

Kiva Crowdfunding Database Analysis

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

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

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

Yasaman Emami
Department of Applied Data
Science
San Jose State University
San Jose, United States
yasaman.emami@sjtu.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 [1]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import warnings
import os
import warnings
warnings.filterwarnings('ignore')

In [1]: df = pd.read_csv('kiva_loans.csv')

In [1]: data = pandas.read_csv('kiva_loans.csv', encoding='utf-8', quotechar='"', delimiter=',')

In [1]: df.info()

In [1]: df.describe()

In [1]: null = df.isnull().sum().sort_values(ascending = False).reset_index()
null

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

In [1]: # This column null values are replaced with the mode
df['funded_time'].mode()

In [1]: df['borrower_genders'].mode()

In [1]: df['funded_time'].fillna(df['funded_time'].mode(), inplace = True)

In [1]: df['borrower_genders'].fillna(df['borrower_genders'].mode(), inplace = True)

In [1]: df.dropna(inplace = True)

In [1]: df.isna().sum()

In [1]: 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.).

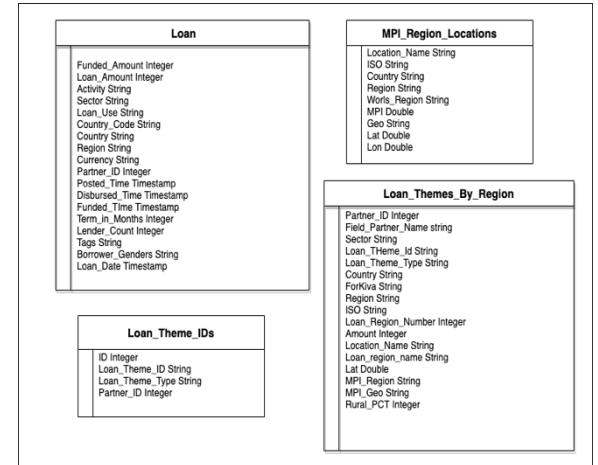


Fig. 2 Data Model

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.

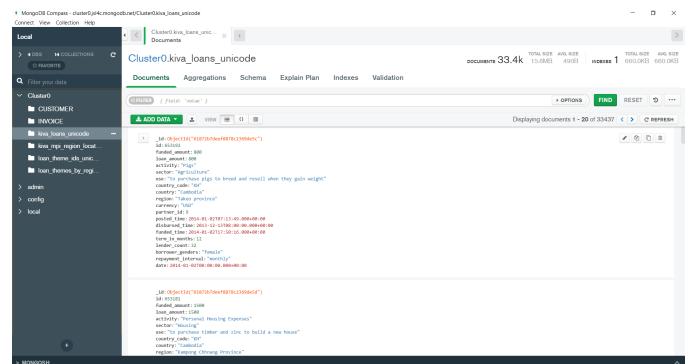


Fig. 3 Kiva_loans

Fig. 4 Kiva_mpi_region

Fig. 5 Kiva_loan_themes_by_region

Fig. 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-5-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": 653193,
  "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": "Chong Chhang Province",
  "currency": "USD",
  "partner_id": 9,
  "posted_time": ISODate("2014-01-02T06:55:54.000Z"),
  "disbursed_time": ISODate("2013-12-13T08:00:00.000Z"),
  "funded_time": ISODate("2014-01-02T21:27:48.000Z"),
  "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": "Personal Housing Expenses",
  "sector": "Housing",
  "use": "To purchase timber and zinc to build a new house",
  "country_code": "KH",
  "country": "Cambodia",
  "region": "Chong Chhang Province",
  "currency": "USD",
  "partner_id": 9,
  "posted_time": ISODate("2014-01-02T06:55:54.000Z"),
  "disbursed_time": ISODate("2013-12-13T08:00:00.000Z"),
  "funded_time": ISODate("2014-01-03T21:27:48.000Z"),
  "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": 64,
  "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

Fig. 6 kiva_mpi_region_locations

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.

```

{
  "_id": ObjectId("6189fd2d495fdaade073045c"),
  "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",
  "region": "Khsach Kandal district",
  "currency": "USD",
  "partner_id": 63,
  "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"),
  "term_in_months": 26,
  "lender_count": 53,
  "borrower_genders": "male",
  "repayment_interval": "monthly",
  "date": ISODate("2014-01-02T08:00:00.000Z")
},
{
  "_id": ObjectId("6189fd2d495fdaade073045c"),
  "id": 653193,
  "funded_amount": 700,
  "loan_amount": 700,
  "activity": "Vehicle",
  "sector": "Personal Use",
  "use": "To purchase a motorbike for his brother to drive to work.",
  "country_code": "KH",
  "region": "Kampong Speu district",
  "currency": "USD",
  "partner_id": 63,
  "posted_time": ISODate("2014-01-02T07:29:53.000Z"),
  "disbursed_time": ISODate("2013-12-16T08:00:00.000Z"),
  "funded_time": ISODate("2014-01-22T12:52:46.000Z"),
  "term_in_months": 14,
  "lender_count": 28,
  "borrower_genders": "male",
  "repayment_interval": "monthly",
  "date": ISODate("2014-01-02T08:00:00.000Z")
},
{
  "_id": ObjectId("6189fd2d495fdaade0730463"),
  "id": 653085,
  "funded_amount": 1100,
  "loan_amount": 1100,
  "activity": "Personal Housing Expenses",
  "sector": "Housing",
  "use": "to build a safe latrine and a water storage unit to improve her family's health",
  "country_code": "KH",
  "region": "Purset",
  "currency": "USD",
  "partner_id": 63,
  "posted_time": ISODate("2014-01-03T07:34:36.000Z"),
  "disbursed_time": ISODate("2013-12-02T08:00:00.000Z"),
  "funded_time": ISODate("2014-01-04T23:37:46.000Z"),
  "term_in_months": 26,
  "lender_count": 32,
  "borrower_genders": "female",
  "repayment_interval": "monthly",
  "date": ISODate("2014-01-03T08:00:00.000Z")
},
{
  "_id": ObjectId("6189fd2d495fdaade0730464"),
  "id": 653364,
  "funded_amount": 600,
  "loan_amount": 600,
  "activity": "Cattle",
  "sector": "Agriculture",
  "use": "To buy a calf to raise. .",
  "country_code": "KH",
  "region": "Kandal",
  "currency": "USD",
  "partner_id": 63,
  "posted_time": ISODate("2014-01-02T08:18:15.000Z"),
  "disbursed_time": ISODate("2013-12-04T08:00:00.000Z"),
  "funded_time": ISODate("2014-01-02T15:12:10.000Z"),
  "term_in_months": 22,
  "lender_count": 20,
  "borrower_genders": "female",
  "repayment_interval": "monthly",
  "date": ISODate("2014-01-02T08:00:00.000Z")
},
{
  "_id": ObjectId("6189fd2d495fdaade073046a"),
  "id": 653364,
  "funded_amount": 500,
}

```

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

```

{
  "_id": ObjectId("61878d1c51c09adb7856b987"),
  "id": 653192,
  "funded_amount": 800,
  "loan_amount": 800,
  "activity": "Pipe",
  "sector": "Agriculture",
  "use": "to purchase pigs to breed and resell when they gain weight",
  "country_code": "KH",
  "region": "Takeo province",
  "currency": "USD",
  "partner_id": 9,
  "posted_time": ISODate("2014-01-02T07:13:49.000Z"),
  "disbursed_time": ISODate("2013-12-19T08:00:00.000Z"),
  "funded_time": ISODate("2014-01-02T15:12:10.000Z"),
  "term_in_months": 12,
  "lender_count": 32,
  "borrower_genders": "female",
  "repayment_interval": "monthly",
  "date": ISODate("2014-01-02T08:00:00.000Z")
},
{
  "_id": ObjectId("61878d1c51c09adb7856b98a"),
  "id": 653189,
  "funded_amount": 600,
  "loan_amount": 600,
  "activity": "Cattle",
  "sector": "Agriculture",
  "use": "To buy a calf to raise. .",
  "country_code": "KH",
  "region": "Cambodia",
  "currency": "Phnom Penh",
  "partner_id": 13,
  "posted_time": ISODate("2014-01-02T08:18:15.000Z"),
  "disbursed_time": ISODate("2013-12-04T08:00:00.000Z"),
  "funded_time": ISODate("2014-01-02T15:12:10.000Z"),
  "term_in_months": 22,
  "lender_count": 22,
  "borrower_genders": "female",
  "repayment_interval": "monthly",
  "date": ISODate("2014-01-02T08:00:00.000Z")
},
{
  "_id": ObjectId("61878d1c51c09adb7856b98b"),
  "id": 653189,
  "funded_amount": 1000,
  "loan_amount": 1000,
  "activity": "Vehicle",
  "sector": "Personal Use",
  "use": "To buy a motorbike for family transportation. .",
  "country_code": "KH",
  "region": "Cambodia",
  "currency": "USD",
  "partner_id": 13,
  "posted_time": ISODate("2014-01-02T07:05:02.000Z"),
  "disbursed_time": ISODate("2013-12-13T08:00:00.000Z"),
  "funded_time": ISODate("2014-01-10T03:22:29.000Z"),
  "term_in_months": 22,
  "lender_count": 22,
  "borrower_genders": "female",
  "repayment_interval": "monthly",
  "date": ISODate("2014-01-02T08:00:00.000Z")
}

```

Fig. 10 loan_amount between 500\$,1500\$

```
Atlas atlas-5alle7-shard-0 [primary] kiva> db.all_kiva.aggregate([{"$group": {"_id": "$loan_amount", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}])
[{"_id": null, "count": 771880},
 {"_id": 125, "count": 3610},
 {"_id": 200, "count": 3590},
 {"_id": 275, "count": 3450},
 {"_id": 350, "count": 3370},
 {"_id": 425, "count": 2073},
 {"_id": 500, "count": 1886},
 {"_id": 575, "count": 1530},
 {"_id": 650, "count": 985},
 {"_id": 725, "count": 799},
 {"_id": 800, "count": 527},
 {"_id": 875, "count": 525},
 {"_id": 950, "count": 519},
 {"_id": 1025, "count": 510},
 {"_id": 1100, "count": 504},
 {"_id": 1175, "count": 461},
 {"_id": 1250, "count": 456},
 {"_id": 1325, "count": 376},
 {"_id": 1400, "count": 342},
 {"_id": 1475, "count": 340},
 {"_id": 1550, "count": 325},
 {"_id": 1625, "count": 277},
 {"_id": 1700, "count": 272},
 {"_id": 1775, "count": 267},
 {"_id": 1850, "count": 241},
 {"_id": 1925, "count": 224},
 {"_id": 2000, "count": 215},
 {"_id": 2075, "count": 215},
 {"_id": 2150, "count": 212},
 {"_id": 2225, "count": 199},
 {"_id": 2300, "count": 191},
 {"_id": 2375, "count": 181},
 {"_id": 2450, "count": 180},
 {"_id": 2525, "count": 167},
 {"_id": 2600, "count": 162},
 {"_id": 2675, "count": 149},
 {"_id": 2750, "count": 148},
 {"_id": 2825, "count": 148}]
Type "it" for more
Atlas atlas-5alle7-shard-0 [primary] kiva> it
[{"_id": 1200, "count": 361},
 {"_id": 886, "count": 349},
 {"_id": 1050, "count": 340},
 {"_id": 300, "count": 385},
 {"_id": 1625, "count": 277},
 {"_id": 1475, "count": 272},
 {"_id": 325, "count": 267},
 {"_id": 875, "count": 241},
 {"_id": 1925, "count": 224},
 {"_id": 389, "count": 215},
 {"_id": 980, "count": 212},
 {"_id": 1886, "count": 199},
 {"_id": 1225, "count": 191},
 {"_id": 775, "count": 181},
 {"_id": 886, "count": 180},
 {"_id": 1650, "count": 167},
 {"_id": 2000, "count": 162},
 {"_id": 875, "count": 149},
 {"_id": 1475, "count": 148},
 {"_id": 1125, "count": 148}]
Type "it" for more
Atlas atlas-5alle7-shard-0 [primary] kiva> it
[{"_id": 725, "count": 145},
 {"_id": 975, "count": 140},
 {"_id": 425, "count": 135},
 {"_id": 1375, "count": 135},
 {"_id": 1350, "count": 116},
 {"_id": 980, "count": 115},
 {"_id": 1275, "count": 108},
 {"_id": 1025, "count": 105},
 {"_id": 1400, "count": 99},
 {"_id": 1375, "count": 95},
 {"_id": 1475, "count": 88},
 {"_id": 1600, "count": 85},
 {"_id": 1675, "count": 82},
 {"_id": 1800, "count": 77},
 {"_id": 1888, "count": 77},
 {"_id": 1150, "count": 72},
 {"_id": 1475, "count": 72},
 {"_id": 1625, "count": 69},
 {"_id": 925, "count": 67},
 {"_id": 1700, "count": 65}]
Type "it" for more
Atlas atlas-5alle7-shard-0 [primary] kiva> ||
```

Fig.11 Count of Each Loan

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

```
[{"_id": ObjectID("458784655059949ab7807504"), "sector": "General Financial Inclusion", "country": "Madagascar", "region": "Tsihoney", "partner_id": 305, "field_partner_name": "Investisseurs & Partenaires", "loan_theme_id": "105600000002XK", "loan_theme_type": "Small Enterprise", "risk": "Low", "forisks": "No", "iso": "MDG", "loan.region_number": "1.8", "amount": 1000000, "location_name": "Tsihoney, Madagascar", "lat: Decimal128="40.18644679", "lon: Decimal128="40.4863993"}, {"_id: ObjectID("458784655059949ab7807505k17"), "sector": "General Financial Inclusion", "country": "Madagascar", "region": "Antananarivo", "partner_id": 305, "field_partner_name": "Vahatra", "loan_theme_id": "105600000002XK", "loan_theme_type": "Extreme Poverty", "risk": "High", "forisks": "No", "iso": "MDG", "loan.region_number": "76.8", "amount": 1000000, "location_name": "Antananarivo, Madagascar", "lat: Decimal128="19.78046481", "lon: Decimal128="47.5152457"}, {"_id: ObjectID("458784655059949ab7807505k27"), "sector": "General Financial Inclusion", "country": "Madagascar", "region": "Antananarivo", "partner_id": 305, "field_partner_name": "Vahatra", "loan_theme_id": "105600000002XK", "loan_theme_type": "Extreme Poverty", "risk": "High", "forisks": "No", "iso": "MDG", "loan.region_number": "76.8", "amount": 1000000, "location_name": "Antananarivo, Madagascar", "lat: Decimal128="19.78046481", "lon: Decimal128="47.5152457"}, {"_id: ObjectID("458784655059949ab7807505k37"), "sector": "General Financial Inclusion", "country": "Madagascar", "region": "Betafika", "partner_id": 305, "field_partner_name": "Vahatra", "loan_theme_id": "105600000002XK", "loan_theme_type": "Extreme Poverty", "risk": "High", "forisks": "No", "iso": "MDG", "loan.region_number": "988.8", "amount": 300000, "location_name": "Betafika, Madagascar", "lat: Decimal128="19.780377", "lon: Decimal128="47.051162"}, {"_id: ObjectID("458784655059949ab7807505k47"), "sector": "General Financial Inclusion", "country": "Madagascar", "region": "Betafika", "partner_id": 305, "field_partner_name": "Vahatra", "loan_theme_id": "105600000002XK", "loan_theme_type": "Extreme Poverty", "risk": "High", "forisks": "No", "iso": "MDG", "loan.region_number": "63.8", "amount": 1000000, "location_name": "Betafika, Madagascar", "lat: Decimal128="19.780377", "lon: Decimal128="46.856752"}, {"_id: ObjectID("458784655059949ab7807505k47"), "sector": "General Financial Inclusion", "country": "Madagascar", "region": "Betafika", "partner_id": 305, "field_partner_name": "Vahatra", "loan_theme_id": "105600000002XK", "loan_theme_type": "Extreme Poverty", "risk": "High", "forisks": "No", "iso": "MDG", "loan.region_number": "203.8", "amount": 1000000, "location_name": "Betafika, Madagascar", "lat: Decimal128="-29.062611", "lon: Decimal128="46.856752"}, {"_id: ObjectID("458784655059949ab7807505k57"), "sector": "General Financial Inclusion", "country": "Madagascar", "region": "Talata", "partner_id": 305, "field_partner_name": "Vahatra",
```

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 NoSQL kiva database.

```

Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.aggregate([{"$match: {term_in_months: $gt:10} }, {"group: {_id:"$activity", cnt:(sum)} } ], {sort:[{"cnt":1} ],{match:{sum:{gt:10}} } })
{
  "_id": "Health", "cnt": 13 },
  "_id": "Consumer Goods", "cnt": 11 },
  "_id": "Fuel/Firewood", "cnt": 11 },
  "_id": "Food/Drinking Water", "cnt": 13 },
  "_id": "Blacksmithing", "cnt": 1 },
  {"_id": "Well digging", "cnt": 16 },
  {"_id": "Swing", "cnt": 16 },
  {"_id": "Transportation", "cnt": 18 },
  {"_id": "Cereals", "cnt": 18 },
  {"_id": "Beauts Salons", "cnt": 20 },
  {"_id": "Education", "cnt": "School costs", "cnt": 21 },
  {"_id": "Crafts", "cnt": 21 },
  {"_id": "Taxi", "cnt": 26 },
  {"_id": "Construction", "cnt": 26 },
  {"_id": "Food Stall", "cnt": 27 },
  {"_id": "Clothing Sales", "cnt": 29 },
  {"_id": "Food Production", "cnt": 29 },
  {"_id": "Transportation", "cnt": 29 },
  {"_id": "Transportation", "cnt": 32 },
  {"_id": "Property", "cnt": 36 },
  {"_id": "Wedding Expenses", "cnt": 36 }
}
Type "it" for more
Atlas atlas-Salle7-shard-0 [primary] kiva> it
{
  "_id": "Land Rental", "cnt": 38 },
  {"_id": "Services", "cnt": 39 },
  {"_id": "Transportation", "cnt": 49 },
  {"_id": "Construction Supplies", "cnt": 49 },
  {"_id": "Beverages", "cnt": 73 },
  {"_id": "Transportation", "cnt": 72 },
  {"_id": "Retail", "cnt": 74 },
  {"_id": "Livestock", "cnt": 78 },
  {"_id": "Construction", "cnt": 77 },
  {"_id": "Weaving", "cnt": 81 },
  {"_id": "Poultry", "cnt": 91 },
  {"_id": "Food Production/Sales", "cnt": 92 },
  {"_id": "Transportation", "cnt": 92 },
  {"_id": "Personal Expenses", "cnt": 99 },
  {"_id": "General Store", "cnt": 114 },
  {"_id": "Transportation", "cnt": 124 },
  {"_id": "Animals Sales", "cnt": 136 },
  {"_id": "Motorcycle Transport", "cnt": 146 },
  {"_id": "Transportation", "cnt": 149 },
  {"_id": "Farm Supplies", "cnt": 269 }
}
Type "it" for more
Atlas atlas-Salle7-shard-0 [primary] kiva> it
{
  "_id": "Grotts", "cnt": 384 },
  {"_id": "Grocery Store", "cnt": 340 },
  {"_id": "Agriculture", "cnt": 386 },
  {"_id": "Pigs", "cnt": 387 },
  {"_id": "Transportation", "cnt": 446 },
  {"_id": "Home Energy", "cnt": 943 },
  {"_id": "Home Appliances", "cnt": 1812 },
  {"_id": "Transportation", "cnt": 1846 },
  {"_id": "Personal Housing Expenses", "cnt": 1936 },
  {"_id": "Farming", "cnt": 7688 }
}
Atlas atlas-Salle7-shard-0 [primary] kiva> 

```

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.

```

Atlas atlas-Salle7-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.16 Update documents set a value on condition

```

Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.deleteMany({loan_theme_type:"Unique"})
{ acknowledged: true, deletedCount: 384876 }
Atlas atlas-Salle7-shard-0 [primary] kiva> 

```

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

	select * from loan_ready	join_theme_ids	join_theme_ids.ready_loan_id	join_loan_details	join_loan_details.ready_loan_id	join_partner	join_partner.ready_loan_id	join_themes	join_themes_by_region	join_themes_by_region.ready_region_id	join_themes_by_region.theme_ids	join_themes_by_region.theme_ids.ready_loan_theme_id
459	select * from loan_ready											
460	join_theme_ids											
461	join_theme_ids.ready_loan_id											
462	join_loan_details											
463	join_loan_details.ready_loan_id											
464	join_partner											
465	join_partner.ready_loan_id											
466	join_themes											
467	join_themes_by_region											
468	join_themes_by_region.ready_region_id											
469	join_themes_by_region.theme_ids											
470	join_themes_by_region.theme_ids.ready_loan_theme_id											

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

	Action	Time	Response	Duration
31 00:00:00	start transaction	2014-01-02T08:00:00Z	0 rows affected	0 seconds
32 00:00:00	insert into loan (loan_id, posted_time, disbursed_time, funded_time, term_in_months, lender_count, borrower_genders, repayment_interval, date) values ('6189fd2d495fdaade073046a', '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z', 26, 1, 'male', 'monthly', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
33 00:00:00	insert into loan_theme (loan_theme_id, loan_id, theme_ids, ready_loan_theme_id) values ('65316', '6189fd2d495fdaade073046a', '100', '100')	1 row inserted	0 seconds	
34 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65320', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
35 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65321', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
36 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65322', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
37 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65323', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
38 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65324', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
39 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65325', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
40 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65326', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
41 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65327', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
42 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65328', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
43 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65329', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
44 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65330', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
45 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65331', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
46 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65332', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
47 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65333', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
48 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65334', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
49 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65335', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
50 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65336', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
51 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65337', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
52 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65338', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
53 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65339', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
54 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65340', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
55 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65341', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
56 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65342', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
57 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65343', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
58 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65344', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
59 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65345', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
60 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65346', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
61 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65347', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
62 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65348', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
63 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65349', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
64 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65350', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
65 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65351', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
66 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65352', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
67 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65353', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
68 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65354', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
69 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65355', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
70 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65356', '6189fd2d495fdaade073046a', '63', '100', '100', 26, '2014-01-02T08:00:00Z', '2013-12-02T08:00:00Z', '2014-01-02T08:00:00Z')	1 row inserted	0 seconds	
71 00:00:00	insert into loan_loan_detail (loan_loan_detail_id, loan_id, partner_id, loan_amount, funded_amount, term_in_months, posted_time, disbursed_time, date) values ('65357', '6189fd2d495fdaade073046a',			

```

[{"t": "atlas-5ale7-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("69945B689132448973B")
    }
  },
  operationTime: Timestamp({ t: 1638765534, i: 3 })
}
Atlas atlas-5ale7-shard-0 [primary] kiva> 

```

Fig. 21 Performance of retrieving all data on MongoDB

Read - Operation 1		
DB	Command	Execution Time(ms)
SQL	<pre> 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_the_ me_id = loan_theme_ids_ready.loan_theme_id; </pre>	1837
NoSQL	db.all_kiva.find()	105

Table 1. Performance and Commands for the first read operation

b) Read operation - 2

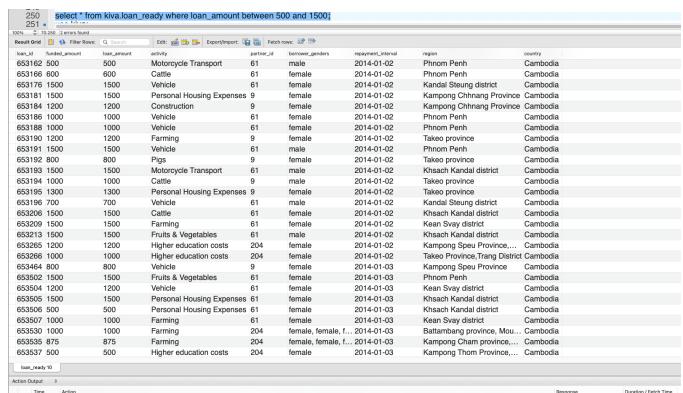


Fig. 22 loan_amount between 500\$,1500\$ on MySQL



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

