Advanced Data Science & Architecture

Loan Data Analysis

- *Under the guidance of Sri Krishnamurthy*

Compiled By,

Mohit Mittal
Sneha Ravikumar
Taj Poovaiah

CONTENTS

ABOUT APPLICATION:

Programming Language used : Python

Workflow Manager User: Luigi

Tasks

A. Downloading Loan Data (GetData.py)
B. Clean Loan Data (CleanData.py)
C. Upload Preprocessed data to Amazon S3 (LoanData.py)

STEPS TO RUN THE PIPELINE

1. Pull the Docker image (mohit914/test:LCv1.05) from the dockerhub
   docker pull mohit914/test:LCv1.05
2. Run the docker image. It will take you into the bash terminal
   docker run -ti mohit914/test:LCv1.05
3. Inside the bash terminal run the python application by using the following command
   python LoanData.py Start –local-scheduler
   python RejectLoanData.py Start –local-scheduler
   This program will run the following tasks automatically:

i. **GetData()** – It will download the LendingClub.com loan dataset. **You will need to provide the following credentials – username: sneha.ravi12@gmail.com, password: Snehar123! Or create your own account**

ii. **ClenaData()** – It will clean and preprocess the LendingClub loan dataset

iii. **Start()** – It will upload the preprocessed data to Amazon S3 Bucket. ***You will need to provide your Amazon Access key and Secret Key to complete this task.***

Docker Trouboulshooting:

1. *If running docker gives No space on the device error, then remove all the images and follow the steps to run the pipeline again. If problem still exists, remove all the images and containers on your device and follow the steps to run the pipeline again.*
2. *If the problem still exists, please email us to let us know the problem so that we can try to troubleshoot the problem.*

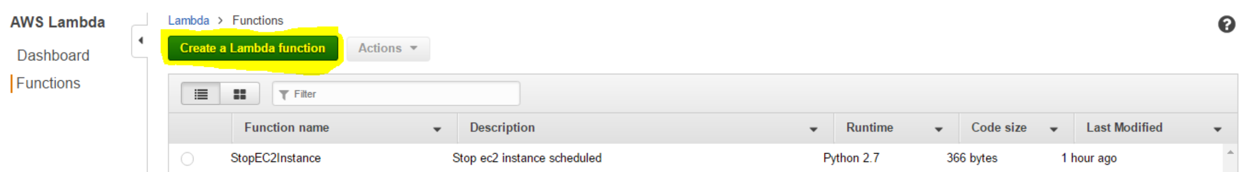
DOCKER ON AMAZON AWS SCHEDULING:

Docker on AWS – Scheduling

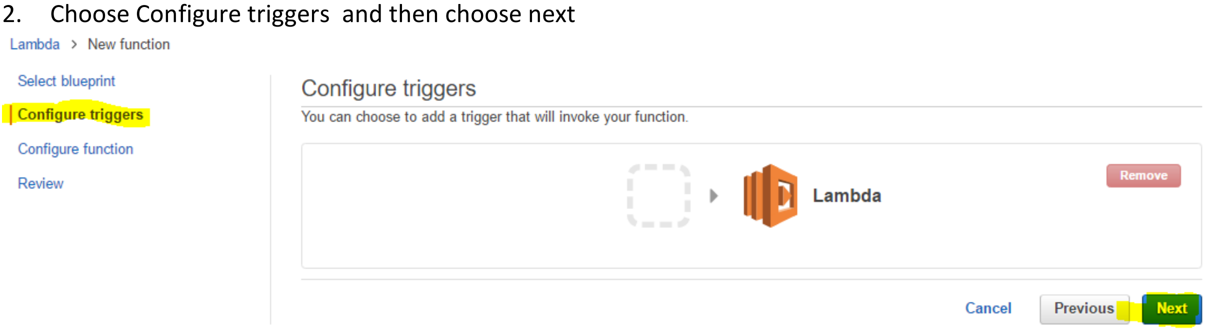
1. Connect to Amazon Linux AMI instance
2. Update the installed package cache on the instance
   sudo yum update -y
3. Install Docker
   sudo yum install -y docker
4. Start the docker service
   sudo service docker start
5. Add the ec2-user to the docker group so that you can execute Docker commands without using **sudo**
   usermod -a -G docker ec2-user
6. Log out and log back in again to pick up the new docker group permissions
7. Pull the docker image from dockerhub
   docker pull mohit914/test:LCv1.05

To schedule the task to start and stop the ec2 instance

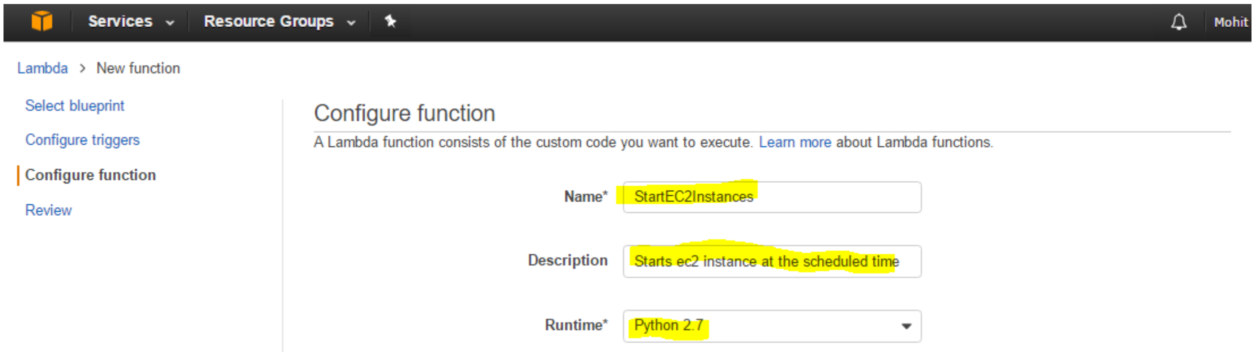
1. Create a Lambda function by going on the AWS Lambda console and clicking on Create Lambda function.



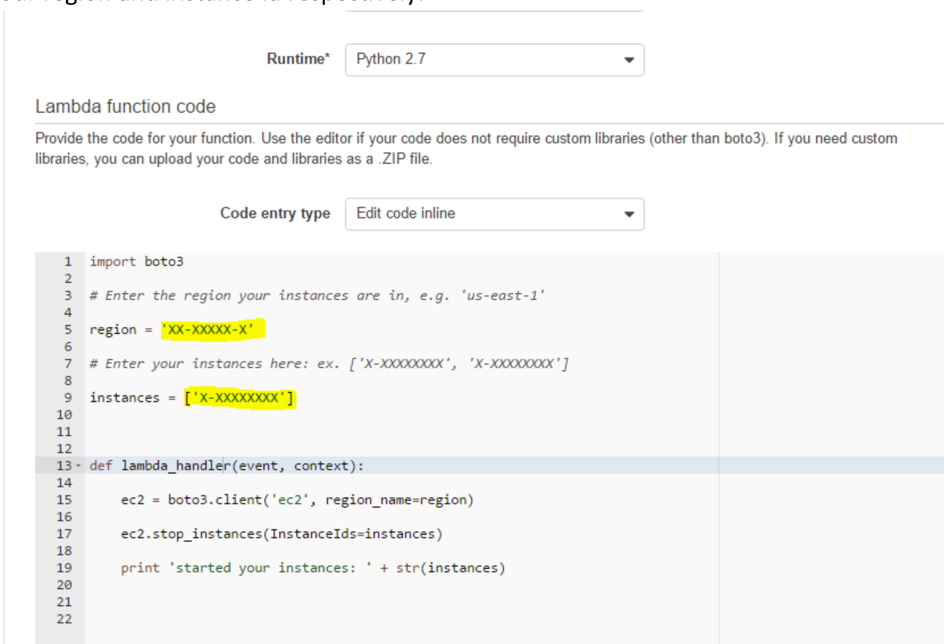
2. Choose Configure triggers and then choose next



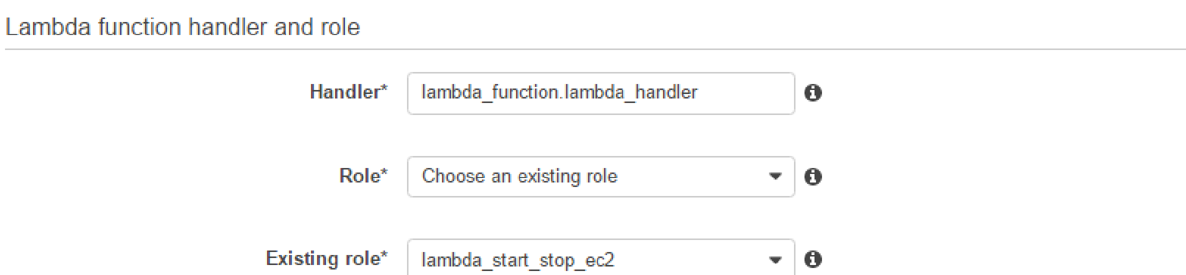
3. Enter the values for name and description of the function and select python 2.7.



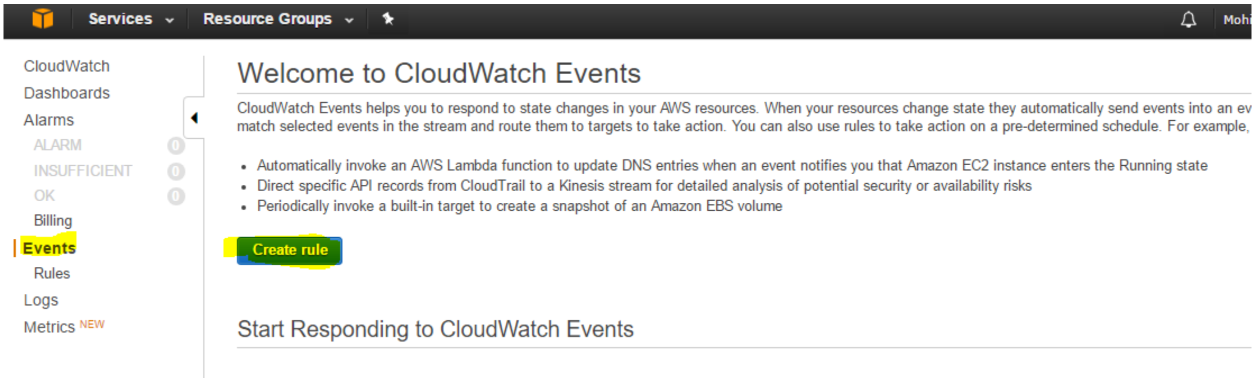
4. Enter the code to start the instance in the "Lambda function code" section below. Replace the highlighted values with your region and instance id respectively.



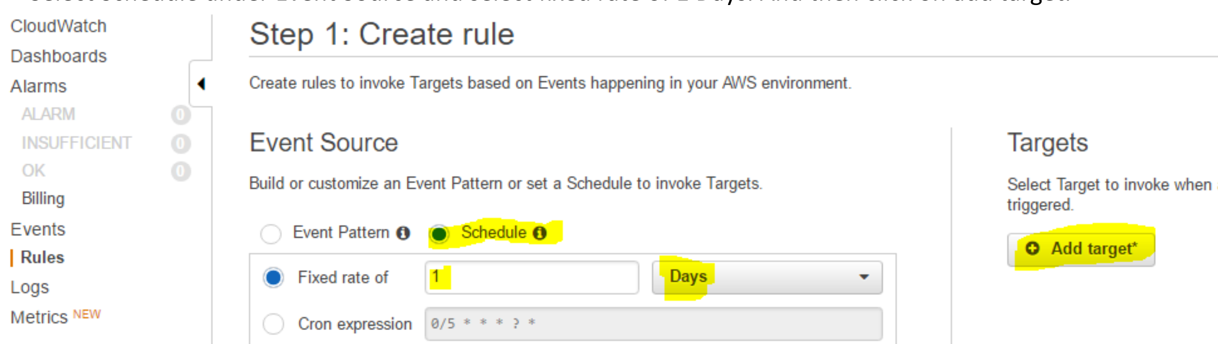
5. Select the role (create if it doesn't exist to start and stop the ec2 instance)



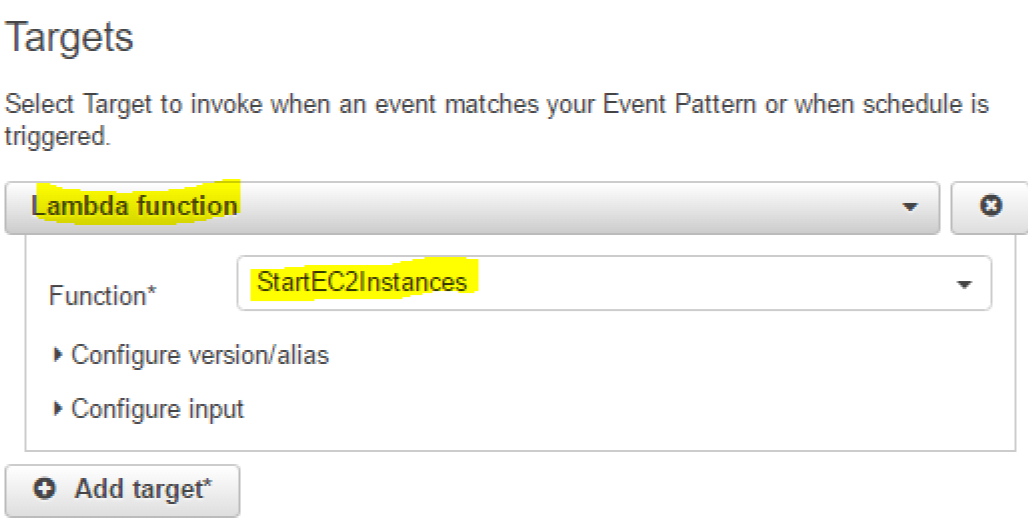
6. Click next and click on Create function
7. Open CloudWatch console and select Events.



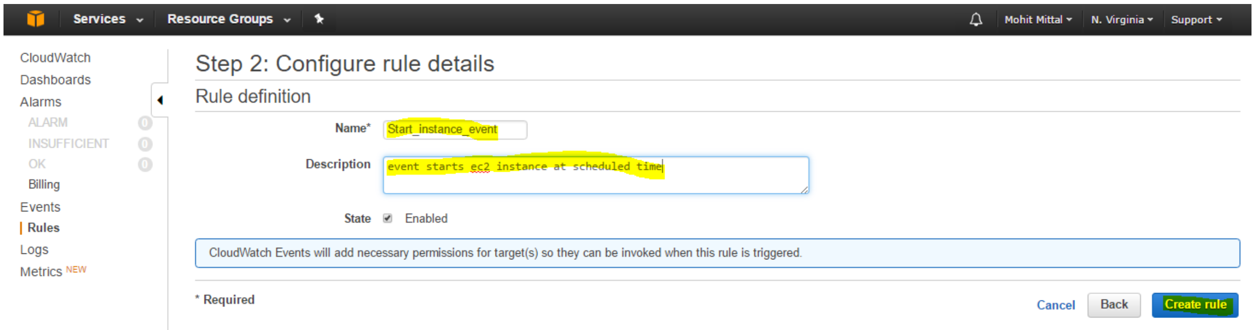
8. Select Schedule under Event Source and select fixed rate of 1 Days. And then click on add target.



