HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

Big Data Integration and Processing

TOPIC:
P2P Lending data

Lecturer: PhD. Đào Thành Chung
Student: Vũ Đình Phượng (20222150M)

Hà Nội, 2023

TABLE OF CONTENTS

1. INTRODUCTION

In the modern era of data-driven decision-making, the convergence of technology and finance has given rise to innovative financial models such as Peer-to-Peer (P2P) lending. P2P lending, often referred to as social lending or crowdfunding, has disrupted the traditional borrowing and lending landscape by facilitating direct interactions between borrowers and investors through online platforms. This digital approach to lending has not only democratized access to credit but has also generated vast amounts of data that hold the potential to transform the financial sector.

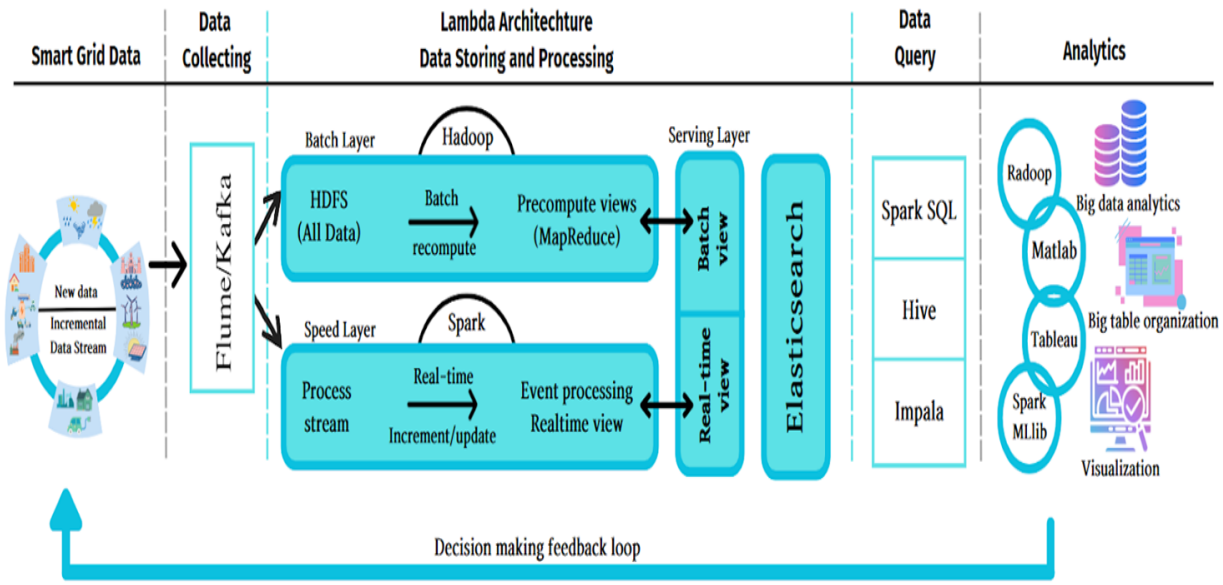
In the pursuit of enhanced financial insights, the integration and processing of big data have become pivotal. Big data, characterized by its volume, velocity, and variety, has opened avenues for understanding borrower behaviors, risk assessment, and investment patterns within the context of P2P lending. This report explores the dynamic relationship between big data integration, processing, and the world of P2P lending.

As we delve into this subject, we uncover the multifaceted nature of P2P lending and the intricate web of data it generates. The evolution of P2P lending platforms has brought forth new opportunities for individuals and businesses to access funding outside of traditional financial institutions. Concurrently, the digital nature of these transactions results in the creation of data points that encapsulate borrower profiles, loan terms, repayment patterns, and investor preferences.