```

● ● ● yasamanemami - mongosh mongodb://data225.5q6p4.mongodb.net/myFirstDatabase -- myFirstDatabase TMPDIR=/var/folders/zj...[redacted]
] Type "it" for more
Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.find({loan_amount:{'$gt':500, '$lt':1500}}).explain("executionStats")
{
  "queryPlanner": {
    "plannerVersion": 1,
    "namespace": "kiva.all_kiva",
    "indexFilterSet": false,
    "parsedQuery": {
      "loan_amount": {
        "$gt": 500,
        "$lt": 1500
      }
    },
    "winningPlan": {
      "stage": "COLLSCAN",
      "filter": {
        "$and": [
          {"loan_amount": {"$lt": 1500}},
          {"loan_amount": {"$gt": 500}}
        ]
      },
      "direction": "forward"
    },
    "rejectedPlans": []
  },
  "executionStats": {
    "executionSuccess": true,
    "nReturned": 10002,
    "executionTimeMillisEstimate": 317,
    "totalKeyExamined: 0,
    "totalDocExamined: 867542,
    "executionStage": "COLLSCAN",
    "filter": {
      "$and": [
        {"loan_amount": {"$lt": 1500}},
        {"loan_amount": {"$gt": 500}}
      ]
    },
    "nReturned: 10003,
    "executionTimeMillisEstimate": 38,
    "works: 807544,
    "advanced: 807542,
    "needValid: 0,
    "saveState: 0,
    "restoreState: 807,
    "isEOF: 1,
    "direction: "forward",
    "docsExamined: 867542
  }
}

serverInfo: {
  host: "data225-shard-00-01.5q6p4.mongodb.net",
  port: 27017,
  version: "4.4.10",
  gitVersion: "58971da1ef93435a9fe2b7a7f088a81713def6e88c"
}
ok: 1
{
  "clusterTime": {
    "clusterTime": timestamp( t : 1636314302, i: 1 ),
    "signature": {
      "hash": Binary(Buffer.from("fa2d8dfaa587be994bacf8dfa1f6893c867759c03", "hex"), 0),
      "keyId": Long("6994586891324489738")
    }
  },
  "operationTime": timestamp( t : 1636314302, i: 1 )
}
Atlas atlas-Salle7-shard-0 [primary] kiva> ||
```

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

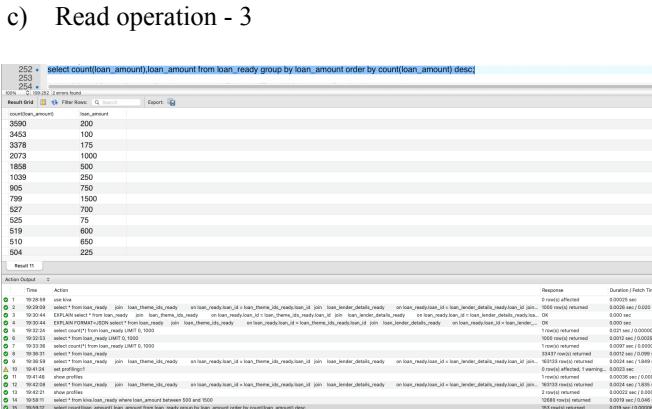


Fig. 25 Count of each loan

```

Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.aggregate([ { $group: { _id: "$loan_amount", count:{$sum:1} } }, { $sort:{"count":1} } ])
{
  "_id": 500,
  "count": 10003
}
{
  "_id": 525,
  "count": 349
}
{
  "_id": 550,
  "count": 338
}
{
  "_id": 575,
  "count": 327
}
{
  "_id": 600,
  "count": 319
}
{
  "_id": 625,
  "count": 318
}
{
  "_id": 650,
  "count": 318
}
{
  "_id": 675,
  "count": 317
}
{
  "_id": 700,
  "count": 316
}
{
  "_id": 725,
  "count": 315
}
{
  "_id": 750,
  "count": 314
}
{
  "_id": 775,
  "count": 313
}
{
  "_id": 800,
  "count": 312
}
{
  "_id": 825,
  "count": 311
}
{
  "_id": 850,
  "count": 310
}
{
  "_id": 875,
  "count": 309
}
{
  "_id": 900,
  "count": 308
}
{
  "_id": 925,
  "count": 307
}
{
  "_id": 950,
  "count": 306
}
{
  "_id": 975,
  "count": 305
}
{
  "_id": 1000,
  "count": 304
}
{
  "_id": 1025,
  "count": 303
}
{
  "_id": 1050,
  "count": 302
}
{
  "_id": 1075,
  "count": 301
}
{
  "_id": 1100,
  "count": 300
}
{
  "_id": 1125,
  "count": 299
}
{
  "_id": 1150,
  "count": 298
}
{
  "_id": 1175,
  "count": 297
}
{
  "_id": 1200,
  "count": 296
}
{
  "_id": 1225,
  "count": 295
}
{
  "_id": 1250,
  "count": 294
}
{
  "_id": 1275,
  "count": 293
}
{
  "_id": 1300,
  "count": 292
}
{
  "_id": 1325,
  "count": 291
}
{
  "_id": 1350,
  "count": 290
}
{
  "_id": 1375,
  "count": 289
}
{
  "_id": 1400,
  "count": 288
}
{
  "_id": 1425,
  "count": 287
}
{
  "_id": 1450,
  "count": 286
}
{
  "_id": 1475,
  "count": 285
}
{
  "_id": 1500,
  "count": 284
}
{
  "_id": 1525,
  "count": 283
}
}

Type "it" for more
Atlas atlas-Salle7-shard-0 [primary] kiva> it
{
  "_id": 1200,
  "count": 36155
}
{
  "_id": 1225,
  "count": 36055
}
{
  "_id": 1250,
  "count": 35955
}
{
  "_id": 1275,
  "count": 35855
}
{
  "_id": 1300,
  "count": 35755
}
{
  "_id": 1325,
  "count": 35655
}
{
  "_id": 1350,
  "count": 35555
}
{
  "_id": 1375,
  "count": 35455
}
{
  "_id": 1400,
  "count": 35355
}
{
  "_id": 1425,
  "count": 35255
}
{
  "_id": 1450,
  "count": 35155
}
{
  "_id": 1475,
  "count": 35055
}
{
  "_id": 1500,
  "count": 34955
}
{
  "_id": 1525,
  "count": 34855
}
{
  "_id": 1550,
  "count": 34755
}
{
  "_id": 1575,
  "count": 34655
}
{
  "_id": 1600,
  "count": 34555
}
{
  "_id": 1625,
  "count": 34455
}
{
  "_id": 1650,
  "count": 34355
}
{
  "_id": 1675,
  "count": 34255
}
{
  "_id": 1700,
  "count": 34155
}
{
  "_id": 1725,
  "count": 34055
}
{
  "_id": 1750,
  "count": 33955
}
{
  "_id": 1775,
  "count": 33855
}
{
  "_id": 1800,
  "count": 33755
}
{
  "_id": 1825,
  "count": 33655
}
{
  "_id": 1850,
  "count": 33555
}
{
  "_id": 1875,
  "count": 33455
}
{
  "_id": 1900,
  "count": 33355
}
{
  "_id": 1925,
  "count": 33255
}
{
  "_id": 1950,
  "count": 33155
}
{
  "_id": 1975,
  "count": 33055
}
{
  "_id": 2000,
  "count": 32955
}
{
  "_id": 2025,
  "count": 32855
}
{
  "_id": 2050,
  "count": 32755
}
{
  "_id": 2075,
  "count": 32655
}
{
  "_id": 2100,
  "count": 32555
}
{
  "_id": 2125,
  "count": 32455
}
{
  "_id": 2150,
  "count": 32355
}
{
  "_id": 2175,
  "count": 32255
}
{
  "_id": 2200,
  "count": 32155
}
{
  "_id": 2225,
  "count": 32055
}
{
  "_id": 2250,
  "count": 31955
}
{
  "_id": 2275,
  "count": 31855
}
{
  "_id": 2300,
  "count": 31755
}
{
  "_id": 2325,
  "count": 31655
}
{
  "_id": 2350,
  "count": 31555
}
{
  "_id": 2375,
  "count": 31455
}
{
  "_id": 2400,
  "count": 31355
}
{
  "_id": 2425,
  "count": 31255
}
{
  "_id": 2450,
  "count": 31155
}
{
  "_id": 2475,
  "count": 31055
}
{
  "_id": 2500,
  "count": 30955
}
{
  "_id": 2525,
  "count": 30855
}
{
  "_id": 2550,
  "count": 30755
}
{
  "_id": 2575,
  "count": 30655
}
{
  "_id": 2600,
  "count": 30555
}
{
  "_id": 2625,
  "count": 30455
}
{
  "_id": 2650,
  "count": 30355
}
{
  "_id": 2675,
  "count": 30255
}
{
  "_id": 2700,
  "count": 30155
}
{
  "_id": 2725,
  "count": 30055
}
{
  "_id": 2750,
  "count": 29955
}
{
  "_id": 2775,
  "count": 29855
}
{
  "_id": 2800,
  "count": 29755
}
{
  "_id": 2825,
  "count": 29655
}
{
  "_id": 2850,
  "count": 29555
}
{
  "_id": 2875,
  "count": 29455
}
{
  "_id": 2900,
  "count": 29355
}
{
  "_id": 2925,
  "count": 29255
}
{
  "_id": 2950,
  "count": 29155
}
{
  "_id": 2975,
  "count": 29055
}
{
  "_id": 3000,
  "count": 28955
}
{
  "_id": 3025,
  "count": 28855
}
{
  "_id": 3050,
  "count": 28755
}
{
  "_id": 3075,
  "count": 28655
}
{
  "_id": 3100,
  "count": 28555
}
{
  "_id": 3125,
  "count": 28455
}
{
  "_id": 3150,
  "count": 28355
}
{
  "_id": 3175,
  "count": 28255
}
{
  "_id": 3200,
  "count": 28155
}
{
  "_id": 3225,
  "count": 28055
}
{
  "_id": 3250,
  "count": 27955
}
{
  "_id": 3275,
  "count": 27855
}
{
  "_id": 3300,
  "count": 27755
}
{
  "_id": 3325,
  "count": 27655
}
{
  "_id": 3350,
  "count": 27555
}
{
  "_id": 3375,
  "count": 27455
}
{
  "_id": 3400,
  "count": 27355
}
{
  "_id": 3425,
  "count": 27255
}
{
  "_id": 3450,
  "count": 27155
}
{
  "_id": 3475,
  "count": 27055
}
{
  "_id": 3500,
  "count": 26955
}
{
  "_id": 3525,
  "count": 26855
}
{
  "_id": 3550,
  "count": 26755
}
{
  "_id": 3575,
  "count": 26655
}
{
  "_id": 3600,
  "count": 26555
}
{
  "_id": 3625,
  "count": 26455
}
{
  "_id": 3650,
  "count": 26355
}
{
  "_id": 3675,
  "count": 26255
}
{
  "_id": 3700,
  "count": 26155
}
{
  "_id": 3725,
  "count": 26055
}
{
  "_id": 3750,
  "count": 25955
}
{
  "_id": 3775,
  "count": 25855
}
{
  "_id": 3800,
  "count": 25755
}
{
  "_id": 3825,
  "count": 25655
}
{
  "_id": 3850,
  "count": 25555
}
{
  "_id": 3875,
  "count": 25455
}
{
  "_id": 3900,
  "count": 25355
}
{
  "_id": 3925,
  "count": 25255
}
{
  "_id": 3950,
  "count": 25155
}
{
  "_id": 3975,
  "count": 25055
}
{
  "_id": 4000,
  "count": 24955
}
{
  "_id": 4025,
  "count": 24855
}
{
  "_id": 4050,
  "count": 24755
}
{
  "_id": 4075,
  "count": 24655
}
{
  "_id": 4100,
  "count": 24555
}
{
  "_id": 4125,
  "count": 24455
}
{
  "_id": 4150,
  "count": 24355
}
{
  "_id": 4175,
  "count": 24255
}
{
  "_id": 4200,
  "count": 24155
}
{
  "_id": 4225,
  "count": 24055
}
{
  "_id": 4250,
  "count": 23955
}
{
  "_id": 4275,
  "count": 23855
}
{
  "_id": 4300,
  "count": 23755
}
{
  "_id": 4325,
  "count": 23655
}
{
  "_id": 4350,
  "count": 23555
}
{
  "_id": 4375,
  "count": 23455
}
{
  "_id": 4400,
  "count": 23355
}
{
  "_id": 4425,
  "count": 23255
}
{
  "_id": 4450,
  "count": 23155
}
{
  "_id": 4475,
  "count": 23055
}
{
  "_id": 4500,
  "count": 22955
}
{
  "_id": 4525,
  "count": 22855
}
{
  "_id": 4550,
  "count": 22755
}
{
  "_id": 4575,
  "count": 22655
}
{
  "_id": 4600,
  "count": 22555
}
{
  "_id": 4625,
  "count": 22455
}
{
  "_id": 4650,
  "count": 22355
}
{
  "_id": 4675,
  "count": 22255
}
{
  "_id": 4700,
  "count": 22155
}
{
  "_id": 4725,
  "count": 22055
}
{
  "_id": 4750,
  "count": 21955
}
{
  "_id": 4775,
  "count": 21855
}
{
  "_id": 4800,
  "count": 21755
}
{
  "_id": 4825,
  "count": 21655
}
{
  "_id": 4850,
  "count": 21555
}
{
  "_id": 4875,
  "count": 21455
}
{
  "_id": 4900,
  "count": 21355
}
{
  "_id": 4925,
  "count": 21255
}
{
  "_id": 4950,
  "count": 21155
}
{
  "_id": 4975,
  "count": 21055
}
{
  "_id": 5000,
  "count": 20955
}
{
  "_id": 5025,
  "count": 20855
}
{
  "_id": 5050,
  "count": 20755
}
{
  "_id": 5075,
  "count": 20655
}
{
  "_id": 5100,
  "count": 20555
}
{
  "_id": 5125,
  "count": 20455
}
{
  "_id": 5150,
  "count": 20355
}
{
  "_id": 5175,
  "count": 20255
}
{
  "_id": 5200,
  "count": 20155
}
{
  "_id": 5225,
  "count": 20055
}
{
  "_id": 5250,
  "count": 19955
}
{
  "_id": 5275,
  "count": 19855
}
{
  "_id": 5300,
  "count": 19755
}
{
  "_id": 5325,
  "count": 19655
}
{
  "_id": 5350,
  "count": 19555
}
{
  "_id": 5375,
  "count": 19455
}
{
  "_id": 5400,
  "count": 19355
}
{
  "_id": 5425,
  "count": 19255
}
{
  "_id": 5450,
  "count": 19155
}
{
  "_id": 5475,
  "count": 19055
}
{
  "_id": 5500,
  "count": 18955
}
{
  "_id": 5525,
  "count": 18855
}
{
  "_id": 5550,
  "count": 18755
}
{
  "_id": 5575,
  "count": 18655
}
{
  "_id": 5600,
  "count": 18555
}
{
  "_id": 5625,
  "count": 18455
}
{
  "_id": 5650,
  "count": 18355
}
{
  "_id": 5675,
  "count": 18255
}
{
  "_id": 5700,
  "count": 18155
}
{
  "_id": 5725,
  "count": 18055
}
{
  "_id": 5750,
  "count": 17955
}
{
  "_id": 5775,
  "count": 17855
}
{
  "_id": 5800,
  "count": 17755
}
{
  "_id": 5825,
  "count": 17655
}
{
  "_id": 5850,
  "count": 17555
}
{
  "_id": 5875,
  "count": 17455
}
{
  "_id": 5900,
  "count": 17355
}
{
  "_id": 5925,
  "count": 17255
}
{
  "_id": 5950,
  "count": 17155
}
{
  "_id": 5975,
  "count": 17055
}
{
  "_id": 6000,
  "count": 16955
}
{
  "_id": 6025,
  "count": 16855
}
{
  "_id": 6050,
  "count": 16755
}
{
  "_id": 6075,
  "count": 16655
}
{
  "_id": 6100,
  "count": 16555
}
{
  "_id": 6125,
  "count": 16455
}
{
  "_id": 6150,
  "count": 16355
}
{
  "_id": 6175,
  "count": 16255
}
{
  "_id": 6200,
  "count": 16155
}
{
  "_id": 6225,
  "count": 16055
}
{
  "_id": 6250,
  "count": 15955
}
{
  "_id": 6275,
  "count": 15855
}
{
  "_id": 6300,
  "count": 15755
}
{
  "_id": 6325,
  "count": 15655
}
{
  "_id": 6350,
  "count": 15555
}
{
  "_id": 6375,
  "count": 15455
}
{
  "_id": 6400,
  "count": 15355
}
{
  "_id": 6425,
  "count": 15255
}
{
  "_id": 6450,
  "count": 15155
}
{
  "_id": 6475,
  "count": 15055
}
{
  "_id": 6500,
  "count": 14955
}
{
  "_id": 6525,
  "count": 14855
}
{
  "_id": 6550,
  "count": 14755
}
{
  "_id": 6575,
  "count": 14655
}
{
  "_id": 6600,
  "count": 14555
}
{
  "_id": 6625,
  "count": 14455
}
{
  "_id": 6650,
  "count": 14355
}
{
  "_id": 6675,
  "count": 14255
}
{
  "_id": 6700,
  "count": 14155
}
{
  "_id": 6725,
  "count": 14055
}
{
  "_id": 6750,
  "count": 13955
}
{
  "_id": 6775,
  "count": 13855
}
{
  "_id": 6800,
  "count": 13755
}
{
  "_id": 6825,
  "count": 13655
}
{
  "_id": 6850,
  "count": 13555
}
{
  "_id": 6875,
  "count": 13455
}
{
  "_id": 6900,
  "count": 13355
}
{
  "_id": 6925,
  "count": 13255
}
{
  "_id": 6950,
  "count": 13155
}
{
  "_id": 6975,
  "count": 13055
}
{
  "_id": 7000,
  "count": 12955
}
{
  "_id": 7025,
  "count": 12855
}
{
  "_id": 7050,
  "count": 12755
}
{
  "_id": 7075,
  "count": 12655
}
{
  "_id": 7100,
  "count": 12555
}
{
  "_id": 7125,
  "count": 12455
}
{
  "_id": 7150,
  "count": 12355
}
{
  "_id": 7175,
  "count": 12255
}
{
  "_id": 7200,
  "count": 12155
}
{
  "_id": 7225,
  "count": 12055
}
{
  "_id": 7250,
  "count": 11955
}
{
  "_id": 7275,
  "count": 11855
}
{
  "_id": 7300,
  "count": 11755
}
{
  "_id": 7325,
  "count": 11655
}
{
  "_id": 7350,
  "count": 11555
}
{
  "_id": 7375,
  "count": 11455
}
{
  "_id": 7400,
  "count": 11355
}
{
  "_id": 7425,
  "count": 11255
}
{
  "_id": 7450,
  "count": 11155
}
{
  "_id": 7475,
  "count": 11055
}
{
  "_id": 7500,
  "count": 10955
}
{
  "_id": 7525,
  "count": 10855
}
{

```

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

Result ID#	Filter Rows	Time	Action	Resource	Duration
View Report Print Report Refresh					
441619	105000000021xW	10:00:00	partner_id		
441619	105000000021xW	10:00:00	partner_name	Higher Education	160
441619	105000000021xW	10:00:00	partner_type	Sustaining Agriculture	336
442596	105000000037xW	10:00:00	partner_id	Extreme Poverty	217
442533	105000000037xW	10:00:00	partner_name	Extreme Poverty	217
442533	105000000037xW	10:00:00	partner_type	Extreme Poverty	217
443477	105000000037xW	10:00:00	partner_id	Underserved	334
443477	105000000037xW	10:00:00	partner_name	Underserved	334
443477	105000000037xW	10:00:00	partner_type	Extreme Poverty	40
446716	105000000021xW	10:00:00	partner_id	Mobile Transactions	219
446716	105000000021xW	10:00:00	partner_name	Mobile Transactions	219
446716	105000000021xW	10:00:00	partner_type	Underserved	334
446717	105000000021xW	10:00:00	partner_id	Underserved	334
446719	105000000021xW	10:00:00	partner_id	Underserved	334
iran_theme_idc_ready *					
1	iran_theme_idc_ready	10:00:00	iran_theme_idc_ready		Action started
2	iran_theme_idc_ready	10:00:00	select * from iban_theme_idc where iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
3	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
4	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
5	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
6	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
7	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
8	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
9	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
10	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
11	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
12	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
13	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
14	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
15	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
16	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
17	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
18	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
19	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)
20	iran_theme_idc_ready	10:00:00	on iban_theme_idc,1 iban_theme_idc_ready = 1, iban_theme_idc_ready = 1	100 rows selected	0.0001 ms (0.0001 ms)

Fig. 28 Members in Specific Region on MySQL

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. 29 Members in Specific Region on MongoDB

291	select loan_ready_loan_id,loan_ready.partner_id,loan_ready.activity,loan_theme_ids_ready,loan_theme_type
292	from loan_ready
293	join loan_theme_ids_ready on loan_ready_loan_id = loan_theme_ids_ready.loan_ready_loan_id
294	where loan_ready.activity like '%Food%' and loan_theme_type like 'General'
295	;
296	
297	
298	
299	
300	
301	
302	
303	
304	
305	
306	
307	
308	
309	
310	
311	
312	
313	
314	
315	
316	
317	
318	
319	
320	
321	
322	
323	
324	
325	
326	
327	
328	
329	
330	
331	
332	
333	
334	
335	
336	
337	
338	
339	
340	
341	
342	
343	
344	
345	
346	
347	
348	
349	
350	
351	
352	
353	
354	
355	
356	
357	
358	
359	
360	
361	
362	
363	
364	
365	
366	
367	
368	
369	
370	
371	
372	
373	
374	
375	
376	
377	
378	
379	
380	
381	
382	
383	
384	
385	
386	
387	
388	
389	
390	
391	
392	
393	
394	
395	
396	
397	
398	
399	
400	
401	
402	
403	
404	
405	
406	
407	
408	
409	
410	
411	
412	
413	
414	
415	
416	
417	
418	
419	
420	
421	
422	
423	
424	
425	
426	
427	
428	
429	
430	
431	
432	
433	
434	
435	
436	
437	
438	
439	
440	
441	
442	
443	
444	
445	
446	
447	
448	
449	
450	
451	
452	
453	
454	
455	
456	
457	
458	
459	
460	
461	
462	
463	
464	
465	
466	
467	
468	
469	
470	
471	
472	
473	
474	
475	
476	
477	
478	
479	
480	
481	
482	
483	
484	
485	
486	
487	
488	
489	
490	
491	
492	
493	
494	
495	
496	
497	
498	
499	
500	
501	
502	
503	
504	
505	
506	
507	
508	
509	
510	
511	
512	
513	
514	
515	
516	
517	
518	
519	
520	
521	
522	
523	
524	
525	
526	
527	
528	
529	
530	
531	
532	
533	
534	
535	
536	
537	
538	
539	
540	
541	
542	
543	
544	
545	
546	
547	
548	
549	
550	
551	
552	
553	
554	
555	
556	
557	
558	
559	
560	
561	
562	
563	
564	
565	
566	
567	
568	
569	
570	
571	
572	
573	
574	
575	
576	
577	
578	
579	
580	
581	
582	
583	
584	
585	
586	
587	
588	
589	
590	
591	
592	
593	
594	
595	
596	
597	
598	
599	
600	
601	
602	
603	
604	
605	
606	
607	
608	
609	
610	
611	
612	
613	
614	
615	
616	
617	
618	
619	
620	
621	
622	
623	
624	
625	
626	
627	
628	
629	
630	
631	
632	
633	
634	
635	
636	
637	
638	
639	
640	
641	
642	
643	
644	
645	
646	
647	
648	
649	
650	
651	
652	
653	
654	
655	
656	
657	
658	
659	
660	
661	
662	
663	
664	
665	
666	
667	
668	
669	
670	
671	
672	
673	
674	
675	
676	
677	
678	
679	
680	
681	
682	
683	
684	
685	
686	
687	
688	
689	
690	
691	
692	
693	
694	
695	
696	
697	
698	
699	
700	
701	
702	
703	
704	
705	
706	
707	
708	
709	
710	
711	
712	
713	
714	
715	
716	
717	
718	
719	
720	
721	
722	
723	
724	
725	
726	
727	
728	
729	
730	
731	
732	
733	
734	
735	
736	
737	
738	
739	
740	
741	
742	
743	
744	
745	
746	
747	
748	
749	
750	
751	
752	
753	
754	
755	
756	
757	
758	
759	
760	
761	
762	
763	
764	
765	
766	
767	
768	
769	
770	
771	
772	
773	
774	
775	
776	
777	
778	
779	
780	
781	
782	
783	
784	
785	
786	
787	
788	
789	
790	
791	
792	
793	
794	
795	
796	
797	
798	
799	
800	
801	
802	
803	
804	
805	
806	
807	
808	
809	
810	
811	
812	
813	
814	
815	
816	
817	
818	
819	
820	
821	
822	
823	
824	
825	
826	
827	
828	
829	
830	
831	
832	
833	
834	
835	
836	
837	
838	
839	
840	
841	
842	
843	
844	
845	
846	
847	
848	
849	
850	
851	
852	
853	
854	
855	
856	
857	
858	
859	
860	
861	
862	
863	
864	
865	
866	
867	
868	
869	
870	
871	
872	
873	
874	
875	
876	
877	
878	
879	
880	
881	
882	
883	
884	
885	
886	
887	
888	
889	
890	
891	
892	
893	
894	
895	
896	
897	
898	
899	
900	
901	
902	
903	
904	
905	
906	
907	
908	
909	
910	
911	
912	
913	
914	
915	
916	
917	
918	
919	
920	
921	
922	
923	
924	
925	
926	
927	
928	
929	
930	
931	
932	
933	
934	
935	
936	
937	
938	
939	
940	
941	
942	
943	
944	
945	
946	
947	
948	
949	
950	
951	
952	
953	
954	
955	
956	
957	
958	
959	
960	
961	
962	
963	
964	
965	
966	
967	
968	
969	
970	
971	
972	
973	
974	
975	
976	
977	
978	
979	
980	
981	
982	
983	
984	
985	
986	
987	
988	
989	
990	
991	
992	
993	
994	
995	
996	
997	
998	
999	
1000	
1001	
1002	
1003	
1004	
1005	
1006	
1007	
1008	
1009	
1010	
1011	
1012	
1013	
1014	
1015	
1016	
1017	
1018	
1019	
1020	
1021	
1022	
1023	
1024	
1025	
1026	
1027	
1028	
1029	
1030	
1031	
1032	
1033	
1034	
1035	
1036	
1037	
1038	
1039	
1040	
1041	

Fig. 31 Countries Like '_am%' on MySQL

