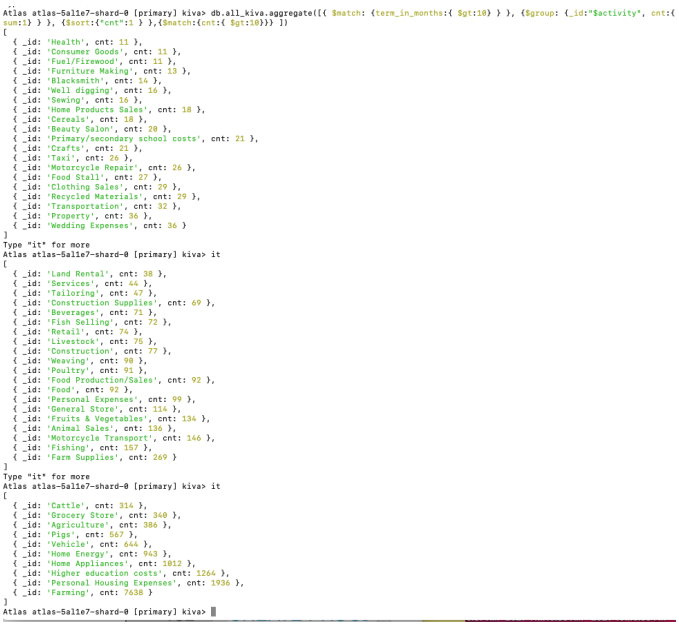


Fig. 16 Display number of times each activity gets funded with terms of bigger 10 months and if they are funded more than 10 times

Figures 17 to 19 below demonstrates pdate Delete and Insert commands in all_kiva collection in MongoDB using Mongo shellupdate commands performed in NoSQL.

```
Atlas atlas-5a11e7-shard-0 [primary] kiva> db.all_kiva.updateMany({loan_theme_type:"General" },{$set:{loan_theme_type:"Unique" }} )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 384876,
  modifiedCount: 384876,
  upsertedCount: 0
}
```

Fig. 17 Update documents set a value on condition

```
Atlas atlas-5a11e7-shard-0 [primary] kiva> db.all_kiva.deleteMany({loan_theme_type:"Unique" })
{ acknowledged: true, deletedCount: 384876 }
Atlas atlas-5a11e7-shard-0 [primary] kiva>
```

Fig. 18 Delete documents based on conditions

Fig. 19 Insert 133 documents to all_kiva collection

V. Data Visualization

The analysis of the Kiva Crowdfunding Dataset gives an insight into the various factors that affect and influence the sustainability of crowdfunded projects. It also reveals the distribution of these projects in the world, gives information about the people involved in the creation of the projects and how the projects are distributed among the different sectors.

- The following pie chart shows the distribution of the number of loans taken by various crowdfunding project owners and their distribution in the different regions.

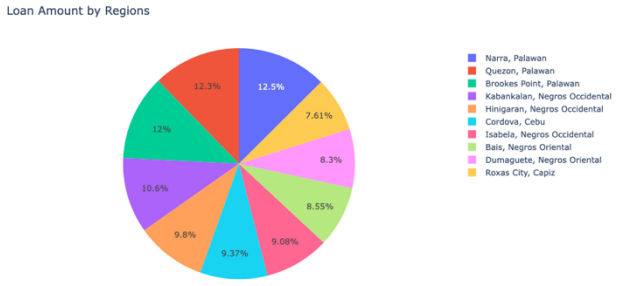


Fig. 20 Loan distribution pie chart

- Analysis of the data reveals the relation between the different projects and the amount that was funded to them with the gender of the borrowers and we see that a greater number of female borrowers as compared to male borrowers and that as the funding amount increases, the number of projects funded for that amount decreases.

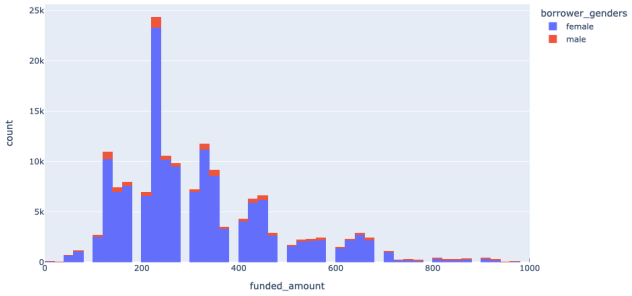


Fig. 21 Funded amounthistogram color-coded by gender

- The below boxplot shows the various sectors of crowdfunded projects and the distribution of the loan amount for each of those sectors as well as the quartiles for the loan amount in those sectors. We see that the wholesale sector has the greatest amount of loans whereas the personal use sector has the lowest.

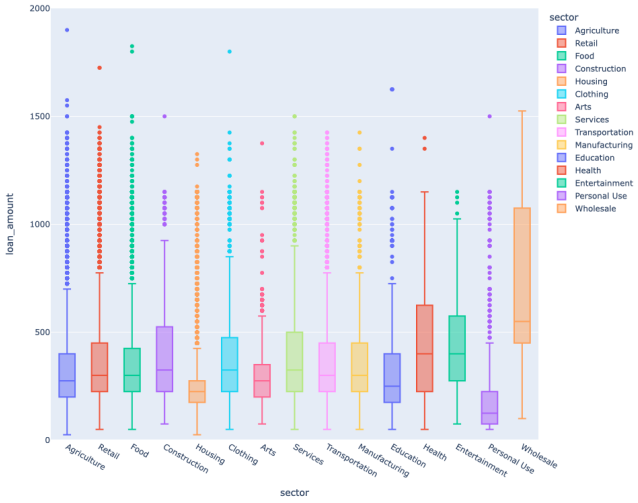


Fig. 22 Project sectors box plot

- The scatter plot shows the cumulative funding of projects in different sectors with respect to the amount of funding they receive. Projects belonging to the retail sector receive the highest funding closely followed by food and agriculture whereas manufacturing projects are the lowest funded.

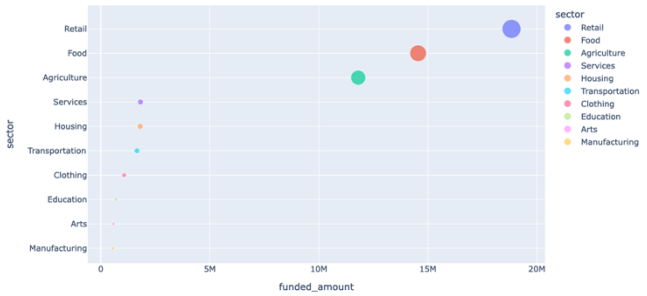


Fig. 23 Funding of projects in different sectors with respect to the amount of funding scatter plot

- The bar graph represents the top ten uses for loans taken by borrowers and their counts. There are about 70,000 different uses for the loans with different numbers of occurrences and we see that the use case 'to build sanitation facilities' is the most seen use.