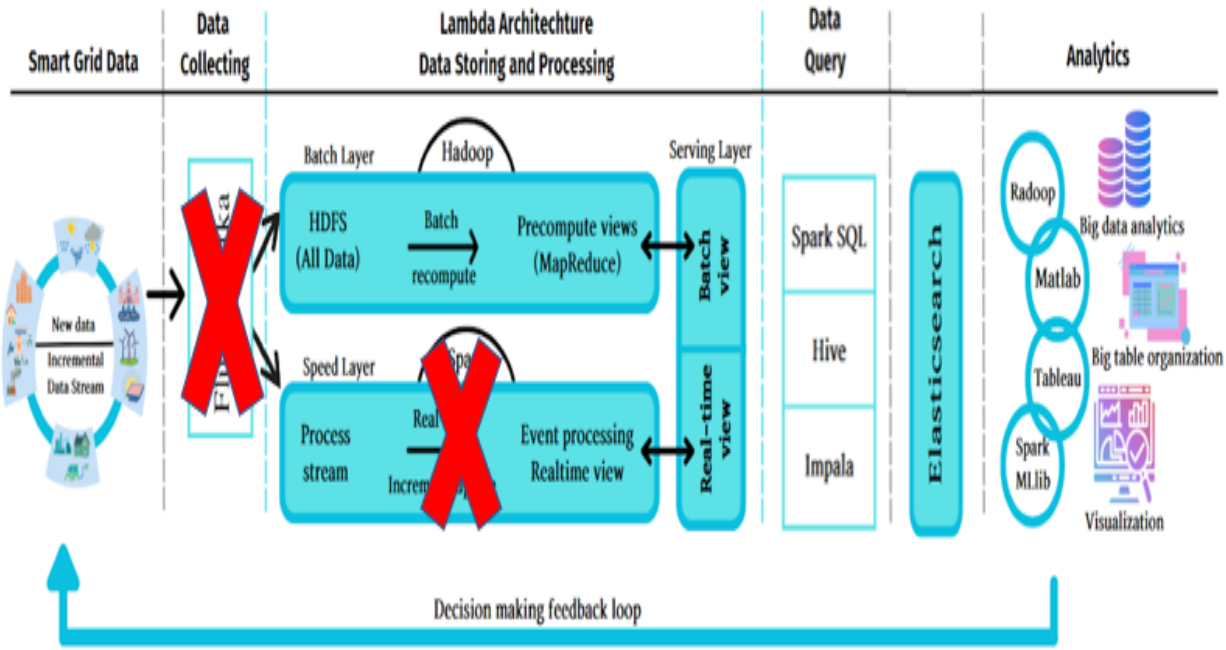
With a focus on the intersection of finance and technology, this report aims to shed light on the significance of big data integration and processing in the realm of P2P lending. By examining the challenges, strategies, and potential outcomes of effectively managing and analyzing large-scale data within this context, we hope to unveil the transformative potential of harnessing data for making informed decisions and optimizing lending practices.

2. ARCHITECTURE

2.1 OVERALL ARCHITECTURE.



Because my topic is not real-time, so I will adjust some things to be appropriate for my project.



What distinctions exist?

1. I won’t utilize Flume, Kafka, or Pub/Sub along with the Speed Layer, as real-time processing isn't a requirement for this project.
2. I will rearrange the sequence of steps, placing the Data Query step before the Elasticsearch step. I intend to employ Spark for data reading and processing before submitting it for storage in Elasticsearch.

2.2 TOOLS, FRAMEWORKS.

So here are these tools, and frameworks that I will use responsive to each step in the architecture.

- Begin by employing Google Cloud Storage (GCS) to store the accumulated data.
- Utilize Spark to retrieve data from GCS, carry out processing and comprehensive data analysis. Subsequently, send the resultant indexes to Elasticsearch.
- Allow Elasticsearch to store the indexes provided during the second step.
- Employ Kibana to create a dashboard with visually engaging representations for effective visualization.

- **Google Cloud Storage** is a scalable, fully managed object storage service provided by Google Cloud Platform (GCP). It allows you to store and retrieve data in a highly available and durable manner. Google Cloud Storage is designed to work seamlessly with other Google Cloud services and can be integrated into various applications and workflows.