9. Select Lambda Functions and the function name.



10. Click on configure details. And enter the name and description for the event.



11. Follow the same steps to create an event to stop the instance on scheduled time.

DOWNLOADING DATA:

File Location: Classes/LoanData/GetData.py

Task Requires no prior tasks to be completed.

Output of the task are all the Loan Data files.

Process:

- Asking user for username and password.
- Creating a browser agent (using the mechanicalsoup library) to store and pass the cookies
- Logging in with the user's credentials
- Checking if the user is successfully logged in or not.
- Landing to the page that contains the list of files and download links
- Putting the table of files in a dataframe
- Iterating through the rows in dataframe for the links that contain sample files and downloading them to a newly created (if it doesn't already exist) "Downloads" directory
- The program also checks if the files are already present in the "Downloads" directory. It skips the downloading if the file already exists.
- Unzipping the downloaded file.

The Downloaded data can be found in the directory : "Data/Downloads/"

CLEANING DATA: LOAN DATA SET

SUMMARY OBSERVATIONS AFTER DOWNLOAD

We obtained the summaries for Loan data after download and after cleaning. We see that the count of missing values have been changed. All the columns after cleaning have the same number of values as seen by the count value. The details of how we are handling missing data in each column is given after "summary of data after cleaning"

	count	mean	std	min	25%	50%	75%	max
member_id	0							
loan_amnt	1321847	14748.22	8622.143	500	8000	12900	20000	40000
funded_amnt	1321847	14739.22	8618.488	500	8000	12850	20000	40000
funded_amnt_inv	1321847	14710.71	8625.729	0	8000	12800	20000	40000
installment	1321847	439.1395	253.2311	14.01	255.9	380.25	578.31	1584.9
annual_inc	1321843	76495.94	69111.92	0	46000	65000	90000	9573072
url	0							
dti	1321847	18.87239	70.98519	-1	12.04	17.83	24.24	9999
delinq_2yrs	1321818	0.329594	0.892043	0	0	0	0	39
inq_last_6mths	1321817	0.650875	0.957663	0	0	0	1	33
mths_since_last_delinq	662677	33.89481	21.89243	0	15	31	49	195
mths_since_last_record	219977	68.9857	26.86579	0	51	70	88	129
open_acc	1321818	11.6559	5.46941	0	8	11	14	97
pub_rec	1321818	0.215028	0.617163	0	0	0	0	86
revol_bal	1321847	16928.01	22933.9	0	6295	11652	20584	2904836
total_acc	1321818	25.02217	11.90355	1	16	23	32	176
out_prncp	1321847	5971.41	7531.106	0	0	2927.55	9990.595	40205.27
out_prncp_inv	1321847	5968.963	7528.321	0	0	2926.1	9986.6	40205.27
total_pymnt	1321847	10065.93	8677.782	0	3677.26	7498.63	13686.11	61501.88
total_pymnt_inv	1321847	10038.57	8667.587	0	3660.27	7472.47	13647.27	61414.02
total_rec_prncp	1321847	7790.732	7367.401	0	2408.46	5319.5	10573.48	40000
total_rec_int	1321847	2192.841	2367.291	0	689.14	1430.24	2798.65	26501.88
total_rec_late_fee	1321847	0.629722	5.518013	0	0	0	0	437.99
recoveries	1321847	81.7311	533.0982	0	0	0	0	39444.37
collection_recovery_fee	1321847	12.08557	89.39024	0	0	0	0	7002.19
last_pymnt_amnt	1321847	2840.73	5556.066	0	298.14	505.6	1327.635	42148.53
collections_12_mths_ex_med	1321702	0.016785	0.146722	0	0	0	0	20
mths_since_last_major_derog	347331	44.41145	22.28005	0	27	44	62	197
policy_code	1321847	1	0	1	1	1	1	1
annual_inc_joint	9300	111563.6	50199.77	11000	78492.5	103000	134000	1050000
dti_joint	9296	18.29942	7.020419	0.32	13.32	18.01	22.9	69.49
acc_now_delinq	1321818	0.005582	0.081198	0	0	0	0	14
tot_coll_amt	1251571	241.1676	8444.208	0	0	0	0	9152545
tot_cur_bal	1251571	140842.8	156363.7	0	30017	80847	209787.5	8000078
open_acc_6m	455717	1.008288	1.188975	0	0	1	2	18
open_il_6m	455718	2.831942	3.052855	0	1	2	3	48
open_il_12m	455718	0.727941	0.978708	0	0	0	1	25
open_il_24m	455718	1.629345	1.657855	0	0	1	2	51
mths_since_rcnt_il	443459	21.27134	26.62149	0	7	13	24	511
total_bal_il	455718	35619.87	42570.6	0	9495	23781	46364	1547285
il_util	395615	70.98847	23.17377	0	58	74	87	1000
open_rv_12m	455718	1.373542	1.536616	0	0	1	2	28

open_rv_24m	455718	2.903416	2.631446	0	1	2	4	60
max_bal_bc	455718	5777.328	5619.377	0	2362	4385	7460	776843
all_util	455695	60.22548	20.15962	0	47	61	74	204
total_rev_hi_lim	1251571	32764.69	36690.14	0	14100	24100	40600	9999999
inq_fi	455718	0.965009	1.497837	0	0	0	1	48
total_cu_tl	455717	1.498709	2.722067	0	0	0	2	111
inq_last_12m	455717	2.166066	2.45905	0	0	1	3	49
acc_open_past_24mths	1271817	4.551259	3.120103	0	2	4	6	64
avg_cur_bal	1251559	13365.83	16004.96	0	3149	7424	18518	958084
bc_open_to_buy	1259244	9547.611	14582.59	0	1345	4359	11520	711140
bc_util	1258527	62.10613	27.51823	0	41.6	65.8	86.2	339.6
chargeoff_within_12_mths	1321702	0.008963	0.10839	0	0	0	0	10
delinq_amnt	1321818	15.61729	812.8777	0	0	0	0	185408
mo_sin_old_il_acct	1214312	127.0696	52.01166	0	100	130	153	724
mo_sin_old_rev_tl_op	1251570	184.3689	94.84415	2	119	167	234	901
mo_sin_rcnt_rev_tl_op	1251570	13.44386	16.80114	0	4	8	16	438
mo_sin_rcnt_tl	1251571	8.086934	9.06588	0	3	6	10	314
mort_acc	1271817	1.689627	2.013544	0	0	1	3	61
mths_since_recent_bc	1260066	24.61952	31.49733	0	6	14	29	639
mths_since_recent_bc_dlq	318765	39.41078	22.75385	0	20	38	58	195
mths_since_recent_inq	1139670	6.833985	5.923765	0	2	5	10	25
mths_since_recent_revol_delinq	448529	35.46095	22.41687	0	17	32	52	197
num_accts_ever_120_pd	1251571	0.505305	1.309245	0	0	0	0	51
num_actv_bc_tl	1251571	3.714223	2.264803	0	2	3	5	47
num_actv_rev_tl	1251571	5.778005	3.334688	0	3	5	7	59
num_bc_sats	1263257	4.745405	2.942177	0	3	4	6	71
num_bc_tl	1251571	8.150501	4.788671	0	5	7	11	79
num_il_tl	1251571	8.514671	7.352151	0	3	7	11	159
num_op_rev_tl	1251571	8.319066	4.541131	0	5	7	11	91
num_rev_accts	1251570	14.68883	8.080868	0	9	13	19	118
num_rev_tl_bal_gt_0	1251571	5.731843	3.253593	0	3	5	7	59
num_sats	1263257	11.70843	5.475699	0	8	11	14	97
num_tl_120dpd_2m	1202848	0.000865	0.031176	0	0	0	0	6
num_tl_30dpd	1251571	0.003876	0.0664	0	0	0	0	4
num_tl_90g_dpd_24m	1251571	0.090736	0.506308	0	0	0	0	39
num_tl_op_past_12m	1251571	2.109639	1.804914	0	1	2	3	32
pct_tl_nvr_dlq	1251418	94.0675	8.79095	0	91.1	97.6	100	100
percent_bc_gt_75	1258823	47.45467	35.83138	0	16.7	50	75	100
pub_rec_bankruptcies	1320482	0.126787	0.372602	0	0	0	0	12
tax_liens	1321742	0.056615	0.418396	0	0	0	0	85
tot_hi_cred_lim	1251571	173672.7	176807.8	0	49789	112389	250854.5	9999999
total_bal_ex_mort	1271817	50299.27	47793.73	0	21430	37888	63251	2921551

total_bc_limit	1271817	21487.9	21304.29	0	7800	15000	28000	1105500
total_il_high_credit_limit	1251571	42211.78	43305.29	0	14825	31728	56712	2101913

SUMMARY OBSERVATIONS POST CLEANING

	count	mean	std	min	25%	50%	75%	max
member_id	0							
loan_amnt	1321847	14748.22	8622.143	500	8000	12900	20000	40000
funded_amnt	1321847	14739.22	8618.488	500	8000	12850	20000	40000
funded_amnt_inv	1321847	14710.71	8625.729	0	8000	12800	20000	40000
installment	1321847	439.1395	253.2311	14.01	255.9	380.25	578.31	1584.9
emp_length	1321847	5.769954	3.726066	0	2	6	10	10
url	0							
zip_code	1321847	51004.38	31154.91	700	22900	47200	80100	99900
delinq_2yrs	1321847	0.329587	0.892034	0	0	0	0	39
inq_last_6mths	1321847	0.65086	0.957657	0	0	0	1	33
mths_since_last_delinq	1321847	16.99237	22.9671	0	0	0	31	195
mths_since_last_record	1321847	11.48035	27.93378	0	0	0	0	129
open_acc	1321847	11.65589	5.469351	0	8	11	14	97
pub_rec	1321847	0.215023	0.617157	0	0	0	0	86
revol_bal	1321847	16928.01	22933.9	0	6295	11652	20584	2904836
total_acc	1321847	25.02163	11.90399	0	16	23	32	176
out_prncp	1321847	5971.41	7531.106	0	0	2927.55	9990.595	40205.27
out_prncp_inv	1321847	5968.963	7528.321	0	0	2926.1	9986.6	40205.27
total_pymnt	1321847	10065.93	8677.782	0	3677.26	7498.63	13686.11	61501.88
total_pymnt_inv	1321847	10038.57	8667.587	0	3660.27	7472.47	13647.27	61414.02
total_rec_prncp	1321847	7790.732	7367.401	0	2408.46	5319.5	10573.48	40000
total_rec_int	1321847	2192.841	2367.291	0	689.14	1430.24	2798.65	26501.88
total_rec_late_fee	1321847	0.629722	5.518013	0	0	0	0	437.99
recoveries	1321847	81.7311	533.0982	0	0	0	0	39444.37
collection_recovery_fee	1321847	12.08557	89.39024	0	0	0	0	7002.19
last_pymnt_amnt	1321847	2840.73	5556.066	0	298.14	505.6	1327.635	42148.53
collections_12_mths_ex_med	1321847	0.016783	0.146714	0	0	0	0	20
policy_code	1321847	1	0	1	1	1	1	1
annual_inc	1321847	76860.81	69223.66	1896	46053.25	65000	91400	9573072
dti	1321847	18.29373	8.379607	-1	12.02	17.8	24.17	69.49
acc_now_delinq	1321847	0.005582	0.081197	0	0	0	0	14
tot_coll_amt	1321847	228.3459	8216.852	0	0	0	0	9152545
tot_cur_bal	1321847	137653.1	152744.7	0	31586	80847	201466	8000078
total_rev_hi_lim	1321847	32304.04	35754.4	0	14600	24100	39300	9999999
acc_open_past_24mths	1321847	4.530395	3.062295	0	2	4	6	64
avg_cur_bal	1321847	16953.34	21720.21	0	3294	8335	21232	958084
bc_open_to_buy	1321847	14811.81	33012.31	0	1455	4796	13512	9999999
chargeoff_within_12_mths	1321847	0.008962	0.108384	0	0	0	0	10
delinq_amnt	1321847	15.61694	812.8688	0	0	0	0	185408
mo_sin_old_il_acct	1321847	127.308	49.8576	0	104	130	151	724

mo_sin_old_rev_tl_op	1321847	183.4455	92.37072	2	121	167	229	901
mo_sin_rcnt_rev_tl_op	1321847	13.15444	16.39398	0	4	8	15	438
mo_sin_rcnt_tl	1321847	7.975982	8.834012	0	3	6	10	314
mort_acc	1321847	1.663525	1.979451	0	0	1	3	61
mths_since_recent_bc	1321847	24.12318	30.83404	0	6	14	28	639
mths_since_recent_bc_dlq	1321847	9.503957	20.22588	0	0	0	0	195
mths_since_recent_inq	1321847	5.892125	5.98367	0	1	4	9	25
mths_since_recent_revol_delinq	1321847	12.03261	21.27011	0	0	0	17	197
num_accts_ever_120_pd	1321847	0.47844	1.279002	0	0	0	0	51
num_actv_bc_tl	1321847	3.516756	2.356072	0	2	3	5	47
num_actv_rev_tl	1321847	5.470818	3.494211	0	3	5	7	59
num_bc_sats	1321847	4.535068	3.037534	0	3	4	6	71
num_bc_tl	1321847	7.71718	5.005622	0	4	7	10	79
num_il_tl	1321847	8.061988	7.404718	0	3	6	11	159
num_op_rev_tl	1321847	7.876783	4.796799	0	5	7	10	91
num_rev_accts	1321847	13.90789	8.52584	0	8	13	18	118
num_rev_tl_bal_gt_0	1321847	5.42711	3.417147	0	3	5	7	59
num_sats	1321847	11.18946	5.870372	0	7	10	14	97
num_tl_120dpd_2m	1321847	0.000787	0.029741	0	0	0	0	6
num_tl_30dpd	1321847	0.00367	0.064617	0	0	0	0	4
num_tl_90g_dpd_24m	1321847	0.085912	0.493085	0	0	0	0	39
num_tl_op_past_12m	1321847	1.99748	1.818943	0	1	2	3	32
pct_tl_nvr_dlq	1321847	89.05552	22.79273	0	89.5	97	100	100
percent_bc_gt_75	1321847	45.19209	36.39951	0	0	44.4	75	100
pub_rec_bankruptcies	1321847	0.126656	0.372432	0	0	0	0	12
tax_liens	1321847	0.05661	0.41838	0	0	0	0	85
tot_hi_cred_lim	1321847	170414.6	172592.2	0	51914	112389	241673.5	9999999
total_bal_ex_mort	1321847	49829.52	46940.33	0	22044	37888	61817	2921551
total_bc_limit	1321847	21242.34	20933.88	0	8000	15000	27200	1105500
total_il_high_credit_limit	1321847	41654.41	42204	0	15624	31728	54762	2101913
90day_worse_rating	1321847	0.262762	0.440134	0	0	0	1	1

HANDLING MISSING DATA: LOAN DATA

Missing Data analysis was undertaken in two phases:

1. Handling Numeric columns
2. Handling text columns

Handling Numeric Columns:
Step 1:
For those columns whose distribution doesn't change mush on replacing with zero, the same was done.
delinq_2yrs has 29 missing observations and, we can replace those with zero, giving lendors the benefit of the doubt they wouldn't forget someone delinquent.
inq_last_6mths was done in a similar manner.

mths_since_last_delinq is also replaced by 0 because of the same reason.