```

Typa "it" for more
Atlas-Sales1>shard0 [primary] kiva_db.all_kiva.find({country: /./, sum_vl: >, forgive:"Yes"})
{
  "_id": ObjectId("4d878bf51c9ad07873fb"),
  "sector": "General Financial Inclusion",
  "country": "Cambodia",
  "region": "Battambang, Cambodia",
  "partner_id": 180,
  "field_partner_name": "Matha Kasekar Limited (HKL)",
  "loan_theme_id": "2d0800000000C2",
  "loan_theme_type": "Green",
  "forgive": "Yes",
  "iso": "KHM",
  "loan_region_number": "12",
  "amount": 2089,
  "location_name": "Battambang, Cambodia",
  "lat": Decimal("13.6475959"),
  "lon": Decimal("102.8975064")
},
{
  "_id": ObjectId("4d878bf51c9ad07873fa"),
  "sector": "General Financial Inclusion",
  "country": "Cambodia",
  "region": "Battambang, Cambodia",
  "partner_id": 180,
  "field_partner_name": "Matha Kasekar Limited (HKL)",
  "loan_theme_id": "1a8800000000C2",
  "loan_theme_type": "Water",
  "forgive": "Yes",
  "iso": "KHM",
  "loan_region_number": "21.0",
  "amount": 27025,
  "location_name": "Battambang, Cambodia",
  "lat": Decimal("13.6475959"),
  "lon": Decimal("102.8975064")
},
{
  "_id": ObjectId("4d878bf51c9ad07873fd"),
  "sector": "General Financial Inclusion",
  "country": "Cambodia",
  "region": "Battambang, Cambodia",
  "partner_id": 180,
  "field_partner_name": "Matha Kasekar Limited (HKL)",
  "loan_theme_id": "1a8800000000C2",
  "loan_theme_type": "Water",
  "forgive": "Yes",
  "iso": "KHM",
  "loan_region_number": "12.0",
  "amount": 27770,
  "location_name": "Battambang, Cambodia",
  "lat": Decimal("13.6475959"),
  "lon": Decimal("102.8922085")
},
{
  "_id": ObjectId("4d878bf51c9ad07873fae"),
  "sector": "General Financial Inclusion",
  "country": "Cambodia",
  "region": "Battambang, Cambodia",
  "partner_id": 180,
  "field_partner_name": "Matha Kasekar Limited (HKL)",
  "loan_theme_id": "2d0800000000C2",
  "loan_theme_type": "Green",
  "forgive": "Yes",
  "iso": "KHM",
  "loan_region_number": "2.0",
  "amount": 2089,
  "location_name": "Battambang, Cambodia",
  "lat": Decimal("13.6475959"),
  "lon": Decimal("102.8922085")
},
{
  "_id": ObjectId("4d878bf51c9ad07873fa"),
  "sector": "General Financial Inclusion",
  "country": "Cambodia",
  "region": "Prey Veng, Cambodia",
  "partner_id": 180,
  "field_partner_name": "Matha Kasekar Limited (HKL)",
  "loan_theme_id": "2d0800000000C2",
  "loan_theme_type": "Green",
  "forgive": "Yes",
  "iso": "KHM",
  "loan_region_number": "12.0",
  "amount": 2089,
  "location_name": "Prey Veng, Cambodia",
  "lat": Decimal("13.6475959"),
  "lon": Decimal("102.8975064")
}

```

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

```
Type "in" for more
Atlas atlas-5-shard-0 [primary] kiva> db.all_kiva.find({country: ".+an.", $or: [{forkiva: "Yes"}, {country: {$regex: ".+an.+"} }]}).explain("executionStats")
{
  queryPlanner:
    {
      planCacheVersion: 1,
      nreturned: 1000000,
      indexesUsed: "kiva.all_kiva",
      indexFilterSet: false,
      parsedQuery: {
        "$or": [
          { "forkiva": { "$eq": "Yes" } },
          { "country": { "$regex": ".+an.+"} }
        ]
      },
      winningPlan: {
        stage: "COLLSCAN",
        filter: {
          "$and": [
            { forkiva: { "$eq": "Yes" } },
            { country: { "$regex": ".+an.+"} }
          ]
        },
        direction: "forward"
      },
      rejectedPlans: []
    },
  executionStats: {
    executionSuccess: true,
    totalKeysExamined: 0,
    executionTimeMillis: 362,
    totalDocsExamined: 0,
    totalBytesExamined: 887542,
    executionStages: [
      {
        stage: "COLLSCAN",
        filter: {
          "$and": [
            { forkiva: { "$eq": "Yes" } },
            { country: { "$regex": ".+an.+"} }
          ]
        },
        returned: 364,
        executionTimeInMilliseconds: 69,
        works: 887544,
        advanced: 264,
        needMoreData: 279,
        needYielded: 8,
        saveState: 887,
        restoreState: 887,
        isEOF: 0,
        direction: "forward",
        docsExamined: 887542
      }
    ],
    serverInfo: {
      host: "data225-shard-00-01.5q6p4.mongodb.net",
      port: 27017,
      version: "4.4.19",
      v: 1,
      vLong: 658971d1ef9343a9f62bf4708a81713def6e88c"
    },
    ok: 1
  },
  operationTime: Timestamp( t: 1634349484, i: 7 )
}
Atlas atlas-5-shard-0 [primary] kiva>
```

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

User: NIG					
	loan_id	partner_id	activity	loan_theme_ids_ready	loan_theme_type
261	select	loan_ready_loan_id,loan_ready.partner_id,loan_ready.activity,loan_theme_ids_ready,loan_theme_type			
262	from	loan_ready			
283	join				
284	loan_theme_ids_ready	on loan_ready.loan_id = loan_theme_ids_ready.loan_id			
285	where	loan_ready.activity like '%Food%' and loan_theme_type<>'General'			
286					
287					
288					
289					
290					
291					
292					
293					
294					
295					
296					
297					
298					
299					
300					
301					
302					
303					
304					
305					
306					
307					
308					
309					
310					
311					
312					
313					
314					
315					
316					
317					
318					
319					
320					
321					
322					
323					
324					
325					
326					
327					
328					
329					
330					
331					
332					
333					
334					
335					
336					
337					
338					
339					
340					
341					
342					
343					
344					
345					
346					
347					
348					
349					
350					
351					
352					
353					
354					
355					
356					
357					
358					
359					
360					
361					
362					
363					
364					
365					
366					
367					
368					
369					
370					
371					
372					
373					
374					
375					
376					
377					
378					
379					
380					
381					
382					
383					
384					
385					
386					
387					
388					
389					
390					
391					
392					
393					
394					
395					
396					
397					
398					
399					
400					
401					
402					
403					
404					
405					
406					
407					
408					
409					
410					
411					
412					
413					
414					
415					
416					
417					
418					
419					
420					
421					
422					
423					
424					
425					
426					
427					
428					
429					
430					
431					
432					
433					
434					
435					
436					
437					
438					
439					
440					
441					
442					
443					
444					
445					
446					
447					
448					
449					
450					
451					
452					
453					
454					
455					
456					
457					
458					
459					
460					
461					
462					
463					
464					
465					
466					
467					
468					
469					
470					
471					
472					
473					
474					
475					
476					
477					
478					
479					
480					
481					
482					
483					
484					
485					
486					
487					
488					
489					
490					
491					
492					
493					
494					
495					
496					
497					
498					
499					
500					
501					
502					
503					
504					
505					
506					
507					
508					
509					
510					
511					
512					
513					
514					
515					
516					
517					
518					
519					
520					
521					
522					
523					
524					
525					
526					
527					
528					
529					
530					
531					
532					
533					
534					
535					
536					
537					
538					
539					
540					
541					
542					
543					
544					
545					
546					
547					
548					
549					
550					
551					
552					
553					
554					
555					
556					
557					
558					
559					
560					
561					
562					
563					
564					
565					
566					
567					
568					
569					
570					
571					
572					
573					
574					
575					
576					
577					
578					
579					
580					
581					
582					
583					
584					
585					
586					
587					
588					
589					
590					
591					
592					
593					
594					
595					
596					
597					
598					
599					
600					
601					
602					
603					
604					
605					
606					
607					
608					
609					
610					
611					
612					
613					
614					
615					
616					
617					
618					
619					
620					
621					
622					
623					
624					
625					
626					
627					
628					
629					
630					
631					
632					
633					
634					
635					
636					
637					
638					
639					
640					
641					
642					
643					
644					
645					
646					
647					
648					
649					
650					
651					
652					
653					

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

```
Atlas: atlas-salsero-0 shard=0 (primary) kivo.$all_kiva.find({activity_id:{$in:loan_ids}, loan_theme_type:{$in:"General"} })
```

{
 id: ObjectId("618bd1c51c99ac07856ab9e*"),
 id: 45351,
 funded_amount: 100,
 loan_amount: 200,
 activity: "Food & Production/Sales",
 sector: "Food",
 use: "To buy a sugar cane juicing machine.",
 country: "Cebu/Philippines",
 region: "Bantayan province, Mung Russey district",
 currency: "USD",
 posted_time: ISODate("2014-01-02T08:10:00.000Z"),
 disbursed_time: ISODate("2013-12-23T08:00:00.000Z"),
 funded_time: ISODate("2014-01-02T10:01:00.000Z"),
 term_in_months: 12,
 lender_count: 20,
 borrower_genders: ["female"],
 repayment_interval: "monthly",
 date: ISODate("2014-01-02T00:00:00.000Z")
},
{
 _id: ObjectId("618bd1c51c99ac07856ab9e*"),
 id: 45351,
 funded_amount: 1000,
 loan_amount: 2000,
 activity: "Food",
 sector: "Food",
 use: "To buy more ingredients for making rice soup and Chinese noodle.",
 country_code: "PHL",
 region: "Battambang province, Mung Russey district",
 currency: "USD",
 posted_time: ISODate("2014-01-03T08:05:18.000Z"),
 disbursed_time: ISODate("2013-12-23T08:00:00.000Z"),
 funded_time: ISODate("2014-01-04T21:02:27.000Z"),
 term_in_months: 12,
 lender_count: 20,
 borrower_genders: ["female", "female", "female"],
 repayment_interval: "monthly",
 date: ISODate("2014-01-03T00:00:00.000Z")
},
{
 _id: ObjectId("618bd1c51c99ac07856ab9e*"),
 id: 45351,
 funded_amount: 100,
 loan_amount: 400,
 activity: "Food",
 sector: "Food",
 use: "To buy beans for resale",
 country: "Cebu/Philippines",
 region: "Bantayan province, Mung Russey district",
 currency: "USD",
 posted_time: ISODate("2014-01-03T08:53:10.000Z"),
 disbursed_time: ISODate("2013-12-05T08:00:00.000Z"),
 funded_time: ISODate("2014-01-04T17:52:04.000Z"),
 term_in_months: 12,
 lender_count: 20,
 borrower_genders: ["female", "female"],
 repayment_interval: "monthly",
 date: ISODate("2014-01-03T00:00:00.000Z")
},
{
 _id: ObjectId("618bd1c51c99ac07856ab9e*"),
 id: 45351,
 funded_amount: 1000,
 loan_amount: 2000,
 activity: "Food",
 sector: "Food",
 use: "To buy vegetables and fish to resell.",
 country_code: "PHL",
 region: "Battambang province, Mung Russey district",
 currency: "USD",
 posted_time: ISODate("2014-01-03T08:53:10.000Z"),
 disbursed_time: ISODate("2013-12-05T08:00:00.000Z"),
 funded_time: ISODate("2014-01-04T17:52:04.000Z"),
 term_in_months: 12,
 lender_count: 20,
 borrower_genders: ["female", "female"],
 repayment_interval: "monthly",
 date: ISODate("2014-01-03T00:00:00.000Z")
}

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

```

Atlas atlas-Saller7-shard-0 [primary] kiva> db.all_kiva.aggregate([{$match: {term_in_months: {$gt:10}}}, {$group: {_id:$activity, cmt:{$sum:cmt}}}, {$sort:[{_id:1}, {_cmt:-1}]}])
[{"_id": "Health", "cmt": 13}, {"_id": "Consumer Goods", "cmt": 11}, {"_id": "Fuel/Firewood", "cmt": 11}, {"_id": "Furniture Making", "cmt": 13}, {"_id": "Handicrafts", "cmt": 13}, {"_id": "Well digging", "cmt": 16}, {"_id": "Sewing", "cmt": 16}, {"_id": "Clothing Sales", "cmt": 18}, {"_id": "Cereals", "cmt": 18}, {"_id": "Beauty Salon", "cmt": 20}, {"_id": "Education", "cmt": "school costs", "cmt": 21}, {"_id": "Crafts", "cmt": 21}, {"_id": "Taxi", "cmt": 25}, {"_id": "Transportation", "cmt": 26}, {"_id": "Food Stall", "cmt": 27}, {"_id": "Clothing Sales", "cmt": 29}, {"_id": "Transportation", "cmt": 30}, {"_id": "Transportation", "cmt": 33}, {"_id": "Property", "cmt": 36}, {"_id": "Wedding Expenses", "cmt": 36}], [{"text": "Type 'it' for more"}]
Atlas atlas-Saller7-shard-0 [primary] kiva> it
[{"_id": "Land Rental", "cmt": 38}, {"_id": "Services", "cmt": 40}, {"_id": "Construction", "cmt": 45}, {"_id": "Construction Supplies", "cmt": 69}, {"_id": "Beverages", "cmt": 71}, {"_id": "Retail", "cmt": 73}, {"_id": "Retail", "cmt": 74}, {"_id": "Livestock", "cmt": 75}, {"_id": "Weaving", "cmt": 76}, {"_id": "Weaving", "cmt": 98}, {"_id": "Poultry", "cmt": 91}, {"_id": "Food", "cmt": "Food supplies", "cmt": 92}, {"_id": "Food", "cmt": 92}, {"_id": "Personal Expenses", "cmt": 99}, {"_id": "Food", "cmt": 100}, {"_id": "Fruits & Vegetables", "cmt": 134}, {"_id": "Animals Sales", "cmt": 139}, {"_id": "Farming", "cmt": 146}, {"_id": "Fishing", "cmt": 157}, {"_id": "Farm Supplies", "cmt": 269}], [{"text": "Type 'it' for more"}]
Atlas atlas-Saller7-shard-0 [primary] kiva> it
[{"_id": "Cattle", "cmt": 314}, {"_id": "Grocery Store", "cmt": 340}, {"_id": "Agriculture", "cmt": 386}, {"_id": "Food", "cmt": 400}, {"_id": "Vehicle", "cmt": 644}, {"_id": "Home Energy", "cmt": 943}, {"_id": "Transportation", "cmt": 1012}, {"_id": "Higher education costs", "cmt": 1264}, {"_id": "Personal Housing Expenses", "cmt": 1936}, {"_id": "Farming", "cmt": 7638}], [{"text": "Atlas atlas-Saller7-shard-0 [primary] kiva"}]

```

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

Read - Operation 6		
DB	Command	Execution Time(ms)
SQL	<pre> 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'; </pre>	24
NoSQL	<pre> db.all_kiva.find({activity:/.*Food.*/, loan_theme_type:{\$ne:"General"} }) </pre>	305

Table 6. Performance and Commands for the fifth read operation

g) Read Operation - 7

```
        nReturned: Long("50"),
        executionTimeMillisEstimate: Long("343")
    )
},  
serverInfo: {  
    host: "127.0.0.1:22555-shard-00-01.5q6p4.mongodb.net",  
    port: 27017,  
    version: "4.0.9",  
    gitVersion: "58971da1ef93435a9f62bf4708a81713defe88c"  
},  
ok: 1,  
$clusterTime: {  
    clusterTime: Timestamp({ t: 1636352470, i: 4 }),  
    signature: {  
        hash: BinaryBuffer.from("708abcf9645b0f2b4d3dbc71bsaca7e0f2f26763c", "hex"),  
        keyId: Long("6994568891324489738")  
},  
operationTime: Timestamp({ t: 1636352470, i: 4 })
```

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

Read - Operation 7		
DB	Command	Execution Time(milliseconds)
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": "Sacrifice", "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.

```
287 • update loan_theme_ids_ready set loan_theme_type='Unique' where loan_theme_type='General';
Action Output 5
Time Action Response Duration / Fetch Time
12 17:58:25 SELECT * FROM kiva_loan_themes_by_region LIMIT 0, 1000
13 17:58:26 select count(*),count(country) from kiva_loan_themes_by_region group by country order by count(country) desc LIMIT 0, 1000
14 17:58:02 select count(*),count(country) from kiva_loan_themes_by_region ready group by country order by count(country) desc LIMIT 0, 1000
15 18:01:40 update loan_theme_ids_ready set loan_theme_type='Unique' where loan_theme_type='General'
16 18:01:40 update loan_theme_ids_ready set loan_theme_type='Unique' where loan_theme_type='General'
17 18:15:37 update loan_theme_ids_ready set loan_theme_type='Unique' where loan_theme_type='General'
18 18:21:49 update loan_theme_ids_ready set loan_theme_type='Unique' where loan_theme_type='General'
19 18:21:49 update loan_theme_ids_ready set loan_theme_type='Unique' where loan_theme_type='General'
20 18:21:59 update count(loan_theme_type) from loan_theme_ids_ready where loan_theme_type='General' LIMIT 0, 1000
21 18:23:06 update loan_theme_ids_ready set loan_theme_type='Unique' where loan_theme_type='General'
```

Fig. 40 Update on Mysql

```
Atlas atlas-Selle7-shard-0 [primary] kiva> db.all_kiva.updateMany({loan_theme_type:"General"}, {$set:{loan_theme_type:"Unique"} })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 384876,
  upsertedCount: 0
}
```

Fig. 41 Update documents set a value on condition

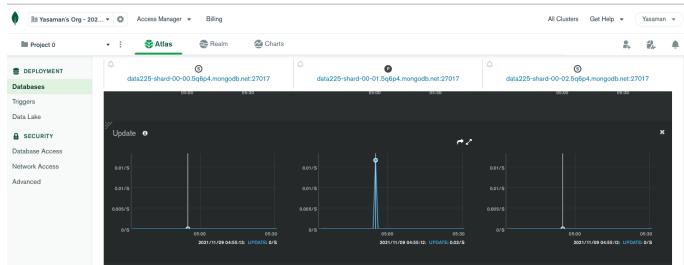


Fig.42 Update Performance on MongoDB

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.

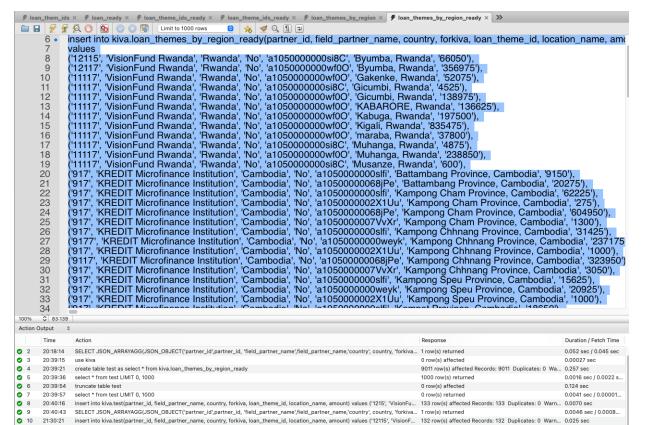


Fig.43 Insert 133 records to MySQL

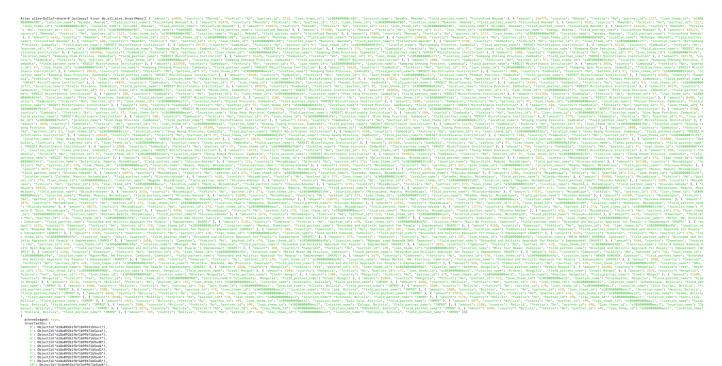


Fig.44 Insert 133 documents to all_kiva collection

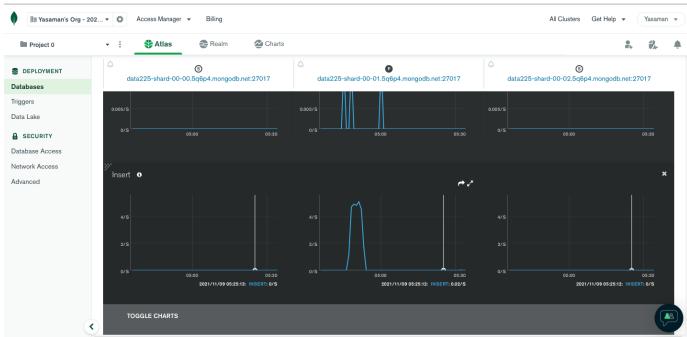


Fig. 45 Performance of Insert on MongoDB

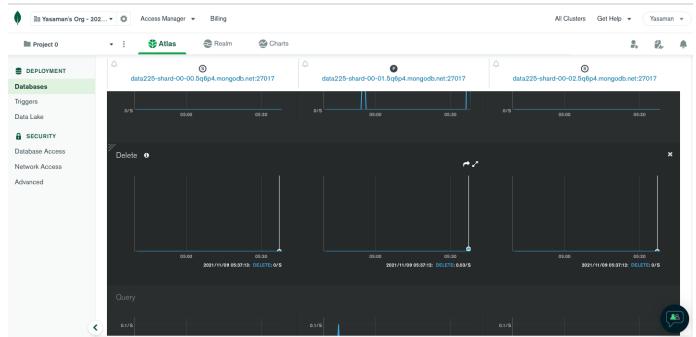


Fig. 48 Performance of delete documents on MongoDB

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.

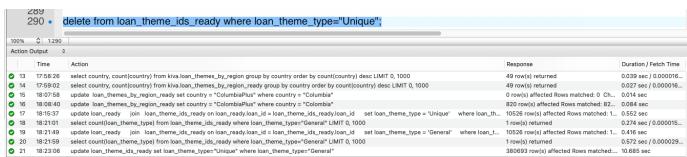


Fig. 46 Delete records based on condition on MySQL