  Key features of Google Cloud Storage include:

  1. **Scalability:** You can store vast amounts of data, ranging from a few bytes to multiple terabytes.
  2. **Durability and Availability:** Google Cloud Storage offers high durability, ensuring that your data is safe even in the event of hardware failures. It also provides high availability, allowing you to access your data whenever you need it.
  3. **Global Reach:** Google Cloud Storage has data centers in various regions around the world, enabling you to store and retrieve data from locations that are geographically closer to your users, reducing latency.
  4. **Object-Based Storage:** Data is stored as objects, which can be images, videos, documents, or any other type of binary data. Each object is associated with a unique key called an Object ID.
  5. **Security:** Google Cloud Storage offers various security features, including encryption at rest and in transit, access controls, and integration with Google Cloud Identity and Access Management (IAM) for fine-grained access control.
  6. **Cost-effective:** Google Cloud Storage offers different storage classes, allowing you to choose the right storage option for your data based on your performance and cost requirements.
  7. **Integration:** It integrates well with other Google Cloud services like BigQuery, Compute Engine, and Google Kubernetes Engine, enabling you to build comprehensive data pipelines and applications.
  8. **Lifecycle Management:** You can set up rules to automatically transition objects between storage classes or even delete objects that are no longer needed, helping you optimize costs.
  9. **Versioning and Metadata:** Google Cloud Storage supports object versioning, allowing you to keep multiple versions of an object. You

can also add custom metadata to objects for better organization and searching.

10. **Data Transfer:** Google Cloud Storage provides tools and APIs to efficiently transfer data to and from the storage service.

To use Google Cloud Storage, you'll need a Google Cloud Platform account, and you can interact with it using the Google Cloud Console (web-based UI), command-line tools, or programmatically through APIs and SDKs. It's commonly used for various purposes such as data storage, backup and archiving, content distribution, and serving static assets for web applications.

- **Apache Spark** is an open-source, distributed data processing and analytics framework designed to handle large-scale data processing tasks with speed and ease. It was developed at the AMPLab at the University of California, Berkeley, and later became an Apache Software Foundation project.

  Key features and characteristics of Apache Spark include:

  1. **In-Memory Processing:** Spark utilizes in-memory computation, which means it stores intermediate data in memory rather than on disk. This greatly speeds up processing, making it suitable for iterative algorithms and interactive data analysis.
  2. **Distributed Computing:** Spark can distribute data and computation across a cluster of machines, enabling parallel processing and efficient utilization of resources.
  3. **Resilient Distributed Datasets (RDDs):** RDDs are the fundamental data structure in Spark. They represent an immutable, fault-tolerant collection of objects that can be processed in parallel. RDDs allow users to perform transformations and actions on data stored in memory.
  4. **High-Level APIs:** Spark offers APIs in multiple languages, including Scala, Java, Python, and R. This makes it accessible to developers with different language preferences.
  5. **Support for Various Workloads:** Spark supports various workloads such as batch processing, iterative algorithms, interactive querying (SQL), machine learning, and graph processing.

6. **Built-in Libraries:** Spark comes with built-in libraries for various tasks, including Spark SQL for querying structured data using SQL-like syntax, MLlib for machine learning, GraphX for graph processing, and Spark Streaming for processing real-time data streams.
7. **Advanced Analytics:** Spark supports more complex analytics tasks like iterative machine learning algorithms, interactive querying, and graph processing, which are not as easily achievable with traditional MapReduce frameworks.
8. **Ease of Use:** Spark provides a user-friendly API that abstracts many of the complexities of distributed computing, making it easier for developers to write efficient and scalable code.
9. **Integration with Big Data Ecosystem:** Spark can be easily integrated with other big data tools and platforms such as Hadoop HDFS, HBase, and various data sources like Apache Cassandra, Amazon S3, and more.
10. **Spark Cluster Manager:** Spark provides its own cluster manager, called the "Standalone Cluster Manager," but it can also be integrated with other cluster managers like Apache Hadoop YARN and Apache Mesos.
11. **Community and Ecosystem:** Spark has a vibrant open-source community that contributes to its development and enhancement. Additionally, there is a growing ecosystem of third-party libraries, connectors, and tools that extend Spark's capabilities.