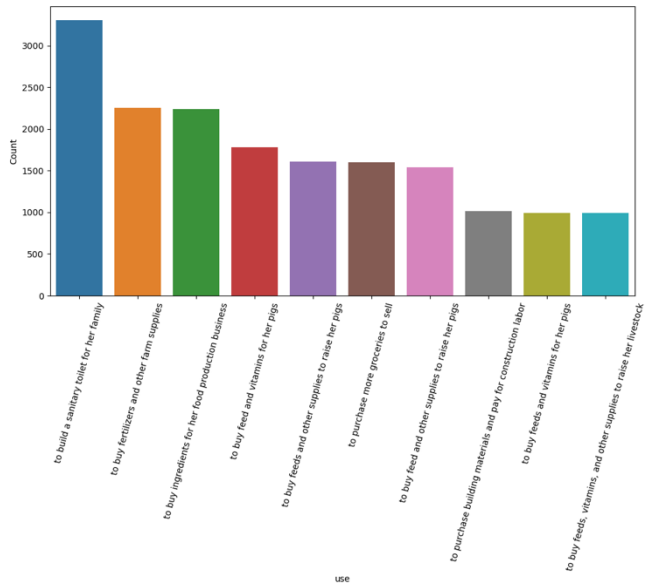


Fig. 24 Top 10 loan use bar chart

- The following heat map shows the degree of relation between the funded amount, loan amount, the term of the loan(in months), and the lender count for each of the projects in the Kiva database.

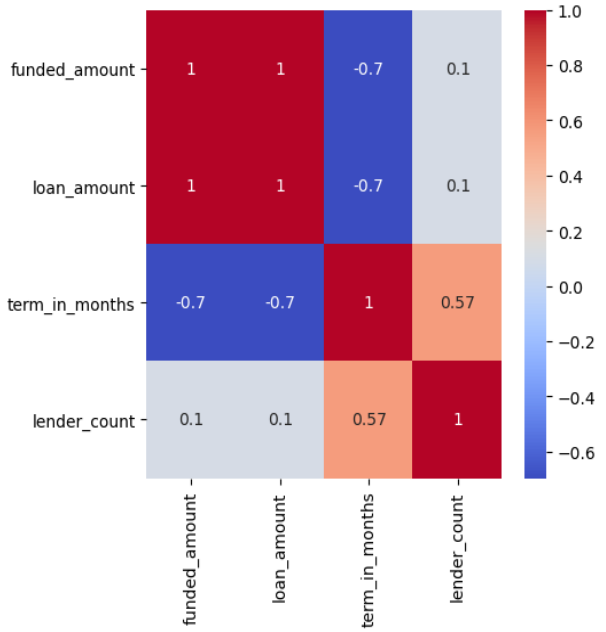


Fig. 25 funded amount, loan amount, the term of the loan(in months), and the lender coun heatmap

- The following map shows the distribution of the crowdfunded projects based on their location. Hovering over the location indicates their latitude, longitude, country name, and region name. The heat map helps us visualize areas with a concentration of these projects. It can be interpreted that South America and West Africa have the highest concentration of crowdfunding projects than anywhere else in the world.

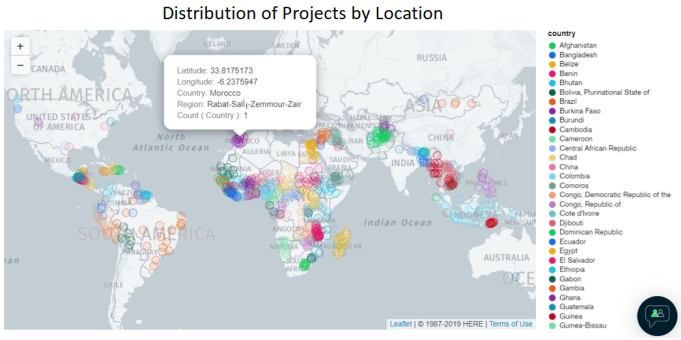


Fig. 26. Distribution of projects by location

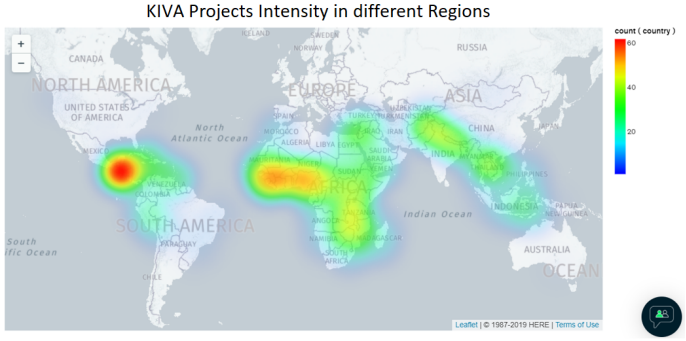


Fig. 27 Kiva project intensity in regions

- The pie chart shows how much funding was received by the projects each month over a period of three years. The greater the area of the sector the more the funding was dispensed during that particular month.

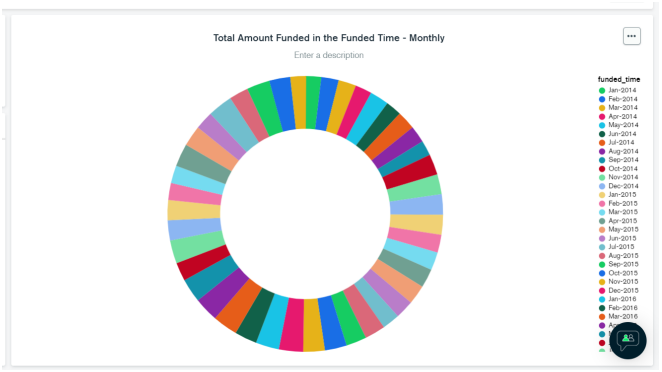


Fig. 28 Monthly funded projects

- Each partner involved in the crowdfunding process contributed to the success of the project and the below scatter plot shows funds contributed by each of the partners to all the projects they were involved with. We can see that contributions by partners range from a few hundred thousand to a few million.

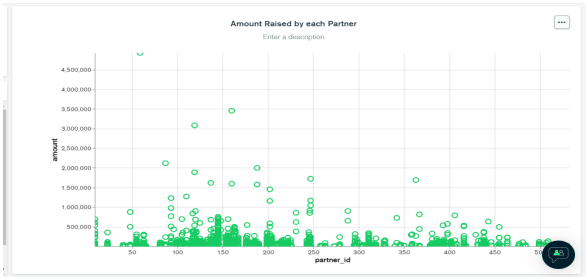


Fig. 29 Partners vs Projects

- The bar graph indicates the loan themes used in each of the sectors. The most different uses for the loan amount are in the General financial Inclusion Sector and the least in the DSE Direct sector. Another graph reveals the distribution of these sectors with their uses in different countries.

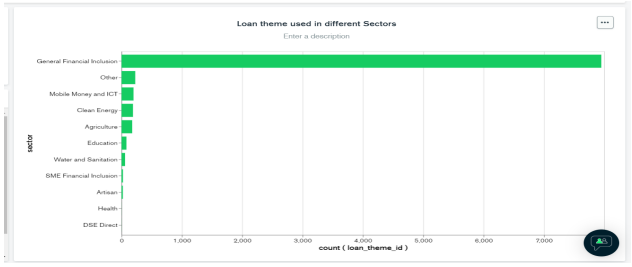


Fig. 30 Kiva loan themes by different sectors

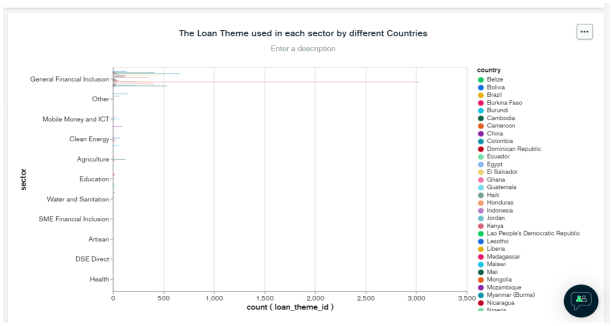


Fig. 31 Kiva loan themes by different countries

VI. Connectivity to Atlas

MongoDB Atlas is a Database-as-a-Service(DBaaS) that gives users the power to create, scale, manage NoSQL databases on the cloud with the flexibility of choosing their own provider(Azure, AWS, and GCP). Deployment of the database on the cloud ensures access, security, timely software updates, and physical hardware independence to database users. For this project, a database 'Kiva' was created on MongoDB and deployed on Atlas.