```
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. 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 Ayny Broosh. “ How to Choose the Right Database”,Sept. 2019, <https://towardsdatascience.com/how-to-choose-the-right-database-afc95541741>.
- [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@jsu.edu

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

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

Yasaman Emami
Department of Applied Data Science
San Jose State University
San Jose, United States
yasaman.emami@jsu.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 [1]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
import warnings
warnings.filterwarnings('ignore')

In [2]: df = pd.read_csv('kiva_loans.csv')

In [3]: data = pandas.read_csv('kiva_loans.csv', encoding='utf-8', quotechar='"', delimiter=',')
In [4]: df.info()
In [5]: df.describe()

In [6]: null = df.isnull().sum().sort_values(ascending = False).reset_index()
null.columns = ['Column', 'Frequency']

In [7]: ## tags column consists of 10000 null values so remove the column.
df.drop('tags', axis=1, inplace=True)
null

In [8]: ## this column null values are replaced with the mode
df['funded_time'].mode()

In [9]: df['borrower_genders'].mode()

In [10]: df.fillna(df['funded_time'].mode(), inplace = True)
df['borrower_genders'].fillna(df['borrower_genders'].mode(), inplace = True)
df.dropna(inplace = True)

In [11]: df.isna().sum()

In [12]: 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.).

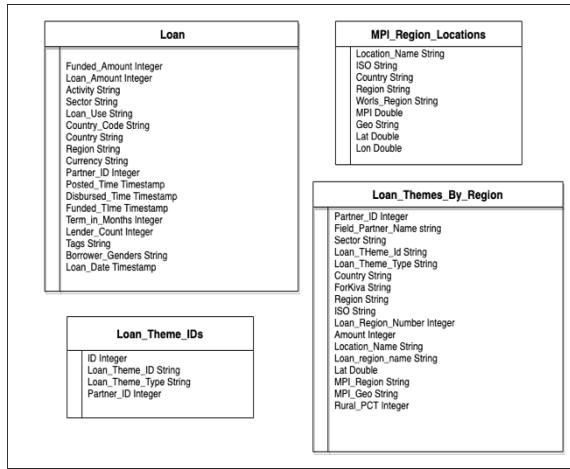


Fig. 2 Data Model - NoSQL

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’.

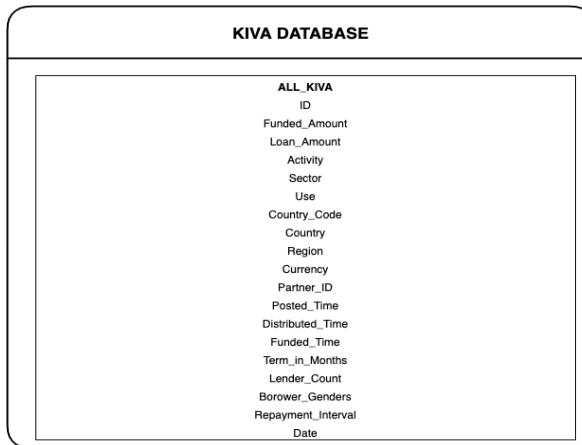


Fig. 3 Document Structure in MongoDB

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.

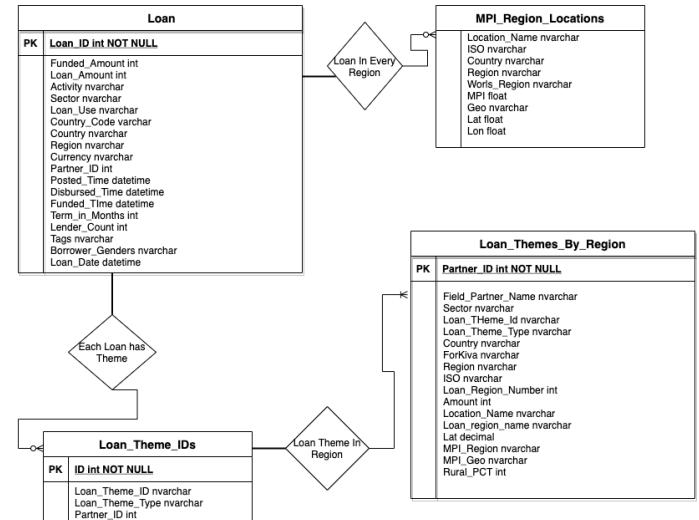


Fig. 4 Denormalized Data in SQL

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.

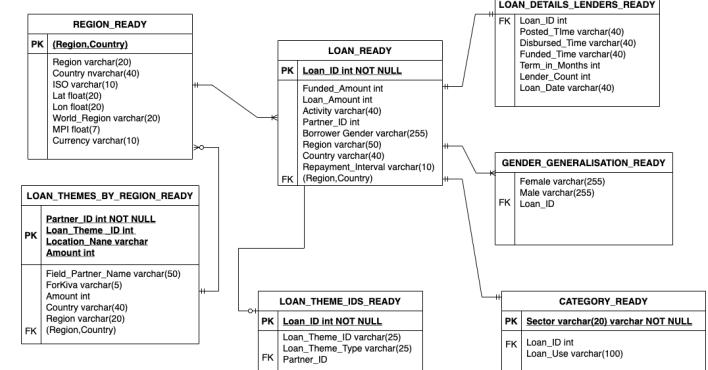


Fig. 5 Normalized Data Model in SQL

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

```

Atlas atlas-5al1e7-shard-0 [primary] kiva> db.all_kiva.find()
{
  {
    _id: ObjectId("6189fd2d495fdaade07304b"),
    id: 65315,
    funded_amount: 1500,
    loan_amount: 1500,
    activity: 'Motorcycle Transport',
    sector: 'Transportation',
    use: 'To purchase motorcycle for his son's commute to school.  ',
    country_code: 'KH',
    country: 'Cambodia',
    region: 'Khasach Kandal district',
    currency: 'USD',
    partner_id: 63,
    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"),
    term_in_months: 24,
    lender_count: 53,
    borrower_genders: 'male',
    repayment_interval: 'monthly',
    date: ISODate("2014-01-02T08:00:00.000Z")
  },
  {
    _id: ObjectId("6189fd2d495fdaade07304c"),
    id: 65319,
    funded_amount: 700,
    loan_amount: 700,
    activity: 'Business',
    sector: 'Personal Use',
    use: 'To purchase a motorbike for his brother to drive to work.',
    country_code: 'KH',
    country: 'Cambodia',
    region: 'Kandal Steung district',
    currency: 'USD',
    partner_id: 63,
    posted_time: ISODate("2014-01-02T07:29:53.000Z"),
    disbursed_time: ISODate("2013-12-16T08:00:00.000Z"),
    funded_time: ISODate("2014-01-22T12:32:48.000Z"),
    term_in_months: 14,
    lender_count: 28,
    borrower_genders: 'male',
    repayment_interval: 'monthly',
    date: ISODate("2014-01-02T08:00:00.000Z")
  },
  {
    _id: ObjectId("6189fd2d495fdaade07304d"),
    id: 65358,
    funded_amount: 1100,
    loan_amount: 1100,
    activity: 'Personal Housing Expenses',
    sector: 'Housing',
    use: 'To build a safe latrine and a water storage unit to improve her family's health',
    country_code: 'KH',
    country: 'Cambodia',
    region: 'Pursat',
    currency: 'USD',
    partner_id: 106,
    posted_time: ISODate("2014-01-03T07:34:36.000Z"),
    disbursed_time: ISODate("2013-12-02T08:00:00.000Z"),
    funded_time: ISODate("2014-01-04T23:37:46.000Z"),
    term_in_months: 26,
    lender_count: 32,
    borrower_genders: 'female',
    repayment_interval: 'monthly',
    date: ISODate("2014-01-03T08:00:00.000Z")
  },
  {
    _id: ObjectId("6189fd2d495fdaade07304c"),
    id: 65316,
    funded_amount: 600,
    loan_amount: 600,
    activity: 'Cattle',
    sector: 'Agriculture',
    use: 'buy a calf to raise. ',
    country_code: 'KH',
    country: 'Cambodia',
    region: 'Kampong Penh',
    currency: 'USD',
    partner_id: 63,
    posted_time: ISODate("2014-01-02T08:18:15.000Z"),
    disbursed_time: ISODate("2013-12-04T08:00:00.000Z"),
    funded_time: ISODate("2014-01-02T15:12:10.000Z"),
    term_in_months: 22,
    lender_count: 22,
    borrower_genders: 'female',
    repayment_interval: 'monthly',
    date: ISODate("2014-01-02T08:00:00.000Z")
  },
  {
    _id: ObjectId("6189fd2d495fdaade07304a"),
    id: 65363,
    funded_amount: 500,
  }
}

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	<pre>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_the_ me_id = loan_theme_ids_ready.loan_theme_id;</pre>	1837
NoSQL	db.all_kiva.find()	105

Table 1. Performance and Commands for the first read operation

b) Read operation - 2

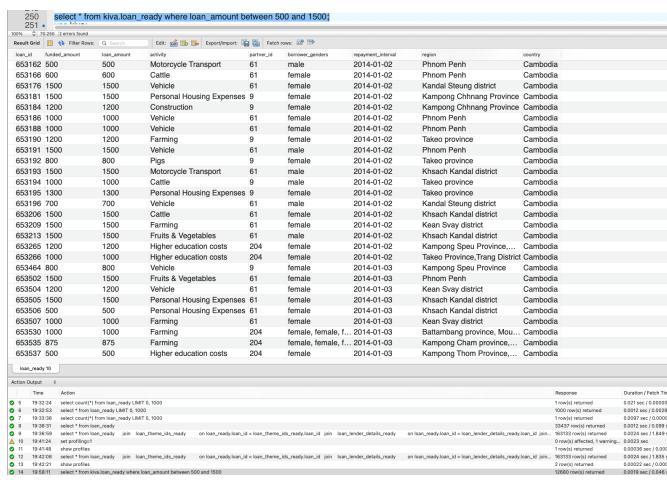


Fig. 7.a loan_amount between 500\$,1500\$ on MySQL

```
yasamanenamami -- mongosh mongodbsrv://data225.5q6p4.mongodb.net/myFirstDatabase -- myFirstDatabase
{
  _id: ObjectId("e1878d1c51c09adb7856b987"),
  id: 653162,
  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("e1878d1c51c09adb7856b98a"),
  id: 653166,
  funded_amount: 600,
  loan_amount: 600,
  activity: 'Calf',
  sector: 'Agriculture',
  use: 'To buy a calf to raise. ',
  country_code: 'KH',
  country: 'Cambodia',
  region: 'Phnom Penh',
  currency: 'USD',
  partner_id: 69,
  posted_time: ISODate("2014-01-02T06:18:15.000Z"),
  disbursed_time: ISODate("2013-12-06T08:00:00.000Z"),
  funded_time: ISODate("2014-01-02T15:12:10.000Z"),
  term_in_months: 22,
  lender_count: 22,
  borrower_genders: 'female',
  repayment_interval: 'monthly',
  date: ISODate("2014-01-02T00:00:00.000Z")
},
{
  _id: ObjectId("e1878d1c51c09adb7856b98b"),
  id: 653168,
  funded_amount: 1800,
  loan_amount: 1800,
  activity: 'Vehicle',
  sector: 'Personal Use',
  use: 'To buy a motorbike for family transportation. ',
  country_code: 'KH',
  country: 'Cambodia',
  region: 'Phnom Penh',
  currency: 'USD',
  partner_id: 61,
  posted_time: ISODate("2014-01-02T07:05:02.000Z"),
  disbursed_time: ISODate("2013-12-13T08:00:00.000Z"),
  funded_time: ISODate("2014-01-10T03:22:29.000Z"),
  term_in_months: 22,
  lender_count: 33,
  borrower_genders: 'female',
  repayment_interval: 'monthly',
  date: ISODate("2014-01-02T00:00:00.000Z")
},
{
  _id: ObjectId("e1878d1c51c09adb7856b98c"),
  id: 653169,
  funded_amount: 1800,
  loan_amount: 1800,
  activity: 'Vehicle',
  sector: 'Personal Use',
  use: 'To buy a motorbike for family transportation. ',
  country_code: 'KH',
  country: 'Cambodia',
  region: 'Phnom Penh',
  currency: 'USD',
  partner_id: 61,
  posted_time: ISODate("2014-01-02T07:05:02.000Z"),
  disbursed_time: ISODate("2013-12-13T08:00:00.000Z"),
  funded_time: ISODate("2014-01-10T03:22:29.000Z"),
  term_in_months: 22,
  lender_count: 33,
  borrower_genders: 'female',
  repayment_interval: 'monthly',
  date: ISODate("2014-01-02T00:00:00.000Z")
},
```

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

```
yasamanenamami -- mongosh mongodbsrv://data225.5q6p4.mongodb.net/myFirstDatabase -- myFirstDatabase TMPDIR=/var/folders/zj/s...
{
  _id: "it"
  for more
Atlas atlas-Sale17-shard-0 [primary] kiva> db.all_kiva.find({loan_amount:{'$gt':500, '$lt':1500}}).explain("executionStats")
{
  queryPlanner:
    planCacheId: 5,
    namespace: "kiva.all_kiva",
    indexFilterSet: false,
    pipeline: [
      {
        '$and': [
          {loan_amount: {'$gt': 500}},
          {loan_amount: {'$lt': 1500}}
        ]
      }
    ],
    winningPlan: {
      stage: "COLLSCAN",
      filter: null,
      pipeline: [
        {loan_amount: {'$gt': 500}},
        {loan_amount: {'$lt': 1500}}
      ],
      direction: "forward"
    },
    rejectedPlans: []
  },
  executionStats: {
    executionTimeMicros: true,
    duration: 18023,
    executionTimeMillisEstimate: 38,
    totalKeysExamined: 0,
    totalDocsExamined: 107542,
    executionStages: [
      {
        stage: "COLLSCAN",
        filter: {
          '$and': [
            {loan_amount: {'$gt': 500}},
            {loan_amount: {'$lt': 1500}}
          ]
        }
      }
    ],
    executionTimeNanos: 18023000000
  },
  serverInfo: {
    host: "data225-shard-0-0-1.5q6p4.mongodb.net",
    port: 27817,
    version: "4.4.18",
    gitVersion: "68971dafe93a38a97f2bf708a81713def6ee88c",
    ok: 1,
    $clusterTime: {
      clusterTime: Timestamp( t: 1636314302, i: 1 ),
      $maxTimeDocument: 0
    },
    operationTime: Timestamp( t: 1636314302, i: 1 )
  }
}
Atlas atlas-Sale17-shard-0 [primary] kiva>
```

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

```
    ( _id: null, count: 774185 ),
    ( _id: 125, count: 3630 ),
    ( _id: 280, count: 3599 ),
    ( _id: 199, count: 3453 ),
    ( _id: 100, count: 3373 ),
    ( _id: 1800, count: 2073 ),
    ( _id: 580, count: 1858 ),
    ( _id: 140, count: 1833 ),
    ( _id: 780, count: 985 ),
    ( _id: 1580, count: 799 ),
    ( _id: 1000, count: 780 ),
    ( _id: 75, count: 628 ),
    ( _id: 600, count: 519 ),
    ( _id: 1200, count: 518 ),
    ( _id: 225, count: 504 ),
    ( _id: 150, count: 461 ),
    ( _id: 425, count: 445 ),
    ( _id: 100, count: 422 ),
    ( _id: 480, count: 419 ),
    ( _id: 525, count: 364 )
  ]
Type "it" for more
Atlas atlas-Salle7-shard-0 [primary] kiva> it
[{"_id": 1280, "count": 361}, {"_id": 800, "count": 349}, {"_id": 1400, "count": 343}, {"_id": 300, "count": 385}, {"_id": 1925, "count": 277}, {"_id": 580, "count": 272}, {"_id": 1000, "count": 261}, {"_id": 875, "count": 243}, {"_id": 550, "count": 224}, {"_id": 1200, "count": 215}, {"_id": 980, "count": 212}, {"_id": 1180, "count": 207}, {"_id": 100, "count": 193}, {"_id": 775, "count": 181}, {"_id": 650, "count": 180}, {"_id": 1250, "count": 171}, {"_id": 2800, "count": 162}, {"_id": 575, "count": 149}, {"_id": 1380, "count": 148}, {"_id": 1175, "count": 144}], Type "it" for more
Atlas atlas-Salle7-shard-0 [primary] kiva> it
[{"_id": 720, "count": 166}, {"_id": 1000, "count": 148}, {"_id": 425, "count": 138}, {"_id": 1470, "count": 124}, {"_id": 100, "count": 123}, {"_id": 990, "count": 115}, {"_id": 1275, "count": 108}, {"_id": 1000, "count": 100}, {"_id": 1380, "count": 99}, {"_id": 1375, "count": 94}, {"_id": 475, "count": 93}, {"_id": 1250, "count": 88}, {"_id": 1875, "count": 82}, {"_id": 1225, "count": 77}, {"_id": 1000, "count": 77}, {"_id": 1150, "count": 72}, {"_id": 1975, "count": 71}, {"_id": 1250, "count": 69}, {"_id": 925, "count": 67}, {"_id": 1760, "count": 65}], Type "it" for more
Atlas atlas-Salle7-shard-0 [primary] kiva> ||
```

Fig. 8.b Count of Each Loan

```

Atlas atlas-5alle7-shard-0 [primary] kiva> db.all_kiva.aggregate([ { $group: { _id: '$loan_amount', count:{$sum:1} } }, { $sort:{'count':-1} } ])
{
  stages: [
    {
      'source': {
        queryPlanner: {
          plannerVersion: 1,
          namespace: '6157659f011433d63ab9f780e_kiva.all_kiva',
          indexFilterString: '',
          parsedQuery: {},
          queryHash: '10878475C',
          planCacheKey: '10878475C',
          winningPlan: {
            stage: 'PROJECTION_SIMPLE',
            transformBy: { loan_amount: 1, _id: 0 },
            inputStage: { stage: 'COLLSCAN', direction: 'forward' }
          },
          rejectedPlans: []
        },
        executionStats: {
          executionSucceeded: true,
          nReturned: 887542,
          executionTimeInMillis: 482,
          totalKeysExamined: 0,
          totalDocsExamined: 887542,
          executionStages: [
            {
              stage: 'PROJECTION_SIMPLE',
              nReturned: 887542,
              executionTimeInMillisEstimate: 133,
              works: 887544,
              advanced: 887542,
              needTime: 1,
              needYield: 0,
              saveState: 886,
              restoreState: 888,
              isEOF: 1,
              transformBy: { loan_amount: 1, _id: 0 },
              inputStage: {
                stage: 'COLLSCAN',
                nReturned: 887542,
                executionTimeInMillisEstimate: 74,
                works: 887544,
                advanced: 887542,
                needTime: 1,
                needYield: 0,
                saveState: 886,
                restoreState: 888,
                isEOF: 1,
                direction: 'forward',
                docsExamined: 887542
              }
            }
          ]
        },
        nReturned: Long("887542"),
        executionTimeInMillisEstimate: Long("495")
      }
    },
    {
      '$group': { '_id': '$loan_amount', count: { '$sum': { '$const': 1 } } },
      nReturned: Long("154"),
      executionTimeInMillisEstimate: Long("678")
    },
    {
      '$sort': { 'count': -1 },
      nReturned: Long("154"),
      executionTimeInMillisEstimate: Long("678")
    }
  ],
  serverInfo: {
    host: 'atlas225-shard-00-01.5qqp4.mongodb.net',
    port: 27017,
    version: '4.4.10',
    gitVersion: '58997da1ef93435a9f62b0f47b8a81713def6e88bc'
  },
  ok: 1,
  $clusterTime: {
    clusterTime: Timestamp({ t: 1636311298, i: 11 }),
    signature: {
      hashKey: '00000000000000000000000000000000',
      keyId: Long("6994586891324489738")
    }
  },
  operationTime: Timestamp({ t: 1636311298, i: 9 })
}
Atlas atlas-5alle7-shard-0 [primary] kiva> 

```

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:"Sloan_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