Apache Spark has gained widespread adoption in the big data and analytics domain due to its speed, ease of use, and flexibility. It is used by organizations for various data processing and analysis tasks, ranging from simple batch processing to complex machine learning applications.

- **Elasticsearch** is an open-source, distributed search and analytics engine designed for real-time data exploration and analysis. It's built on top of the Apache Lucene library and is known for its powerful text search capabilities, speed, scalability, and ease of use. Elasticsearch is often used to build

applications that require fast and accurate search functionality across large volumes of data.

Key features and aspects of Elasticsearch include:

1. **Full-Text Search:** Elasticsearch excels at full-text search, allowing users to perform complex queries on text data with high accuracy and speed.
2. **Distributed and Scalable:** Elasticsearch is designed for distributed computing and can easily scale horizontally by adding more nodes to a cluster. This makes it suitable for handling large datasets and high query loads.
3. **Real-Time Search:** Elasticsearch provides real-time search capabilities, meaning that as soon as data is indexed, it becomes searchable. This makes it ideal for applications where up-to-date information is crucial.
4. **Schema-less:** Elasticsearch is schema-less, meaning you can index documents without having to define a rigid structure beforehand. This provides flexibility when dealing with evolving or varied data.
5. **JSON-Based:** Data in Elasticsearch is stored in JSON format, which is a widely used and easy-to-understand data interchange format.
6. **RESTful API:** Elasticsearch exposes a comprehensive RESTful API that allows developers to interact with the system using HTTP requests. This makes it easy to integrate Elasticsearch into various applications.
7. **Rich Query DSL:** Elasticsearch offers a powerful Query DSL (Domain Specific Language) that allows you to construct complex queries for filtering, searching, and aggregating data.
8. **Aggregation and Analytics:** Elasticsearch provides aggregation capabilities that allow you to summarize and analyze data, making it useful for generating reports and insights.
9. **Multi-Tenancy:** Elasticsearch supports multi-tenancy, meaning you can index and search data from multiple sources or users within the same cluster while maintaining data isolation.

10. **Built-in Replication and Fault Tolerance:** Elasticsearch automatically handles data replication across nodes for fault tolerance and high availability.
11. **Plugins and Ecosystem:** Elasticsearch has a rich ecosystem of plugins and extensions that enhance its functionality. These include plugins for security, monitoring, visualization (Kibana), and more.
12. **Use Cases:** Elasticsearch is commonly used for a variety of applications, including website search engines, log and event data analysis, monitoring and observability, e-commerce product search, and text mining.

Elasticsearch is part of the larger Elastic Stack, which also includes Logstash (for data processing and transformation) and Kibana (for data visualization and exploration). Together, these components create a comprehensive solution for ingesting, processing, analyzing, and visualizing data.

- **Kibana** is an open-source data visualization and exploration platform that works seamlessly with Elasticsearch. It is part of the larger Elastic Stack (formerly known as the ELK Stack), which also includes Elasticsearch for data storage and retrieval, and Logstash for data processing and transformation. Kibana provides a user-friendly web interface for creating and sharing dynamic visualizations, dashboards, and reports based on data stored in Elasticsearch.

  Key features and capabilities of Kibana include:

  1. **Data Visualization:** Kibana allows you to create a wide range of visualizations such as line charts, bar charts, pie charts, maps, heatmaps, and more. These visualizations help you understand and analyze your data effectively.
  2. **Dashboard Creation:** Kibana enables you to combine multiple visualizations onto a single dashboard, providing a comprehensive view of your data. Dashboards can be customized and arranged to suit your specific needs.