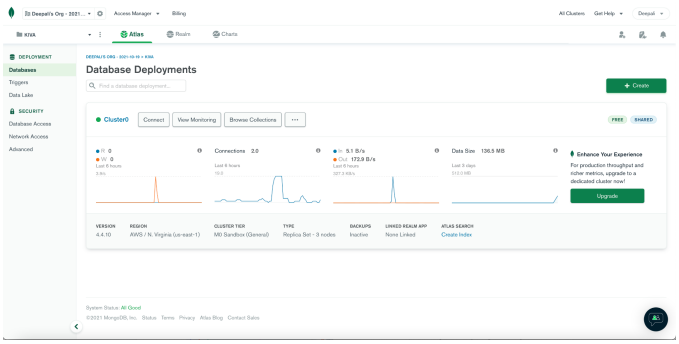


Fig. 32 MongoDB Atlas Connectivity-1

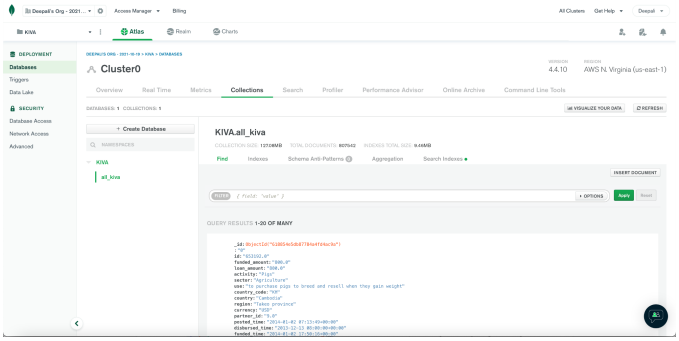


Fig. 33 MongoDB Atlas Connectivity-2

VII. NoSQL Performance Measurement

NoSQL is a non-relational database that can handle semi-structured and unstructured data and enables the user to query the data at speed. The prioritisation of speed, adaptability

and availability over consistency makes NoSQL well suited for big data manipulation over other querying languages. The query performance here is measured using the explain() function available in MongoDB. This method returns a document containing the query plan as well as the execution statistics.

Fig. 34 below shows loan_amounts between $500-$1500 performance statistics. Also, the count of each loan_amount and then ordering the results in descending format is shown in Fig. 35 the '_id' here representing the loan_amount value.

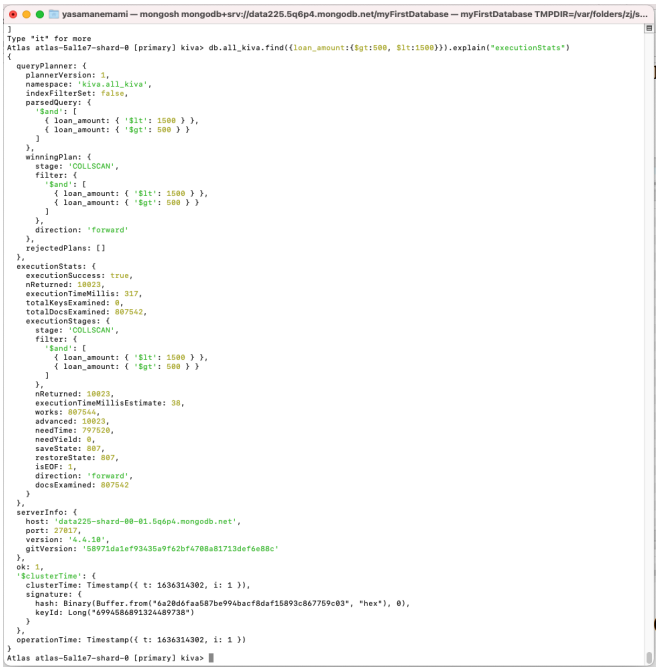


Fig. 34 Loans between $50-$1500 query performance statistics

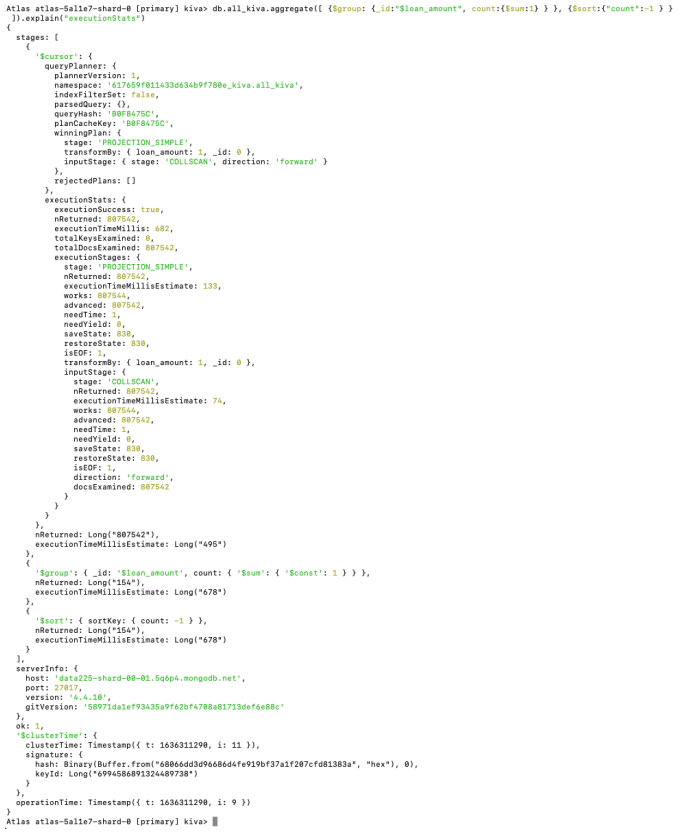


Fig. 35 Loan counts performance statistics

Fig. 36 represents the performance statistics for a query of countries whose name has "am" and they are applying for kiva.

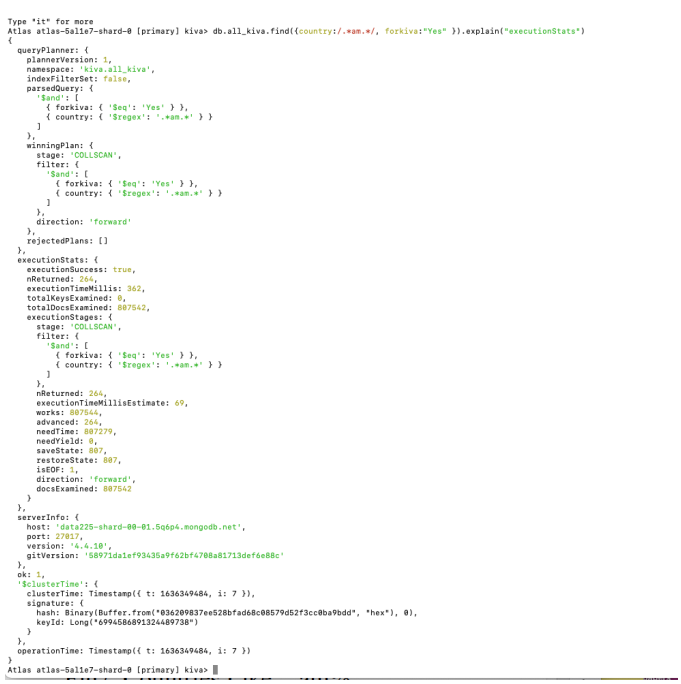


Fig. 36 Country like ".*am.*" statistics

Fig. 37 shows the performance for specific countries and then ordering desc by country performance.