```
Atlas:atlas$allt7-hard-0 [primary] kiva> db.all_kiva.find({country: { $in: ['Yemen', 'Egypt', 'Madagascar'] }}).sort({country:-1}).explain('executionStats')
{
  queryPlanner:
    {
      plannerVersion: 1,
      namespace: 'db.all_kiva',
      indexFilterSet: false,
      parsedQuery: { country: { '$in': [ 'Egypt', 'Madagascar', 'Yemen' ] } },
      winningPlan:
        {
          stage: 'SORT',
          sortPattern: { country: -1 },
          memLimit: 1350422,
          type: 'simple',
          inputStage: {
            stage: 'COLLSCAN',
            filter: { country: { '$in': [ 'Egypt', 'Madagascar', 'Yemen' ] } },
            direction: 'forward'
          }
        },
      rejectedPlans: []
    },
  executionStats:
    {
      executionSuccess: true,
      nreturned: 3,
      executionTimeMillis: 348,
      totalKeysExamined: 0,
      totalDocsExamined: 807542,
      executionPlan:
        {
          stage: 'SORT',
          nReturned: 3,
          executionTimeMillisEstimate: 54,
          works: 807602,
          advanced: 50,
          needTime: 807444,
          needield: 0,
          saveState: 807,
          restoreState: 807,
          isEOF: 1,
          sortPattern: { country: -1 },
          memLimit: 1350422,
          type: 'simple',
          totalDataSizeSorted: 14821,
          usedMemory: 0
        },
      inputStage:
        {
          stage: 'COLLSCAN',
          filter: { country: { '$in': [ 'Egypt', 'Madagascar', 'Yemen' ] } },
          nReturned: 0,
          executionTimeMillisEstimate: 51,
          works: 807602,
          advanced: 57,
          needTime: 807486,
          needield: 0,
          saveState: 807,
          restoreState: 807,
          isEOF: 0,
          direction: 'forward',
          docExamine: 807542
        }
    },
  serverInfo: {
    host: 'data2255-shard-00-01.Sq6p4.mongodb.net',
    port: 27017,
    version: '4.0.16',
    gitVersion: '18871date:f93435a9f62bf4708a81713defe688c'
  },
  ok: 1,
  '$clusterTime':
    {
      clusterTime: timestamp( t: 1636354413, i: 7 ),
      signature:
        {
          hash: BinaryBuffer.from("912fe470081fd4c0852bc52699c689fa387cc6", "hex"),
          keyId: Long("699458689132468738")
        }
    },
  operationTime: timestamp( t: 1636354413, i: 7 )
}
....
```

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.*/'

```
Type 'it' for more
Atlas atlas-501e7-shard-0 [primary] kiva> db.all_kiva.find({country:/.+am./, forkiva:"Yes"}).explain("executionStats")
{
  queryPlanner: {
    plannerVersion: 1,
    namespace: 'kiva.all_kiva',
    indexFilterSet: false,
    parsedQuery: {
      '$and': [
        { forkiva: { '$eq': 'Yes' } },
        { country: { '$regex': '.+am.' } }
      ]
    },
    winningPlan: {
      stage: 'COLLSCAN',
      filter: {},
      '$and': [
        { forkiva: { '$eq': 'Yes' } },
        { country: { '$regex': '.+am.' } }
      ],
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 264,
    executionTimeMillisEstimate: 69,
    totalKeysExamined: 362,
    totalDocsExamined: 0,
    totalDocsSelected: 887542,
    executionStages: {
      stage: 'COLLSCAN',
      filter: {},
      '$and': [
        { forkiva: { '$eq': 'Yes' } },
        { country: { '$regex': '.+am.' } }
      ]
    },
    nReturned: 264,
    executionTimeMillisEstimate: 69,
    workTime: 807279,
    advanced: 264,
    needTime: 807279,
    needTime: 807279,
    saveState: 807,
    restoreState: 807,
    iops: 0,
    direction: 'forward',
    docsExamined: 887542
  },
  serverInfo: {
    host: 'atlas-501e7-shard-00-01.5q6p4.mongodb.net',
    port: 27017,
    version: '4.4.19',
    gitVersion: '69971da1ef93435a9f62bf4788a81713def6e8bc'
  },
  ok: 1,
  $clusterTime: {
    $clusterTime: timestamp({ t: 1636349484, i: 7 }),
    signature: {
      hash: Binary(Buffer.from('036209837ee528bfad8c08579d52f3cc0ba9bdd', 'hex'), 0),
      keyId: Long('699456897324499738')
    }
  },
  operationTime: timestamp({ t: 1636349484, i: 7 })
}
Atlas atlas-501e7-shard-0 [primary] kiva> //
```

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

use kiva						
loan_id	partner_id	activity	loan_theme_type	Response	Duration / Fetch Time	Action Output
261	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';					
262						
263						
264						
265						
266						
267						
100%	78-265	1 Error found				
Result Grid						
Filter Rows: <input type="text"/> Search Export:						
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			
100...	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

```

atlas:atlas-Galle7-shard0001:PRIMARY> db.al1.alva.find({activity_id:$eq:61, loan_type:{ $in: ["General"] } })
{
  {
    "_id": ObjectId("61878d1c519ad9785689e*"),
    "id": 45051,
    "funded_amount": 200,
    "loan_amount": 200,
    "activity": "Food, Agriculture/Production/Sales",
    "sector": "Food",
    "use": "buying a sugar cane juicing machine.",
    "country_code": "KH",
    "country": "Cambodia",
    "region": "Kampong Speu province",
    "currency": "USD",
    "partner": "Kampong Speu Provincial Government",
    "posted_time": ISODate("2024-01-04T08:18:59.000Z"),
    "disbursed_time": ISODate("2023-12-23T08:00:00.000Z"),
    "funded_time": ISODate("2024-01-04T08:00:00.000Z"),
    "term_in_months": 14,
    "lender_count": 26,
    "borrower_genders": "female",
    "repayment_interval": "monthly",
    "date": ISODate("2024-01-02T00:00:00.000Z")
  },
  {
    "_id": ObjectId("61878d1c519ad9785689e*"),
    "id": 45052,
    "funded_amount": 1000,
    "loan_amount": 1000,
    "activity": "Food",
    "sector": "Food",
    "use": "to buy more ingredients for making rice soup and Chinese noodle.",
    "country_code": "KH",
    "country": "Cambodia",
    "region": "Rattambon province, Moun Ruessey district",
    "currency": "USD",
    "partner": "Rattambon Provincial Government",
    "posted_time": ISODate("2024-01-07T08:13:18.000Z"),
    "disbursed_time": ISODate("2023-12-06T08:00:00.000Z"),
    "funded_time": ISODate("2024-01-04T15:23:00.000Z"),
    "term_in_months": 15,
    "lender_count": 26,
    "borrower_genders": "female, female, female",
    "repayment_interval": "monthly",
    "date": ISODate("2024-01-03T00:00:00.000Z")
  },
  {
    "_id": ObjectId("61878d1c519ad9785689e*"),
    "id": 45053,
    "funded_amount": 400,
    "loan_amount": 400,
    "activity": "Food",
    "sector": "Food",
    "use": "buying more ingredients for resale",
    "country_code": "KH",
    "country": "Cambodia",
    "region": "Kampong Speu province, Moun Ruessey district",
    "currency": "KHM",
    "partner": "Kampong Speu Provincial Government",
    "posted_time": ISODate("2024-01-06T08:53:10.000Z"),
    "disbursed_time": ISODate("2023-12-27T08:00:00.000Z"),
    "funded_time": ISODate("2024-01-04T17:32:44.000Z"),
    "term_in_months": 12,
    "lender_count": 15,
    "borrower_genders": "female, female",
    "repayment_interval": "monthly",
    "date": ISODate("2024-01-03T00:00:00.000Z")
  },
  {
    "_id": ObjectId("61878d1c519ad9785689e*"),
    "id": 45054,
    "funded_amount": 1000,
    "loan_amount": 1000,
    "activity": "Food",
    "sector": "Food",
    "use": "to buy vegetables and fish to resell.",
    "country_code": "KH",
    "country": "Cambodia",
    "region": "Kampong Speu province, Moun Ruessey district",
    "currency": "KHM",
    "partner": "Kampong Speu Provincial Government"
  }
}

```

Read - Operation 6		
DB	Command	Execution Time(ms)
SQL	<pre> 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'; </pre>	24
NoSQL	<pre> db.all_kiva.find({activity:/.*Food.*/, loan_theme_type:{\$ne:"General"} }) </pre>	305

Table 6. Performance and Commands for the fifth read operation

f) Read Operation - 7

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

```

type = "kiva"
host = "atlas-salle1-shard-0 [primary] kiva"
db.all_kiva.find({activity:{$regex: "#Food.*"}, loan_theme_type:{'$eq': 'General'}}).explain("executionStats")
{
  queryPlanner: {
    planVersion: 3,
    namespace: 'kiva.all_kiva',
    indexFilterSet: false,
    parseTimeNs: 0,
    "sort": {},
    "limit": 0,
    "activity": [
      { '$match': { 'activity': { '$regex': '#Food.*' } },
        'loan_theme_type': { '$not': { '$eq': 'General' } }
      }
    ],
    winningPlan: {
      stage: 'COLLSCAN',
      filter: {
        '$and': [
          { 'activity': { '$regex': '#Food.*' } },
          { 'loan_theme_type': { '$not': { '$eq': 'General' } } }
        ]
      },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 232,
    executionTimeMillis: 305,
    totalKeysExamined: 1,
    totalDocExamined: 807542,
    executionStages: {
      stage: 'COLLSCAN',
      filter: {
        '$and': [
          { 'activity': { '$regex': '#Food.*' } },
          { 'loan_theme_type': { '$not': { '$eq': 'General' } } }
        ]
      },
      nReturned: 232,
      executionTimeMillisEstimate: 37,
      works: 807544,
      advance: 232,
      needTime: 807311,
      needYield: 0,
      restoreState: 807,
      isEOF: 1,
      direction: 'forward',
      docSizeExamined: 807462
    }
  },
  serverInfo: {
    host: 'data225-shard-00-01.Sqdp4.mongodb.net',
    port: 20171,
    version: '4.4.19',
    gitversion: '68971da1ef93a35a9f62bf4788a81713defe88c',
    ok: 1,
    $clusterTime: {
      clusterTime: Timestamp( t: 1636349295, i: 5 ),
      signature: {
        hash: Binary(Buffer.from('17321cae08fb0f174c8afff79d25d8e41ca638de5c'), 'hex'),
        keyid: Long('6594586891324a89738')
      }
    },
    operationTime: Timestamp( t: 1636349295, i: 5 )
  }
}
atlas-salle1-shard-0 [primary] kiva

```

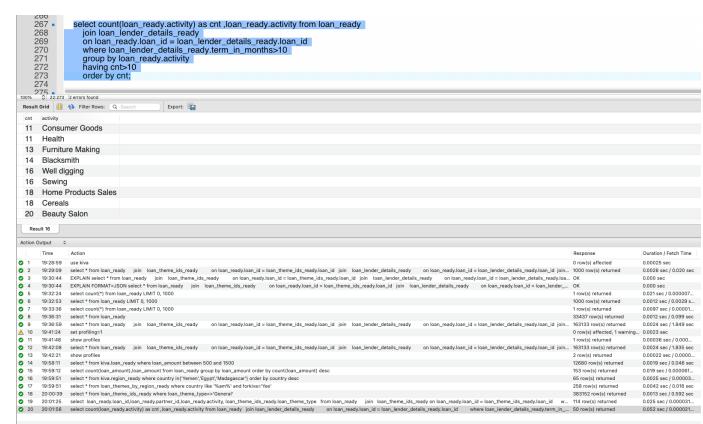


Fig.12.a # of times each activity gets funded with terms of bigger 10 months
and if they are funded more than 10 times

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

```

Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.aggregate([{$match: {term_in_months: {$gt:10}}}, {$group: {_id:"$activity", cnt:{$sum:1}}}, {$sort:{'cnt':-1}}, {$match:{'cnt:{$gt:10}}}], {allowDiskUse: true})
{
  "_id": "Hunting", "cnt": 11 },
  "_id": "Consumer Goods", "cnt": 11 },
  "_id": "Fuel/Firewood", "cnt": 11 },
  "_id": "Transportation", "cnt": 13 },
  "_id": "Blacksmith", "cnt": 13 },
  "_id": "Well digging", "cnt": 16 },
  "_id": "Sewing", "cnt": 16 },
  "_id": "Food Processing", "cnt": 18 },
  "_id": "Cereals", "cnt": 18 },
  "_id": "Beef/Bacon", "cnt": 20 },
  {"$match": {"loan_theme_type": "School costs", "cnt": 21}},
  {"_id": "Crafts", "cnt": 21 },
  {"_id": "Taxi", "cnt": 26 },
  {"_id": "Business Services", "cnt": 26 },
  {"_id": "Food Stall", "cnt": 27 },
  {"_id": "Clothing Sales", "cnt": 29 },
  {"_id": "Weaving", "cnt": 30 },
  {"_id": "Transportation", "cnt": 32 },
  {"_id": "Property", "cnt": 36 },
  {"_id": "Wedding Expenses", "cnt": 36 }

Type "it" for more
Atlas atlas-Salle7-shard-0 [primary] kiva> it
{
  "_id": "Land Rental", "cnt": 38 },
  {"_id": "Services", "cnt": 47 },
  {"_id": "Tailoring", "cnt": 47 },
  {"_id": "Construction Supplies", "cnt": 69 },
  {"_id": "Beverages", "cnt": 73 },
  {"_id": "Clothing", "cnt": 72 },
  {"_id": "Retail", "cnt": 74 },
  {"_id": "Livestock", "cnt": 76 },
  {"_id": "Meat/Poultry", "cnt": 76 },
  {"_id": "Weaving", "cnt": 90 },
  {"_id": "Poultry", "cnt": 91 },
  {"_id": "Food Processing", "cnt": 92 },
  {"_id": "Food", "cnt": 92 },
  {"_id": "Personal Expenses", "cnt": 99 },
  {"_id": "Farming", "cnt": 100 },
  {"_id": "Fruits & Vegetables", "cnt": 134 },
  {"_id": "Animal Sales", "cnt": 136 },
  {"_id": "Motorcycle Transport", "cnt": 146 },
  {"_id": "Farming", "cnt": 150 },
  {"_id": "Farm Supplies", "cnt": 269 }

Type "it" for more
Atlas atlas-Salle7-shard-0 [primary] kiva> it
{
  {"_id": "Cattle", "cnt": 314 },
  {"_id": "Grocery Store", "cnt": 340 },
  {"_id": "Agriculture", "cnt": 386 },
  {"_id": "Clothing", "cnt": 400 },
  {"_id": "Vehicle", "cnt": 644 },
  {"_id": "Home Energy", "cnt": 943 },
  {"_id": "Business Services", "cnt": 1012 },
  {"_id": "Higher education costs", "cnt": 1264 },
  {"_id": "Personal Housing Expenses", "cnt": 1936 },
  {"_id": "Farming", "cnt": 7638 }

Atlas atlas-Salle7-shard-0 [primary] kiva>

```

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

```

yasamanemami -- mongosh mongo:db:src:///data225.5q@p4.mongodb.net/myFirstDatabase -- myFirstDatabase TMDFDIR=/var/folders/rjz/specs
Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.aggregate([{$match: {term_in_months: {$gt:10}}}, {$group: {_id:"$activity", cnt:{$sum:1}}}, {$sort:{'cnt':-1}}, {$match:{'cnt:{$gt:10}}}], {allowDiskUse: true})
{
  "stages": [
    {
      "bson": {
        "queryPlanner": {
          "plannerVersion": 1,
          "namespace": "kiva.all_kiva",
          "indexFilterSet": false,
          "parsedQuery": { "term_in_months": { "$gt": 10 } },
          "queryHash": "A22EE0AB",
          "planCacheKey": "A22EE0AB",
          "winningPlan": {
            "stage": "PROJECTION_SIMPLE",
            "transformby": { "activity": { "$id": 0 } },
            "inputStage": {
              "stage": "COLLSCAN",
              "filter": { "term_in_months": { "$gt": 10 } },
              "direction": "forward"
            }
          },
          "rejectedPlans": []
        }
      }
    },
    {
      "executionStats": {
        "executionSuccess": true,
        "nReturned": 17684,
        "executionTimeMs": 347,
        "totalKeysetsExamined": 0,
        "totalDocsExamined": 807542,
        "executionTimeMillisEstimate": 72,
        "nReturned": 17684,
        "executionTimeMsEstimate": 81,
        "works": 807544,
        "advanced": 17684,
        "needToRead": 769059,
        "needToFetch": 0,
        "saveState": 888,
        "restoreState": 888,
        "isEOF": 1,
        "transformby": { "activity": { "$id": 0 } },
        "inputStage": {
          "stage": "COLLSCAN",
          "filter": { "term_in_months": { "$gt": 10 } },
          "direction": "forward",
          "executionTimeMillisEstimate": 72,
          "works": 807544,
          "advanced": 17684,
          "needToRead": 769059,
          "needToFetch": 0,
          "saveState": 888,
          "restoreState": 888,
          "isEOF": 1,
          "direction": "forward",
          "docsExamined": 807542
        }
      }
    },
    {
      "nReturned": Long("17684"),
      "executionTimeMillisEstimate": Long("341")
    },
    {
      "group": { "_id": "$activity", "cnt": { "$sum": { '$const': 1 } } },
      "nReturned": Long("112"),
      "executionTimeMillisEstimate": Long("343")
    },
    {
      "match": { "cnt": { "$gt": 10 } },
      "nReturned": Long("56"),
      "executionTimeMillisEstimate": Long("343")
    },
    {
      "sort": { "sortKey": { "cnt": 1 } },
      "nReturned": Long("68"),
      "executionTimeMillisEstimate": Long("343")
    }
  ],
  "serverInfo": {
    "host": "data225-shard-00-01.5q@p4.mongodb.net",
    "port": 27017,
    "version": "5.0.10",
    "gitVersion": "f88971datef93435a9762bf4708a81713defe88c",
    "ok": 1,
    "$clusterTime": {
      "clusterTime": Timestamp({ t: 1636352470, i: 4 }),
      "signature": {
        "hash": BinaryBuffer.from("7b8abc9f94580f2b4d3db71baaca7e02f26763c", "hex"),
        "keyId": Long("699a58e891324a8738")
      }
    },
    "operationTime": Timestamp({ t: 1636352470, i: 4 })
  }
}

```

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(milliseconds)
SQL	<pre> 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; </pre>	52
NoSQL	<pre> db.all_kiva.aggregate([{\$match: {term_in_months: {\$gt:10}}}, {\$group: {_id: "\$activity", cnt: {\$sum:1}}}, {\$sort:{'cnt':1}}, {\$match:{'cnt:{\$gt:10}}}], {allowDiskUse: true}) </pre>	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.

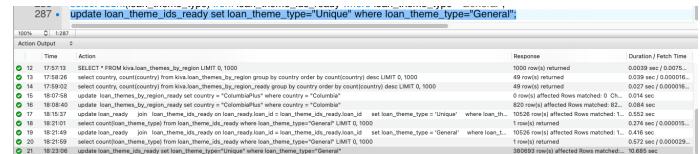


Fig. 13.a Update on Mysql

```

Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.updateMany({loan_theme_type:"General"}, {$set:{loan_theme_type:"Unique"}}, {allowDiskUse: true})
{
  "acknowledged": true,
  "insertedCount": 0,
  "matchedCount": 384876,
  "modifiedCount": 384876,
  "upsertedCount": 0
}

```

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

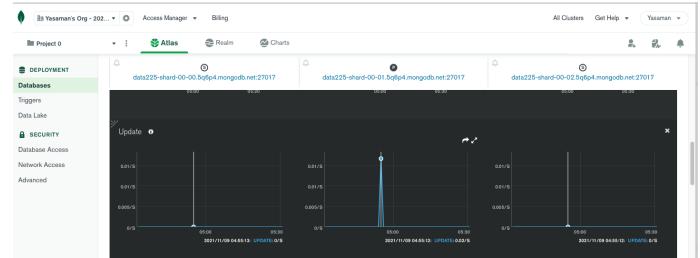


Fig. 13.c Update Performance on MongoDB

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: e: "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.

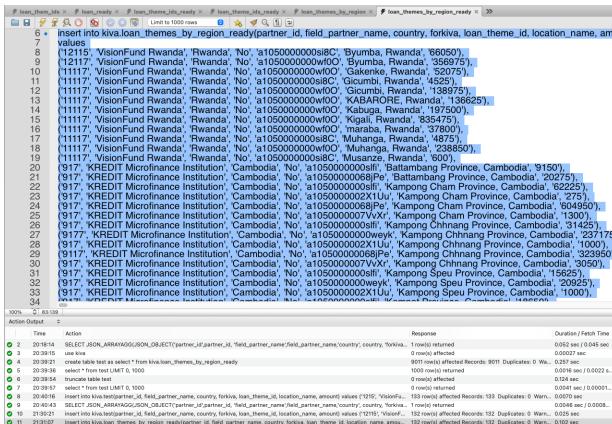


Fig. 14.a Insert 133 records to MySQL



Fig. 14.b Insert 133 documents to all_kiva collection

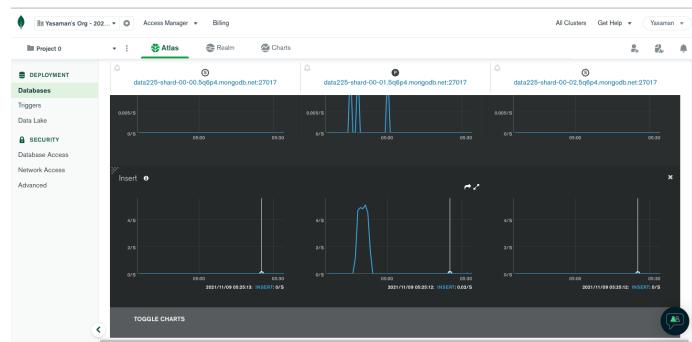


Fig. 14.c Performance of Insert on MongoDB

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", "partner_id": 1215, "loan_theme_id": "a1050000000si8C", "location_name": "Byumba, Rwanda", "field_partner_name": "VisionFund Rwanda", "region": "Battambang Province, Cambodia", "partner_genders": "No", "activity": "61", "repayment_interval": "male", "borrower_genders": "2014-01-02", "region": "Phnom Penh", "country": "Cambodia" }, ... , 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.

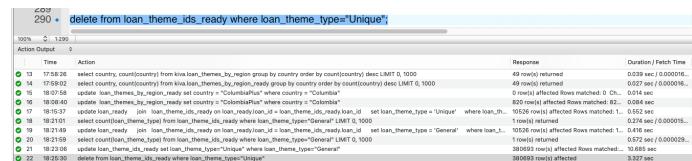


Fig. 15.a Delete records based on condition on MySQL

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

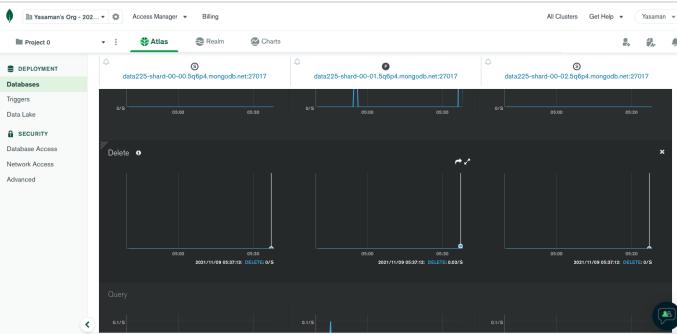


Fig. 15.c Performance of delete documents on MongoDB

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.

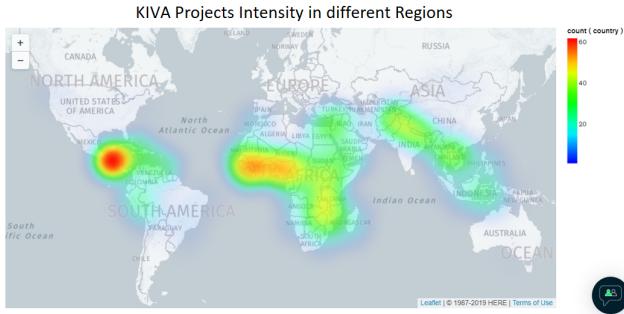


Fig. 16 Heat Map of Distribution of Crowdfunded Projects

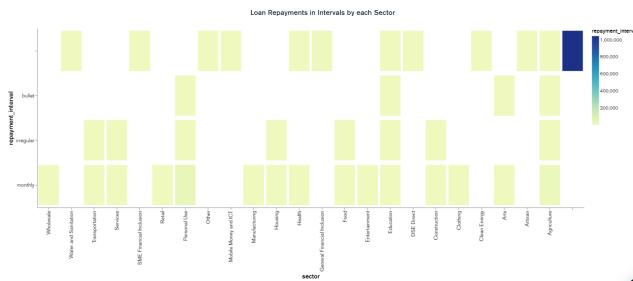


Fig. 17 Distribution of loan repayment interval over various sectors

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.

disadvantages. The selection of NoSQL for this project comes with a compromise on some of the features MySQL provides that MongoDB cannot.

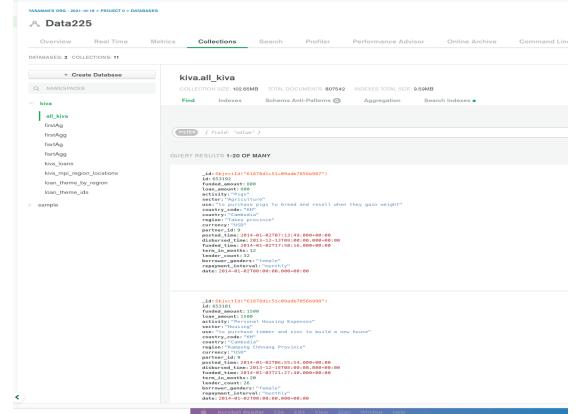


Fig. 18 Collection in MongoDB - 'all_kiva'

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-afc95541741>.
- [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@jsu.edu

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

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

Yasaman Emami
Department of Applied Data Science
San Jose State University
San Jose, United States
yasaman.emami@jsu.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—*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 [1]: import pandas as pd
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

In [2]: df = pd.read_csv('kiva_loans.csv')
data = pandas.read_csv('kiva_loans.csv', encoding='utf-8', quotechar='"', delimiter=',')
df.info()

In [3]: df.describe()

In [4]: null = df.isnull().sum().sort_values(ascending = False).reset_index()
null.columns = ['Column', 'Frequency']

In [5]: ## temp column consists of 10000 null values so remove the column.
df.drop('temp', axis=1, inplace=True)

In [6]: ## this column null values are replaced with the mode
df['funded_time'].mode()

In [7]: df['borrower_genders'].mode()

In [8]: df['funded_time'].fillna(df['funded_time'].mode(), inplace = True)
df['borrower_genders'].fillna(df['borrower_genders'].mode(), inplace = True)

In [9]: df.dropna(inplace = True)

In [10]: df.info()

In [11]: 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.).

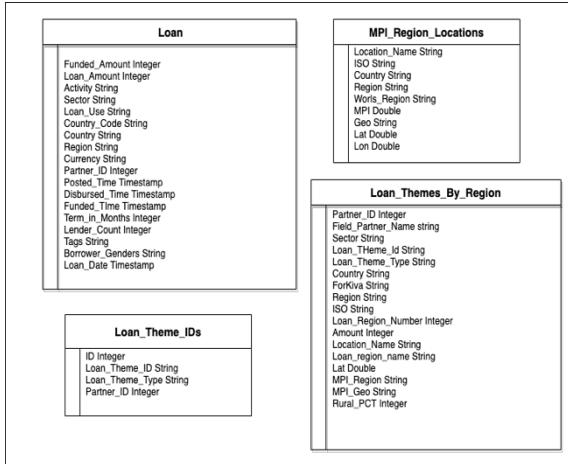


Fig. 2 Data Model

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

YASAMAN'S ORG - 2021-10-19 > PROJECT 0 > DATABASES

Data225

- Overview
- Real Time
- Metrics
- Collections**
- Search
- Profiler
- Performance Advisor
- Online Archive
- Command Line

DATABASES: 2 COLLECTIONS: 11

+ Create Database

NAMESPACES

kiva

- all_kiva
- firstAgg
- firstAgg
- firstAgg
- firstAgg
- firstAgg
- kiva_loans
- kiva_mpi_region_locations
- loan_theme_by_region
- loan_theme_ids
- sample

kiva.kiva_mpi_region_locations

COLLECTION SIZE: 165.1KB TOTAL DOCUMENTS: 892 INDEXES TOTAL SIZE: 24KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

FILTER { field: 'value' }

QUERY RESULTS 1-20 OF MANY