3. **Interactive Exploration:** Kibana provides an interactive way to explore and analyze your data. Users can filter, drill down, and interact with visualizations to gain deeper insights.
4. **Search and Querying:** Kibana's search and query capabilities allow you to perform advanced searches on your indexed data stored in Elasticsearch. This is particularly useful for retrieving specific information and patterns from your data.
5. **Time-Series Analysis:** Kibana is especially powerful for time-series data analysis. It offers tools for visualizing trends and patterns over time, making it valuable for monitoring and observability use cases.
6. **Canvas:** Kibana's Canvas feature lets you create dynamic, pixel-perfect infographics and presentations. It's useful for designing custom visualizations and reports that go beyond traditional charting.
7. **Machine Learning Integration:** Kibana integrates with Elasticsearch's machine learning capabilities, allowing you to create anomaly detection and predictive analytics visualizations.
8. **Saved Searches and Visualizations:** You can save your searches and visualizations for reuse or sharing with others. This is helpful for collaboration and creating consistent reporting.
9. **User Access Control:** Kibana supports user authentication and access control through integration with security features in Elasticsearch. This allows you to control who can access and modify the visualizations and dashboards.
10. **Plugin Ecosystem:** Kibana has a plugin ecosystem that lets you extend its functionality. There are various community-contributed and official plugins available for additional visualization types and integrations.

Kibana is often used in conjunction with Elasticsearch for various use cases such as log and event analysis, application performance monitoring, business intelligence, and operational dashboards. Its user-friendly interface makes it accessible to a wide range of users, including data analysts, business users, and technical teams.

3. EXPERIMENT AND RESULT

3.1. DATASET

Lending Club is a peer-to-peer Lending company based in the US. They match people looking to invest money with people looking to borrow money. When investors invest their money through Lending Club, this money is passed onto borrowers, and when borrowers pay their loans back, the capital plus the interest passes on back to the investors. It is a win for everybody as they can get typically lower loan rates and higher investor returns.

The Lending Club dataset contains complete loan data for all loans issued through the 2007-2020Q3, including the current loan status (Current, Late, Fully Paid, etc.) and latest payment information. Features (aka variables) include credit scores, number of finance inquiries, address including zip codes and state, and collections among others. Collections indicates whether the customer has missed one or more payments and the team is trying to recover their money. The file is a matrix of about 2925493 observations and 151 variables.

3.2. EXPERIMENT AND RESULT

3.2.1 PREPROCESSING DATA.

Section Goals:

- Remove unnecessary or repetitive features.
  - We just keep 27 importance features
  - We only keep the loans with status "Current" or "Fully Paid" or "Charged Off."
  - If emp_title is omitted, it means Borrowers don't want to offer their work or maybe they are unemployed. So we will fill Other to them.
- Remove or fill any missing data.
  - Drop rows with null values in the "issue_d" column.
  - Drop rows with null values in the "loan_amnt" column.

3.2.2 ANALYZE DATA

- Count and Sum loan amount by loan_status.

```
loan_status_agg = df.groupby("loan_status").agg(
        count(col("*")).alias("loan_amount_sum_by_status"),
        sum(col("loan_amnt")).alias("loan_amount_sum_by_status")
    )

#loan_status_agg.show()
```

- **Risk-Return Analysis by Grade:** This analysis combines average interest rates and average loan amounts for each loan grade. It provides insights into the relationship between the risk associated with each grade and the potential return (higher interest rate) investors can expect.

```
# Extract numeric values from int_rate column
int_rate_numeric = regexp_extract(col("int_rate"), r"([0-9]+.[0-9]+)", 1).cast("float")

risk_return_analysis = df.withColumn("int_rate_numeric", int_rate_numeric) \
    .groupBy("grade") \
    .agg(avg("int_rate_numeric").alias("avg_interest_rate"), avg("loan_amnt").alias("avg_loan_amount"))
# Format the avg_interest_rate to 2 decimal places
risk_return_analysis = risk_return_analysis.withColumn("avg_interest_rate", format_number("avg_interest_rate", 2))

#risk_return_analysis.show()
```

- **By quarter:** Calculates the count of loan applications and sum loan amounts for each quarter and year. It groups the data by the year and quarter extracted from the "issue_d" column.

```
loan_data = df.withColumn("parsed_date", to_date("issue_d", "MMM-yyyy"))
loan_data = loan_data.withColumn("year", year("parsed_date"))
loan_data = loan_data.withColumn("quarter", quarter("parsed_date"))

loan_amount_count_by_quarter = loan_data.groupBy("year", "quarter").agg(count("*").alias("loan_amount_count_by_quarter"))
loan_amount_sum_by_quarter = loan_data.groupBy("year", "quarter").agg(sum(col("loan_amnt")).alias("loan_amount_sum_by_quarter"))
```

