Implementing a Database Management System for a P2P Lending Platform

Group 1

Penglin Du

Xinrui Dong

Suzy Gao

Yumeng Gu

Jia Yang

Implementing a Database Management System for a P2P Lending Platform

Table of Contents

1 Consultant / Client Scenario

1.1 The choice for the scenario

Implementing a database management system for a P2P lending platform.

1.2 Reason, motivation, and research

The peer-to-peer(P2P) lending marketplace works through a simple online platform, which connects borrowers and lenders. Since P2P are wide-area and large-scale systems that provide content sharing and storage services, having a well-designed database is crucial for them to manage the massive amount of information. To be specific, P2P systems do not lend their own funds but act as facilitators to both the borrower and the lender. With the rising popularity of peer-to-peer lending platforms, competition and products have increased as well. While these marketplaces operate on the same basic principle, they vary in terms of eligibility criteria, loan rates, amounts, and tenures as well as offerings. Some focus on personal loans, and a few target students and young professionals, while some cater exclusively to business needs.

1.3 The initial plan of action

The initial plan of actions after proposing the scenario and finding data includes but are not limited to fully exploring the datasets, deciding and assigning tasks, normalizing the raw datasets, developing a relational schema, drawing the ER diagrams, designing the ETL process, conducting analysis on the relational database, obtaining insights for the P2P lending platform, constructing interactive dashboards, and self-evaluation and future improvement. The initial design for our database involves three main entities: the investors who lend money, the borrowers who request it, and load information which may include loan fulfillment and repayment data.

1.4 Improve the company's decision-making process and other benefits

By designing a clear database for automatically storing and easily retrieving the data, our work will help the company keep track of basic transactions, provide information on borrowers and investors, and generate insights through various analytical procedures that will help the company run the business more efficiently, help managers and employees make better decisions, mitigate investment risks, and at last improve the whole decision-making process.

2 Team Contract / Dataset

2.1 Team contract

TEAM CONTRACT:

Project Name: **Database Management System Design For a P2P Lending Platform**
Team Name: **Project Group 1**
Team members: Xinrui Dong, Penglin Du, Suzy Gao, Yumeng Gu, Jia Yang

Roles and Responsibilities		Responsible Persons	Due Date
Client Scenario Justifications	• Describing consultant/client scenario. • Providing detailed reasoning behind the choice and motivation. • Showing research that has been performed in making the decision and initial plan of action. • Explaining how the project will improve the decision-making for the client. • Presenting other possible benefits of this project	Suzy Gao, Yumeng Gu	April 1st
Normalization and ER diagrams	• Providing sample data and link to full data. Explaining the choice of dataset. • Presenting the normalization plan with details and ER diagrams. • Providing SQL code for normalized tables • Explaining all works for normalization	Penglin Du, Jia Yang, Yumeng Gu	April 15th
ETL Process	• Presenting the ETL process in detail • Formatting the code properly • Providing Github repo or Gist • Explaining all works for ETL	Jia Yang , Suzy Gao, Penglin Du	April 22nd
Analytical Procedure	• Showcasing at least 10 analytical procedures • Supporting each one with its corresponding code listing that delivers the results	Xinrui Dong, Yumeng Gu	April 25th
Justification of Methodology and Tools	• Presenting specific plans on how customer will interact with the database system	Yumeng Gu, Xinrui Dong	April 29th
Metabase Dashboards	• Showcasing dashboards created in Metabase • Pasting screenshots and explaining what it presents and the benefits of it. • Providing links to the dashboards	Xinrui Dong, Suzy Gao	May 3rd
Report/Presentation Style and Formatting	• Properly formatting codes with links • Using 12pt Arial or Times New Roman font, spaced at 1.2 lines • Submitting as a single PDF with all relevant attachments	Penglin Du	May 5th

2.2 Project dataset

2.2.1 Dataset

The dataset we choose is an open dataset from Prosper, a P2P lending platform.

Link to the dataset: https://docs.google.com/spreadsheets/d/1edot-MfpCHP-DgHhS36L-U7C4if-it6kukOwHx_Mm-k/edit?usp=sharing

Link for downloading the original data source: https://s3.amazonaws.com/udacity-hosted-downloads/ud651/prosperLoanData.csv

2.2.2 Reasons for choosing the data

The original dataset has a rich record of 81 variables and 113,979 observation for each loan list data from 2005 to 2014, which are suitable for database design. The database mainly contains four groups of data:

1. Loan Status: The status of the loan list, such as Cancelled, Charged off, Completed, Current, Defaulted, Final Payment In Progress, Past Due.
2. Borrower Data: Basic properties about borrowers such as income, occupation, employment status, etc.
3. Loan Data: Basic properties about the loan such as length of the loan(term), Borrower APR, etc.
4. Credit Risk Metrics: Metrics measured the risk of loans, such as Credit grade, Prosper Score, bank card utilization, etc.

However, the above data with only the information of borrower and loan is not enough for comprehensive database design. Since the data of investor is not open to the public, in order to make the dataset more suitable for database design, information about the investor is generated and linked to the borrower by loan information. A sample dataset with 1,000 rows is used for the following procedures.

3 Normalization Plan / ER Diagram

3.1 Create schema

3.1.1 Investors section

```
#CREATE TABLE
stmt <- 'CREATE TABLE phones (
phone_id char(10),
phone_number varchar(20),
PRIMARY KEY(phone_id));'
```

```
dbGetQuery(con, stmt)

stmt<- 'CREATE TABLE nominee(
nominee_id varchar(50),
nominee_first_name varchar(50),
nominee_last_name varchar(50),
nominee_relationship_with_investor varchar(30),
nominee_date_of_birth date,
PRIMARY KEY (nominee_id));'
dbGetQuery(con, stmt)

stmt<- 'CREATE TABLE payment_bank(
payment_bank_id char(10),
payment_bank_name varchar(50),
PRIMARY KEY (payment_bank_id));'
dbGetQuery(con, stmt)
```

(Please check the detailed codes on GitHub, we only post a sample code for creating table in this report, GitHub link: https://github.com/suzygaoo/P2P-lending-platform-database-system)

3.1.2 Borrowers section

```
#CREATE TABLES
stmt <- 'CREATE TABLE b_phones (
phone_id char(10),
phone_number varchar(20),
PRIMARY KEY(phone_id));'
dbGetQuery(con, stmt)

stmt <- 'CREATE TABLE borrower (
borrower_id varchar(10),
first_name varchar(50),
last_name varchar(50),
state varchar(50),
city varchar(50),
street_address varchar(50),
zip_code varchar(10),
PRIMARY KEY (borrower_id));'
dbGetQuery(con, stmt)

stmt <- 'CREATE TABLE loan_ticket (
loan_ticket_id varchar(10),
borrower_id varchar(10),
loan_amount numeric(10,2),
loan_tenure_in_month numeric,
reason_for_loan text,
```

borrow_rate decimal(8,6),
borrower_apr decimal(8,6),
loan_status varchar(50),
PRIMARY KEY (loan_ticket_id),
FOREIGN KEY (borrower_id) REFERENCES borrower);'
dbGetQuery(con, stmt)