Fig. 37 Country specific query performance

Figure 38 is showing the applications in the food industry excluding General purpose applicants' query performance.

Fig. 38 Food industry applicants query performance

Fig. 39 shows the performance for the query of the loan activities group by where they have more than 10 months terms and also the count of the group is more than 10 at the end the result is ordered by count.

Fig. 39 Display number of times each activity gets funded with terms of bigger 10 months and if they are funded more than 10 times query performance

Up to this point, all of the queries were equivalent to select queries in Relational databases and the statistics display that queries run faster on MySQL in comparison with NoSQL takes more time.

Now we perform Update and Delete commands on both MySQL and MongoDB which shows this type of data manipulation takes more time on MySQL than MongoDB.

Fig40 and Fig41 illustrate the results of performing the same commands on MongoDB and MySQL workbench.

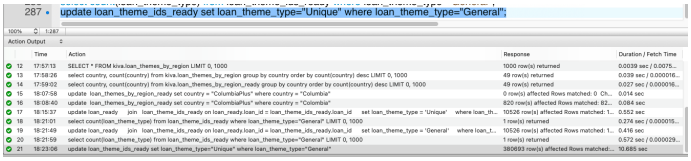


Fig.40 Update on 380693records MySQL

Fig. 41 Update 384876 records MongoDB. Execution Time for Query

Query\Database	SQL	NoSQL
Loan between $50-$500 Query	2.6 ms	317 ms
Loan Counts	32 ms	682 ms
Country Like “.*am.*” Query	17 ms	362 ms
Country-Specific Query	20 ms	340 ms
Food Industry Applicants Query	72 ms	305 ms
Funding of Activity With Term >10 Months	81 ms	347 ms
Insert Query	102 ms	20 ms
Update Query	10.685s	20 ms
Delete Query	3.327 s	30 ms

Table 1 Execution Time: SQL vs. NoSQL

Both MySQL and MongoDB have their advantages and disadvantages. Being relational-oriented, MySQL is scalable and has consistency, it supports ACID transactions. Whereas, MongoDB is flexible and faster. MySQL DBM systems need to maintain a consistent state and impact the performance a bit. Whereas, MongoDB stores data as documents or key-value pairs and thus performs data manipulation operations better.

The performance of the database systems depends on the context of the data. NoSQL performs better with unstructured data such as document-oriented, graph-oriented, etc. whereas SQL works best with structured data. In this project, the data was structured and the structure was considered unlikely to be changed, hence SQL takes a shorter time to execute than NoSQL. Also, NoSQL works faster with big data as well as rapid growth data, however, the data used here is not huge. Selecting between NoSQL databases versus Relational databases depends on the priority of the project. For instance, if there are a great number of connection calls at the same time to a database on one server, it's better to use NoSQL structures to be able to handle the connections since Relational databases are capable of handling a few calls at once. Therefore it can be said that NoSQL is not faster than MySQL and MySQL is not faster than NoSQL.

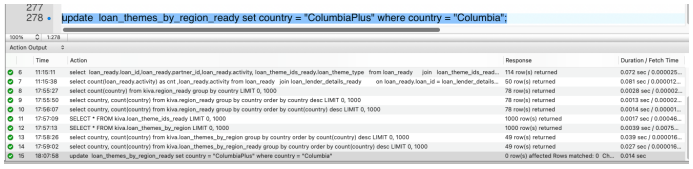


Fig. 38 Update Command MySQL

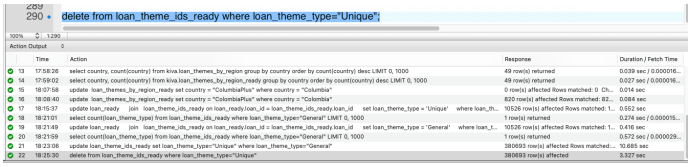


FIg. 39 Delete Command MYSQL

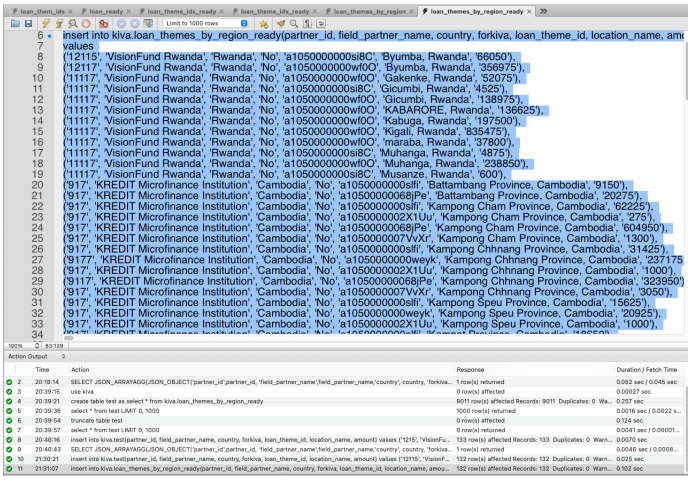


Fig. 40 Insert Command MySQL

Fig. 41 Update Command NoSQL

Fig. 42 Insert Command NoSQL

Fig. 43 Delete Command NoSQL

Acknowledgment

We would like to express our sincere gratitude to Professor Simon Shim for his guidance that aided in the completion of this project and helped us understand database concepts using practical examples. We would also like to thank Shiva Abhishek Varma Penmetsa and Rushikesh Jagtap for their continued help and support throughout the development of the project.

References

[1] S. Yu, Crowdfunding and regional entrepreneurial investment: an application of the CrowdBerkeley database, Research Policy, Volume 46, Issue 10, 2017, Pages 1723-1737, ISSN 0048-7333

[2] Burtch, G., Ghose, A., Wattal, S., 2014. Cultural Differences and geography as determinants of online prosocial lending. MIS Q. 38 (3), 773–794.

[3] Mollick, E., Nanda, R., 2015. Wisdom or Madness? Comparing Crowds with Expert Evaluation in Funding the Arts. Manage. Sci. 62 (6), 1533–1553.Mollick, E.R., 2014. The Dynamics of Crowdfunding: An Exploratory Study. J. Bus.Venturing 29 (January (1)), 1–16.

[4] Smith, Tim. “Crowdfunding.” Investopedia, Investopedia, 13Sept.2021,www.investopedia.com/terms/c/crowdfunding .

PROJECT - 1 REPORT

Kiva Crowdfunding Dataset Analysis

Poojitha Katta
Department of Applied Data Science
San Jose State University
San Jose, United States
poojitha.katta@sjsu.edu

Purnima Bhukya
Department of Applied Data Science
San Jose State University
San Jose, United States
purnima.bhukya@sjsu.edu

Deepali Zutshi
Department of Applied Data Science
San Jose State University
San Jose, United States
deepali.zutshi@sjsu.edu

Yasaman Emami
Department of Applied Data Science
San Jose State University
San Jose, United States
yasaman.emami@sjsu.edu

Abstract—The main aim of this project is to create a database for the crowdfunding platform-Kiva, to analyze and evaluate factors which influence various aspects of a crowdfunded project and draw conclusions about them.