- **Loan Term Analysis:** Analyzing loan terms helps you understand how borrowers prefer different loan durations ("36 months", "60 months"). This can provide insights into borrower preferences and how loan term options influence loan demand.

```
loan_term_analysis = df.groupBy("term").agg(count("*").alias("loan_count"))
#loan_term_analysis.show()
```

- **Emp_length:** These analyses on the emp_length column can provide insights into how borrowers' employment stability influences their loan behavior, loan performance, interest rates, and other loan-related aspects.

  This analysis provides the distribution of loans across different employment length categories. It helps you understand the prevalence of various employment lengths among borrowers.

Analyze how average interest rates change based on employment length categories. This analysis can reveal whether employment length influences the interest rates offered to borrowers.

```
emp_length_distribution = df.withColumn("int_rate_numeric", int_rate_numeric) \
    .groupBy("emp_length").agg(
        count("*").alias("loan_count"), avg("int_rate_numeric").alias("avg_interest_rate")
    )
#emp_length_distribution.show()
```

- **Emp_title:** Find the emp_title with the highest sum of loan amounts.

```
# Group by emp_title and calculate the sum of loan_amnt
emp_loan_sum = df.groupBy("emp_title").agg(sum("loan_amnt").alias("total_loan_amount"))

# Order by total_loan_amount in descending order to find the highest sum
highest_loan_emp = emp_loan_sum.orderBy(desc("total_loan_amount"))
#highest_loan_emp.show()
```

3.2.3 SEND THE INDEXES TO ELASTICSEARCH.

```
index_to_es(loan_status_agg, index_name = "loan_status_agg")
```

```
100%|██████████| 3/3 [00:00<00:00, 3838.59it/s]
```

```
index_to_es(risk_return_analysis, index_name = "risk_return_analysis")
```

```
21it [00:00, 20417.34it/s]
```

```
index_to_es(loan_amount_count_by_quarter, index_name = "loan_amount_count_by_quarter")
```

```
1431it [00:00, 380165.25it/s]
```

```
index_to_es(loan_amount_sum_by_quarter, index_name = "loan_amount_sum_by_quarter")
```

```
1431it [00:00, 367998.10it/s]
```

```
index_to_es(loan_term_analysis, index_name = "loan_term_analysis")
```