(Please check the detailed codes on GitHub, we only post a sample code for creating table in this report, GitHub link: https://github.com/suzygaoo/P2P-lending-platform-database-system)

3.1.3 Investor proposal section

CREATE TABLE Investor_proposal(
investor_proposal_id varchar(10),
Investor_id char(10),
Loan_ticket_id varchar(10),
proposal_amount numeric(9,2),
proposal_date date,
cancel_date date,
PRIMARY KEY(investor_proposal_id),
FOREIGN KEY(investor_id) REFERENCES investor(investor_id),
FOREIGN KEY(loan_ticket) REFERENCES investor(loan_ticket_id));

3.2 Normalization

3.2.1 Normalization process with Borrower sample dataset

1NF												
loan table												
loan_ticket_id (pk)	borrower_id	first_name	last_name	state	city	street_address	zip_code	reason_for_l	loan_amount	loan_ter	borrower_apr	borrow_rate
lt62393	b49002	Kattie	Furley	Virginia	Roanoke	6 Mallory Road	24048	Debt consoli	9,425.00	36	0.16516	0.15
lt62393	b49002	Kattie	Furley	Virginia	Roanoke	6 Mallory Road	24048	Debt consoli	9,425.00	36	0.16516	0.15
lt62393	b49002	Kattie	Furley	Virginia	Roanoke	6 Mallory Road	24048	Debt consoli	9,425.00	36	0.16516	0.15
lt48635	b71211	Dyann	Masi	Oklahoma	Oklahoma City	7075 Riverside Park	73147	Debt consoli	10,000.00	36	0.12016	0.09
lt48635	b71211	Dyann	Masi	Oklahoma	Oklahoma City	7075 Riverside Park	73147	Debt consoli	10,000.00	36	0.12016	0.09
lt48635	b71211	Dyann	Masi	Oklahoma	Oklahoma City	7075 Riverside Park	73147	Debt consoli	10,000.00	36	0.12016	0.09
borrower phones table												
phone_id	phone_number											
p#####	3038795473											
p#####	8606388472											
p#####	5856317298											
p#####	2623610087											
borrower_phone table												
phone_id	borrower_id	phone_type										
p#####	i####											
p#####	i####											
p#####	i####											
p#####	i####											

The original borrower dataset is not in 1NF, as the value of phone_numbers and credit agency are not atomic. Making it into 1NF by separating the non-atomic value and bringing out the data of phone numbers and make an entity phone table and

borrower_phone relational table.

2NF

loan table

loan_ticket_id (pk)	borrower_id	first_name	last_name	state	city	street_address	zip_code	reason_for_l	loan_amount	loan_ter	borrower_apr	borrow_rate	loan_status	occupation	employment_status
lt62393	b49002	Kattie	Furley	Virginia	Roanoke	6 Mallory Road	24048	Debt consoli	9,425.00	36	0.16516	0.158		Other	Self-employed
lt48635	b71211	Dyann	Masi	Oklahoma	Oklahoma City	7075 Riverside Park	73147	Debt consoli	10,000.00	36	0.12016	0.092	current	Professional	Employed

credit rating agency table

credit_rating_agency_id	credit_rating_agency	establish_year
c#####	Equifax	1898
c#####	TransUnion	1968
c#####	Experian	1996

credit info table

credit_info_id	borrower_id	current_cr	credit_sc	credit_rating_agency
ci#####	b49002	2	640	Equifax
ci#####	b49002	2	650	TransUnion
ci#####	b49002	2	659	Experian
ci#####	b71211	44	680	Equifax
ci#####	b71211	44	690	TransUnion
ci#####	b71211	44	699	Experian

borrower phones table

phone_id	phone_number
p#####	3038795473
p#####	8606388472
p#####	5856317298
p#####	2623610087

borrower_phone table

phone_id	borrower_id	phone_type
p#####	i#####	
p#####	i#####	
p#####	i#####	
p#####	i#####	

After the first step, the dataset is now in 1NF. Next is to achieve the 2NF by getting rid of the partial dependency in credit rating agencies and add a credit rating agency table, a credit info table.

3NF

transitive dependencies exist in loan inforamtion, credit rating agency, and emoloyement , etc

borrower table

borrower_id	first_name	last_name	state	city	street	zip_code
b49002	Kattie	Furley	Virginia	Roanoke	6 Mallory Road	24048
b71211	Dyann	Masi	Oklahoma	Oklahoma Cit	7075 Riverside F	73147

loan ticket table

loan_ticket_id	borrower_id	loan_amount	loan_tenur	reason for loar	borrow_rate	borrower_apr	loan_status
lt######	b49002	9,425.00	36	Debt consolida	0.158	0.16516	
lt######	b71211	10,000.00	36	Debt consolida	0.158	0.12016	current

eployement table

employment_id	borrower_id	employment	occupation	employment_status
e#####	b49002	2	Other	Self-employed
e#####	b71211	44	Professiona	Employed

credit rating agency table

credit_rating_agency_id	credit_rating_agency	establish_year
c#####	Equifax	1898
c#####	TransUnion	1968
c#####	Experian	1996

credit info table

credit_info_id	borrower_id	current_cr	credit_sc	credit_rating_agency
ci#####	b49002	2	640	Equifax
ci#####	b49002	2	650	TransUnion
ci#####	b49002	2	659	Experian
ci#####	b71211	44	680	Equifax
ci#####	b71211	44	690	TransUnion
ci#####	b71211	44	699	Experian

borrower_phone table

phone_id	borrower_id	phone_type
p#####	i#####	
p#####	i#####	
p#####	i#####	
p#####	i#####	

prosper index

prosper_inde	borrower_id	prosper_	prosper_score
p####	b49002		
p####	b71211	6	7

borrow liability table

borrow_liabi	loan_ticket_id	monthly_	liability_start_da	liability_end_date
bl#####	lt######	330.43	9/12/07	8/14/09
bl#####	lt######	318.93	3/3/14	

delinquency history table

delinquency	borrower_id	current_	amount_delqinquency
d####	b49002	2	472
d####	b71211		

income info table

income_info	borrower_id	income_	income_range_t	income_verifia	stated_monthly_income
i####	b49002	25000	49,999	TRUE	3083.333
i####	b71211	50000	74,999	TRUE	318.93

borrower phones table

phone_id	phone_number
p#####	3038795473
p#####	8606388472
p#####	5856317298
p#####	2623610087

At this point, the dataset is already in 2NF. Moving further to 3NF, the transitive dependency of loan, employment, delinquency, income should be removed by creating a loan_ticke_table, an employment table, a prosper index table, a liability table, a delinquency table, and an income info table. By normalizing the original dataset into 3NF, the dataset is now broken up into 11 tables.