Other columns imputed with zeroes are - **open_acc, pub_rec, total_acc, collections_12_mths_ex_med, acc_now_delinq**

Step 2:
Numeric columns where missing values can be replaced by median without changing the distribution of the data, we can replace it by median.

Example:
annual_inc has only 4 missing observations so we do a median value imputation with this feature.

tot_coll_amt will involve a median value imputation.
fullData ['tot_coll_amt'] = fullData ['tot_coll_amt'].fillna(fullData ['tot_coll_amt'].median())

Other columns for median replacement:

Step 3:
We also included a few derived columns to get some necessary insights and information:
mths_since_last_major_derog will be changed to a new variable where missing values = 0 for no derogs and non-missing = 1 for atleast 1 derog. feature will be named 90day_worse_rating

fullData ['90day_worse_rating'] = np.where(fullData ['mths_since_last_major_derog'].isnull(), 0, 1)Joint Account Type and Individual Account Type were mutually exclusive. So were the incomes. Hence they were put together to form a single column

open to buy = credit limit - (sum of holds and outstanding balance) assuming that sum of holds and outstanding balance is zero in this case
fullData['bc_open_to_buy'].fillna(fullData['tot_hi_cred_lim'], inplace=True)

With the high & low credit ranges, we were able to calculate the risk score
fullData['risk_score'] = fullData[['fico_range_low', 'fico_range_high']].mean(axis=1)

Step 4: Delete those insignificant columns where most of the column is empty and will not let us analyze anything

features below are being dropped due to their significantly high proportion of missing values or they are date values.

fullData = fullData.drop(['earliest_cr_line', 'last_pymnt_d', 'next_pymnt_d', 'last_credit_pull_d',
'annual_inc_joint','dti_joint', 'verification_status_joint', 'open_acc_6m', 'open_il_6m',
'open_il_12m', 'open_il_24m', 'mths_since_rcnt_il',
'total_bal_il', 'il_util', 'open_rv_24m', 'open_rv_12m', 'max_bal_bc', 'all_util','total_cu_tl',
'mths_since_last_record', 'mths_since_last_major_derog']

Handling Missing & Cleaning for Text Columns:

STEP 1: Check all the text columns that are categorical. We noticed that there weren't any missing values

->Split Issue Date to obtain Month and Year columns.

->Home ownership had about 7 categories. Merged 'other', 'none' and 'any' into a single category called none.

->Employment length was changed to contain numeric values

->The same was done with term and interest rate columns

STEP 2: A lot of columns seemed like they couldnt give any important information.
These were dropped - url,dech, emp title

CLEANING: DECLINED LOAN DATA SET

After downloading the Declined Loan Data Set, we observed that there were 9 columns:

Amount
Requested
Application Date
Loan Title
Risk Score
Debt-To-Income
Ratio
Zip Code
State
Employment
Length
Policy Code

1) **Loan Title**: We removed the Loan Title Column since we found a lot of titles that didn't make any sense and realized that it wouldn't help with any of our further analysis
2) **Application Date**: We split the application date into Year, Month and Day since we plan on using these columns for future Analysis
3) **State**: We replaced all the missing State values with 'NA'
4) **Zip Code**: We replaced the the XX's in Zip Code with 00's since there's no way of knowing the last 2 digits. We plan on removing the column in future since it doesn't make much sense
5) **Risk Score**: We replaced the missing values with '999' since more than 50% of the data was missing and replacing the values with either a mean or median would change the distribution by a large margin

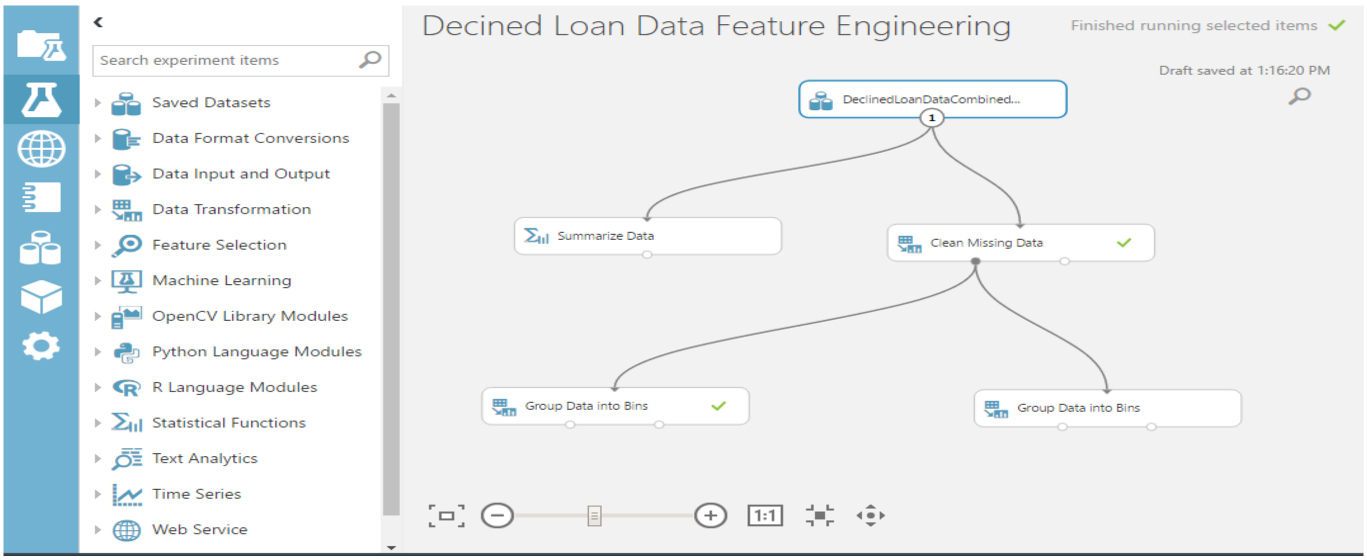
SUMMARY OBSERVATIONS MADE AFTER REPLACING THE MISSING VALUES OF THE RISK SCORE WITH MEDIAN AND MEAN. REPLACED THE MISSING VALUES WITH 999

	Amount Requested	Risk_Score	Employment Length	Policy Code
count	11079386.000000	4676607.000000	11079386.000000	11079386.000000
mean	13391.543411	623.382936	1.707245	0.005543
std	16196.710716	108.408128	1.884364	0.105140
min	0.000000	0.000000	0.000000	0.000000
25%	4500.000000	591.000000	1.000000	0.000000
50%	10000.000000	640.000000	1.000000	0.000000
75%	20000.000000	678.000000	1.000000	0.000000
max	1400000.000000	990.000000	10.000000	2.000000

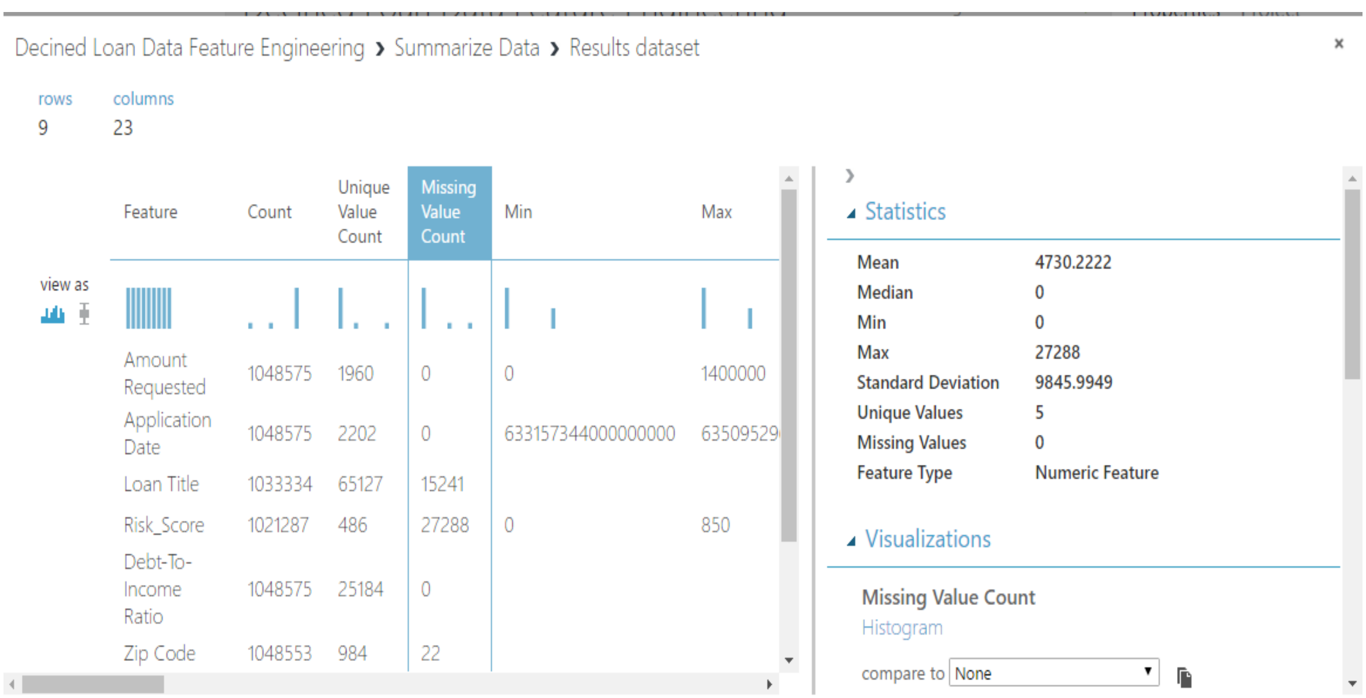
	Amount Requested	Risk_Score	Employment Length	Policy Code
count	11079386.000000	11079386.000000	11079386.000000	11079386.000000
mean	13391.543411	632.985940	1.707245	0.005543
std	16196.710716	70.908453	1.884364	0.105140
min	0.000000	0.000000	0.000000	0.000000
25%	4500.000000	640.000000	1.000000	0.000000
50%	10000.000000	640.000000	1.000000	0.000000
75%	20000.000000	640.000000	1.000000	0.000000
max	1400000.000000	990.000000	10.000000	2.000000

	Amount Requested	Risk_Score	Employment Length	Policy Code
count	11079386.000000	11079386.000000	11079386.000000	11079386.000000
mean	13391.543411	632.985940	1.707245	0.005543
std	16196.710716	70.908453	1.884364	0.105140
min	0.000000	0.000000	0.000000	0.000000
25%	4500.000000	640.000000	1.000000	0.000000
50%	10000.000000	640.000000	1.000000	0.000000
75%	20000.000000	640.000000	1.000000	0.000000
max	1400000.000000	990.000000	10.000000	2.000000

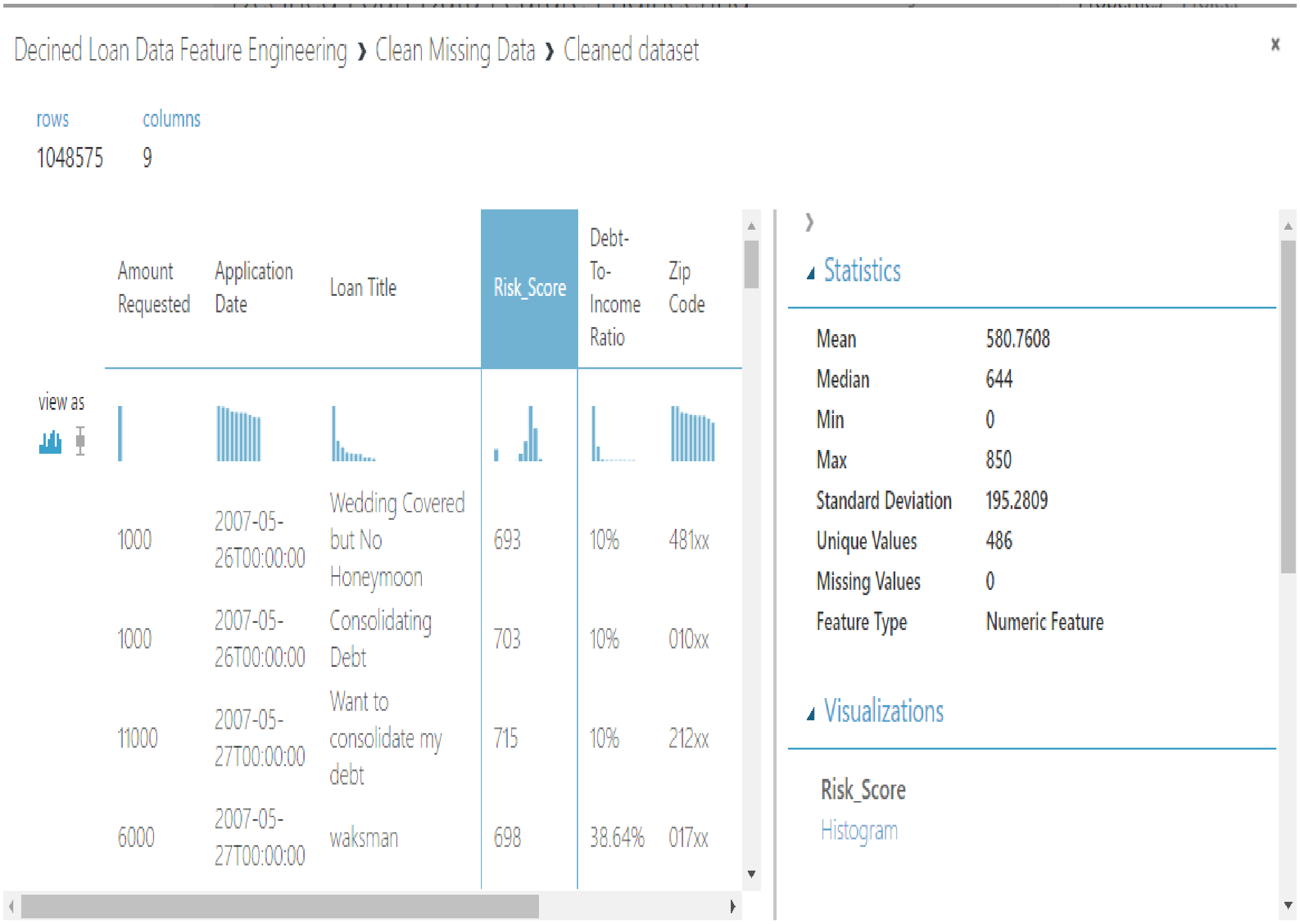
MICROSOFT AZURE ML STUDIO TO CROSS VALIDATE OUR FINDINGS AND VISUALIZE CLEANING, SUMMARY, AND FEATURE ENGINEERING