```

_id: ObjectID("61a772d151cd99db7840bc38")
locationName:"Babushtan, Afghanistan"
ISO:"AF"
country:"Afghanistan"
region:"Babushtan"
worldRegion:"South Asia"
lat: 35.734725
lon: 63.781993

```



```

_id: ObjectID("61a772d151cd99db7840bc31")
locationName:"Badghis, Afghanistan"
ISO:"AF"
country:"Afghanistan"
region:"Badghis"
worldRegion:"South Asia"
lat: 35.167139
lon: 63.769384

```



```

_id: ObjectID("61a772d151cd99db7840bc32")
locationName:"Baghlan, Afghanistan"
ISO:"AF"
country:"Afghanistan"
region:"Baghlan"
worldRegion:"South Asia"
lat: 35.107139
lon: 69.287753

```



```

_id: ObjectID("61a772d151cd99db7840bc33")

```

Fig.6 kiva_mpi_region_locations

YASAMAN'S ORG - 2021-10-19 > PROJECT 0 > DATABASES

Data225

- Overview
- Real Time
- Metrics
- Collections**
- Search
- Profiler
- Performance Advisor
- Online Archive
- Command Line

DATABASES: 2 COLLECTIONS: 11

+ Create Database

NAMESPACES

kiva

- all_kiva
- firstAgg
- firstAgg
- firstAgg
- firstAgg
- firstAgg
- kiva_loans
- kiva_mpi_region_locations
- loan_theme_by_region
- loan_theme_ids
- sample

kiva.all_kiva

COLLECTION SIZE: 102.69MB TOTAL DOCUMENTS: 807542 INDEXES TOTAL SIZE: 9.59MB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

FILTER { field: 'value' }

QUERY RESULTS 1-20 OF MANY

```

_id: ObjectID("61a772d151cd99db7856993")
id: 653192
funded_amount: 100
loan_amount: 100
activity: "Pigs"
sector: "Agriculture"
use: "to purchase pigs to breed and resell when they gain weight"
country_code: "AF"
country_name: "Afghanistan"
region: "Takhar province"
partner: 146
posted_time: 2014-01-07T01:13:09.000+00:00
disbursed_time: 2013-12-17T01:00.00-00:00
funded_time: 2014-01-07T15:58:16.000+00:00
term_in_months: 12
lender_count: 1
borrower_genders: "Female"
repayment_start_time: "2014-01-07T00:00:00+00:00"
date: 2014-01-07T00:00:00.000+00:00

```



```

_id: ObjectID("61a772d151cd99db7856998")
id: 653181
funded_amount: 1500
loan_amount: 1500
activity: "Personal Housing Expenses"
sector: "Housing"
use: "to buy timber and zinc to build a new house"
country_code: "AF"
country_name: "Afghanistan"
region: "Kangar Chahar Province"
current: 100
partner: 146
posted_time: 2014-01-07T01:55:54.000+00:00
disbursed_time: 2013-12-17T01:00.00-00:00
funded_time: 2014-01-07T15:58:16.000+00:00
term_in_months: 20
lender_count: 1
borrower_genders: "Female"
repayment_start_time: "2014-01-07T00:00:00+00:00"
date: 2014-01-07T00:00:00.000+00:00

```

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.

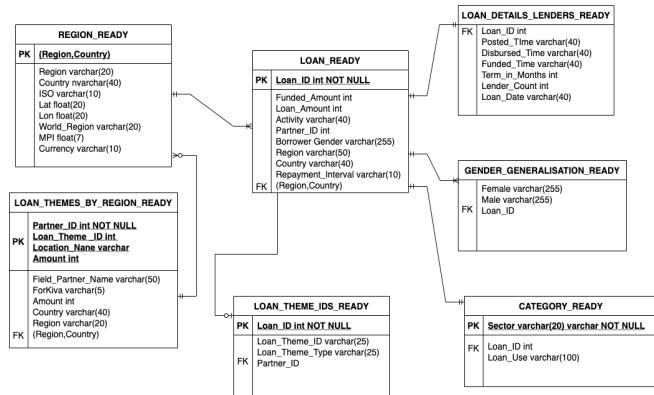


Fig. 8 Normalized Data (Project-1)

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

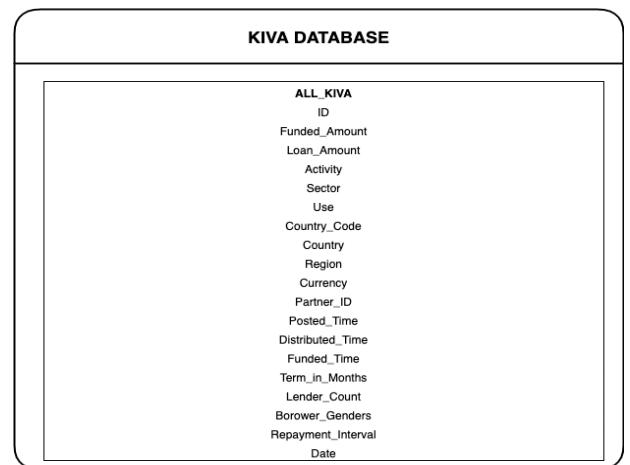


Fig. 9 Document Structure of Kiva- MongoDB

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’.

```

[Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.find()
{
  _id: ObjectId("61878d1c51c09adb7856b987"),
  id: 65317,
  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: 65318,
  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: 65319,
  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"),
  term_in_months: 22,
  lender_count: 22,
  borrower_genders: 'female',
  repayment_interval: 'monthly',
  date: ISODate("2014-01-02T00:00:00.000Z")
},
{
  _id: ObjectId("61878d1c51c09adb7856b98a"),
  id: 65316,
  funded_amount: 600,
  loan_amount: 600,
  activity: 'Cattle',
  sector: 'Agriculture',
  use: 'To buy a calf to raise. ',
  country_code: 'KH',
  country: 'Cambodia',
  region: 'Phnom Penh',
  currency: 'USD',
  partner_id: 60,
  posted_time: ISODate("2014-01-02T06:18:15.000Z"),
  disbursed_time: ISODate("2013-12-06T08:00:00.000Z"),
  funded_time: ISODate("2014-01-02T15:12:10.000Z"),
  term_in_months: 22,
  lender_count: 22,
  borrower_genders: 'female',
  repayment_interval: 'monthly',
  date: ISODate("2014-01-02T00:00:00.000Z")
},
{
  _id: ObjectId("61878d1c51c09adb7856b98b"),
  id: 65315,
  funded_amount: 3000,
  loan_amount: 3000,
  activity: 'Vehicle',
  sector: 'Personal Use',
  use: 'To buy a motorbike for family transportation. ',
  country_code: 'KH',
  country: 'Cambodia',
  region: 'Phnom Penh',
  currency: 'USD',
  partner_id: 61,
  posted_time: ISODate("2014-01-02T07:05:02.000Z"),
  disbursed_time: ISODate("2013-12-13T08:00:00.000Z"),
  funded_time: ISODate("2014-01-10T03:22:29.000Z"),
  term_in_months: 22,
  lender_count: 33,
  borrower_genders: 'female',
  repayment_interval: 'monthly',
  date: ISODate("2014-01-02T00:00:00.000Z")
},
{
  _id: null, count: 774185 },
  { _id: 125, count: 212 },
  { _id: 289, count: 3598 },
  { _id: 189, count: 3453 },
  { _id: 175, count: 3351 },
  { _id: 187, count: 1873 },
  { _id: 580, count: 1858 },
  { _id: 256, count: 1839 },
  { _id: 159, count: 1533 },
  { _id: 1680, count: 799 },
  { _id: 760, count: 527 },
  { _id: 121, count: 527 },
  { _id: 480, count: 519 },
  { _id: 656, count: 510 },
  { _id: 120, count: 479 },
  { _id: 158, count: 463 },
  { _id: 625, count: 455 },
  { _id: 100, count: 443 },
  { _id: 488, count: 393 },
  { _id: 525, count: 364 }
]

```

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 MongoShel 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.

```

[Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.find({loan_amount:{$gt:500, $lt:1500}})
{
  _id: ObjectId("61878d1c51c09adb7856b987"),
  id: 65317,
  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: 65318,
  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: 65319,
  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"),
  term_in_months: 22,
  lender_count: 22,
  borrower_genders: 'female',
  repayment_interval: 'monthly',
  date: ISODate("2014-01-02T00:00:00.000Z")
},
{
  _id: ObjectId("61878d1c51c09adb7856b98a"),
  id: 65316,
  funded_amount: 600,
  loan_amount: 600,
  activity: 'Cattle',
  sector: 'Agriculture',
  use: 'To buy a calf to raise. ',
  country_code: 'KH',
  country: 'Cambodia',
  region: 'Phnom Penh',
  currency: 'USD',
  partner_id: 60,
  posted_time: ISODate("2014-01-02T06:18:15.000Z"),
  disbursed_time: ISODate("2013-12-06T08:00:00.000Z"),
  funded_time: ISODate("2014-01-02T15:12:10.000Z"),
  term_in_months: 22,
  lender_count: 22,
  borrower_genders: 'female',
  repayment_interval: 'monthly',
  date: ISODate("2014-01-02T00:00:00.000Z")
},
{
  _id: ObjectId("61878d1c51c09adb7856b98b"),
  id: 65315,
  funded_amount: 3000,
  loan_amount: 3000,
  activity: 'Vehicle',
  sector: 'Personal Use',
  use: 'To buy a motorbike for family transportation. ',
  country_code: 'KH',
  country: 'Cambodia',
  region: 'Phnom Penh',
  currency: 'USD',
  partner_id: 61,
  posted_time: ISODate("2014-01-02T07:05:02.000Z"),
  disbursed_time: ISODate("2013-12-13T08:00:00.000Z"),
  funded_time: ISODate("2014-01-10T03:22:29.000Z"),
  term_in_months: 22,
  lender_count: 33,
  borrower_genders: 'female',
  repayment_interval: 'monthly',
  date: ISODate("2014-01-02T00:00:00.000Z")
},
{
  _id: null, count: 500 }
]

```

Fig. 11 loan_amount between 500\$,1500\$

```

[Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.aggregate([
  { $group: { _id: "$loan_amount", count:{$sum:1} } },
  { $sort:{'count':-1} }
])
{
  _id: null, count: 774185 },
  { _id: 125, count: 212 },
  { _id: 289, count: 3598 },
  { _id: 189, count: 3453 },
  { _id: 175, count: 3351 },
  { _id: 187, count: 1873 },
  { _id: 580, count: 1858 },
  { _id: 256, count: 1839 },
  { _id: 159, count: 1533 },
  { _id: 1680, count: 799 },
  { _id: 760, count: 527 },
  { _id: 121, count: 527 },
  { _id: 480, count: 519 },
  { _id: 656, count: 510 },
  { _id: 120, count: 479 },
  { _id: 158, count: 463 },
  { _id: 625, count: 455 },
  { _id: 100, count: 443 },
  { _id: 488, count: 393 },
  { _id: 525, count: 364 }
]
Type 'it' for more
Atlas atlas-Salle7-shard-0 [primary] kiva> it
{
  _id: 1280, count: 361 },
  { _id: 880, count: 349 },
  { _id: 275, count: 338 },
  { _id: 120, count: 333 },
  { _id: 1825, count: 277 },
  { _id: 456, count: 272 },
  { _id: 100, count: 271 },
  { _id: 875, count: 243 },
  { _id: 550, count: 224 },
  { _id: 386, count: 213 },
  { _id: 120, count: 203 },
  { _id: 1180, count: 207 },
  { _id: 1280, count: 195 },
  { _id: 120, count: 193 },
  { _id: 850, count: 189 },
  { _id: 1888, count: 167 },
  { _id: 120, count: 165 },
  { _id: 575, count: 149 },
  { _id: 1380, count: 148 },
  { _id: 1125, count: 148 }
]
Type 'it' for more
Atlas atlas-Salle7-shard-0 [primary] kiva> it
{
  _id: 725, count: 146 },
  { _id: 975, count: 146 },
  { _id: 120, count: 145 },
  { _id: 1475, count: 124 },
  { _id: 1380, count: 116 },
  { _id: 120, count: 115 },
  { _id: 1275, count: 108 },
  { _id: 675, count: 106 },
  { _id: 120, count: 105 },
  { _id: 1375, count: 99 },
  { _id: 475, count: 86 },
  { _id: 120, count: 85 },
  { _id: 1875, count: 83 },
  { _id: 1225, count: 77 },
  { _id: 1888, count: 77 },
  { _id: 120, count: 75 },
  { _id: 1975, count: 71 },
  { _id: 1525, count: 69 },
  { _id: 120, count: 68 }
]
Type 'it' for more
Atlas atlas-Salle7-shard-0 [primary] kiva> it

```

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”.

```
Type "list" for more
Atlas atlas-Sales1-shard0 [primary] kiva.$all_kiva.find({country:"+vn", $or: [ {forkiva:"Yes"} ]})

{
  _id: ObjectId("63f878d3f51c09ad078573fa"),
  sector: "General Financial Inclusion",
  country: "Cambodia",
  region: "Banteay Meanchey",
  partner_name: "Hathaa Kasekkr Limited (HKL)",
  location_name: "Banteay Meanchey, Cambodia",
  lon_them:type: "Green",
  forkiva: "Yes",
  lat: Decimal128(+13.467599),
  lon: Decimal128(+104.497589)
},
{
  _id: ObjectId("63f878d3f51c09ad078573fa"),
  sector: "General Financial Inclusion",
  country: "Cambodia",
  region: "Banteay Meanchey",
  partner_name: "Hathaa Kasekkr Limited (HKL)",
  location_name: "Banteay Meanchey, Cambodia",
  lon_them:type: "Green",
  forkiva: "Yes",
  lat: Decimal128(+13.467599),
  lon: Decimal128(+104.497589)
},
{
  _id: ObjectId("63f878d3f51c09ad078573fa"),
  sector: "General Financial Inclusion",
  country: "Cambodia",
  region: "Banteay Meanchey",
  partner_name: "Hathaa Kasekkr Limited (HKL)",
  location_name: "Banteay Meanchey, Cambodia",
  lon_them:type: "Green",
  forkiva: "Yes",
  lat: Decimal128(+13.467599),
  lon: Decimal128(+104.497589)
},
{
  _id: ObjectId("63f878d3f51c09ad078573fa"),
  sector: "General Financial Inclusion",
  country: "Cambodia",
  region: "Battambang",
  partner_name: "Hathaa Kasekkr Limited (HKL)",
  location_name: "Battambang, Cambodia",
  lon_them:type: "Green",
  forkiva: "Yes",
  lat: Decimal128(+13.46523),
  lon: Decimal128(+103.392285)
},
{
  _id: ObjectId("63f878d3f51c09ad078573fa"),
  sector: "General Financial Inclusion",
  country: "Cambodia",
  region: "Battambang",
  partner_name: "Hathaa Kasekkr Limited (HKL)",
  location_name: "Battambang, Cambodia",
  lon_them:type: "Green",
  forkiva: "Yes",
  lat: Decimal128(+13.46523),
  lon: Decimal128(+103.392285)
},
{
  _id: ObjectId("63f878d3f51c09ad078573fa"),
  sector: "General Financial Inclusion",
  country: "Cambodia",
  region: "Kampot Chhn",
  partner_name: "Hathaa Kasekkr Limited (HKL)",
```

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” .

```

Atlas atlas-Salut shard=0 [primary] kive> db.all_kiva.find({activity_id:{$in:[Food,+Food]}, loan_theme:type({$in:[General]}) })
{
  {
    _id: ObjectId("61e78d1c51c9a07b785698f8"),
    funded_amount: 200,
    loan_amount: 1000,
    activity: "Food Production/Sales",
    sector: "Food",
    user: "To buy a sugarcane Juicing machine.",
    country_code: "MH",
    currency: "INR",
    region: "Pune/Feni",
    current_loan: 1000,
    partner_id: 91,
    posted_time: ISODate("2014-05-02T18:10:00.000Z"),
    disbursed_time: ISODate("2013-12-23T18:00:00.000Z"),
    funded_time: ISODate("2014-05-02T18:11:03.000Z"),
    term_in_months: 12,
    lender_count: 1,
    borrower_gender: "female",
    repayment_interval: "monthly",
    date: ISODate("2014-05-02T00:00:00.000Z")
  },
  {
    _id: ObjectId("61e78d1c51c9a07b785699e9"),
    funded_amount: 1000,
    loan_amount: 1000,
    activity: "Food",
    sector: "Food",
    user: "To buy ingredients for making rice soup and Chinese noodle.",
    country_code: "MH",
    currency: "INR",
    region: "Battambang province, Mung Hussey district",
    current_loan: 1000,
    partner_id: 294,
    posted_time: ISODate("2014-05-04T18:00:00.000Z"),
    disbursed_time: ISODate("2013-12-06T18:00:00.000Z"),
    funded_time: ISODate("2014-05-04T21:52:17.000Z"),
    term_in_months: 12,
    lender_count: 26,
    borrower_gender: "female, female, female",
    repayment_interval: "monthly",
    date: ISODate("2014-05-03T00:00:00.000Z")
  },
  {
    _id: ObjectId("61e78d1c51c9a07b78569a68"),
    id: 45543,
    funded_amount: 400,
    loan_amount: 400,
    activity: "Food",
    sector: "Food",
    user: "To buy beans for resale",
    country_code: "MH",
    currency: "INR",
    region: "Cebu/Bohol",
    current_loan: 400,
    partner_id: 295,
    posted_time: ISODate("2014-05-03T08:00:00.000Z"),
    disbursed_time: ISODate("2013-12-05T08:00:00.000Z"),
    funded_time: ISODate("2014-05-04T17:15:00.000Z"),
    term_in_months: 12,
    lender_count: 1,
    borrower_gender: "female, female",
    repayment_interval: "monthly",
    date: ISODate("2014-05-03T00:00:00.000Z")
  },
  {
    _id: ObjectId("61e78d1c51c9a07b78569a99"),
    id: 45554,
    funded_amount: 1000,
    loan_amount: 1000,
    activity: "Food",
    sector: "Food",
    user: "To buy vegetables and fish to resell",
    country_code: "MH",
    currency: "INR",
    region: "Mumbai"
  }
}

```

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 NoSQLI kiva database.