Keywords—MySQL, Crowdfunding, Database, Kiva

I. Introduction

Crowdfunding is a system that enables individuals or ventures to seek small investments, contributions, or loans from a variety of funders online. Kiva is a crowdfunding platform that offers a new financing channel for small and micro businesses as well as individuals. The aim of the project is to build a database management system for Kiva platform to analyze the factors that influence crowdfunded projects by estimating the welfare level of partners in specific regions, based on shared financial and demographic aspects. Technologies such as MySQL workbench would be used to create the database management system and its schema including relationships based on region, funding, project type, etc. Using Tableau, a powerful visualization tool, we can observe, understand, and draw conclusions on better funded categories, borrowing patterns and regional analysis from the data. The system can be deployed on cloud-based platforms such as AWS using Python for better accessibility and security. This project can help improve access to crowdfunding, assess borrower welfare levels, by analyzing the growth of previously funded projects and benefit Kiva with a better database system to enhance their platform.

II. Dataset

A. Source

The data was obtained from Kaggle, an online community of data scientists and machine learning practitioners. It was made available by Kiva, an online crowdfunding platform, for the "Data Science for Goof" Challenge and invited people to help them build a localized model to estimate various metrics in regions where Kiva has active loans.

B. Dataset Description

The dataset contains 4 csv files with 54 attributes total, which includes 30 string, 7 decimal, 4 datetime and 13 other data types. The first table consists of 20 columns, detailing the id, funded amount, loan amount, country code, country, currency, region, etc. Similarly, the second, third and fourth table outlines data snapshot and can be matched to the loan theme regions to get a loan's location and provides details for id, loan theme id, loan theme type, partner id and MPI (Multidimensional Poverty Index). Extracting several insights from the historical micro-loans over a period and correlating the regional averages by gender, sector, or borrowing behavior to estimate the welfare rate is to be followed.

C. Data Cleaning

The data required cleaning and correcting to be processed and stored into the database. Many of the tables had missing data which was replaced by the mode of the data in that column. Attributes with a majority of the data missing were removed from the tables altogether. Many of the tables also contained attributes which were duplicated in the same table as well as across multiple tables. These were removed and the data was normalized to create tables for columns which were repeated across several tables.

```
In [ ]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')

In [ ]: df = pd.read_csv('kiva_loans.csv')

In [ ]: data = pandas.read_csv('kiva_loans.csv', encoding='utf-8', quotechar='"', delimiter=',')

In [ ]: df.info()

In [ ]: df.describe()

In [ ]: null  = df.isnull().sum().sort_values(ascending = False).reset_index()
        null.columns = ['Column', 'Frequency']
        null

In [ ]: ## tags column consists of 10000 null values so remove the column.
        df.drop('tags', axis=1,inplace=True)

In [ ]: ## this column null values are replaced with the mode
        df['funded_time'].mode()

In [ ]: df['borrower_genders'].mode()

In [ ]: df['funded_time'].fillna(df['funded_time'].mode(), inplace = True)

In [ ]: df['borrower_genders'].fillna(df['borrower_genders'].mode(), inplace = True)

In [ ]: df.dropna(inplace = True)

In [ ]: df.isna().sum()

In [ ]: data.to_csv('kiva_mpi_region_locations', encoding='utf-8', index = False)
```

Fig. 1 Jupyter Notebook for Data Cleaning

III. ENTITY RELATIONSHIP DIAGRAM

An entity relationship diagram is a flowchart that explains how the entities (place or object or person) are related to one another within an organization. An ER diagram is essential in the modelling of data of any organization in a graphical manner that is easily understood by the users of the database.

A. Denormalized ER Diagram

- The data as available in its raw form online was not normalized. The tables contained repeated attributes with a few columns missing the data.
- The ER diagram for the raw data was created as follows:

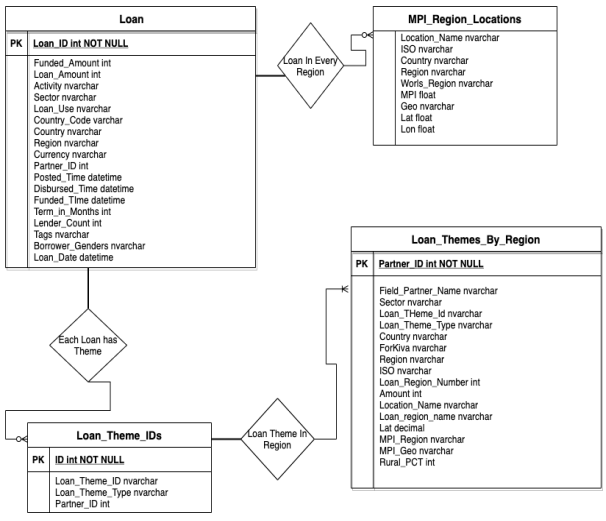


Fig. 2 ER Diagram (1)

B. Normalization

- The process of normalization structures the database to the "normal forms" in order to reduce data redundancy, as well as to improve the integrity of the data.
- The raw dataset contained 4 tables which were further broken down to create 7 tables and improve the overall quality of data.
- The attributes 'ISO ', 'country', 'latitude', 'longitude', 'world_region', 'location' were common in many of the entities. These were extracted to create a separate entity table called 'Region' which was then linked to the table 'Loan' to define the loan details for each region.
- The entity 'Loan' contained the 'borrower genders' attribute which was multi-values and was thus moved to create a separate table called 'Gender Generalization' to specify the genders of borrowers for each of the projects.
- Similarly, an entity 'Category' was created to relate the various project sectors to the loan ID number and how the loan amount was being used.

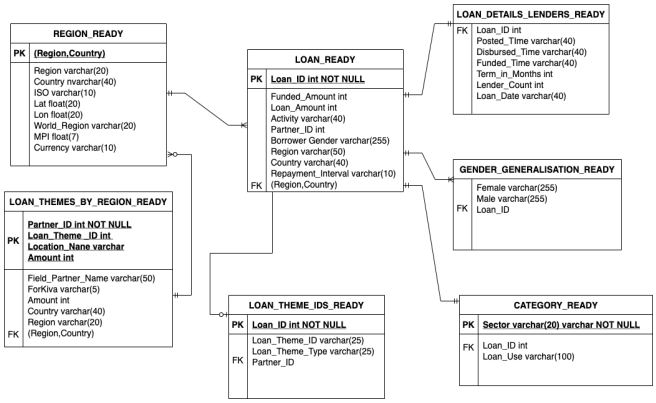


Fig. 3 ER Diagram (2)

C. Primary Keys & Foreign Keys

- Primary keys are attributes in each entity table that are used to uniquely identify each entry in the entity table.
- Foreign keys are attributes in a table that refer to the primary keys of another table. A primary key, foreign key presence defines a relation between the two entity tables.
- In the database that was created, attributes such as 'Loan_ID', 'Loan_Theme_ID', 'Sector' are used to create primary key-foreign key relations among the entities.
- Composite keys are a set of attributes in a table that uniquely define the data for each row of the table. For the entity, 'Loan_Themes_By_Region_Ready' a composite key containing 'Partner_ID', 'Loan_Theme_ID, 'Location_Name', and 'Amount' was created.
- The composite key '(Region,Country)' was defined for the 'Region_Ready' entity table and used as a foregin composite key in the 'Loan_Ready' table.