CHECKING FOR MISSING VALUES USING SUMMARIZE DATA USING MICROSOFT AZURE ML STUDIO



CLEANING DATA AND CHECKING FOR MISSING VALUES USING MICROSOFT AZURE ML STUDIO

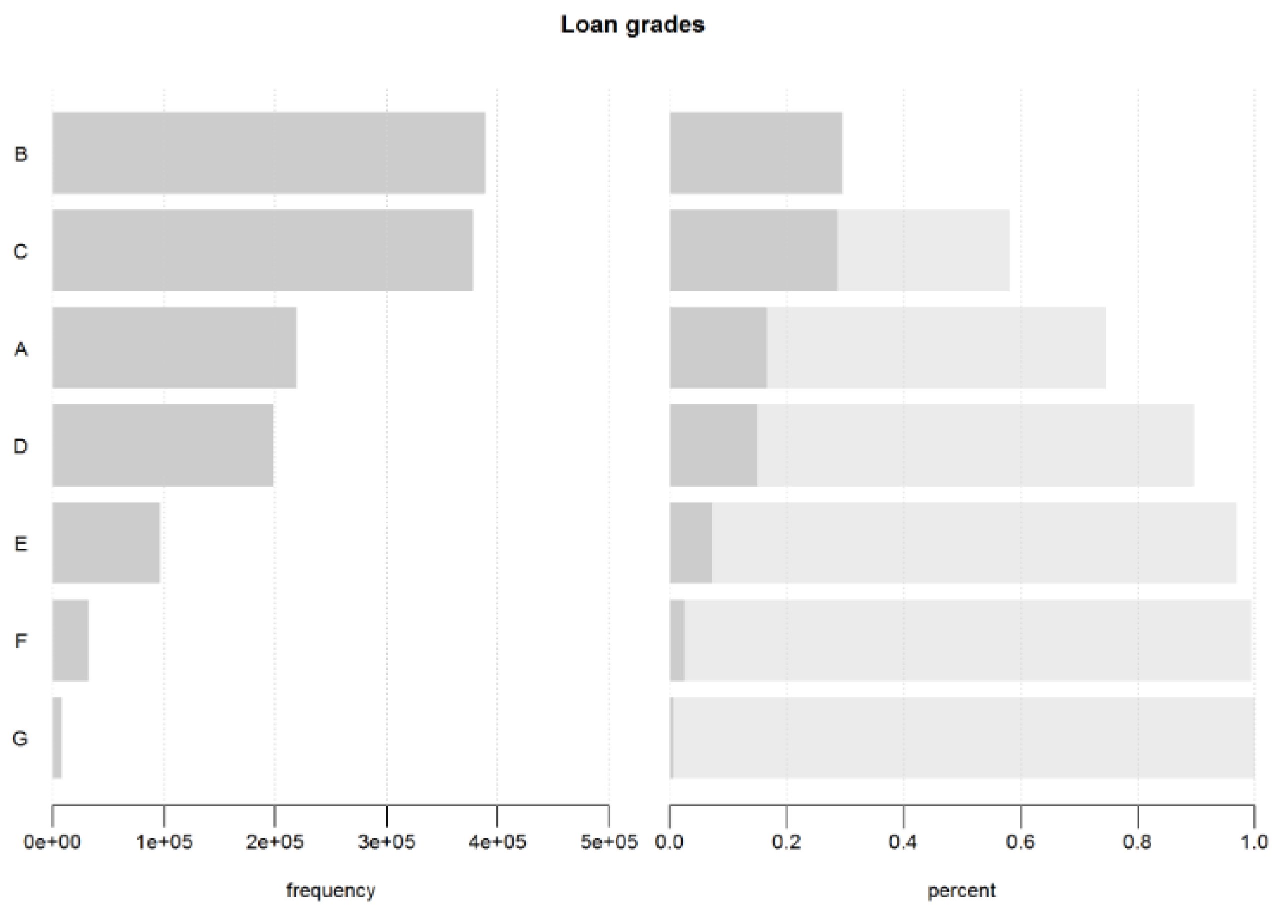


EXPLORATORY DATA ANALYSIS: LOAN DATA

USING JUPYTER NOTEBOOK AND R PUBLISHED LINK: HTTP://RPUBS.COM/PALECANDA

ANALYSIS FOR LOAN DATA: GRADES

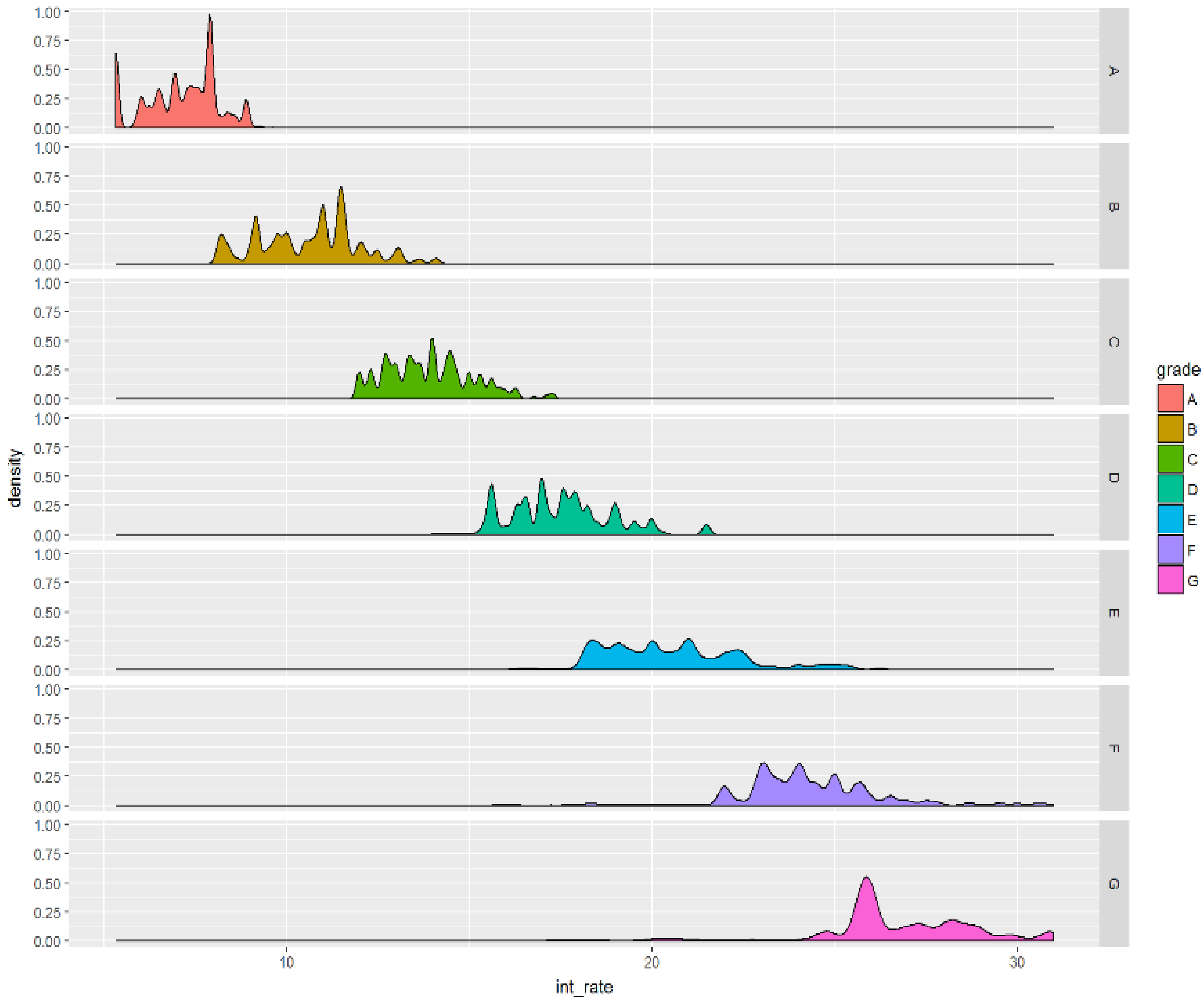
LOAN DATA SET GRADE FREQUENCY




```
library(plotly)
library(ggplot2)
library(plyr)
install.packages("DescTools")
library(DescTools)

#Loan Status Frequency
Desc(data1$loan_status, plotit = T)

#Loan Grade Frequency
Desc(data1$grade, main = "Loan grades", plotit = TRUE)
```

By checking the frequency of the grade, we see that people with higher grade (F/G) are very less as compared to people with lower grades (A,B,C). Hence, people with a good credit score are very few.

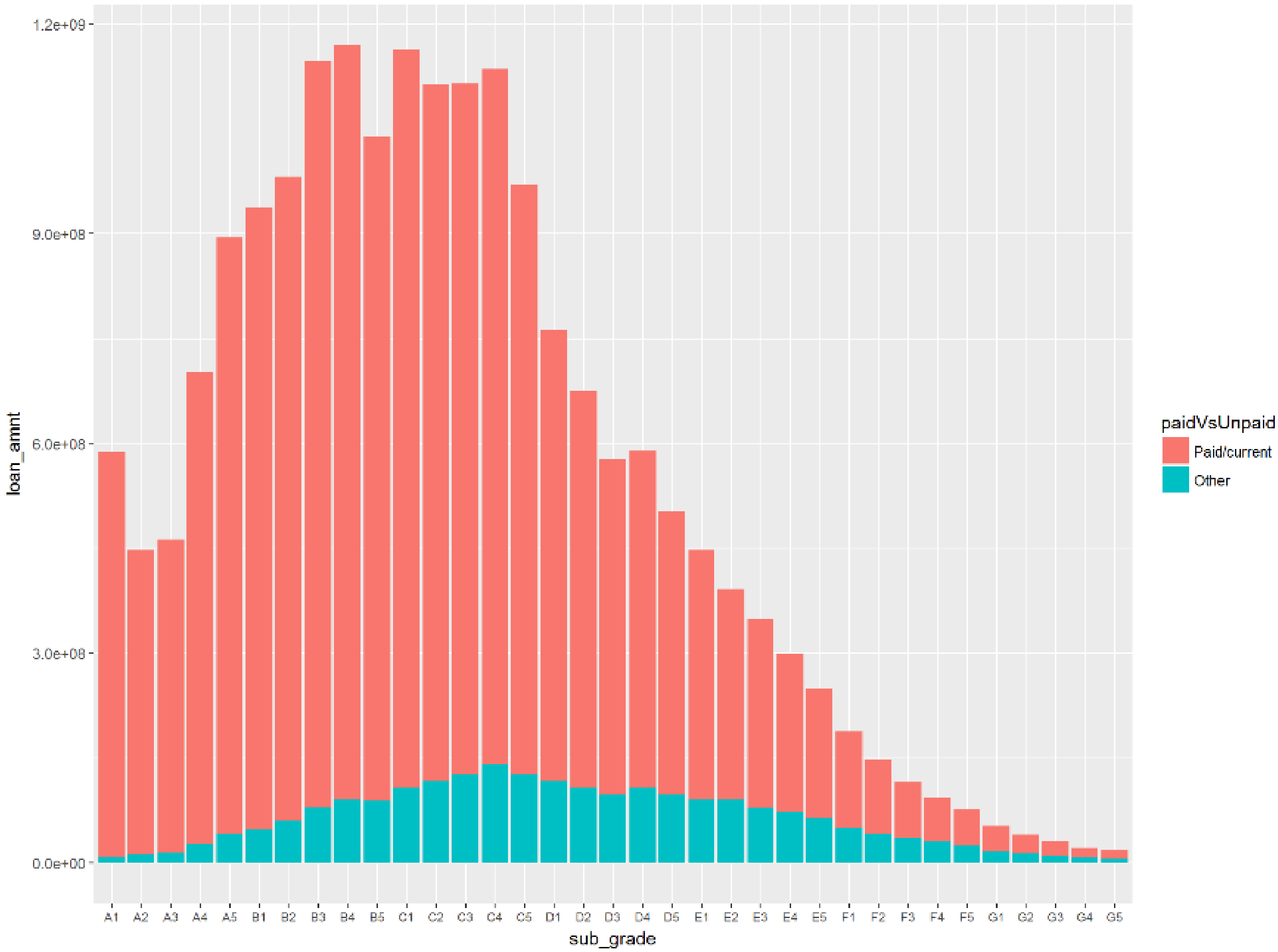
EXPLORING INTEREST RATES BASED ON GRADES ASSIGNED BY LENDING CLUB




```
library(ggplot2)
ggplot(data1, aes(int_rate, fill = grade)) + geom_density() + facet_grid(grade ~ .)
```

Grades are assigned based on risk, and so interest rates go up as the risk goes up. Here we see that the interest rate is the highest for grades F and G, since their credit score, hence risk score is low