3.2.1 Normalization process with Investor sample dataset

1NF

investor_table

investor_id	firstname	lastname	ssn	date_of_birth	zip_code	address	city	country	kyc_complet	escrow_acc	i
i####	Anthiathia	Dilliway	459-75-4785	12/20/75	80217	628 Vernon F	Denver	United State:	N	000-3430801	
i####	Ella	Dunnaway	095-56-6959	2/5/81	53205	208 Grayhaw	Milwaukee	United State:	Y	000-5380465	
i####	Anthiathia	Dilliway	459-75-4785	12/20/75	80217	628 Vernon F	Denver	United State:	N	000-3430801	
i####	Ella	Dunnaway	095-56-6959	2/5/81	53205	208 Grayhaw	Milwaukee	United State:	Y	000-5380465	
i####	Anthiathia	Dilliway	459-75-4785	12/20/75	80217	628 Vernon F	Denver	United State:	N	000-3430801	
i####	Ella	Dunnaway	095-56-6959	2/5/81	53205	208 Grayhaw	Milwaukee	United State:	Y	000-5380465	
i####	Anthiathia	Dilliway	459-75-4785	12/20/75	80217	628 Vernon F	Denver	United State:	N	000-3430801	
i####	Ella	Dunnaway	095-56-6959	2/5/81	53205	208 Grayhaw	Milwaukee	United State:	Y	000-5380465	
i####	Anthiathia	Dilliway	459-75-4785	12/20/75	80217	628 Vernon F	Denver	United State:	N	000-3430801	
i####	Ella	Dunnaway	095-56-6959	2/5/81	53205	208 Grayhaw	Milwaukee	United State:	Y	000-5380465	
i####	Anthiathia	Dilliway	459-75-4785	12/20/75	80217	628 Vernon F	Denver	United State:	N	000-3430801	
i####	Ella	Dunnaway	095-56-6959	2/5/81	53205	208 Grayhaw	Milwaukee	United State:	Y	000-5380465	

investor phones table

phone_id	phone_number
p#####	3038795473
p#####	8606388472
p#####	5856317298
p#####	2623610087

investor_phone table

phone_id	investor_id	phone_type
p#####	i####	
p#####	i####	
p#####	i####	
p#####	i####	

The original investor dataset is not in 1NF, as the value of phone_numbers and payment account, bank info is not atomic. Making it into 1NF by separating the non-atomic value and bringing out the data of phone numbers and make an entity phone table and investor_phone relational table.

2NF

payment bank table

payment_bank_id	payment_bank_name
pb####	TD Bank
pb####	Wells Fargo
pb####	Bank of America

payment method table

payment_methond_id	investor_id	payment_account_number	payment_account_holder_name	payment_bank_id
pm####	i####	3574989067924360	Amelita Dilliway	pb####
pm####	i####	4917742384425210	Martha Dunnaway	pb####
pm####	i####	201707071634668	Amelita Dilliway	pb####
pm####	i####	3543260464892610	Martha Dunnaway	pb####
pm####	i####	6706420524492310000	Amelita Dilliway	pb####
pm####	i####	3550058897278910	Martha Dunnaway	pb####
pm####	i####	3574989067924360	Amelita Dilliway	pb####
pm####	i####	4917742384425210	Martha Dunnaway	pb####
pm####	i####	201707071634668	Amelita Dilliway	pb####
pm####	i####	3543260464892610	Martha Dunnaway	pb####
pm####	i####	6706420524492310000	Amelita Dilliway	pb####
pm####	i####	3550058897278910	Martha Dunnaway	pb####

phones table

phone_id	phone_number
p#####	3038795473
p#####	8606388472
p#####	5856317298
p#####	2623610087

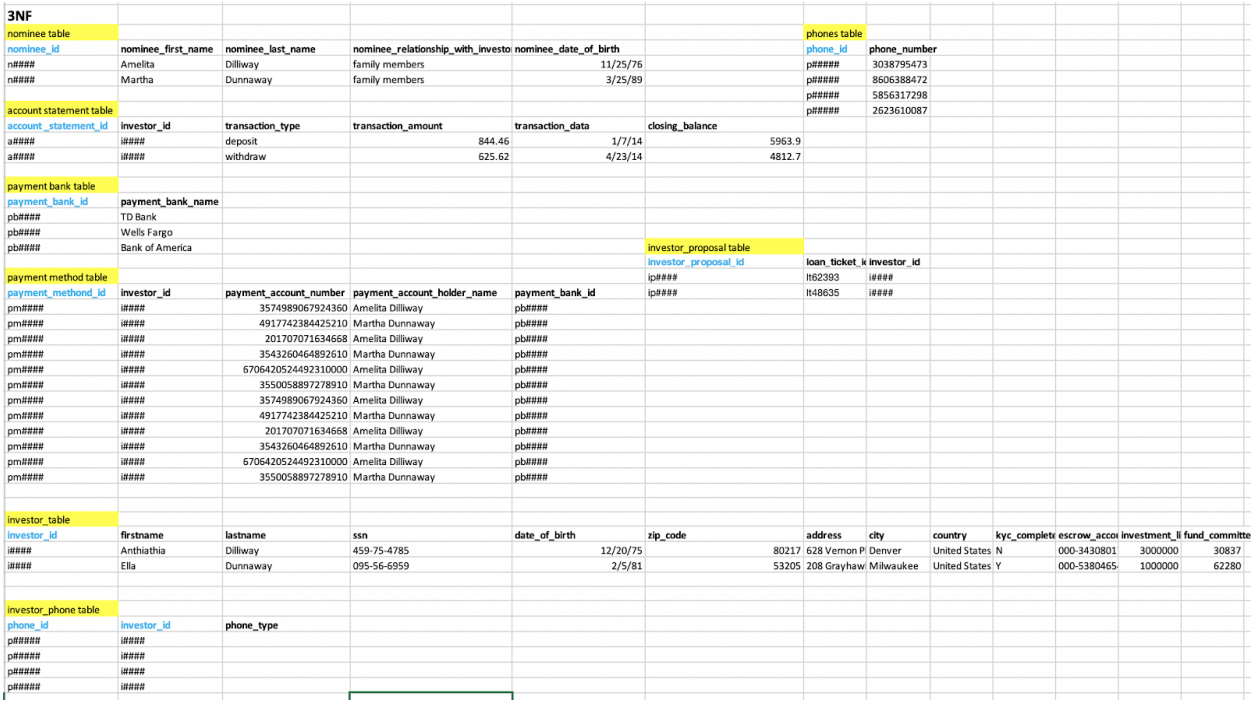
investor_phone table

phone_id	investor_id	phone_type
p#####	i####	
p#####	i####	
p#####	i####	
p#####	i####	

investor_table

investor_id	firstname	lastname	ssn	date_of_birth	zip_code	address	city	country	kyc_complet	escrow_acc	i
i####	Anthiathia	Dilliway	459-75-4785	12/20/75	80217	628 Vernon F	Denver	United State	N	000-3430801	
i####	Ella	Dunnaway	095-56-6959	2/5/81	53205	208 Grayhaw	Milwaukee	United State	Y	000-5380465	