```

        {
            "_id": "Fish_Selling", "cnt": 72 },
            {"_id": "Fruit_Selling", "cnt": 70 },
            {"_id": "Livestock", "cnt": 70 },
            {"_id": "Construction", "cnt": 77 },
            {"_id": "Weaving", "cnt": 90 },
            {"_id": "Food", "cnt": 91 },
            {"_id": "Food_Production/Sales", "cnt": 92 },
            {"_id": "Food*", "cnt": 92 },
            {"_id": "Food_Sales", "cnt": 99 },
            {"_id": "General_Store", "cnt": 114 },
            {"_id": "Fruits & Vegetables", "cnt": 134 },
            {"_id": "Animal_Sales", "cnt": 136 },
            {"_id": "Food_Sales", "cnt": 146 },
            {"_id": "Fishing", "cnt": 157 },
            {"_id": "Farm_Supplies", "cnt": 269 }
        }
    }

    type "it" for more
Atlas:atlas-SaleId=shard-0 [primary] kiva> it
{
    {
        "_id": "Grocery_Store", "cnt": 314 },
        {"_id": "Grocery_Store", "cnt": 340 },
        {"_id": "Agriculture", "cnt": 386 },
        {"_id": "Pigs", "cnt": 400 },
        {"_id": "Food_Sales", "cnt": 644 },
        {"_id": "Home_Energy", "cnt": 943 },
        {"_id": "Home_Applications", "cnt": 1812 },
        {"_id": "Food_Sales", "cnt": 1840 },
        {"_id": "Personal_Housing_Expenditure", "cnt": 1924 },
        {"_id": "Farming", "cnt": 768 }
    }
}

```

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-5allie7-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-5allie7-shard-0 [primary] kiva> db.all_kiva.deleteMany({loan_theme_type:"Unique"})
{
  acknowledged: true,
  deletedCount: 384876
}
Atlas atlas-5allie7-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.

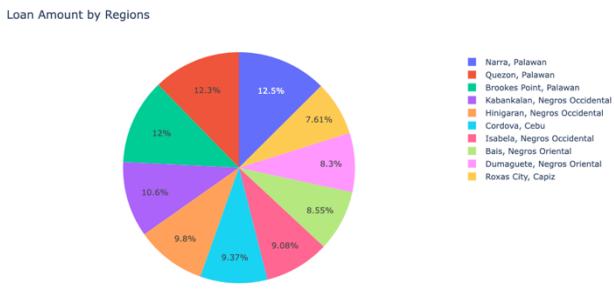


Fig. 20 Loan distribution pie chart

- 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.

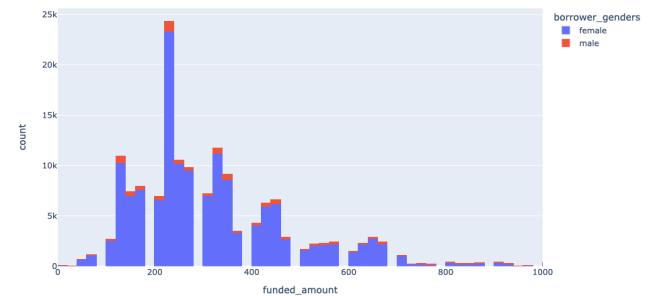


Fig. 21 Funded amounthistogram color-coded by gender

- 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.

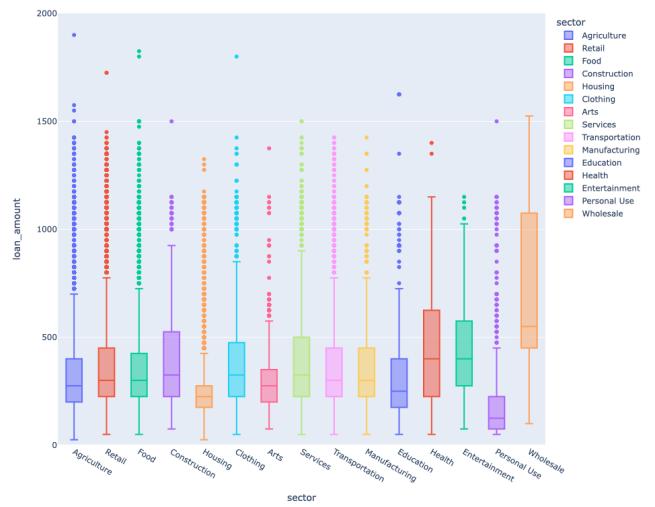


Fig. 22 Project sectors box plot

- 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.

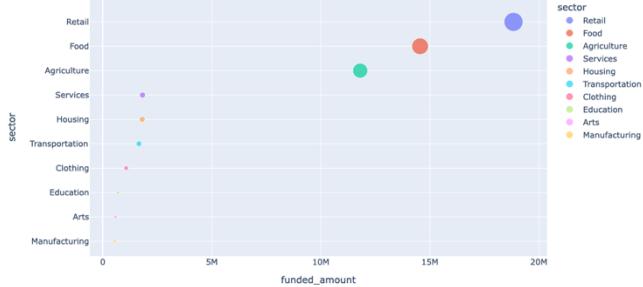


Fig. 23 Funding of projects in different sectors with respect to the amount of funding scatter plot

- 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.

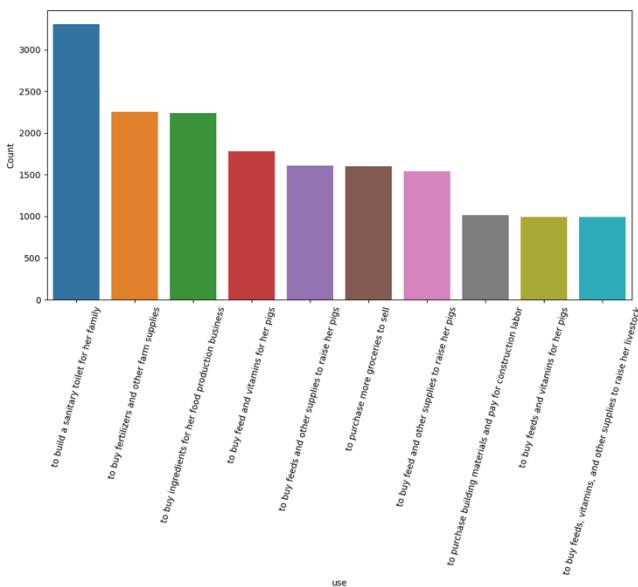


Fig. 24 Top 10 loan use bar chart

- 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.

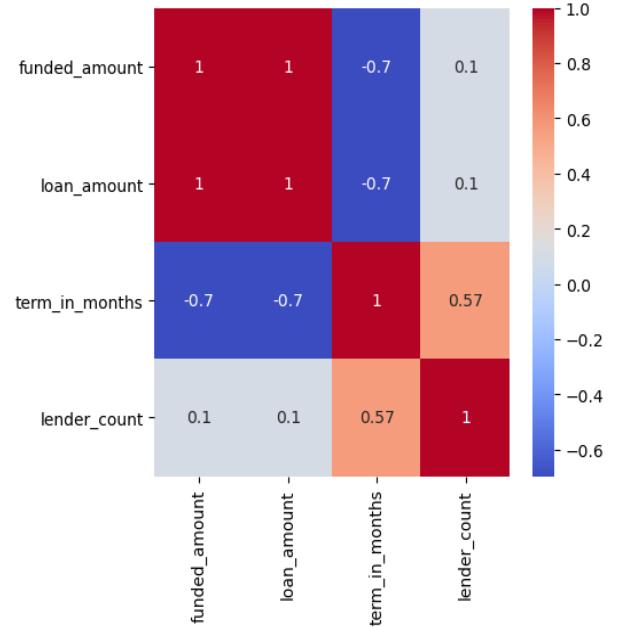


Fig. 25 funded amount, loan amount, the term of the loan(in months), and the lender count heatmap

- The following map shows the distribution of the crowdfunding 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

KIVA Projects Intensity in different Regions

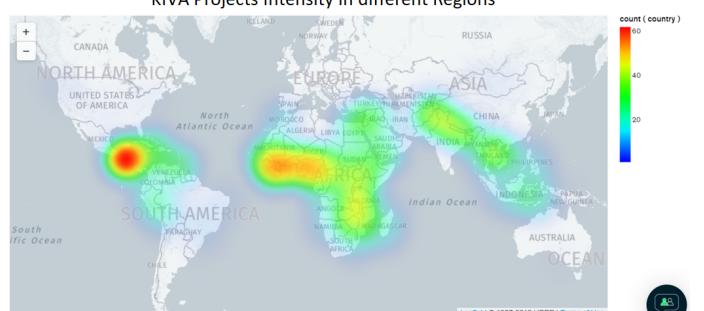


Fig. 27 Kiva project intensity in regions

- 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.

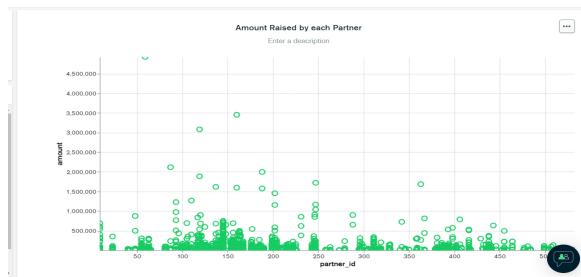


Fig. 29 Partners vs Projects

- 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.

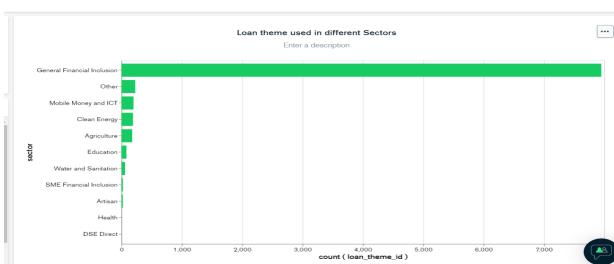


Fig. 30 Kiva loan themes by different sectors

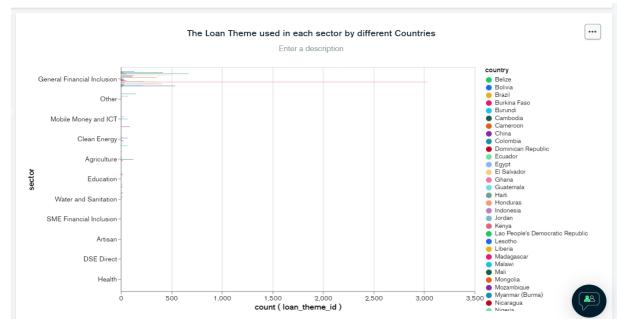


Fig. 31 Kiva loan themes by different countries

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.

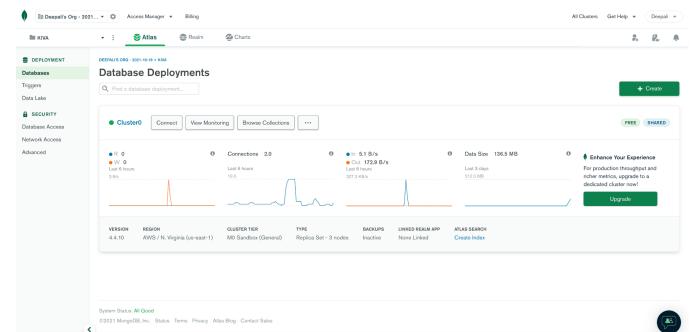


Fig. 32 MongoDB Atlas Connectivity-1

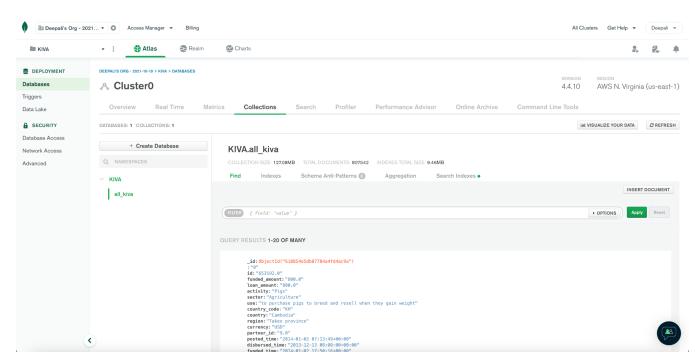


Fig. 33 MongoDB Atlas Connectivity-2

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.

```
  type "it" for more
  Alias alias-Sale7-shard-0 [primary] kiva> db.all_kiva.find({loan_amount:{'$gt':500, '$lt':1500}}).explain("executionStats")
{
  queryPlanner: {
    stage: "PRIMARY",
    keyPattern: {},
    namespace: "kiva.all_kiva",
    indexFilterSet: false,
    parsedQuery: {
      '$gt': 500,
      '$lt': 1500
    },
    queryHashObj: "54545454545454545454545454545454"
  },
  winningPlan: {
    stage: "COLLSCAN",
    filter: {},
    isEOF: false,
    documents: [
      { loan_amount: { '$lt': 1500 } },
      { loan_amount: { '$gt': 500 } }
    ],
    direction: "forward"
  },
  rejectedPlans: []
},
executionStats: {
  executionSuccess: true,
  nReturned: 10023,
  executionTimeMillis: 317,
  totalKeysExamined: 0,
  totalDocsExamined: 8,
  totalDocsSelected: 807542,
  executionPlan: {
    stage: "COLLSCAN",
    filter: {},
    isEOF: false,
    documents: [
      { loan_amount: { '$lt': 1500 } },
      { loan_amount: { '$gt': 500 } }
    ],
    direction: "forward"
  },
  nReturned: 10023,
  executionTimeMillisEstimate: 38,
  works: 807544,
  advanced: 10023,
  needMoreResults: 0,
  needYield: 0,
  saveState: 887,
  restoreState: 887,
  isEOF: false,
  direction: "forward",
  docsExamined: 807542
},
serverInfo: {
  host: "data225-shard-00-01.5qp64.mongodb.net",
  port: 27817,
  version: "4.4.8",
  gitVersion: "59d971daef93435a9f7e2bf47f88a81713def6e88c",
  type: "mongos"
},
ok: 1,
$clusterTime: {
  clusterTime: Timestamp({ t: 1636314302, i: 1 }),
  signature: "0000000000000000000000000000000000000000"
},
getVersion: "59d971daef93435a9f7e2bf47f88a81713def6e88c",
operationTime: Timestamp({ t: 1636314302, i: 1 })
}
Atlas alias-Sale7-shard-0 [primary] kiva> ||
```

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

```
Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.aggregate([ { $group: { '_id': '$loan_amount', 'count': { '$sum': { '$const': 1 } } } }, { $sort: { 'count': -1 } } ], { explain: 'executionStats' })
{
  "stages": [
    {
      "cursor": {
        "queryPlanner": {
          "plannerVersion": 1,
          "namespace": "kiva.all_kiva",
          "indexFilterSet": false,
          "parsedQuery": {}
        },
        "query": "db.all_kiva.find({})",
        "planCacheKey": "B0F8475C",
        "winningPlan": {
          "stage": "COLLSCAN",
          "transformby": { '_id': 1, '_id': 0 },
          "inputStage": { "stage": "COLLSCAN", "direction": "forward" }
        },
        "rejectedPlans": []
      },
      "executionStats": {
        "executionTimeMillis": true,
        "nReturned": 807542,
        "executionTimeMillisEstimate": 133,
        "works": 807542,
        "advanced": 807542,
        "needTime": 1,
        "needFieldId": 0,
        "saveState": 830,
        "restoreState": 830,
        "isEOF": 1,
        "transMod": { '_loan_amount': 1, '_id': 0 },
        "inputStage": {
          "stage": "COLLSCAN",
          "direction": "forward",
          "executionTimeMillisEstimate": 74,
          "works": 807544,
          "advanced": 807542,
          "needTime": 1,
          "needFieldId": 0,
          "saveState": 830,
          "restoreState": 830,
          "isEOF": 1,
          "direction": "forward",
          "docsExamined": 807542
        }
      }
    },
    {
      "nReturned": Long("807542"),
      "executionTimeMillisEstimate": Long("496")
    }
  ],
  "executionStats": {
    "nGrouped": 1,
    "nReturned": Long("1"),
    "executionTimeMillisEstimate": Long("678")
  },
  "collation": {
    "sortKey": { 'count': -1 },
    "nReturned": Long("1"),
    "executionTimeMillisEstimate": Long("678")
  }
},
"serverInfo": {
  "host": "data225-shard-00-01.5qpq4.mongodb.net",
  "port": 27017,
  "version": "4.4.10",
  "gitVersion": "58971daef93425a9ff62bf780a81713de6e88c"
},
"ok": 1,
"clusterTime": {
  "clusterTime": Timestamp( t: 1636311298, i: 1 ),
  "signature": {
    "hash": "5994f586991324497738",
    "keyId": Long("169864dd8d94686d4fe919bf37af287cfdb81383a", "hex")
  }
},
"operationTime": Timestamp( t: 1636311298, i: 9 )
}
Atlas atlas-Salle7-shard-0 [primary] kiva> ||
```

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.

```
Type 'it' for more
Atlas atlas-5Sale7-shard-0 [primary] kiva> db.all_kiva.find({country:{$regex:'.+am.+'}, forkiva:'Yes'}).explain("executionStats")
{
  queryPlanner: {
    plannerVersion: 1,
    namespace: 'db.all_kiva',
    indexFilterSet: false,
    parsedQuery: {
      '$and': [
        { forkiva: { '$eq': 'Yes' } },
        { country: { '$regex': '.+am.' } }
      ],
      '$or': [
        { forkiva: { '$eq': 'Yes' } },
        { country: { '$regex': '.+am.' } }
      ]
    },
    winningPlan: {
      stage: 'COLLSCAN',
      filter: {},
      '$and': [
        { forkiva: { '$eq': 'Yes' } },
        { country: { '$regex': '.+am.' } }
      ],
      '$or': [
        { forkiva: { '$eq': 'Yes' } },
        { country: { '$regex': '.+am.' } }
      ],
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 264,
    executionTimeMillis: 362,
    totalKeysExamined: 0,
    totalDocsExamined: 887542,
    executionStage: {
      stage: 'COLLSCAN',
      filter: {},
      '$and': [
        { forkiva: { '$eq': 'Yes' } },
        { country: { '$regex': '.+am.' } }
      ],
      '$or': [
        { forkiva: { '$eq': 'Yes' } },
        { country: { '$regex': '.+am.' } }
      ]
    },
    nReturned: 264,
    executionTimeMillisEstimate: 69,
    works: 887544,
    advanced: 264,
    needYield: 0,
    needKeyFromDisk: 279,
    needFieldId: 0,
    saveState: 887,
    restoreState: 887,
    isEOF: 1,
    direction: 'forward',
    docsExamined: 887542
  },
  serverInfo: {
    host: 'data25-shard-00-01.5qbq4.mongodb.net',
    port: 27017,
    version: '4.4.19',
    gitVersion: '58971d1aef93435a9ff62bf4708a81713defde88c'
  },
  ok: 1
}
clusterTime: {
  timestamp( t: 1636349484, i: 7 ),
  signature: {
    hash: BinaryBuffer.from('08d20983fe528bfad68c08579d2f7cc0ba9bdd', 'hex'),
    keyId: Long('199648689813224489738')
  }
},
operationTime: Timestamp( t: 1636349484, i: 7 )
Atlas atlas-5Sale7-shard-0 [primary] kiva> 
```

Fig. 36 Country like “.*am.*” statistics

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

```
Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.find({country: { $in: ["Yemen", "Egypt", "Madagascar"] }}).sort({country:-1}).explain("executionStats")
{
  queryPlanner: {
    plannerVersion: 1,
    namespace: "db.all_kiva",
    indexFilterSet: false,
    parsedQuery: { country: { '$in': [ 'Egypt', 'Madagascar', 'Yemen' ] } },
    winningPlan: {
      stage: "SORT",
      sortPatten: { country: { '$in': [ 'Egypt', 'Madagascar', 'Yemen' ] } },
      type: "simple",
      inputStage: {
        stage: "COLLSCAN",
        filter: { country: { '$in': [ 'Egypt', 'Madagascar', 'Yemen' ] } },
        direction: "forward"
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 37,
    executionTimeMillis: 348,
    totalKeyExamined: 6,
    totalDocExamined: 807542,
    executionStages: {
      stage: "SORT",
      nReturned: 37,
      executionTimeMillisEstimate: 54,
      works: 807602,
      advanced: 57,
      needTime: 7584,
      needield: 0,
      saveState: 807,
      restoreState: 807,
      isEOF: 1,
      sortPattern: { country: -1 },
      memLimit: 3588432,
      type: "simple",
      totalDataSizeSorted: 14821,
      usedMemory: 0,
      inputStage: {
        stage: "COLLSCAN",
        filter: { '$in': [ 'Egypt', 'Madagascar', 'Yemen' ] },
        executionTimeMillisEstimate: 51,
        works: 807486,
        advanced: 57,
        needTime: 807486,
        needield: 0,
        saveState: 807,
        restoreState: 807,
        isEOF: 1,
        direction: "forward",
        docsExamined: 807542
      }
    }
  },
  serverInfo: {
    host: "data225-shard-00-01.5q6p4.mongodb.net",
    port: 27017,
    version: "4.4.10",
    gitVersion: "58971da1ef93435a9f62bf4788a81713defe88c"
  },
  ok: 1,
  clusterTime: {
    clusterTime: Timestamp( t: 1636354413, i: 7 ),
    signature: {
      hash: Binary(Buffer.from("9212fe4770901ef64dc9520d52699c689a307cc6", "hex"), 0),
      keyId: Long("699486891324489738")
    }
  },
  operationTime: Timestamp( t: 1636354413, i: 7 )
}
-----
```

Fig. 37 Country specific query performance

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

```
type: "text" | more
Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.find({activity: { $not: { $eq: "General" } } }).explain("executionStats")
{
  queryPlanner: {
    plannerVersion: 1,
    namespace: "db.all_kiva",
    indexFilterSet: false,
    parsedQuery: { activity: { '$not': { '$eq': 'General' } } },
    winningPlan: {
      stage: "COLLSCAN",
      filter: { $not: { activity: { '$eq': 'Food' } }, loan_theme_type: { '$not': { $eq: 'General' } } },
      direction: "forward"
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 232,
    executionTimeMillis: 385,
    totalKeyExamined: 6,
    totalDocExamined: 807542,
    executionStages: {
      stage: "COLLSCAN",
      filter: {
        $and: [
          { activity: { '$not': { '$eq': 'Food' } } },
          { loan_theme_type: { '$not': { $eq: 'General' } } }
        ]
      },
      nReturned: 232,
      executionTimeMillisEstimate: 37,
      works: 807311,
      needTime: 232,
      saveState: 807,
      restoreState: 807,
      isEOF: 1,
      direction: "forward",
      docsExamined: 807542
    }
  },
  serverInfo: {
    host: "data225-shard-00-01.5q6p4.mongodb.net",
    port: 27017,
    version: "4.4.10",
    gitVersion: "58971da1ef93435a9f62bf4788a81713defe88c"
  },
  ok: 1,
  clusterTime: {
    clusterTime: Timestamp( t: 1636349925, i: 5 ),
    signature: {
      hash: Binary(Buffer.from("17321c4e78bf174c8aff979d25d8e41ca638de5c", "hex"), 0),
      keyId: Long("699486891324489738")
    }
  },
  operationTime: Timestamp( t: 1636349925, i: 5 )
}
Atlas atlas-Salle7-shard-0 [primary] kiva> 
```

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.