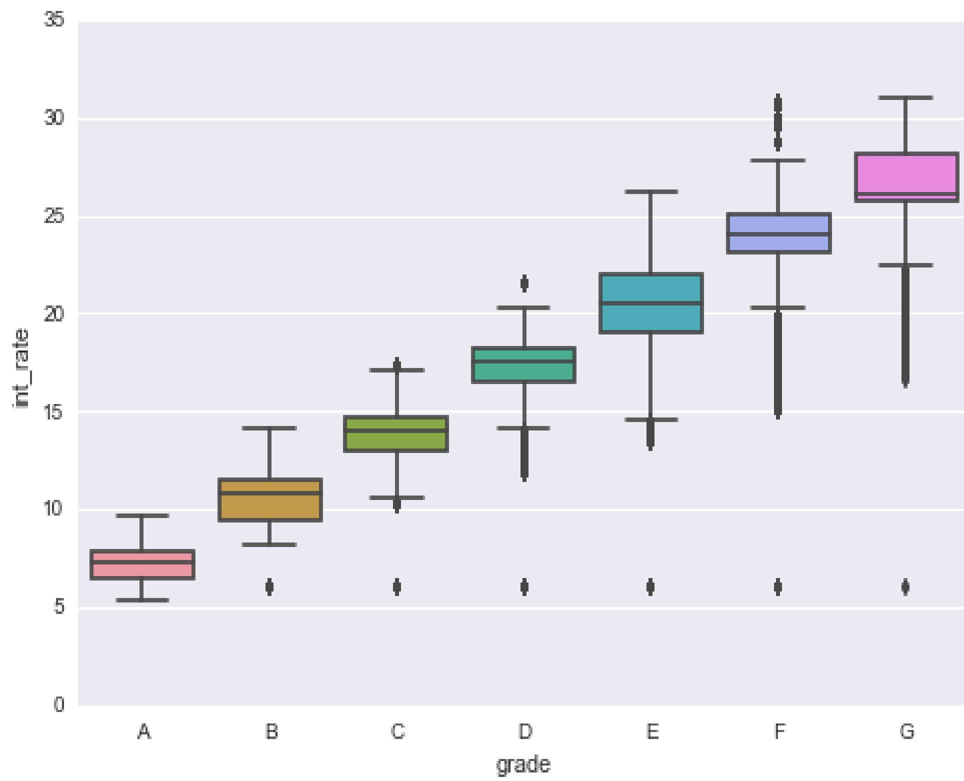
PAID VS. UNPAID LOAN AMOUNT OVER THE GRADES




```
data1$paidVsUnpaid <- "Other"
data1$paidVsUnpaid[which(data1$loan_status == "Fully Paid" | data1$loan_status == "Current" | data1$loan_status ==
                  "Does not meet the credit policy. Status:Fully Paid") ] <- "Paid/current"
data1$paidVsUnpaid <- factor(data1$paidVsUnpaid)
data1$paidVsUnpaid <- factor(data1$paidVsUnpaid, levels = rev(levels(data1$paidVsUnpaid)))
table(data1$paidVsUnpaid)
ggplot(data1, aes(paidVsUnpaid, loan_amnt, fill = paidVsUnpaid)) + geom_bar(stat = "identity")

#Paid Vs. Unpaid loan amount over the Grades
loan_by_grade <- aggregate(loan_amnt ~ sub_grade + paidVsUnpaid, data = data1, sum)
gbar <- ggplot(loan_by_grade, aes(sub_grade, loan_amnt, fill = paidVsUnpaid))
gbar + geom_bar(stat = "identity") + theme(axis.text.x=element_text(size=7))
```

Here, we have created new column with 2 factor levels. 1) “Paid/current” - Represents the status is Current or Fully Paid. 2) “Other” - Represents defaults, charger-off and other status

INTEREST RATES AGAINST GRADES

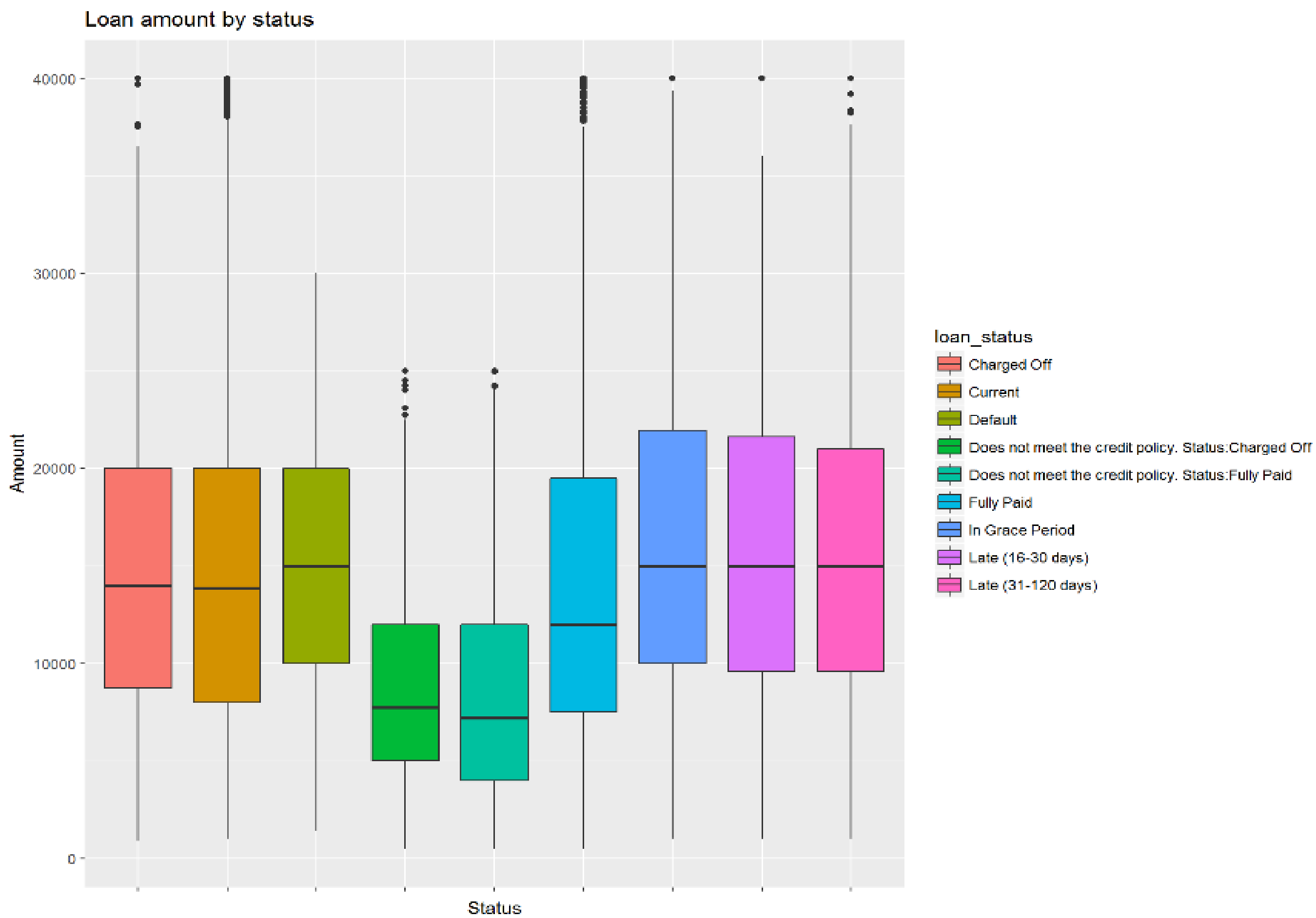



```
import seaborn as sns
import matplotlib.pyplot as plt

ax1 = sns.boxplot(x='grade', y='int_rate', data=loan,
                  order=sorted(loan['grade'].unique()))
ax1.set_ybound(lower=0)
plt.show()
```

We observe that the loans have a higher rate of interest until one gets to the higher grades such as A and B

ANALYSIS FOR LOAN DATA: LOAN AMOUNT

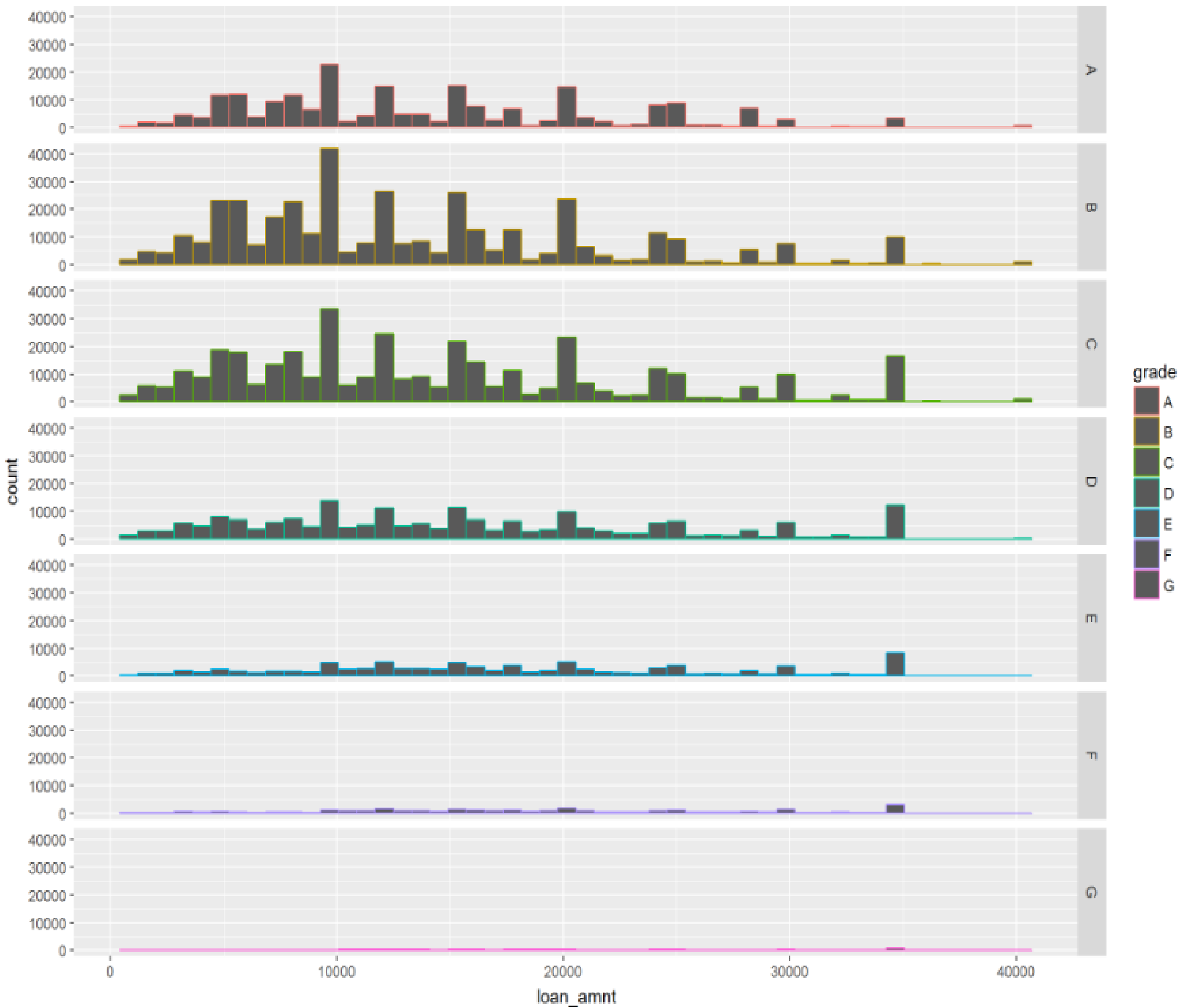
LOAN AMOUNT AGAINST LOAN STATUS




```
#checking the distribution of loan amounts by status.
library(ggplot2)
box_status <- ggplot(data1, aes(loan_status, loan_amnt))
box_status + geom_boxplot(aes(fill = loan_status)) +
  theme(axis.text.x = element_blank()) +
  labs(list(
    title = "Loan amount by status",
    x = "Status",
    y = "Amount"))
```

By plotting the loan amount against the loan status, we see that most of the loans are in grace period, or are late

COUNT OF LOAN AMOUNT AGAINST GRADE

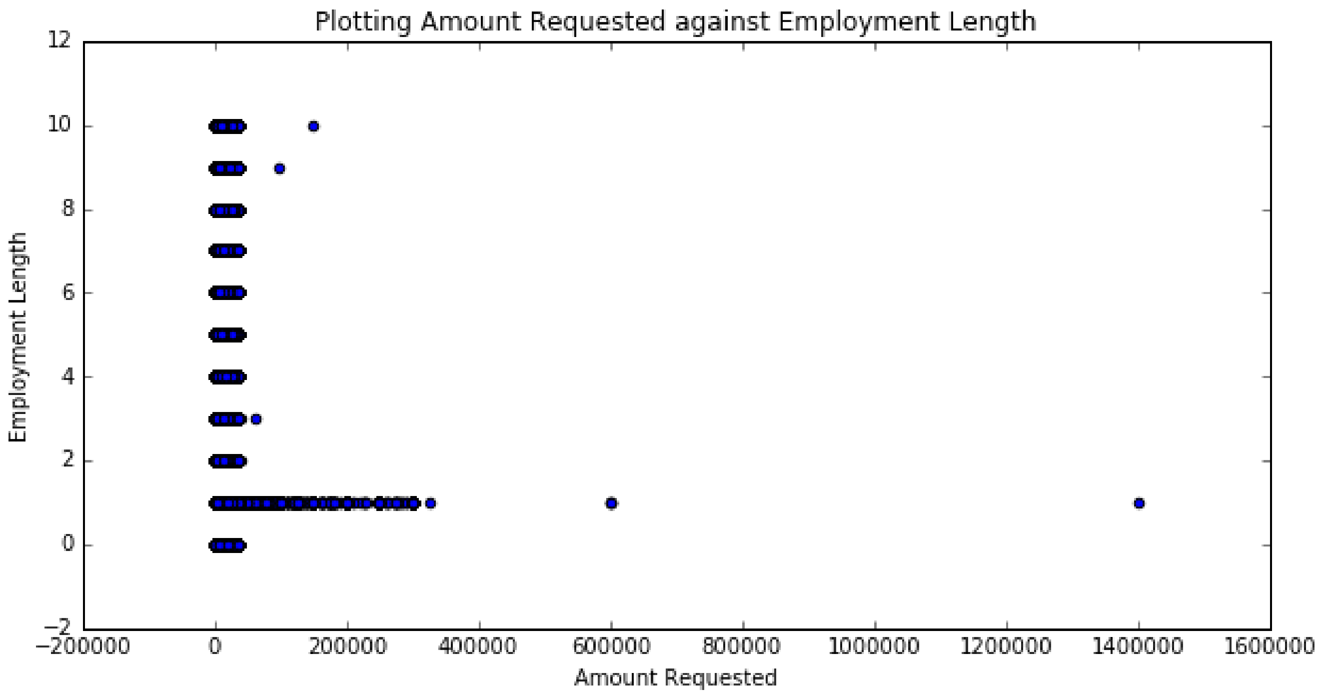


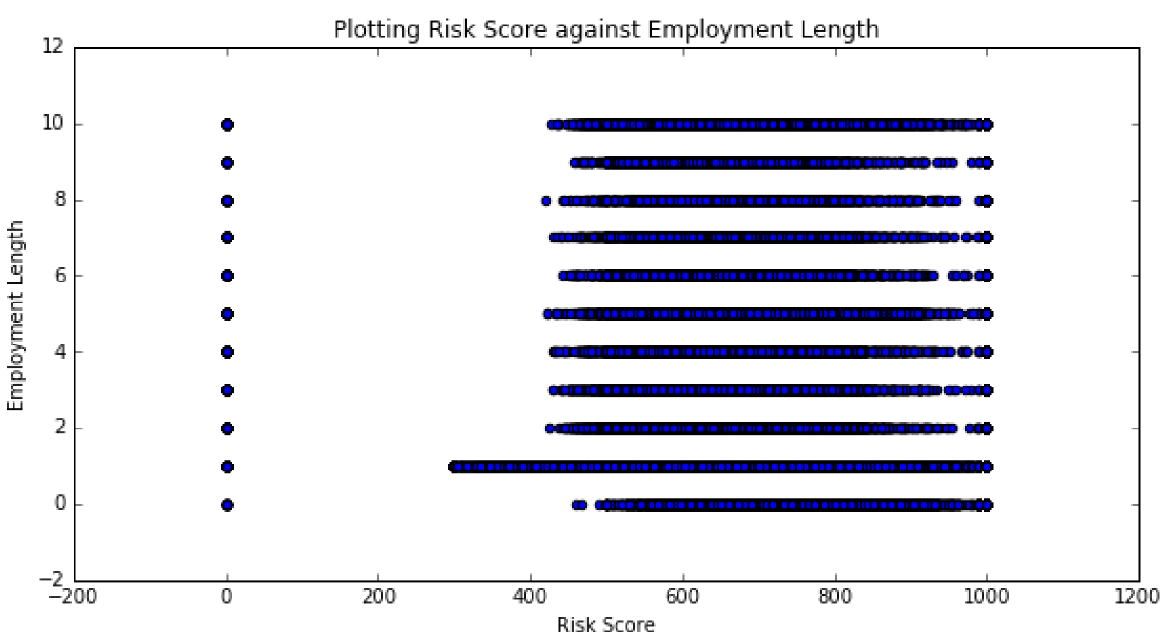

```
library(ggplot2)
library(dplyr)
library(plotly)
library(ggplot2)

#Loan amount Distribution based on Grades assigned by Lending Club
#Those with higher grades (A, B, C and D) have received more loans compared to those with lower grades (E, F and G)
library(ggplot2)
ggplot(data1, aes(loan_amnt, col = grade)) + geom_histogram(bins = 50) + facet_grid(grade ~ .)
```