After the first step, the dataset is now in 1NF. Next is to achieve the 2NF by getting rid of the partial dependency in payment account type, payment account name, etc. and add a payment bank table, a payment method table.



3NF

nominee table

nominee_id	nominee_first_name	nominee_last_name	nominee_relationship_with_investo	nominee_date_of_birth
n####	Amelita	Dilliway	family members	11/25/76
n####	Martha	Dunnaway	family members	3/25/89

account statement table

account_statement_id	investor_id	transaction_type	transaction_amount	transaction_data	closing_balance
a####	i####	deposit	844.46	1/7/14	5963.9
a####	i####	withdraw	625.62	4/23/14	4812.7

payment bank table

payment_bank_id	payment_bank_name
pb####	TD Bank
pb####	Wells Fargo
pb####	Bank of America

payment method table

payment_methond_id	investor_id	payment_account_number	payment_account_holder_name	payment_bank_id
pm####	i####	3574989067924360	Amelita Dilliway	pb####
pm####	i####	4917742384425210	Martha Dunnaway	pb####
pm####	i####	201707071634668	Amelita Dilliway	pb####
pm####	i####	3543260464892610	Martha Dunnaway	pb####
pm####	i####	6706420524492310000	Amelita Dilliway	pb####
pm####	i####	3550058897278910	Martha Dunnaway	pb####
pm####	i####	3574989067924360	Amelita Dilliway	pb####
pm####	i####	4917742384425210	Martha Dunnaway	pb####
pm####	i####	201707071634668	Amelita Dilliway	pb####
pm####	i####	3543260464892610	Martha Dunnaway	pb####
pm####	i####	6706420524492310000	Amelita Dilliway	pb####
pm####	i####	3550058897278910	Martha Dunnaway	pb####

investor_table

investor_id	firstname	lastname	ssn	date_of_birth	zip_code	address	city	country	kyc_complete	escrow_accor	investment_li	fund_committe
i####	Anthiathia	Dilliway	459-75-4785	12/20/75	80217	628 Vernon P	Denver	United States	N	000-3430801	3000000	30837
i####	Ella	Dunnaway	095-56-6959	2/5/81	53205	208 Grayhaw	Milwaukee	United States	Y	000-5380465	1000000	62280

investor_phone table

phone_id	investor_id	phone_type
p#####	i####	
p#####	i####	
p#####	i####	
p#####	i####	

phones table

phone_id	phone_number
p#####	3038795473
p#####	8606388472
p#####	5856317298
p#####	2623610087

investor_proposal table

investor_proposal_id	loan_ticket_i	investor_id
ip####	lt62393	i####
ip####	lt48635	i####

At this point, the dataset is already in 2NF. Moving further to 3NF, the transitive dependency of nominee and transaction should be removed by creating a nominee table and an account_statement table. By normalizing the original dataset into 3NF, the dataset is now broken up into 7 tables. Adding one more relational table that connects the investor with loan tickets, the dataset in total is normalized into 19 tables.

3.3 ER Diagram

Please check the Appendix 1 for the ER Diagram.

4 ETL Process (Plan and Execution) / GitHub Repo

4.1 ETL process

4.1.1 Extract

In the ETL process, the first step is to extract data, which is the process of reading data from a database. In this stage, the data is collected, often from multiple and different types of sources. In order to obtain border aspects of information on the Peer-to-peer platform, three sets of data were used in this project: Lending Club Load Data, Borrower data, and 10-Year Treasury Constant Maturity Rate. The CSV file for the prosper loan data was downloaded from the Prosper website(prosper.com). The borrower data was randomly generated from Mockaroo website (https://mockaroo.com/), which includes

mostly geographic information of the borrowers. The 10-year US treasury data was gathered from the Interworks website. (https://wdc.portals.interworks.com/fred_20/)

After successfully extract necessary data from the original sources, the next step is to merge two datasets and convert them into a single format for standardized processing. In this project, the information about investor and loan was extracted from Prosper dataset, combined with the borrower information, would be able to contain sufficient data about each loan the company lent.

The 10-Year Treasury Constant Maturity Rate will not be merged as it will only be used for visualization at the end of the project.

4.1.2 Transform

Transform is the process of converting the extracted data from its previous form into the form it needs to be in so that it can be placed into another database. Transformation occurs by using rules or lookup tables or by combining the data with other data. Our transformation process includes removing extraneous or erroneous data (cleaning), applying business rules, checking data integrity, and creating aggregates as necessary. Below are the steps for data transformation:

1. Splitting: Splitting a single column into multiple columns.

For columns that have multiple entries in the same row, first we need to separate them in R using function 'seperate_row': credit_rating_agency, establish_year, credit_score since they have multiple entries in the same row, phone_numbers, payment_account_types, payment_account_holder_names, payment_account_numbers,payment_bank_names

2. Deduplication: Identifying and removing duplicate records, create unique identifiers.

For columns that need to separate from the original dataset and create new tables in the database in order to achieve 3NF, we need to assign a unique identifier for each of them: employment_id, credit_rating_agency_id, credit_info_id, prosper_index_id, borrow_liability_id, delinquency_history_id, income_info_id, nominee_id, phone_id, payment_bank_id, payment_method_id, investor_id, account_statement_id

3. Cleaning: convert data types to maintain data format consistency.

Data in the investor sections are simple text data, which is not suitable for our schema and may cause trouble loading in. Thus, we need to convert them into proper data types according: date_of_birth, transaction_date, transaction_amount, and closing_date.

4. Missing Values.

For some of the columns that contain NA values, we need to find the best way to deal with those missing values. Our dataset contains few missing values, also, according to our scenario, there is no need to fill all the missing values because the real-world data is not perfect most of the time, and it is reasonable that some fields are empty in the dataset. For example, ssn, phone_number, zip_code, etc., these features are meaningless for the following process of conducting analysis and getting insights.