D. Business Rules

- Kiva is a crowdfunding platform which has lots of borrowers and lenders as members.
- The first business requirement is that a member cannot be resident of Crimea, Cuba, Iran, Syria,

North Korea, or Sudan, which is stored in several attributes like country, country_code and region in the first table in the database and should be checked against this rule.

IV. SQL QUERIES

After the designing and creation of the database and loading the data into the tables, the following SQL queries were performed to update the data and get insights from it.

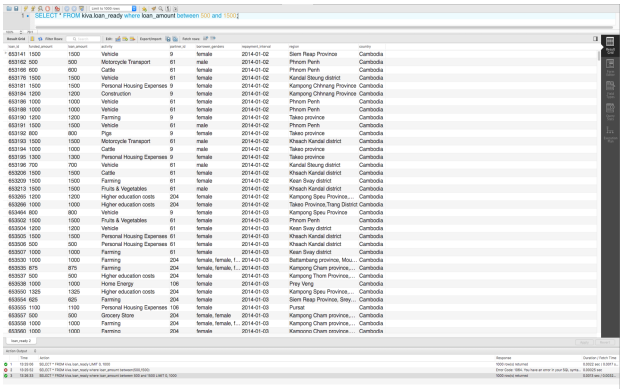


Fig 4. Loan Amount Between (500,1500)

In Fig. 4, all of the attributes like activity, loan amount, borrower gender, region,.. for the records of the table loan_ready that the loan_amount value is between 500 and 1500 are displayed.

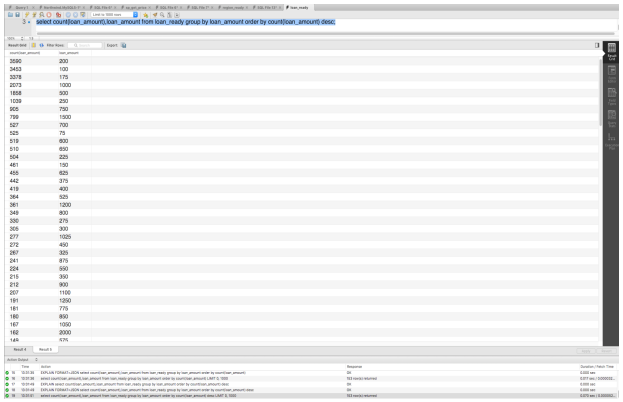


Fig 5. Count of Each Loan

Fig. 5 is displaying how many members applied for the same amount of loan regardless of region and other attributes.

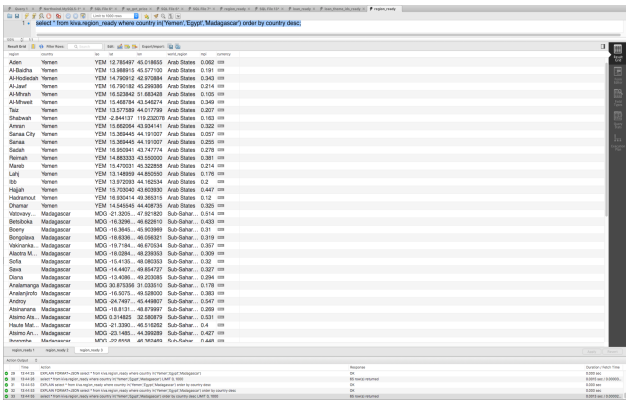


Fig 6. Members in Specific Region

In Fig. 6, all the members who are from specific countries which includes 'Yemen, Egypt or Madagascar' are shown.

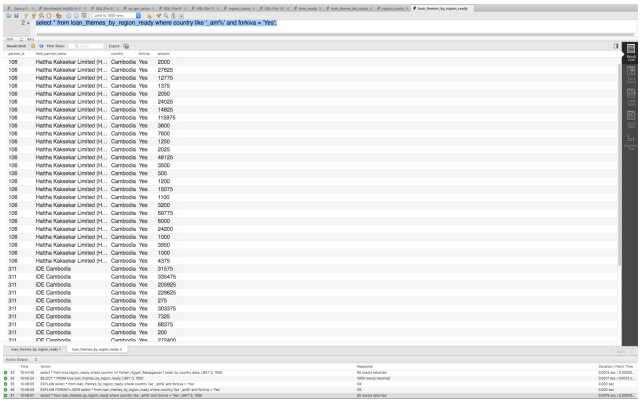


Fig7. Countries Like '_am%'

In Fig. 7 all the members that their country name contains letters 'am' as their second and third letter and they are applying for kiva are displayed.

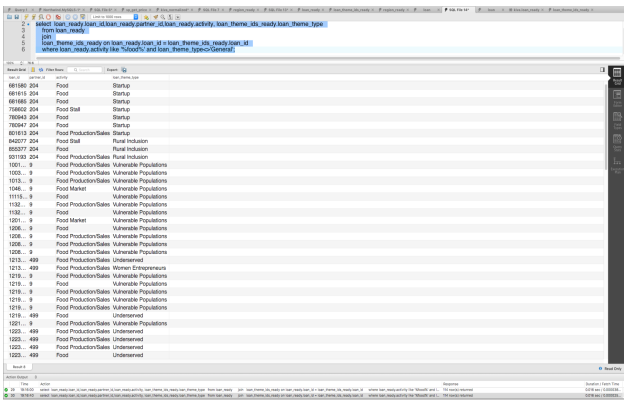


Fig8. Display records from join of two tables to show the loans in food industry specific type (exclude general types)

Fig. 8 is showing the records in which loan_id along with the partner_id their activity and the loan theme which might be different for each partner and the data comes from joining two tables of loan_ready and themes_ids_ready for partners which they are active in food industry and the loan_theme_type is not 'General' they are applying for a specific type.

```
select count(loan_ready.activity) as cnt ,loan_ready.activity from loan_ready
    join loan_lender_details_ready
    on loan_ready.loan_id = loan_lender_details_ready.loan_id
    where loan_lender_details_ready.term_in_months>10
    group by loan_ready.activity
    having cnt>10
    order by cnt;
```

cnt	activity
11	Fuel/Firewood
11	Consumer Goods
11	Health
13	Furniture Making
14	Blacksmith
16	Well digging
16	Sewing
18	Home Products Sales
18	Cereals
20	Beauty Salon
21	Primary/secondary…
21	Crafts
26	Taxi
26	Motorcycle Repair
27	Food Stall
29	Clothing Sales
29	Recycled Materials
32	Transportation
36	Property
36	Wedding Expenses
38	Land Rental
44	Services
47	Tailoring
69	Construction Supplies
71	Beverages
72	Fish Selling
74	Retail
75	Livestock

Fig. 9 Display number of times each activity get funded with terms of bigger 10 months and if they are funded more than 10 times

In Fig. 9, the activities which were funded more than 10 times and they are getting funded with the terms more than 10 months from lenders are shown the result is coming from the source of two tables and filtered for above a threshold.