We observe that, those with higher grades (A, B, C and D) have received more loans compared to those with lower grades (E, F and G)

EXPLORATORY ANALYSIS: DECLINED LOAN DATA

REJECTED LOAN AMOUNT AGAINST EMPLOYMENT LENGTH VS RISK SCORE





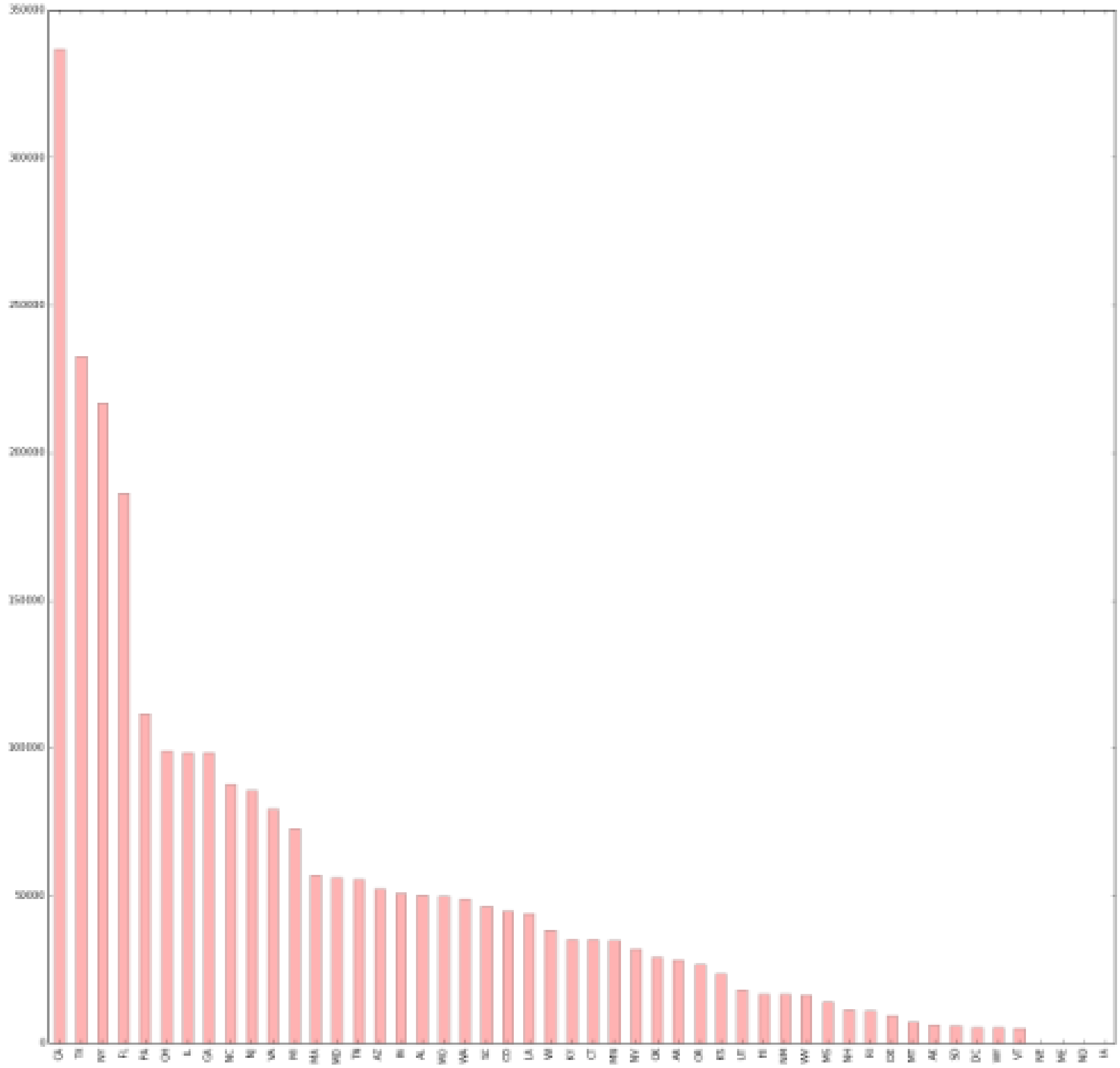
In this Analysis for Declined Loan Data, we have analyzed Employment length against the Amount Requested and Employment length against the Risk Score. We can see that the Loan Requested has mostly been declined for people with an Employment Length between 0-2 years. This may be due to various reasons

1) Their credit scores are low since they just started working
2) After plotting the employment length against the risk score, the above point is pretty evident since the risk score is calculated using the credit scores (FICO scores/2)

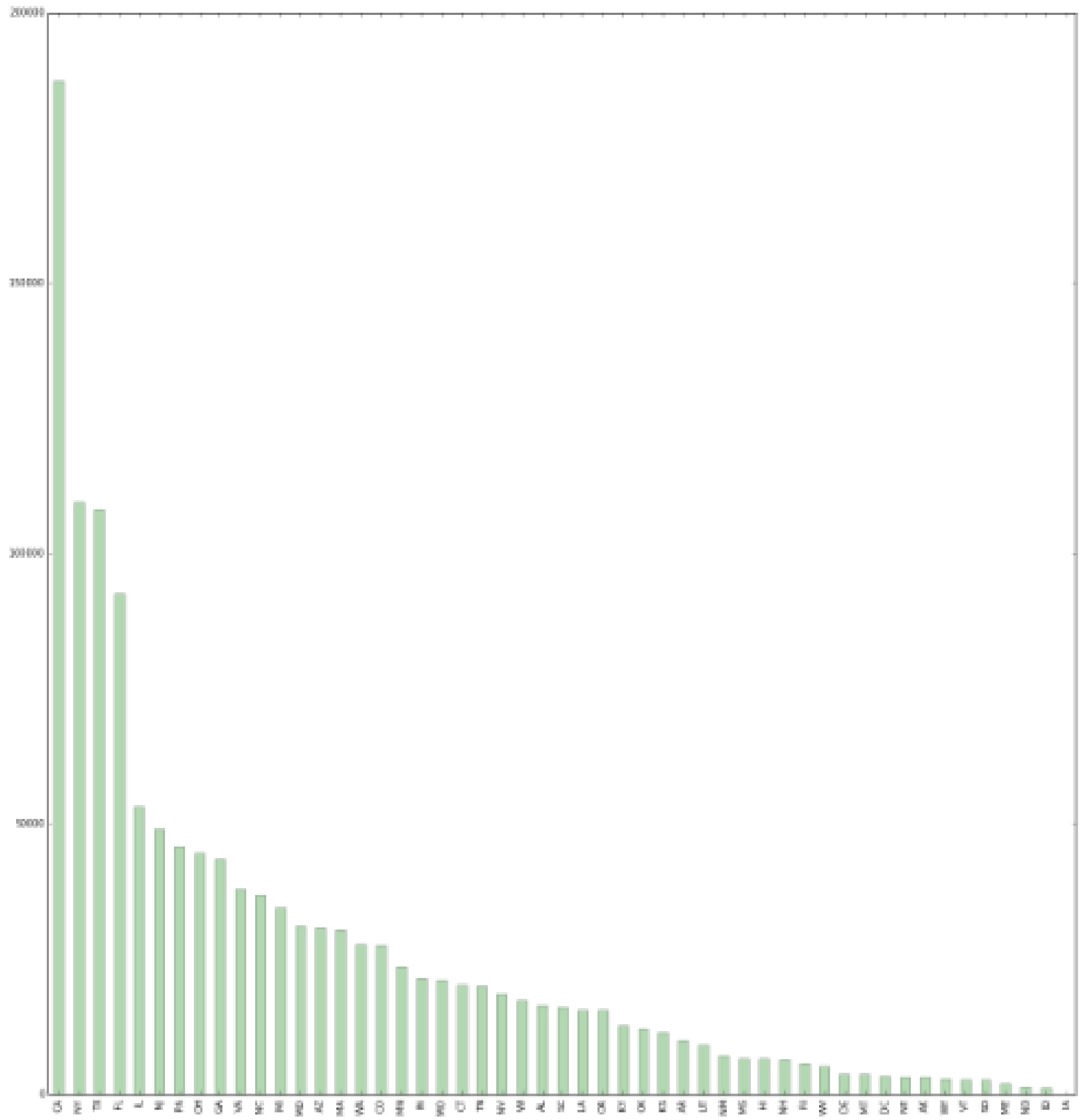
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

#Plotting the Amount Requested against Employment Length
%matplotlib inline
plt.figure(figsize=(10,5))
plt.scatter(fullData['Amount Requested'], fullData['Employment Length'])
plt.title("Amount Requested against Employment Length")
plt.ylabel('Employment Length')
plt.xlabel('Amount Requested')
plt.show()
```

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

#Plotting Risk Score against Employment Length
%matplotlib inline
plt.figure(figsize=(10,5))
plt.scatter(fullData['Risk_Score'], fullData['Employment Length'])
plt.title("Plotting Risk Score against Employment Length")
plt.ylabel('Employment Length')
plt.xlabel('Risk Score')
plt.show()
```

COUNT OF ACCEPTED AND REJECTED LOAN AMOUNT AGAINST STATES



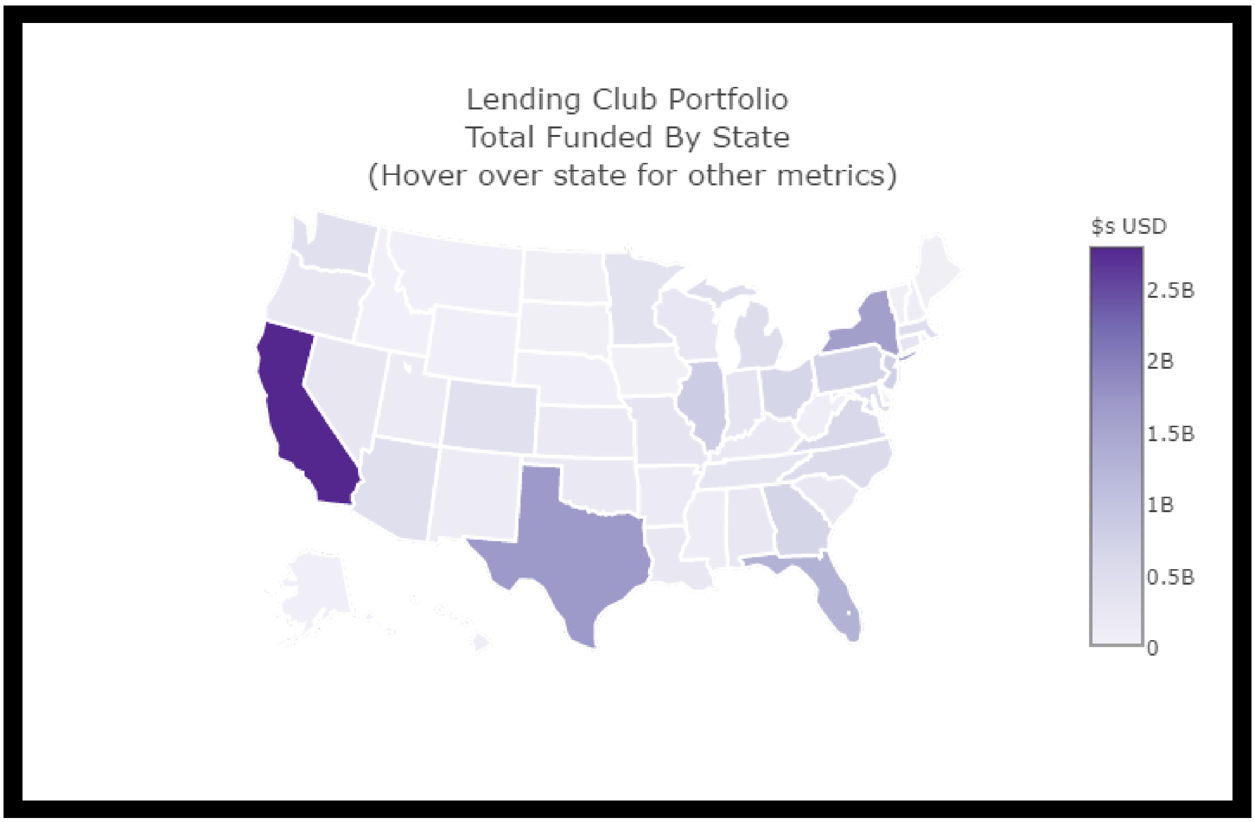
```
fig = plt.figure(figsize=(20, 20), dpi=100)
fullData['State'].value_counts().plot(kind='bar',alpha=.30,color='red')
```



The above 2 graphs shows that the count of the accepted and rejected loan are high in CA, TX, NY and FL. T the reason for this could be that these states were the first to be eligible to get a loan from lending club.

```
import pandas as pd
import numpy as np
from datetime import datetime
from matplotlib import pyplot as plt

%matplotlib inline
loandata = pd.read_csv("C:/Users/taj/Desktop/cleaned_loandata.csv",encoding = "ISO-8859-1",low_memory=False)

fig = plt.figure(figsize=(20, 20), dpi=100)
loandata['addr_state'].value_counts().plot(kind='bar',alpha=.30,color='green')
```

TOTALS FUNDED BY STATE



The above graph gives a detailed analysis of the Average Balance per Borrower and Average Income per borrower in each state. We see that the Averages are the highest in CA

```
import plotly.plotly as py
import plotly.graph_objs as graph_objs

for col in df_plot.columns:
    df_plot[col] = df_plot[col].astype(str)

    scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,143)']]

df_plot['text'] = df_plot['code'] + '<br>' +\
    'Avg Balance Per Borrower ($ USD): '+df_plot['Average_Balance']+'<br>'+\
    'Avg Annual Income Per Borrower ($ USD): '+df_plot['Average_Income']


data = [ dict(
        type='choropleth',
        colorscale = scl,
        autocolorscale = False,
        locations = df_plot['code'],
        z = df_plot['Loan_Funding'],
        locationmode = 'USA-states',
        text = df_plot['text'],
        marker = dict(
            line = dict (
                color = 'rgb(255,255,255)',
                width = 2
            ) ),
        colorbar = dict(
            title = "$s USD")
        ) ]

layout = dict(
        title = 'Lending Club Portfolio<br> Total Funded By State <br> (Hover over state for other metrics)',
        geo = dict(
            scope='usa',
            projection=dict( type='albers usa' ),
            showlakes = True,
            lakecolor = 'rgb(255, 255, 255)'),
             )

fig = dict( data=data, layout=layout )
iplot( fig, filename='d3-cloropleth-map' )
```

FEATURE ENGINEERING:

We plan on building models and predicting the best features that we can use to calculate the interest rates.