4.1.3 Load

Loading data to the target multidimensional structure is the final step in the ETL process. In this step, extracted and transformed data is written into the dimensional structures actually accessed by the end users and application systems, where they can be integrated, rearranged, and consolidated, creating a new type of unified information base for reports and reviews.

Below is the code for loading data into our database:

```
###import investor data
library(readxl)
investor <- read_excel("~/Downloads/SQL_Project_Group1.xlsx",
                       sheet = "Investor", col_types = c("numeric",
                                                         "text", "text", "text", "date", "text",
                                                         "text", "text", "text", "text", "text",
                                                         "text", "text", "text", "text", "text",
                                                         "text", "text", "text", "text", "date",
                                                         "text", "text", "numeric", "date",
                                                         "numeric"))library(dplyr)
library(tidyr)
#assign account_statement_id
investor$X__1 <- NULL
investor <- bind_cols('account_statement_id' = sprintf('a%09d',1:nrow(investor)), investor)

#separate phone column
investor <- investor%>%
  separate_rows(phone_numbers)

#separate rows
investor <- investor %>%
```

```
  separate_rows(payment_account_types, payment_account_holder_names,
payment_account_numbers,
                payment_bank_names, sep= '\\|')
investor <- filter(investor, payment_account_types != '')
investor<- filter(investor, phone_numbers !='')
df<-investor

#load data
#nominee table
df1 <- df %>% select(nominee_first_name, nominee_last_name, nominee_date_of_birth,
nominee_relationship_with_investor) %>% distinct()
df2 <- bind_cols('nominee_id' = sprintf('n%09d', 1:nrow(df1)),df1)
dbWriteTable(con,name = 'nominee',value = df2, row.names = FALSE,append=TRUE)
df <- df %>% inner_join(y = df2, by = c('nominee_first_name','nominee_last_name'))

#phone table
df1 <- df %>% select('phone_number'= phone_numbers) %>% distinct()
df2 <- bind_cols('phone_id' = sprintf('p%09d', 1:nrow(df1)),df1)
dbWriteTable(con,name = 'phones',value = df2, row.names = FALSE,append=TRUE)
df <- df %>% inner_join(y = df2, by = c('phone_numbers'='phone_number'))

#payment_bank_name
df1 <- df %>% select('payment_bank_name'= payment_bank_names) %>% distinct()
df2 <- bind_cols('payment_bank_id' = sprintf('B%09d', 1:nrow(df1)),df1)
dbWriteTable(con,name = 'payment_bank',value = df2, row.names = FALSE,append=TRUE)
df <- df %>% inner_join(y = df2, by = c('payment_bank_names'='payment_bank_name'))
```

(Please check the detailed codes on GitHub, we only post a sample code for creating table in this report, GitHub link: https://github.com/suzygaoo/P2P-lending-platform-database-system)

```
###import borrower data
df <- read_xlsx("~/Desktop/SPRING 19/SQL/project/borrower_new.xlsx")
df <- separate_rows(df, credit_rating_agency,establish_year,credit_score,
                    sep = '\\|')

#separate phone column
df <- df%>%
  separate_rows(phone_numbers)

#phone table
df1 <- df %>% select('phone_number'= phone_numbers) %>% distinct()
df2 <- bind_cols('phone_id' = sprintf('p%09d', 1:nrow(df1)),df1)
dbWriteTable(con,name = 'b_phones',value = df2, row.names = FALSE,append=TRUE)
df <- df %>% inner_join(y = df2, by = c('phone_numbers'='phone_number'))

#borrower
df1 <- df %>% select(borrower_id,first_name,last_name,state,city,
```

```
        street_address,zip_code) %>% distinct()
dbWriteTable(con,name = 'borrower', value = df1, row.names = FALSE, append = TRUE)

#loan_ticket
df1 <- df %>% select(loan_ticket_id,borrower_id,loan_amount,loan_tenure_in_month,reason_for_loan,
        borrow_rate,borrower_apr,loan_status) %>% distinct()
dbWriteTable(con,name = 'loan_ticket', value = df1, row.names = FALSE, append = TRUE)
```

(Please check the detailed codes on GitHub, we only post a sample code for creating table in this report, GitHub link: https://github.com/suzygaoo/P2P-lending-platform-database-system)

```
###import investor_proposal
SQL_Project_investor_proposal <- read_csv("investor_proposal.csv")
df <- SQL_Project_Investor_proposal
df1 <- df %>% select(loan_ticket_id, investor_id) %>% distinct()
df2 <- bind_cols('investor_proposal_id' = sprintf('ip%09d', 1:nrow(df1)), df1)
dbWriteTable(con, name = 'investor_proposal', value = df2, row.names = FALSE, append = TRUE)
```

4.2 GitHub repo

GitHub link: https://github.com/suzygaoo/P2P-lending-platform-database-system

5 Analytical Procedures / SQL and R Code

Now we have formed the final database, and we plan to help our customers to interact with the database system we designed.

For analysts, we will help them learn how to make changes to the database, and how to utilize our database for analyzing the operational conditions and problems of the company. The tool we use is SQL, to be specific, analysis is conducted from the borrower side and investor side, and the implements and queries are as follow.

5.1 Analytical procedures and code for Borrower section

1. Find the borrowers whose monthly income is ranked in the top 5 for each of the states with the top 5 numbers of borrowers. These people are the target borrowers of the marketing campaign.

```
SELECT STATE, BORROWER, INCOME
FROM
(SELECT i.state AS STATE, i.borrower_id AS BORROWER,
i.stated_monthly_income AS INCOME, RANK()OVER(PARTITION BY i.state ORDER BY
i.stated_monthly_income DESC) AS b_rank, row_number
```

```
FROM (SELECT i.stated_monthly_income, b.borrower_id, b.state
FROM income_info i JOIN borrower b on i.borrower_id = b.borrower_id) AS i JOIN
(SELECT state, ROW_NUMBER()over(ORDER BY a_rank ASC) AS row_number
FROM
(SELECT state, RANK()OVER(ORDER BY COUNT(borrower_id) DESC) AS a_rank
FROM borrower
GROUP BY state) AS FOO
WHERE a_rank <= 5) AS c ON i.state = c.state) AS FOO
WHERE b_rank <= 5
ORDER BY row_number ASC, INCOME DESC;
```