V. Triggers & Procedures

- Procedures are a set of statements that are stored and called upon to be run multiple times and help reduce network traffic and computations.
- We have created several stored procedures to load the data from original tables into the normalized form tables.

```
DELIMITER $$
CREATE PROCEDURE load_data_into_loan_lender_details_ready()
BEGIN
    INSERT INTO loan_lender_details_ready(posted_time, disbursed_time,
    funded_time, term_in_months, lender_count, loan_date, loan_id)
    select posted_time, disbursed_time, funded_time,
    term_in_months, lender_count, loan_date,loan_id from loan;
END $$
DELIMITER ;

call load_data_into_loan_lender_details_ready();
```

Action Output

	Time	Action	Response	Duration / Fetch Time
71	20:12:59	TRUNCATE `kiva`.`loan_lender_details_ready`	OK	0.000 sec
72	20:13:10	call load_data_into_loan_lender_details_ready()	33437 row(s) affected	0.463 sec

Fig. 10 sp1 load_data_lender_details_ready()

- In Fig. 10, we are inserting data from the loan table into loan_lender_details_ready and calling the stored procedure would pull 33437 tuples of data into our normalized table.
- Fig. 11 is shows the data load procedure for another table which loads 892 records from the original mpi_region_locations table into the region_ready.

```
DELIMITER $$
CREATE PROCEDURE region_ready()
BEGIN
    INSERT INTO region_ready (region,country, iso, lat, lon, world_region,
    mpi) SELECT region,country, iso, lat, lon, world_region,
    mpi FROM mpi_region_locations;
END $$
DELIMITER ;

call region_ready();
```

Action Output

	Time	Action	Response	Duration / Fetch Time
92	20:27:29	CREATE PROCEDUR...	0 row(s) affected	0.067 sec
93	20:27:45	call region_ready()	892 row(s) affected	0.043 sec

Fig. 11 sp2 region_ready()

- In Fig. 12, using stored procedure to insert data into loan_themes_by_region_ready and by calling the loan_themes stored procedure we were able to insert 8876 records for the required attribute from the original loan_them_by_region table.

```
DELIMITER $$
CREATE PROCEDURE loan_themes()
BEGIN
    INSERT INTO loan_themes_by_region_ready (
    partner_id, field_partner_name, country, forkiva,location_name,loan_theme_id,amount)
    SELECT partner_id, field_partner_name, country , forkiva , location_name ,
    loan_theme_id,amount FROM loan_themes_by_region;
END $$
DELIMITER ;
call loan_themes();
```

Action Output

	Time	Action	Response	Duration / Fetch Time
96	20:40:10	CREATE PROCEDUR...	0 row(s) affected	0.103 sec
97	20:40:25	call loan_themes()	8876 row(s) affected	0.181 sec

Fig. 12 sp3 loan_themes()

Similarly, we have stored procedures for all the tables after 3NF so calling those we load data into different tables.

- Triggers are database objects that are activated when specified events occur.

- In our crowdfunding project there is a restriction for some countries that cannot be a member as a lender or borrower in Kiva platform so every time a new

member need to register the location is needed to be check so we make sure that the new member is not in the restricted area, to satisfy this goal we created a trigger to check the region before insertion to the database as described in Fig. 13.

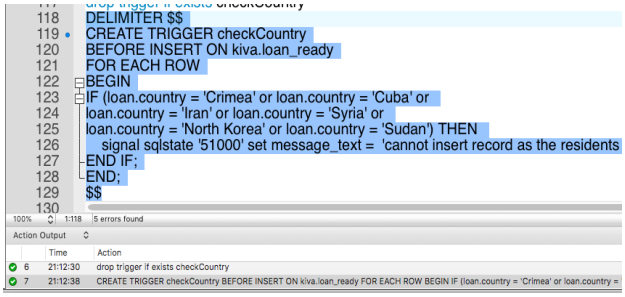


Fig. 13 checkCountry trigger

VI. Data Visualization

The analysis of the Kiva Crowdfunding Dataset gives an insight into the various factors that affect and influence the sustainability of crowdfunded projects. It also reveals the distribution of these projects in the world, gives information about the people involved in the creation of the projects and how the projects are distributed among the different sectors.

- Borrower Gender Distribution

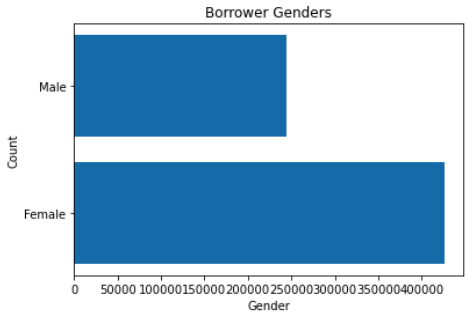


Fig. 14 Borrower gender Distribution

The above bar graph represents the genders of the loan borrowers with the number of women being approximately 40,000 and the men being close to 25,000. This implies that a far greater number of females seek loans from lenders as compared to males. One of the possible reasons for this is that the process of crowdfunding itself attracts more female lenders than the traditional venture capitalists.

- Repayment Method

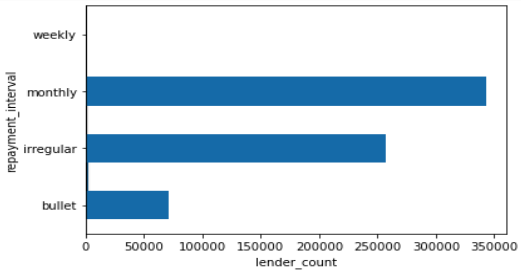


Fig. 15 Repayment Methods

The dataset reveals that there are majorly four types of repayment methods adopted by borrowers for the repayment of the loans, namely, ‘Bullet’, ’Irregular’, ’Monthly’ , and ‘Weekly’. A bullet method involves paying back the entire loan amount in one go at the time of maturity. The method involves gathering large sums and is hence not very popular. On the other hand the methods of monthly and irregular are much more popular whereas the weekly method is not used by anyone at all, perhaps because it involves payments to be made fairly quicker than the other methods.

- Project Sectors

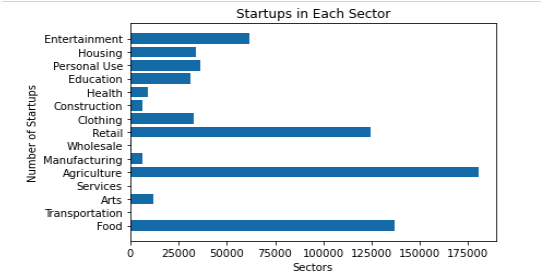


Fig. 16 Project Sectors

On further analysis of the data, it is observed that the projects are categorized into fifteen major sectors such as Food, Arts, Manufacturing, etc with Agriculture having the greatest number of projects. Agriculture involves a large capital investment which can be acquired through crowdfunding rather than by small investment companies. Retail and food sector closely trail agriculture in the number of projects.

- World Region

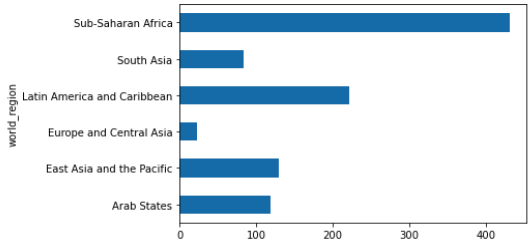


Fig. 17 World Region Distribution

The above bar graph represents the distribution of the projects in different regions of the world. The region of Sub-Saharan Africa sees the maximum number of crowdfunded projects. Because of the lack of alternate sources of financing, crowdfunding has seen a growing popularity here.