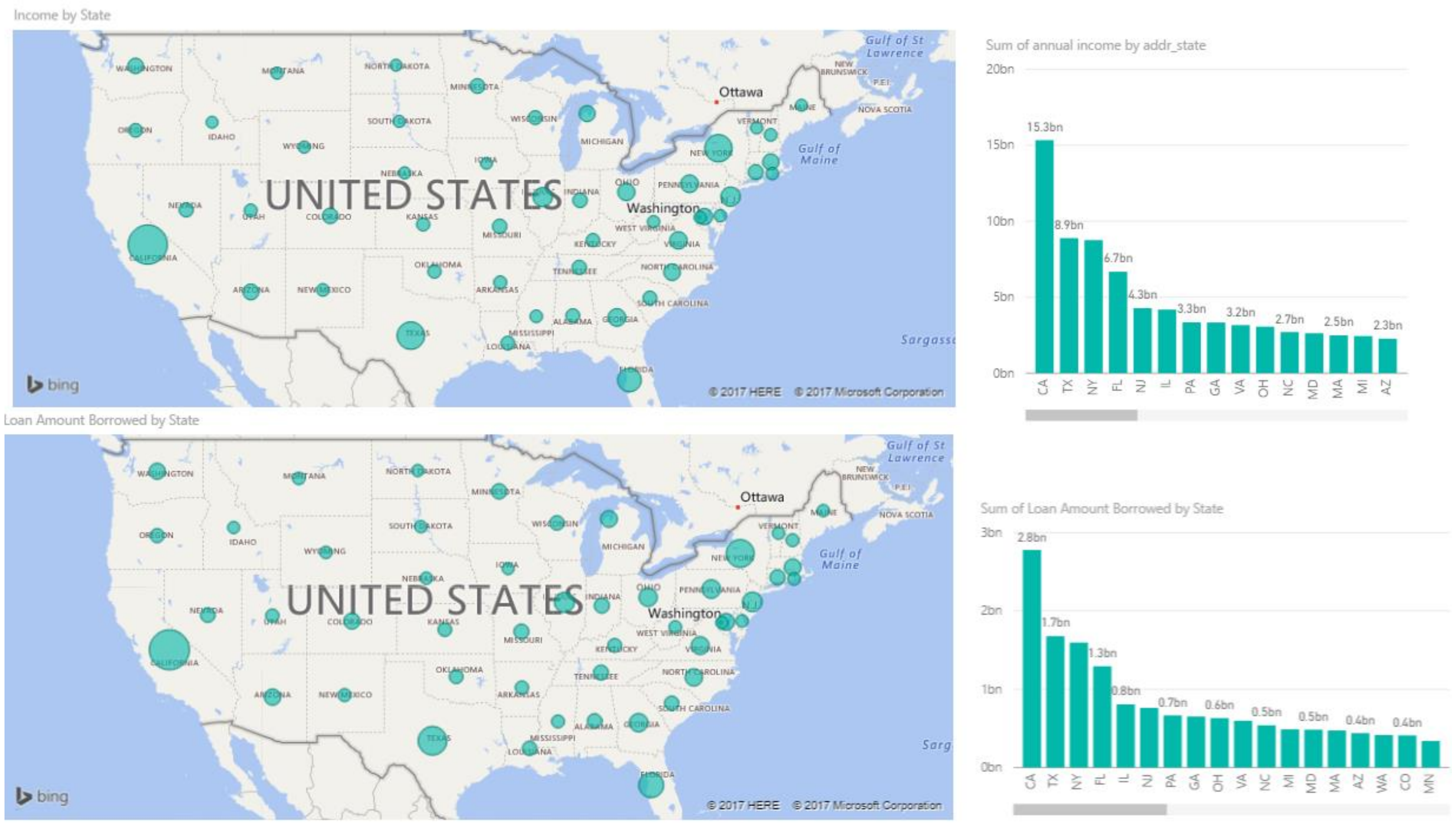
As of now, we plan on selecting the following features:

1) Minimum Credit Score (FICO) – To calculate the Risk Score
2) Maximum Credit Score (FICO) – To calculate the Risk Score
3) Employment Length – Since we've seen that the employment length affects the credit score
4) DTI (Debt-to-Income) – Since we have this in both Loan and Declined Loan Data Set and since the DTI is a vital deciding factor
5) State – Since we have this information in both the data sets, and since we've seen that the interest rates, income, etc varies with state (May or May not be included)

Power BI Analysis

Our aim was to see how the following parameters influence interest rates

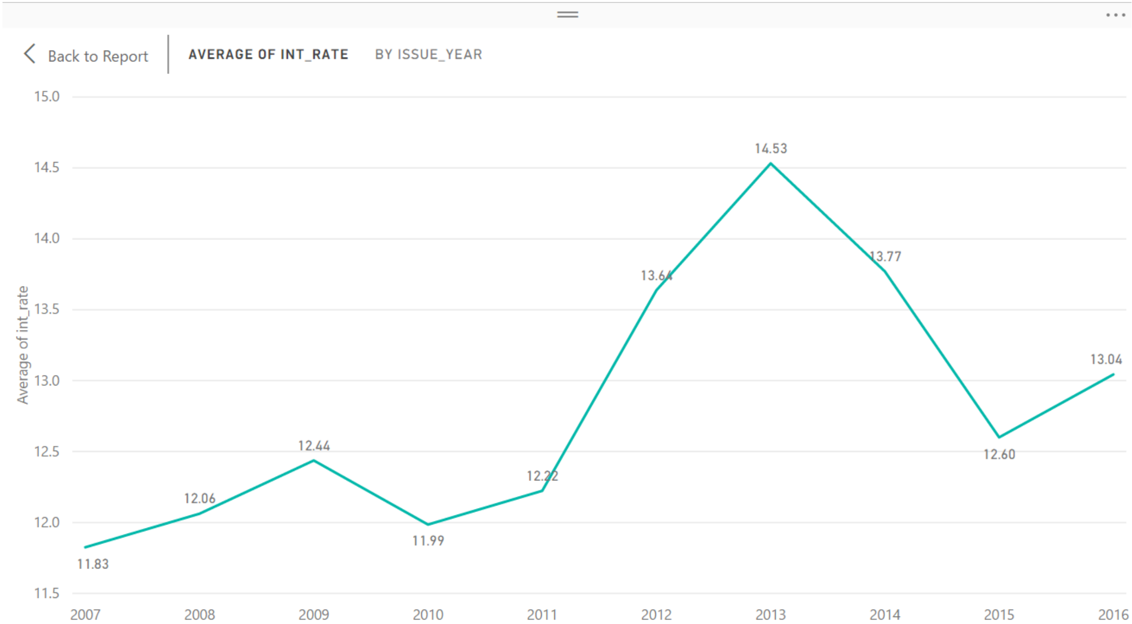
1. Income Analysis



The above dashboard has four graphs to establish the relationship between the the land demography parameter -Income Level to the nature of the loan amout.

As expected, the state with the highest annual income (15.3 b) has the highest loan borrowed (2.8bn). Our hypothesis is also that California, being THE "Silicon Valley", required more investments.

In the east coast, the financial capital, NY is the state with the most loan amount, as expected.



We see an interesting trend in the average interest rate across years.

What we observe is an unusual spike in interest rate to a high of 14.5%. As we dug deeper, we understood that this was the first time in 30 years.

30-year mortgage rate hits 3.81%,

Marcy Gordon, AP Business Writer Published 10:11 a.m. ET May 30, 2013 | **Updated 11:**

(Photo: Elaine Thompson AP)

CONNECT TWEET in 22 LINKEDIN COMMENT EMAIL MORE

WASHINGTON (AP) — Average U.S. rates on fix mortgages jumped this week to their highest level a year, signaling slightly higher costs for homebuyers. But rates remain low by historical standards.

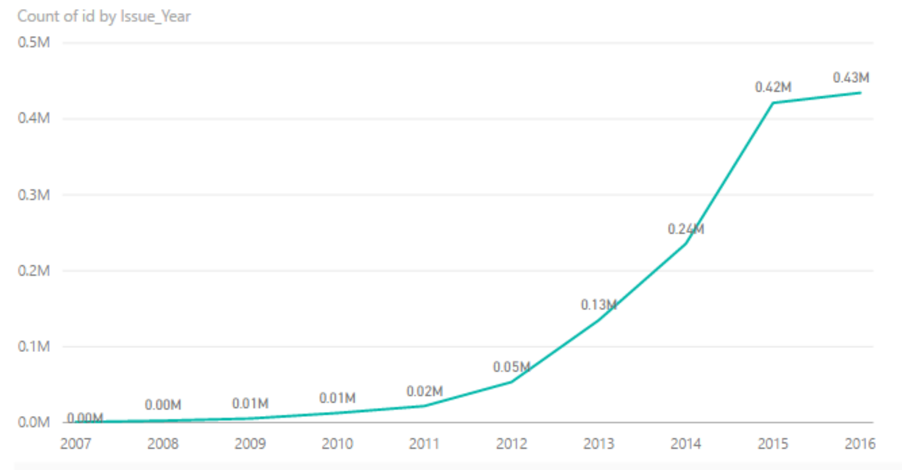
Mortgage buyer Freddie Mac says the average ra for the 30-year loan rose to 3.81%, up from 3.59% last week. That's still not far from the 3.31% rate reached in November, the lowest on records datir to 1971.

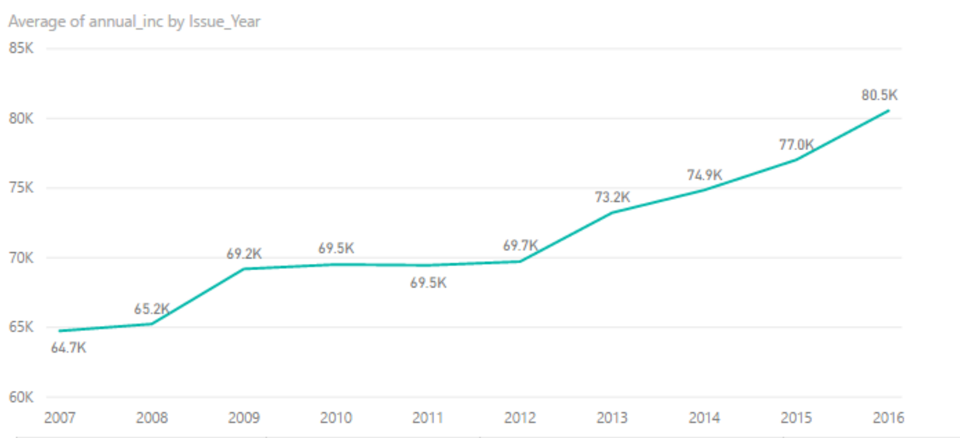
HOME BUYERS: Mortgage rates may be headec

STORY HIGHLIGHTS

- **Freddie Mac says average rate for 30-year loan rose to 3.81%**
- **15-year loan rose almost 3%**

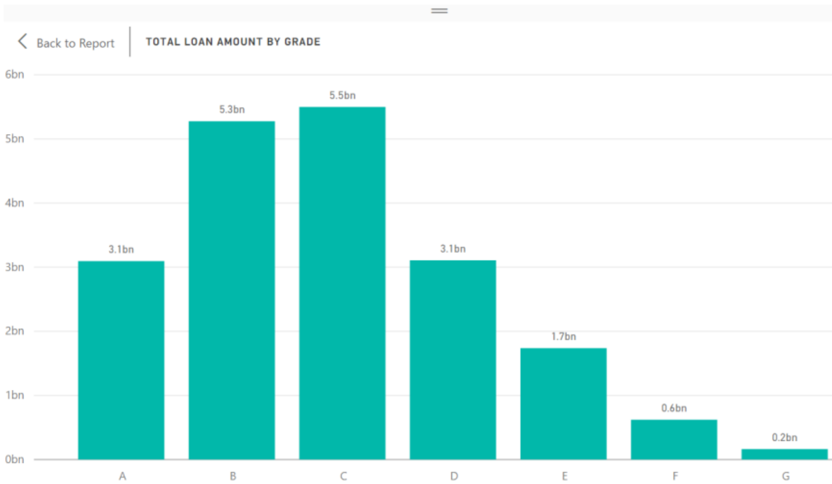
We tried to see if this unusual spike brought about any changes in the number of loans borrowed or the income levels





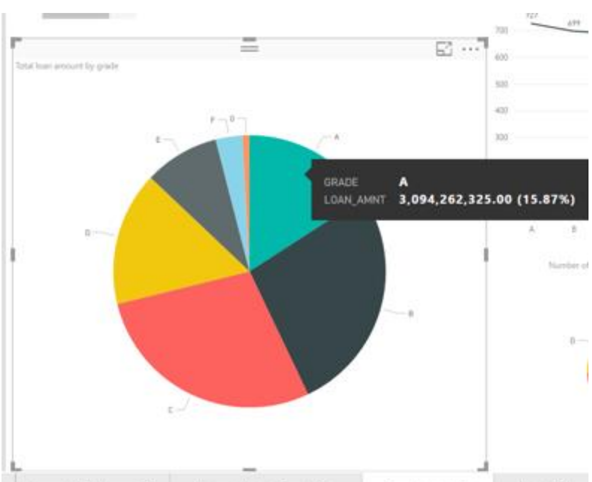
There is no change in the trend. The number of borrowed loans shows an increasing trend as did the annual income

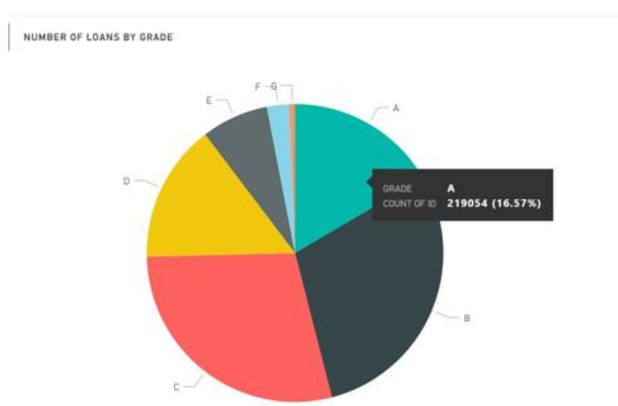
2. Loan Grade



This is the distribution of sum of loan amounts across grades. Since grade A holders are the ones with the highest loan granted, we'd expect their total loan amount to be high.