2. Find the borrowers whose monthly income is ranked in the lowest 5 for each of the states with the top 5 numbers of borrowers. These people should be regularly managed and checked in case they do not have enough money to pay back the loan on time.

```
SELECT STATE, BORROWER, INCOME
FROM
(SELECT i.state AS STATE, i.borrower_id AS BORROWER,
i.stated_monthly_income AS INCOME, RANK()OVER(PARTITION BY i.state ORDER BY
i.stated_monthly_income ASC) AS b_rank, row_number
FROM (SELECT i.stated_monthly_income, b.borrower_id, b.state
FROM income_info i JOIN borrower b on i.borrower_id = b.borrower_id) AS i JOIN
(SELECT state, ROW_NUMBER()over(ORDER BY a_rank ASC) AS row_number
FROM
(SELECT state, RANK()OVER(ORDER BY COUNT(borrower_id) DESC) AS a_rank
FROM borrower
GROUP BY state) AS FOO
WHERE a_rank <= 5) AS c ON i.state = c.state) AS FOO
WHERE b_rank <= 5
ORDER BY row_number ASC, INCOME ASC;
```

3. Count the average loan amount for each occupation. This help to recognize the target occupations, for instance, if students are in large need of loans, Prosper could organize its marketing campaign in schools.

```
SELECT e.occupation, ROUND(avg(l.loan_amount),0) AS loan_amount
FROM employment e JOIN loan_ticket l on e.borrower_id = l.borrower_id
GROUP BY e.occupation
ORDER BY loan_amount DESC;
```

4. The average score of each credit rating agency. This helps recognize the different rating standards for each credit rating agency.

```
SELECT credit_rating_agency,
ROUND(AVG(credit_score),0) AS credit_score
```

```
FROM credit_info
JOIN credit_rating_agency USING(credit_rating_agency_id)
GROUP BY credit_rating_agency;
```

5. Top 5 states and cities which have the highest borrower average monthly income.

```
SELECT state, city, COUNT(borrower_id),ROUND(AVG(stated_monthly_income)) AS monthly_income
FROM borrower
JOIN income_info USING(borrower_id)
GROUP BY state,city
ORDER BY COUNT(borrower_id) DESC
LIMIT 5;
```

6. Reason for loans which generate the most delinquent.

```
SELECT reason_for_loan, ROUND(AVG(amount_delinquent),0)
FROM loan_ticket
JOIN delinquency_history USING(borrower_id)
GROUP BY reason_for_loan
ORDER BY AVG(amount_delinquent) DESC;
```

7. The gap between borrower's monthly payment and loan amount each month. This helps to ensure the security of cash flow.

```
SELECT EXTRACT (MONTH FROM liability_start_date ) AS month, SUM(monthly_repayment_amount)
AS monthly_repayment, SUM(loan_amount) AS loan_amount
FROM borrow_liability
JOIN loan_ticket USING(loan_ticket_id)
GROUP BY month
ORDER BY month ASC;
```

8. Transform credit rating scores into 8 levels(AA, A, B, C, D, E, F, N).

```
CREATE OR REPLACE FUNCTION score_to_letter (prosper_rating numeric (2,1))
RETURNS varchar(2) AS
$fun$
DECLARE rating_lvl varchar(2);
BEGIN
 IF $1 = '1' THEN rating_lvl = 'F';
 ELSEIF $1 = '2' THEN rating_lvl = 'E';
 ELSEIF $1 = '3' THEN rating_lvl = 'D';
 ELSEIF $1 = '4' THEN rating_lvl = 'C';
 ELSEIF $1 = '5' THEN rating_lvl = 'B';
 ELSEIF $1 = '6' THEN rating_lvl = 'A';
 ELSEIF $1 = '7' THEN rating_lvl = 'AA';
```

```
 ELSE rating_lvl ='N';
 END IF;
RETURN rating_lvl;
END;
$fun$
language plpgsql;

SELECT *, score_to_letter (prosper_rating) Rating_Lvl
FROM prosper_index;
```

9. Transform the income information into High/Moderate/Low groups and calculate the size of each group.

```
CREATE OR REPLACE FUNCTION income_to_lvl (stated_monthly_income numeric (10,2))
RETURNS varchar(2) AS
$fun$
DECLARE income_lvl varchar(2);
BEGIN
 IF $1 < 3000 THEN income_lvl = 'L';
 ELSEIF $1 < '6000' THEN income_lvl = 'M';
 ELSE income_lvl ='H';
 END IF;
RETURN income_lvl;
END;
$fun$
language plpgsql;

SELECT *, income_to_lvl (stated_monthly_income) Income_Lvl
FROM income_info;

SELECT COUNT( income_to_lvl (stated_monthly_income)) AS H
FROM (SELECT *,income_to_lvl (stated_monthly_income) Income_Lvl
 FROM income_info) AS a
WHERE Income_Lvl = 'H';

SELECT COUNT( income_to_lvl (stated_monthly_income)) AS M
FROM (SELECT *,income_to_lvl (stated_monthly_income) Income_Lvl
 FROM income_info) AS a
WHERE Income_Lvl = 'M';

SELECT COUNT( income_to_lvl (stated_monthly_income)) AS L
FROM (SELECT *,income_to_lvl (stated_monthly_income) Income_Lvl
FROM income_info) AS a
WHERE Income_Lvl = 'L';

SELECT COUNT( income_to_lvl (stated_monthly_income)) AS H
```

```
FROM (SELECT *,income_to_lvl (stated_monthly_income) Income_Lvl
FROM income_info) AS a
WHERE Income_Lvl = 'H';
```

10. Top 5 states with the largest numbers of borrowers. For where to set new branches and expand business in the future, these states are the first choice.

```
SELECT b.state, COUNT(borrower_id)
FROM borrower b
GROUP BY b.state
ORDER BY COUNT(borrower_id) DESC
LIMIT 5;
```

5.2 Analytical procedures and code for Borrower section

1. Average investment limit for investors in different aging ranges.

```
CREATE OR REPLACE FUNCTION dob_to_age(date_of_birth date)
RETURNS varchar(20) AS
$fun$
DECLARE age varchar(20);
BEGIN
  IF $1 BETWEEN '1960-01-01' AND '1969-12-31' THEN age = '60s';
  ELSEIF $1 BETWEEN '1970-01-01' AND '1979-12-31' THEN age = '70s';
  ELSEIF $1 BETWEEN '1980-01-01' AND '1989-12-31' THEN age = '80s';
  ELSEIF $1 BETWEEN '1990-01-01' AND '1999-12-31' THEN age = '90s';
  END IF;
 RETURN age;
 END;
$fun$
language plpgsql;

SELECT *, dob_to_age(date_of_birth)
FROM investor;

SELECT dob_to_age(date_of_birth), ROUND(AVG(investment_limit), 2)
FROM investor
GROUP BY dob_to_age(date_of_birth)
ORDER BY ROUND(AVG(investment_limit), 2) DESC;
```

2. Top 5 cities with the largest number of investors.

```
SELECT city, COUNT(investor_id)
FROM investor
GROUP BY city
ORDER BY COUNT(investor_id) DESC
```

3. The number of investors of each payment bank. If the result shows a large difference between different payment banks, Prosper could recognize the payment bank which has the most investors and consider to have a long-term cooperative relationship with this bank.