- Projects Funded Through Kiva

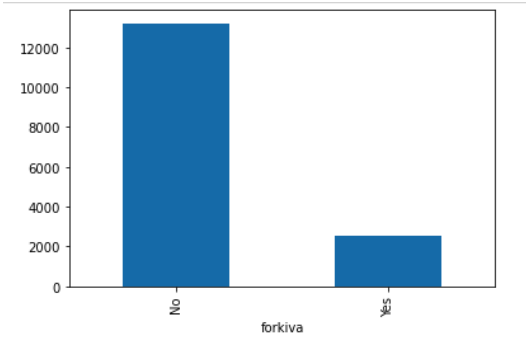


Fig. 18 Kiva Funded Projects

The dataset provided by Kiva contains projects funded not only through their own platform but also by numerous other sources. The graph gives a distribution of the projects that used Kivas' online platform to appeal to people to invest in their idea.

VII. Connectivity to AWS

Amazon Web Services (AWS) is a secure online cloud storage platform that provides various functionalities to businesses such as storage, content delivery, and compute power to help them scale and grow.

Amazon Relational Database Service (RDS) is a service that helps users create, operate on and scale databases on the cloud. RDS not only allows users to create isolated database environments but also provides the necessary security required for the data.

MySQL Workbench when connected to the AWS RDS provides a way for users to deploy their database on the Amazon cloud and manage the creation and manipulation of database entities through Python.

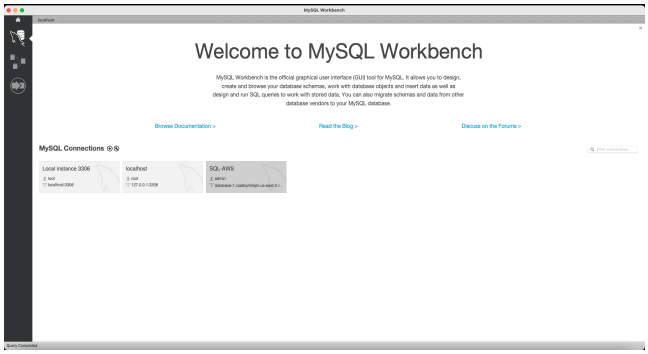


Fig. 19 MySQL-AWS Connection

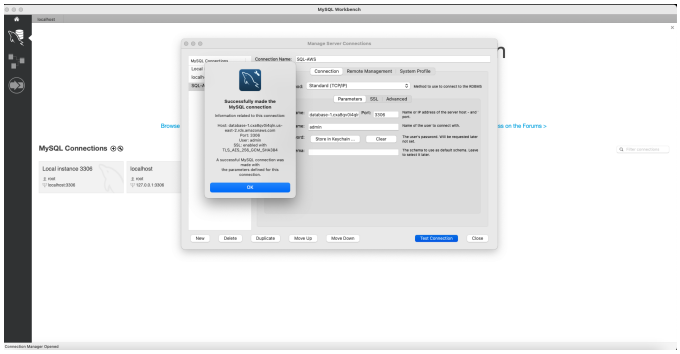


Fig. 20 MySQL-AWS Connection Successful

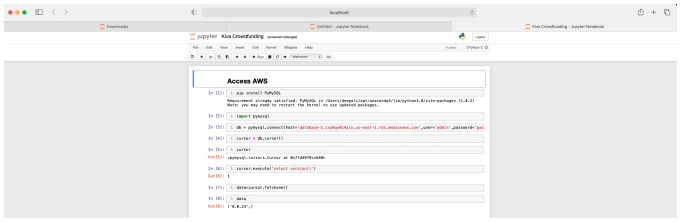


Fig. 21 Python-AWS Connection

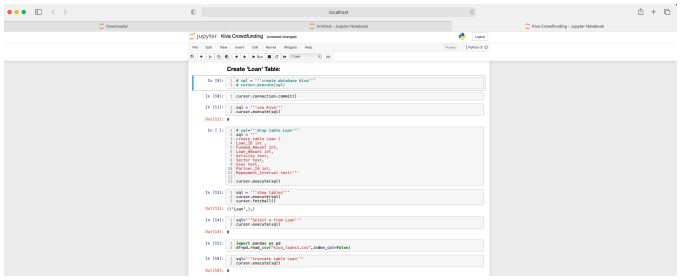


Fig. 22 Database Table Created Using PyMySQL

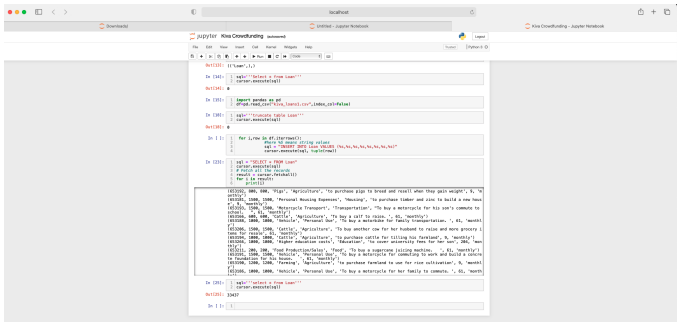


Fig. 23 Querying the data in Python

VIII. SQL Performance Measurement

The SQL performance was measured by analysing the time it took to execute various queries on the database. On average, DML commands require a much longer time to execute as compared to DDL commands. Queries that required data to be selected, updated, altered and joined had a greater execution time. The time taken to run and output the results of triggers and procedures also took more time when compared to the performance of the other data definition commands. The DML and DDL commands are having a very small execution times all the commands are taking less than a second to get executed for example fetching 33437 records into the lender details table was taking only 0.463 second even for pulling data from several tables its very fast for example for the last query performed in the sql query section which returned 50 records taking 0.044 sec and the execution plan is shown in Figure below.

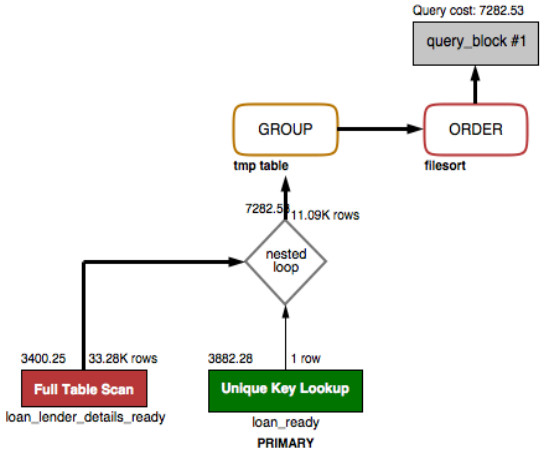


Fig. 24 MySQL Performance Analysis for a sample complex query on Kiva database

Acknowledgment

We would like to express our sincere gratitude to Professor Simon Shim for his guidance that aided in the completion of this project and helped us understand database concepts using practical examples. We would also like to thank Shiva Abhishek Varma Penmetsa and Rushikesh Jagtap for their continued help and support throughout the development of the project.

References

[1] S. Yu, Crowdfunding and regional entrepreneurial investment: an application of the CrowdBerkeley database, Research Policy, Volume 46, Issue 10, 2017, Pages 1723-1737, ISSN 0048-7333

[2] Burtch, G., Ghose, A., Wattal, S., 2014. Cultural Differences and geography as determi- nants of online prosocial lending. MIS Q. 38 (3), 773–794.

[3] Mollick, E., Nanda, R., 2015. Wisdom or Madness? Comparing Crowds with Expert Evaluation in Funding the Arts. Manage. Sci. 62 (6), 1533–1553.Mollick, E.R., 2014. The Dynamics of Crowdfunding: An Exploratory Study. J. Bus.Venturing 29 (January (1)), 1–16.

[4] Smith, Tim. “Crowdfunding.” Investopedia, Investopedia, 13Sept.2021,www.investopedia.com/terms/c/crowdfunding.