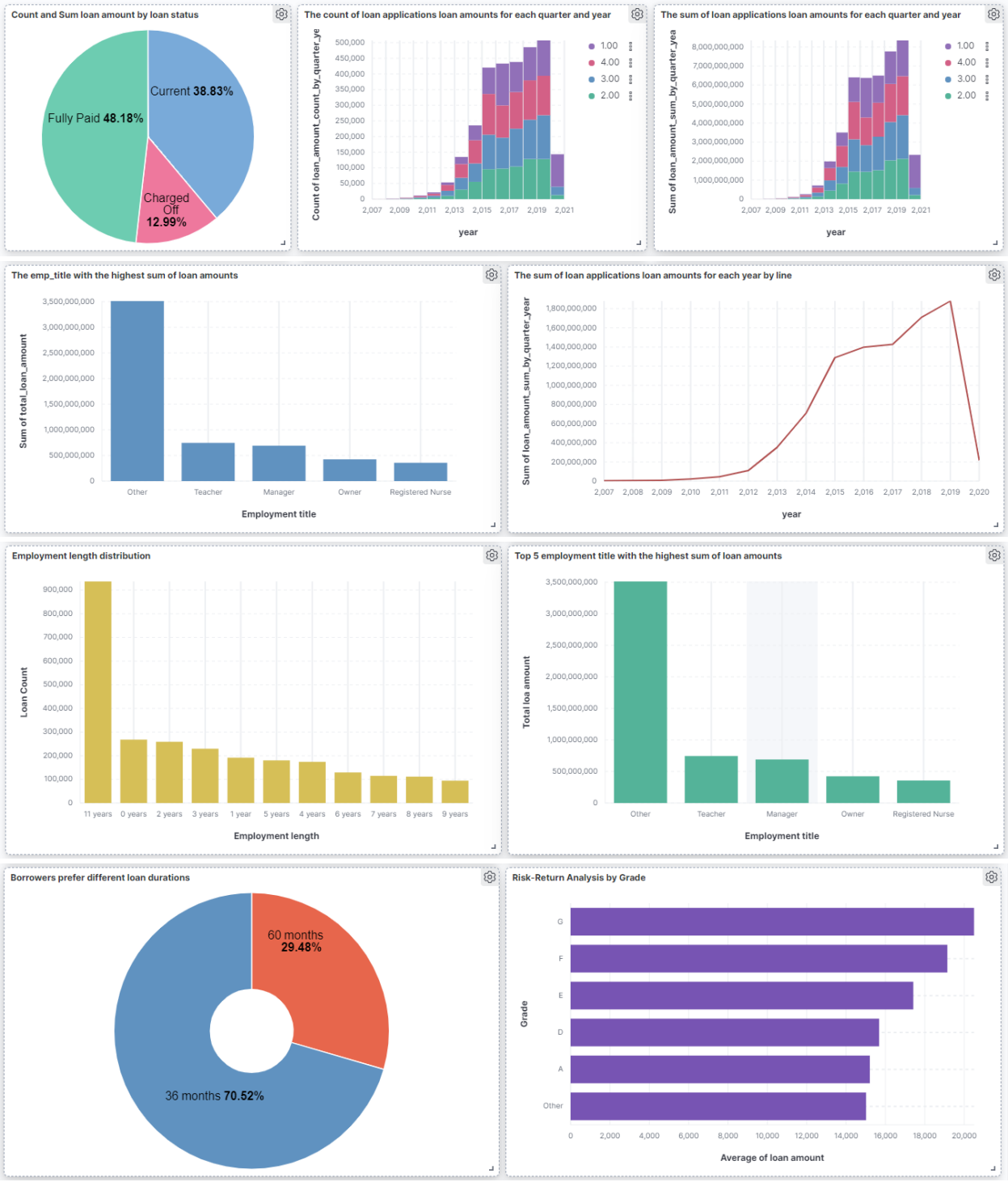
```
 50%|█████     | 1/2 [00:00<00:00, 1897.88it/s]
```

```
index_to_es(emp_length_distribution, index_name = "emp_length_distribution")
```

```
66it [00:00, 49090.98it/s]
```

```
index_to_es(highest_loan_emp, index_name = "highest_loan_emp")
```

3.2.4 RESULT


Count and Sum loan amount by loan status
Current 38.83%
Fully Paid 48.18%
Charged Off 12.99%
The count of loan applications loan amounts for each quarter and year
Count of loan_amount_count_by_quarter_ye
year
The sum of loan applications loan amounts for each quarter and year
Sum of loan_amount_sum_by_quarter_yea
year
1.00
4.00
3.00
2.00
The emp_title with the highest sum of loan amounts
Sum of total_loan_amount
Other
Teacher
Manager
Owner
Registered Nurse
Employment title
The sum of loan applications loan amounts for each year by line
Sum of loan_amount_sum_by_quarter_year
year
Employment length distribution
Loan Count
11 years
0 years
2 years
3 years
1 year
5 years
4 years
6 years
7 years
8 years
9 years
Employment length
Top 5 employment title with the highest sum of loan amounts
Total loa amount
Other
Teacher
Manager
Owner
Registered Nurse
Employment title
Borrowers prefer different loan durations
60 months 29.48%
36 months 70.52%
Risk-Return Analysis by Grade
Grade
G
F
E
D
A
Other
Average of loan amount

4. BIBLIOGRAPHY

1. Dataset https://www.kaggle.com/datasets/ethon0426/lending-club-20072020q1
2. LendingClub https://www.lendingclub.com
3. Google cloud https://cloud.google.com
4. Elasticsearch https://www.elastic.co
5. Kibana https://www.elastic.co/kibana
6. Apache Spark https://spark.apache.org