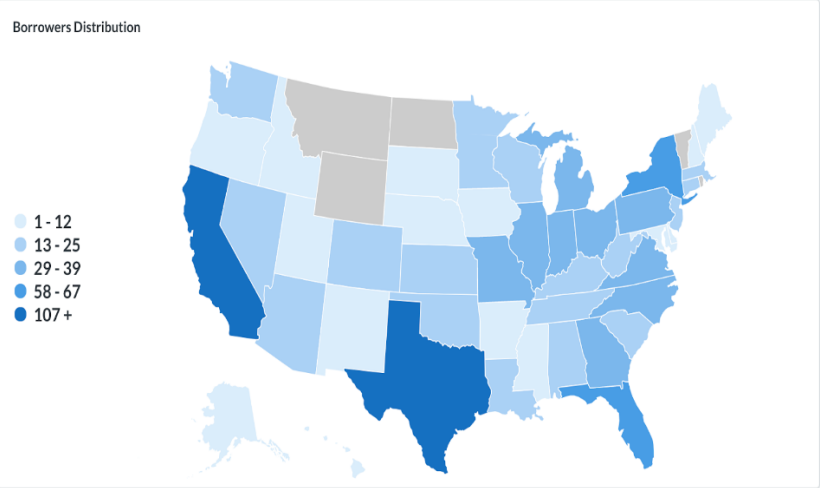
```
SELECT payment_bank_name, COUNT(investor.investor_id)
FROM investor
JOIN payment_method ON investor.investor_id = payment_method.investor_id
JOIN payment_bank ON payment_method.payment_bank_id = payment_bank.payment_bank_id
GROUP BY payment_bank_name
ORDER BY COUNT(investor.investor_id) DESC;
```

5.3 Conclusion for analytical procedures

In the database we design, our clients can check the investors' and borrowers' detailed information which covers the aspects of income, credit, account statement, demographic info, etc. The analytics team can simply check or filter data by SELECT, FROM, or CREATE VIEW function to define a view of a query. Furthermore, our database allows them to update or delete data for their real-time requirements. We have also designed an interactive dashboard on metabase which allows our client to interact with visualized diagrams. This dashboard will help the "C" level officers to intuitively understand the key features we have identified for borrowers and investors, as well as some insightful information that we think might be helpful for company managers in making business strategies.

In addition, we have proposed 13 analytical procedures including the customer portals of borrowers and investors, the gap between loan and payment in each month, geographical portals and income portals of borrowers and investors, and information about credit rating agencies. These analytical procedures can give a comprehensive knowledge for Prosper analysts team to understand the current situation of its users and the market by running the query in PostgreSQL query tool or in programming language R.

What's more, based on our analytics, we have built a dashboard for the "C" level officers and generated applicable strategies for management. For instance, by comparing the monthly loan out amount and repayment amount, we can find an obvious gap between them which might cause cash flow problem or funding mismatch, executives can easily find this problem by visualization, and immediately propose strategies to solve it.



Additionally, from geographical data, we find that most of the borrowers live in New York, Texas, California, and Florida and there are 4 other states with no borrowers. Therefore, Prosper should consider extending its business to these 4 unoccupied states to earn a leading position in the local market. Take these analyses as examples, we could create an authentic report for the C suits and assist them in making strategic decisions and plans.

Although our database is well designed enough for all non-technical people to interact with due to the metabase dashboard visualization for all users even without a technical background, we still choose to use R as the main programming language, the benefit of performing database actions with programming language is to conduct actions faster and easier to modify.

Considering the possible redundancy and performance problems, we test the compliance of data when needing to add new data into the database, and we have transformed some features from scattered distributing values to categories and this has benefited our analysis and revealing process. We do have the same data stored in multiple tables and some of the information in our charts are repetitive but can generate different insights based on a different scenario. The efficiency of the data retrieval process is relatively high because of the logical design. Our clients can approach this database on the cloud by connecting with Cloud SQL, they can also make an on-premise database by setting up an enterprise internal database center.

We have also set up an interactive dashboard on Tableau, which does an excellent job in visualization.

5.4 Create view

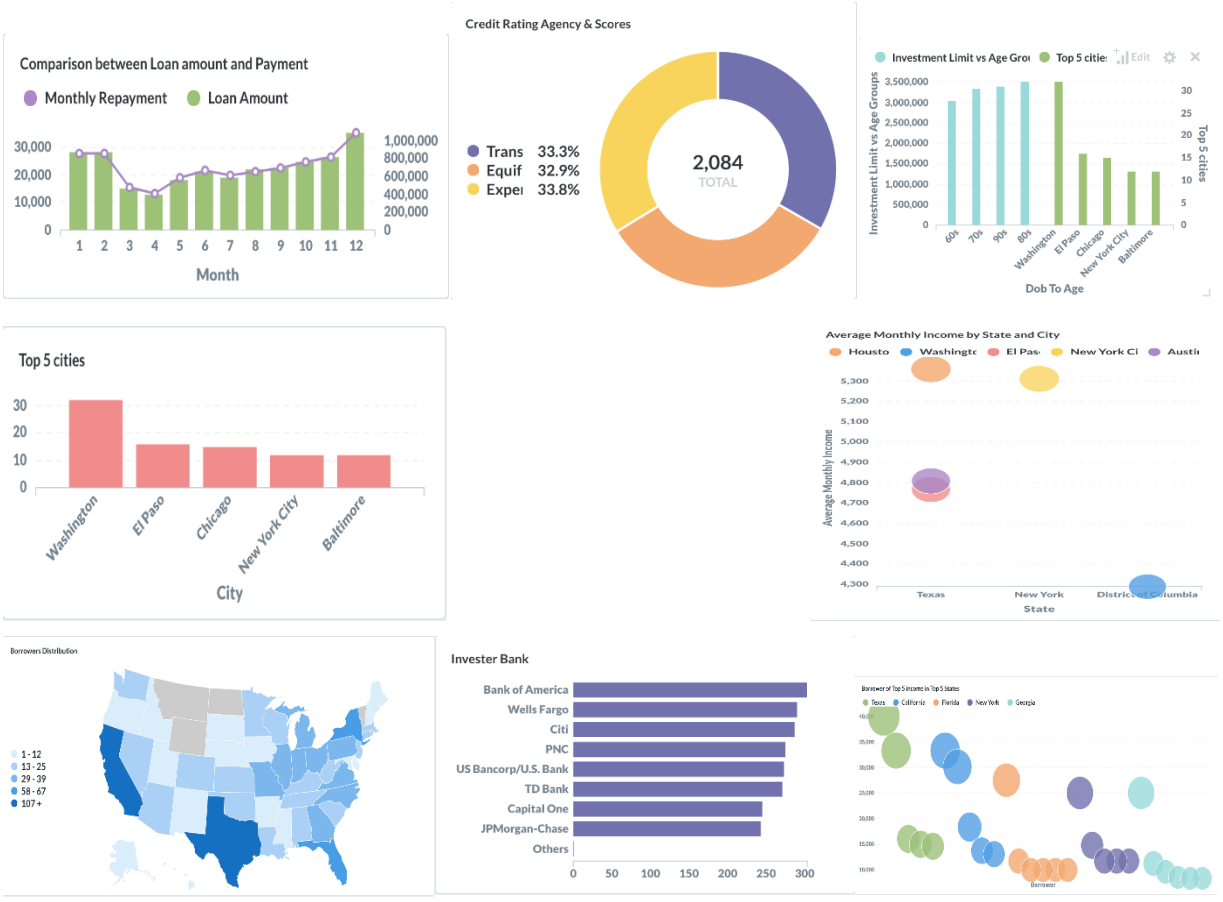
Creating view is a useful method commonly used for reporting purposes and ensuring data security at the same time. This helps analysts pull out necessary data from the database more efficiently, it allows employees to create read-only views to expose read-only data to specific users. Below is the SQL code for creating a view that shows borrower's information as well as their credit rating information:

```
CREATE VIEW borrower_income_credit_rating_details AS
        SELECT b.borrower_id, b.first_name, b.last_name, b.zip_code, e.occupation,
e.employment_status_duration,
                        c.current_credit_lines, c.credit_score,r.credit_rating_agency
        FROM borrower b
        JOIN employment e ON b.borrower_id = e.borrower_id
        JOIN income_info i ON b.borrower_id = i.borrower_id
        JOIN credit_info c ON b.borrower_id = c.borrower_id
        JOIN credit_rating_agency r ON c.credit_rating_agency_id = r.credit_rating_agency_id;
```