```
Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.aggregate([{$match: {term_in_months: { $gt: 10 } }}, {$group: { _id: "$activity", cnt: { $sum: 1 } }}, {$sort: {cnt: -1}}]).explain("executionStats")
{
  stages: [
    {
      "source": {
        "queryPlanner": {
          "plannerVersion": 1,
          "namespace": "db.all_kiva",
          "indexFilterSet": false,
          "parsedQuery": { term_in_months: { '$gt': 10 } },
          "winningPlan": {
            "stage": "PROJECTION_SIMPLE",
            "transformBy": { activity: 1 },
            "inputStage": {
              "stage": "COLLSCAN",
              "filter": { term_in_months: { '$gt': 10 } },
              "direction": "forward"
            }
          },
          "rejectedPlans: []"
        }
      },
      "executionStats: {
        "executionSuccess": true,
        "nReturned": 17684,
        "executionTimeMillis": 347,
        "totalKeyExamined: 6,
        "totalDocExamined: 807542,
        "executionStages: {
          stage: "PROJECTION_SIMPLE",
          nReturned: 17684,
          executionTimeMillisEstimate: 81,
          works: 807644,
          advanced: 17684,
          needTime: 799859,
          needield: 0,
          saveState: 808,
          restoreState: 808,
          isEOF: 1,
          direction: "forward",
          docsExamined: 807542
        }
      }
    },
    {
      "stage": "executionStats",
      "executionSuccess": true,
      "nReturned": 17684,
      "executionTimeMillis": 17684,
      "executionTimeMillisEstimate: 72,
      "totalKeyExamined: 347,
      "totalDocExamined: 807542,
      "executionStages: {
        stage: "COLLSCAN",
        filter: { term_in_months: { '$gt': 10 } },
        direction: "forward"
      }
    },
    {
      "stage": "executionStats",
      "executionSuccess": true,
      "nReturned": 17684,
      "executionTimeMillis": 341,
      "executionTimeMillisEstimate: 72,
      "totalKeyExamined: 347,
      "totalDocExamined: 807542,
      "executionStages: {
        stage: "COLLSCAN",
        filter: { _id: "$activity", cnt: { '$sum: 1 } } },
        nReturned: Long("112"),
        executionTimeMillisEstimate: Long("343")
      }
    },
    {
      "stage": "executionStats",
      "executionSuccess": true,
      "nReturned": Long("112"),
      "executionTimeMillis": Long("343"),
      "executionTimeMillisEstimate: Long("343")
    },
    {
      "stage": "executionStats",
      "executionSuccess": true,
      "nReturned": Long("112"),
      "executionTimeMillis": Long("343"),
      "executionTimeMillisEstimate: Long("343")
    }
  ],
  serverInfo: {
    host: "data225-shard-00-01.5q6p4.mongodb.net",
    port: 27017,
    version: "4.4.10",
    gitVersion: "58971da1ef93435a9f62bf4788a81713defe88c"
  },
  ok: 1,
  clusterTime: {
    clusterTime: Timestamp( t: 1636352470, i: 4 ),
    signature: {
      hash: Binary(Buffer.from("7bbabcf94a5b0f2b4d3db71baca7e0f2f26763c", "hex"), 0),
      keyId: Long("699486891324489738")
    }
  },
  operationTime: Timestamp( t: 1636352470, i: 4 )
}
-----
```

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.

Time	Action	Response	Duration / Fetch Time
12:17:26	SELECT * FROM kiva_loans WHERE _id = 1000	1000 rows returned	0.0039 sec / 0.0075...
12:17:26	select count(*) from kiva_loans WHERE _id >= 1000 group by country order by count(*) desc LIMIT 0, 1000	49 rows returned	0.039 sec / 0.0050...
12:17:26	update kiva_loans set ready = 1 where _id >= 1000 group by country order by count(*) desc LIMIT 0, 1000	0 rows affected Rows matched: 0 Ch..	0.0001 sec
16:18:07	update kiva_loans set ready = 0 where _id >= 1000 group by country order by count(*) desc LIMIT 0, 1000	0 rows affected Rows matched: 0 Ch..	0.014 sec
16:18:07	update kiva_loans set ready = 1 where _id >= 1000 group by country order by count(*) desc LIMIT 0, 1000	820 rows affected Rows matched: 820	0.084 sec
17:18:21	select count(*) from kiva_loans WHERE _id >= 1000 group by country order by count(*) desc LIMIT 0, 1000	1 rows returned	0.0001 sec
18:18:21	update kiva_loans set ready = 1 where _id >= 1000 group by country order by count(*) desc LIMIT 0, 1000	10526 rows affected Rows matched: 10526	0.416 sec
18:18:21	update kiva_loans set ready = 0 where _id >= 1000 group by country order by count(*) desc LIMIT 0, 1000	1 rows returned	0.0029 sec
18:18:21	update kiva_loans set ready = 1 where _id >= 1000 group by country order by count(*) desc LIMIT 0, 1000	384876 rows affected Rows matched: 384876	0.648 sec

Fig.40 Update on 380693records MySQL

```
Atlas atlas-Salle7-shard-0 [primary] kiva> db.all_kiva.updateMany({loan_theme_type: "General"}, { $set: {loan_theme_type: "Unique" } })
{
  acknowledged: true,
  insertedId: null,
  matcheCount: 384876,
  modifiedCount: 384876,
  upsertedCount: 0
}
-----
```

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.

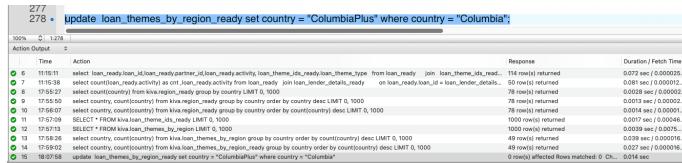


Fig. 38 Update Command MySQL



Fig. 39 Delete Command MySQL

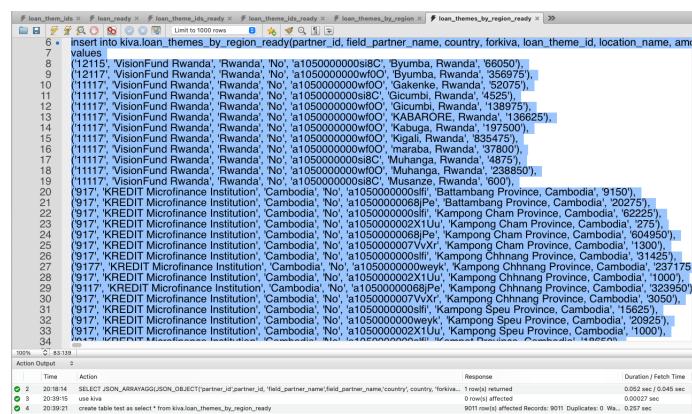


Fig. 40 Insert Command MySQL



Fig. 41 Update Command NoSQL

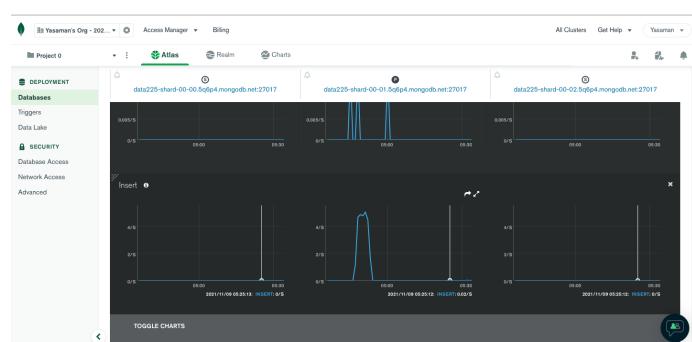


Fig. 42 Insert Command NoSQL

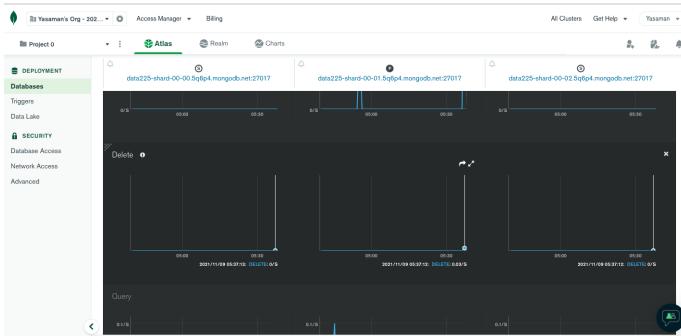


Fig. 43 Delete Command NoSQL

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@sjtu.edu

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

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

Yasaman Emami
Department of Applied Data Science
San Jose State University
San Jose, United States
yasaman.emami@sjtu.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 [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

In [1]: df = pd.read_csv('kiva_loans.csv')

In [1]: data = pandas.read_csv('kiva_loans.csv', encoding='utf-8', quotechar='"', delimiter=',')

In [1]: df.info()

In [1]: df.describe()

In [1]: null = df.isnull().sum().sort_values(ascending = False).reset_index()
null.columns = ['Column', 'Frequency']
null

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

In [1]: ## this column null values are replaced with the mode
df['funded_time'].mode()

In [1]: df['borrower_genders'].mode()

In [1]: df['funded_time'].fillna(df['funded_time'].mode(), inplace = True)

In [1]: df['borrower_genders'].fillna(df['borrower_genders'].mode(), inplace = True)

In [1]: df.dropna(inplace = True)

In [1]: df.isna().sum()

In [1]: df.to_csv('kiva_mpl_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:

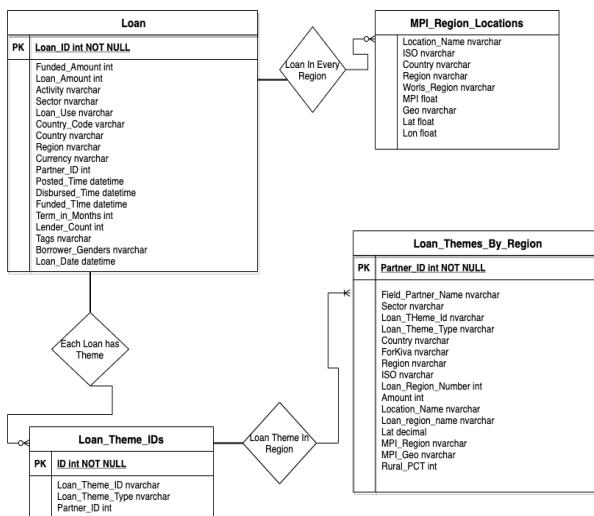


Fig. 2 ER Diagram (1)

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.

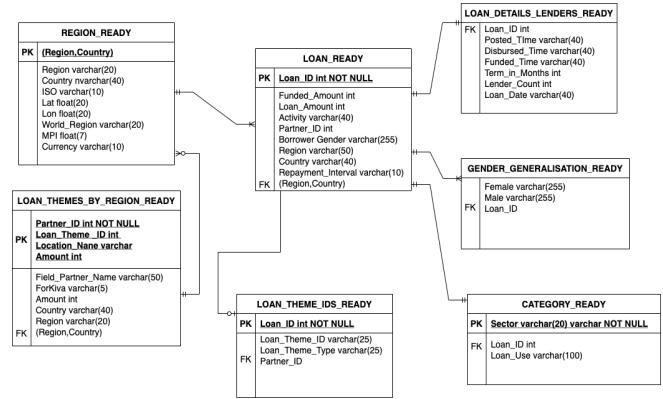


Fig. 3 ER Diagram (2)

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 foreign 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.

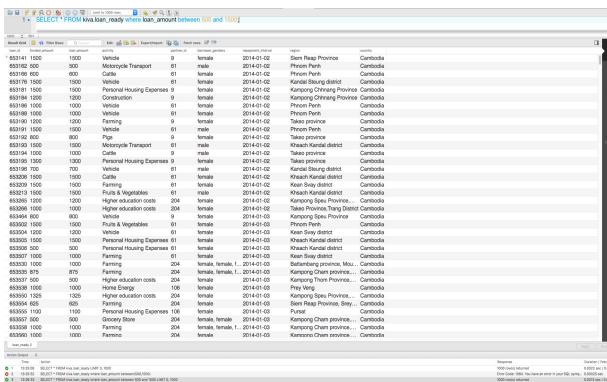


Fig 4. Loan Amount Between (500,1500)

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.

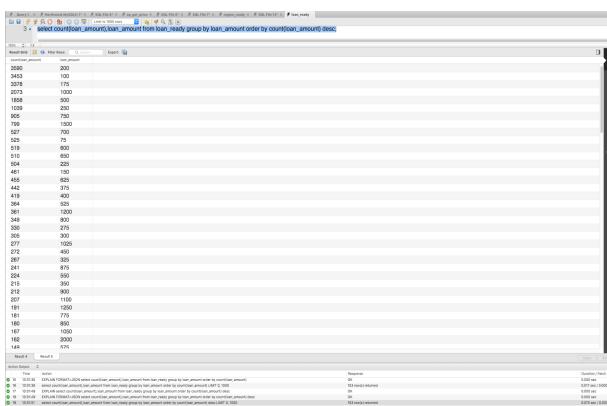


Fig 5. Count of Each Loan

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

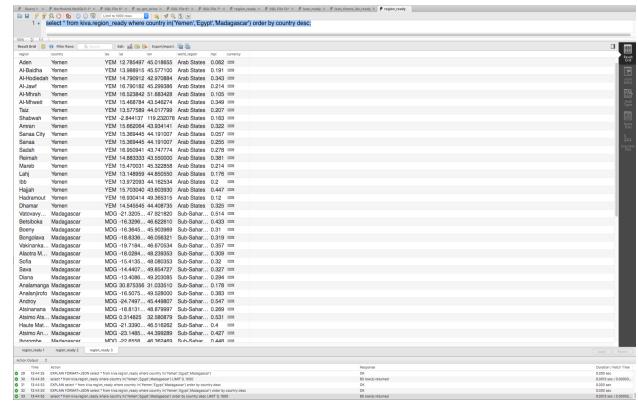


Fig 6. Members in Specific Region

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

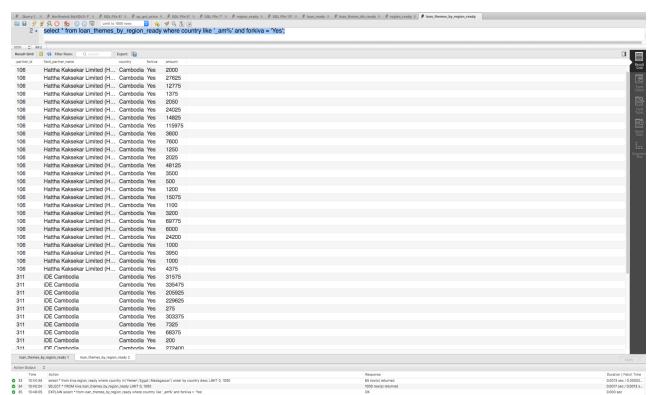


Fig7. Countries Like ‘_am%’

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.

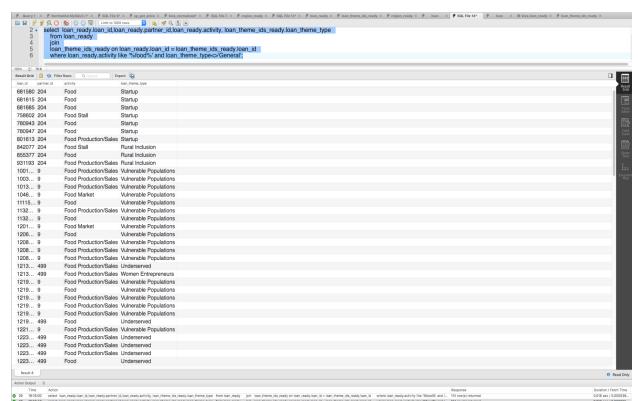


Fig8. Display records from join of two tables to show the loans in food industry specific type (exclude general types)

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.

```

7
8 • select count(loan_ready.activity) as cnt ,loan_ready.activity from loan_ready
9   join loan_lender_details_ready
10  on loan_ready.loan_id = loan_lender_details_ready.loan_id
11  where loan_lender_details_ready.term_in_months>10
12  group by loan_ready.activity
13  having cnt>10
14  order by cnt;
15

```

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 than 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.

```

192
193  DELIMITER $$*
194 • CREATE PROCEDURE load_data_into_loan_lender_details_ready()
195  BEGIN
196  INSERT INTO loan_lender_details_ready(posted_time, disbursed_time,
197  funded_time, term_in_months, lender_count, loan_date, loan_id)
198  select posted_time, disbursed_time, funded_time,
199  term_in_months, lender_count, loan_date, loan_id from loan;
200  END $$*
201  DELIMITER ;
202
203 • call load_data_into_loan_lender_details_ready();
204
205
206

```

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.

```

130
131  DELIMITER $$*
132 • CREATE PROCEDURE region_ready()
133  BEGIN
134  INSERT INTO region_ready (region,country,iso,lat,lon,world_region,
135  mpi) SELECT region,country,iso,lat,lon,world_region,
136  mpi FROM mpi_region_locations;
137  END $$*
138  DELIMITER ;
139
140 • call region_ready();
141
142

```

Action Output

Time	Action	Response	Duration / Fetch Time
92 20:27:29	CREATE PROCEDURE...	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.

```

130
131  DELIMITER $$*
132 • CREATE PROCEDURE loan_themes()
133  BEGIN
134  INSERT INTO loan_themes_by_region_ready (
135  partner_id,field_partner_name,country,forkiva,location_name,loan_theme_id,amount
136  ) SELECT partner_id,field_partner_name,country,forkiva,location_name ,
137  loan_theme_id,amount FROM loan_themes_by_region;
138  END $$*
139  DELIMITER ;
140

```

Action Output

Time	Action	Response	Duration / Fetch Time
96 20:40:10	CREATE PROCEDURE...	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 checked 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.

```

118  DELIMITER $$ 
119  CREATE TRIGGER checkCountry
120  BEFORE INSERT ON kiva.loan_ready
121  FOR EACH ROW
122  BEGIN
123  IF (loan.country = 'Crimea' or loan.country = 'Cuba' or
124  loan.country = 'Iran' or loan.country = 'Syria' or
125  loan.country = 'North Korea' or loan.country = 'Sudan') THEN
126  signal sqlstate '51000' set message_text = 'cannot insert record as the residents
127  END IF;
128  END;
129  $$
```

Action Output: 5 errors found

Time	Action
6 21:12:30	drop trigger if exists checkCountry
7 21:12:38	CREATE TRIGGER checkCountry BEFORE INSERT ON kiva.loan_ready FOR EACH ROW BEGIN IF (loan.country = 'Crimea' or loan.country =

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

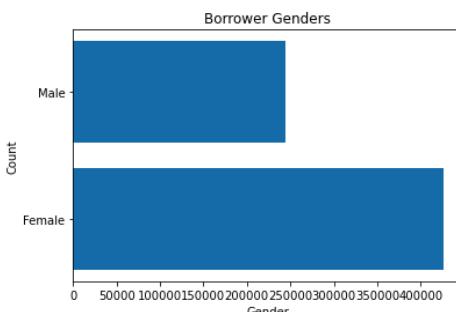


Fig. 14 Borrower gender Distribution

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

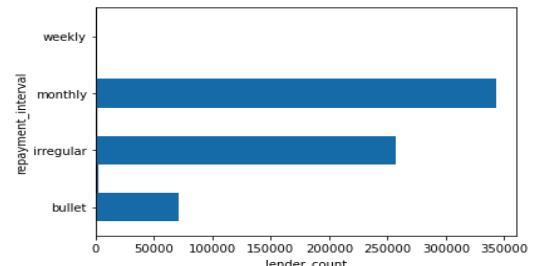


Fig. 15 Repayment Methods

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

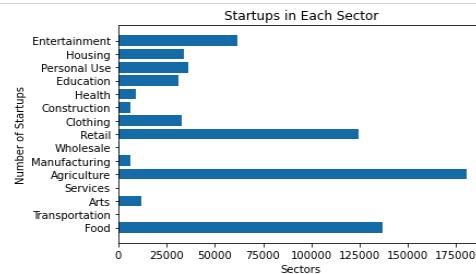


Fig. 16 Project Sectors

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

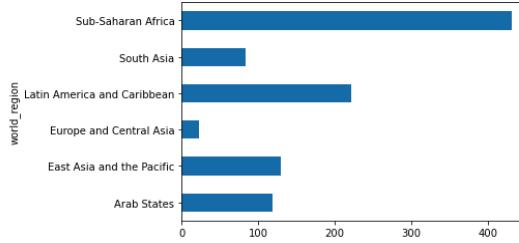


Fig. 17 World Region Distribution

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 crowdfunding projects. Because of the lack of alternate sources of financing, crowdfunding has seen a growing popularity here.

- Projects Funded Through Kiva

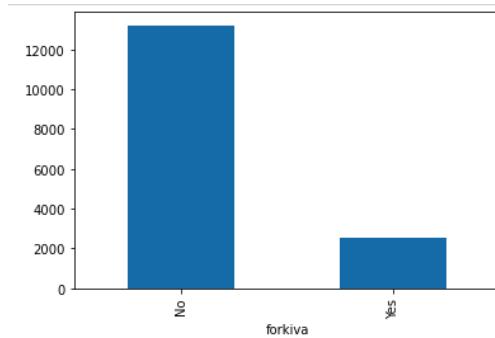


Fig. 18 Kiva Funded Projects

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.

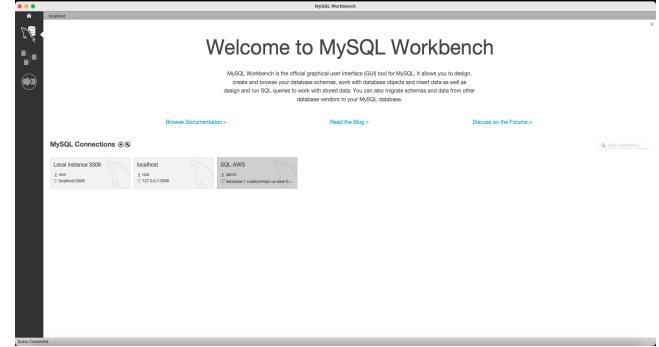


Fig. 19 MySQL-AWS Connection

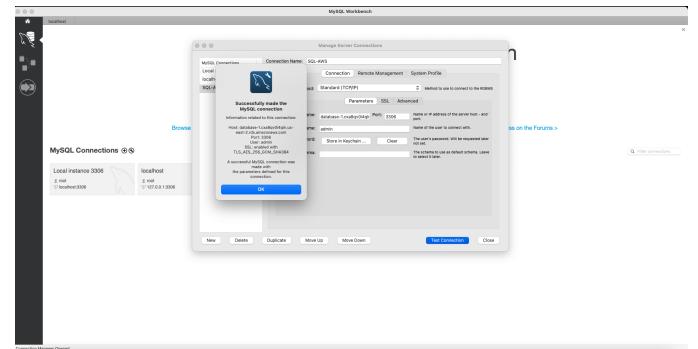


Fig. 20 MySQL-AWS Connection Successful

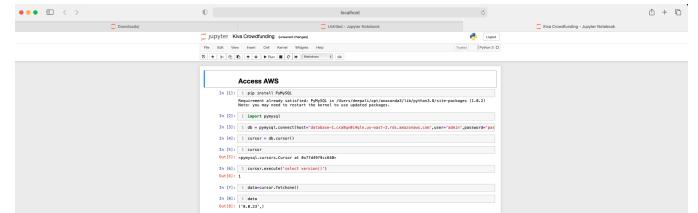


Fig. 21 Python-AWS Connection

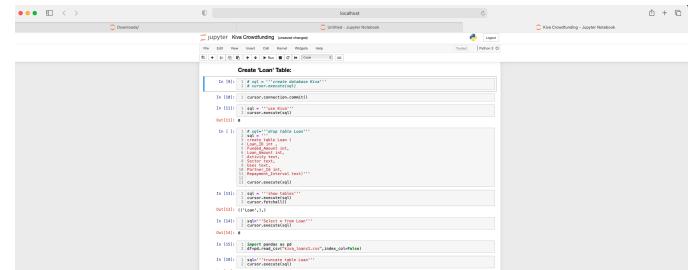


Fig. 22 Database Table Created Using PyMySQL

The screenshot shows a Jupyter Notebook window titled "Kiva Crowdfunding". It contains several code cells and their corresponding output. The code includes various SQL statements such as creating tables, inserting data, and running queries like "SELECT * FROM loans". The output displays the results of these queries, which consist of multiple rows of data.

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.

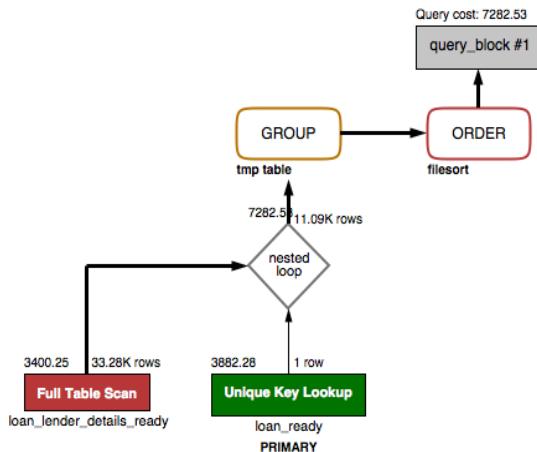


Fig. 24 MySQL Performance Analysis for a sample complex query on Kiva database

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 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/crowdfundin g.