6 Dashboards / Conclusion / Recommendation

6.1 Dashboards

6.1.1 Metabase dashboard



Metabase Dashboard: http://s19db.apan5310.com:3201/public/dashboard/ee8a48c2-7d83-4229-ba12-619efa7586ce

6.1.2 Tableau dashboard

Tableau Dashboard: https://public.tableau.com/profile/jia.yang#!/

6.1.3 Insights from dashboards

Our dashboard focus on two kinds of information: demographic and risk-related information of borrowers and investors.

Figure 1 shows the distribution of our borrowers in different states and their average credit score. Another part of Figure 1 shows the distribution of our investors and the fund they invest.

Figure 2 demonstrates a variety of risk metrics of the users who borrow more than $100,000 in Prosper. The risk metrics include credit score, number of delinquents, amount of delinquent and borrow apr.

Figure 3 shows the income condition of our borrowers, which is an important financial metrics to know our customers

Figure 4 shows the capital flow of Prosper. The bar chart means the inflow of capital from investors while the blue line shows the outflow of the loan amount to our borrowers. This We know the balance of our cash every year.

Figure 5 shows the age distribution of our borrowers, which could help us know the demographic information of our borrowers.

Figure 6 shows the 10-year treasury maturity rate, which is an important financial index that affects loan rate and our business.

6.2 Conclusion

Our goal is to store, organize and analyze all the data of Prosper and get useful insights for the company to know the clients and manage the loan risk. In RDBMS, data are stored in columns and rows, which makes it easy to access and manage data. RDBMS can preserve the integrity of data and provide a review which can hide sensitive tables and protect our data. The process of ETL is to integrate data and combine different data in a meaningful way so we can conduct analysis on them.

A better understanding of our customers, such as their age, incomes, and locations, could help Prosper to know who are our customers, hot to choose better media channels to reach our customers, and attract more clients.

To a peer to peer lending company, risk management is the most essential part of the whole business.

The risk metrics in our dashboard, which include the loan amount, credit score and delinquent information, help Prosper to measure and monitor the risk of their loans. With these dashboards, Prosper can easily find out the borrowers with a high risk of default. For those high-risk borrowers, such as borrowers have a large amount of delinquents, low credit scores and large loan, Prosper could take actions to mitigate this risk.

6.3 Recommendation

1. Since the company does not have a business in Wyoming, North Dakota, Montana, and Vermont, so the management team should work on creating strategies to extend its business into uncovered states.

2. There are serious cash flow gaps between loan out amount and repayment amount which might cause a capital mismatch. Therefore, the Prosper management team should take care of the cash flow condition and prepare strategies for solving this problem.

3. For those borrowers with a high probability of default, such as b46587 and b12654, whose credit score is around 500 and enjoy below average loan rate, we should charge them higher rates to hedge the default risk.

4. Based on the analytic, we found most of our investors' ages are between 20-40 so Prosper could use more online media channels such as social media and PPC (Par per Click) marketing channels to reach more target investors. On the other side, we could use the demographic data to know our target borrowers and adjust media plan. Given the fact that Prosper doesn't have the age data of our borrowers, Prosper should collect those data in the future to conduct data analytics and create a more efficient marketing strategy.

7 References

[1] https://www.sciencedirect.com/science/article/pii/S131915781100019X

[2] https://www.vertabelo.com/blog/technical-articles/connecting-borrowers-and-lenders-a-peer-to-peer-lending-platform-data-model

[3] https://towardsdatascience.com/p2p-lending-platform-data-analysis-exploratory-data-analysis-in-r-part-1-32eb3f41ab16

8 Appendix

[1] ER Diagram

9 Links

[1] Dataset: https://docs.google.com/spreadsheets/d/1edot-MfpCHP-DgHhS36L-U7C4if-it6kukOwHx_Mm-k/edit?usp=sharing

[2] Original data downloading: https://s3.amazonaws.com/udacity-hosted-downloads/ud651/prosperLoanData.csv

[3] GitHub: https://github.com/suzygaoo/P2P-lending-platform-database-system)

[4] Lucidchart: https://www.lucidchart.com/invitations/accept/20158b3f-86ce-4ca7-a47a-e8a8bb58b225

[5] Metabase Dashboard: http://s19db.apan5310.com:3201/public/dashboard/ee8a48c2-7d83-4229-ba12-619efa7586ce

[6] Tableau Dashboard: https://public.tableau.com/profile/jia.yang#!/

Appendix 1 ER Diagram