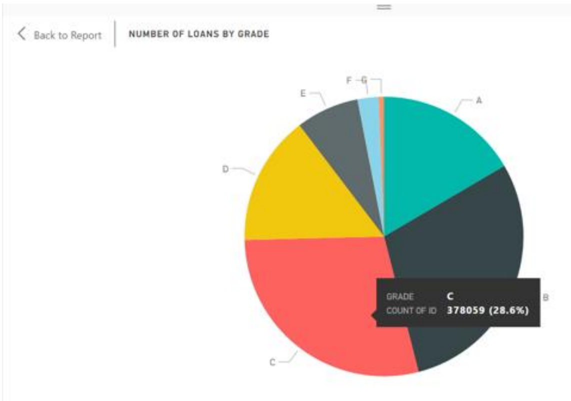
However, this graph shows otherwise. To understand why, take a closer look at this graph:

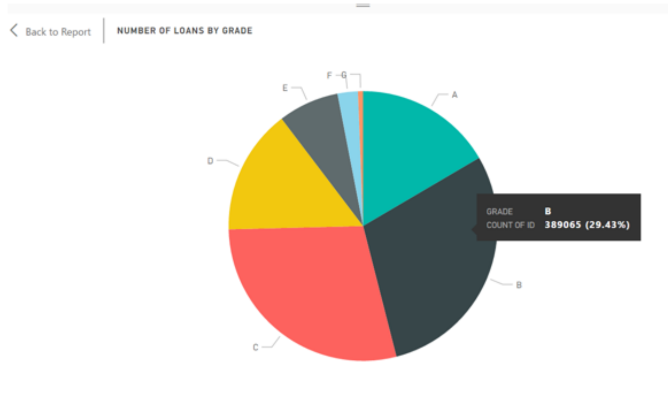




In terms of contribution, Grade A loans contribute only to 16% of the total number of loans borrowed. Reflected by the total loan amount also contributing to 15.8% of the total amount.

While Grades B and C contribute to about 60% of the total number of loans borrowed. Hence, the loan amounts are the highest in grades B and C than in A

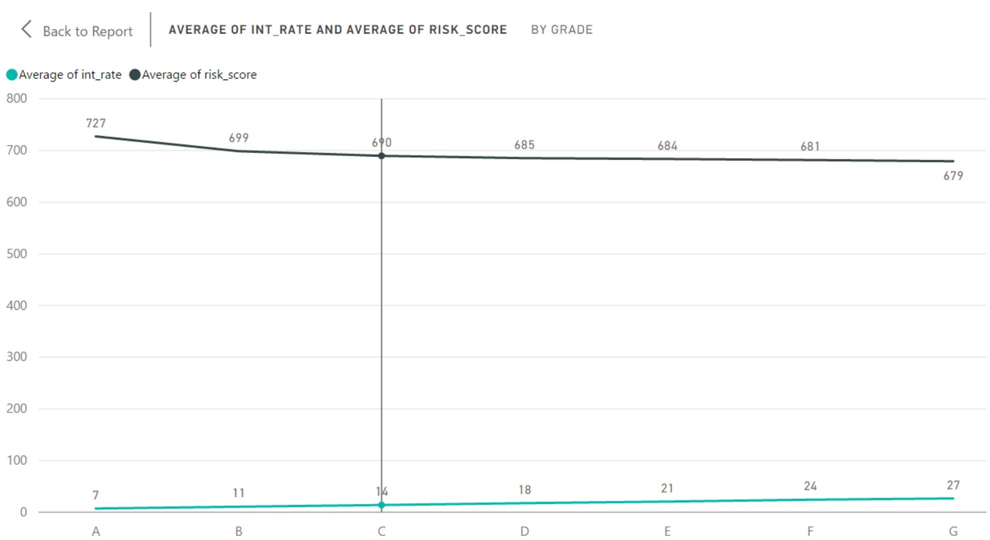




Since the grades are primarily calculated using the risk score, we tried to check if the average interest rates acorss grades varied as expected. *We were right!*

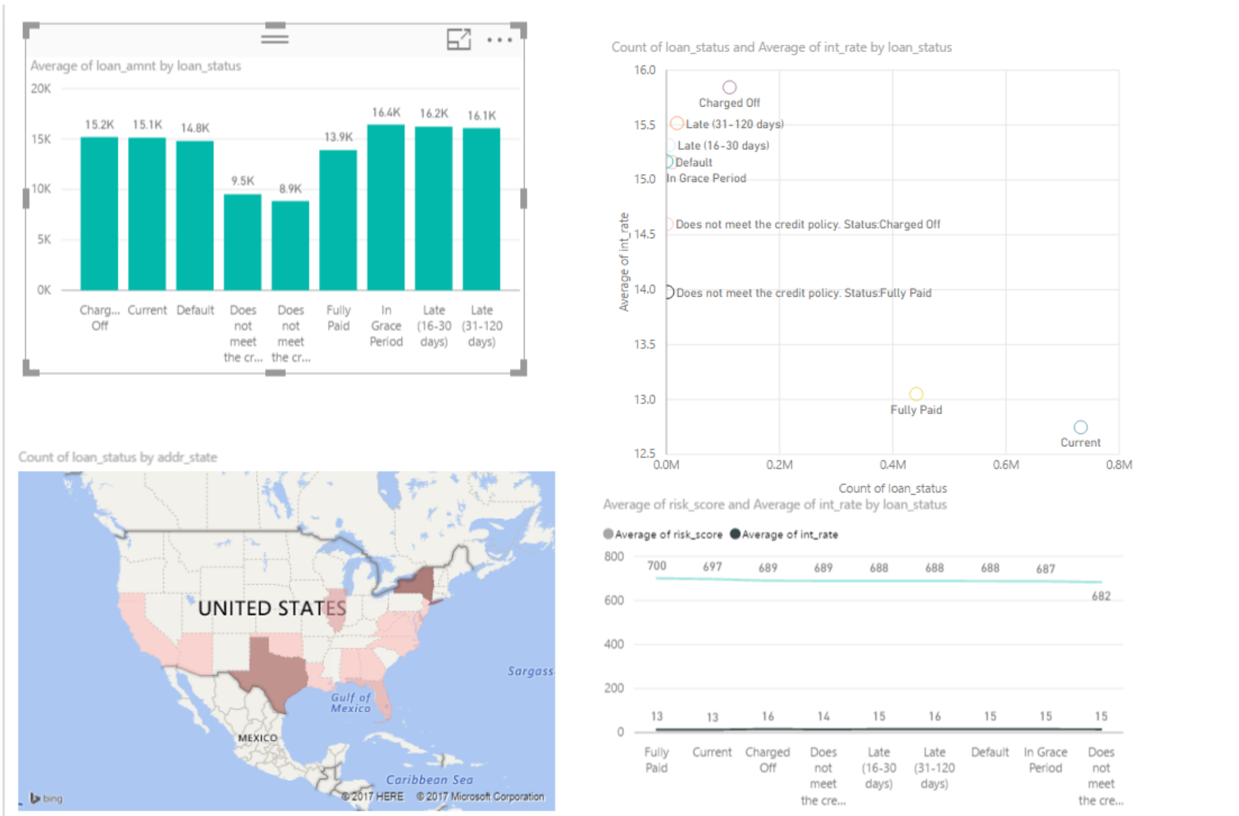
Grade A had the highest average risk score (727), hence the lowest interest rate (7.2)

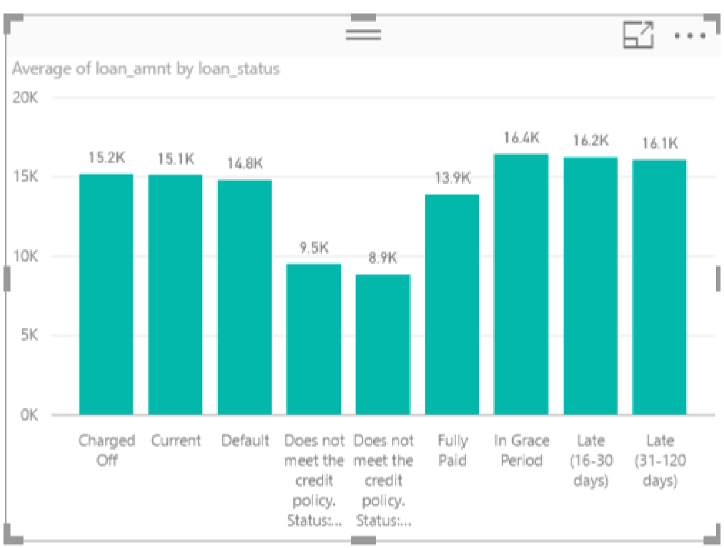
The trend lines are as expected.



LOAN STATUS

This dashboard summarizes various trends with respect to loan status

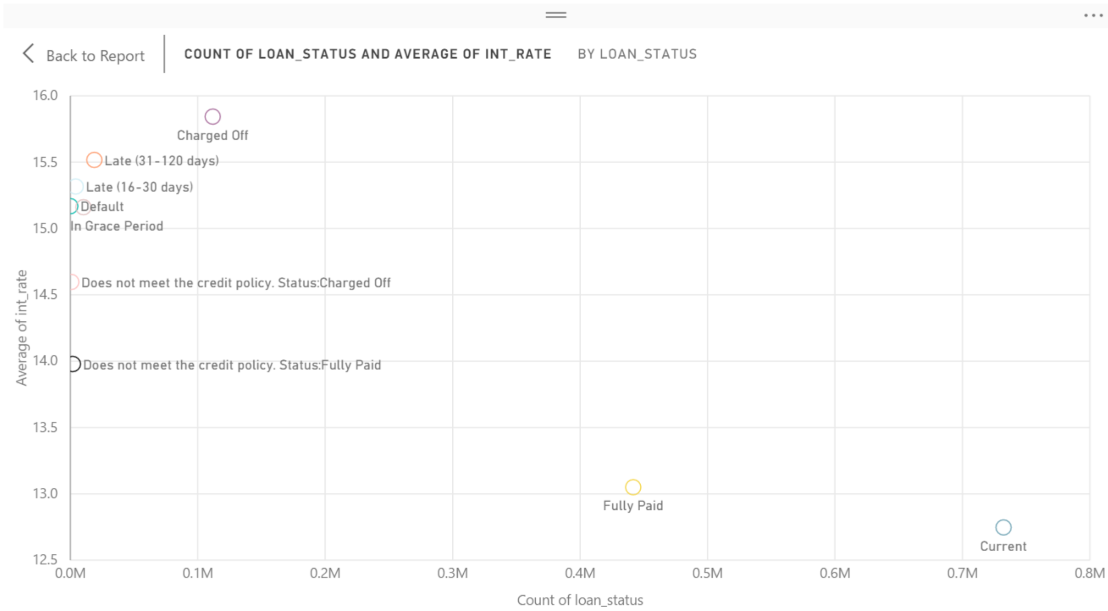




This is the distribution of average loan amount over different loan status in the data

As per the graph, the loans which didn't meet the credit policy seems to have had an average loan amount of 9k - 9.5k

This interesting graph summarizes the Count of loans with various status and how they vary across the average interest rates. There are some important insights we get of this:



loan_status	Count of loan_status	Average of int_rate
Charged Off	111740	15.84
Current	732250	12.75
Default	26	15.17
Does not meet the credit policy. Status:Charged Off	761	14.60
Does not meet the credit policy. Status:Fully Paid	1988	13.98
Fully Paid	441663	13.05
In Grace Period	10297	15.16
Late (16-30 days)	4274	15.32
Late (31-120 days)	18848	15.52

Careful observations show the following resuts:

The current and the fully paid loans, are the highest in number (732,250 and 441,663) contributing to over 60% of the total number of loans, have a very low average interest rates (12.75 and 13.05 respectively)

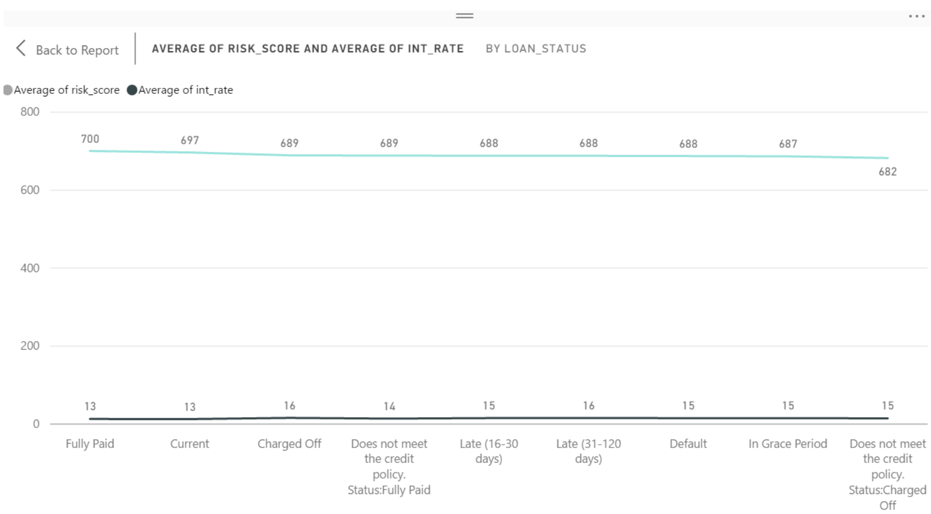
However, the Default and Charged off loans have high interest rates (15.75 and 15.84 respectively)

The most number of default loans are in Texas, followed by New York, Illinois etc.

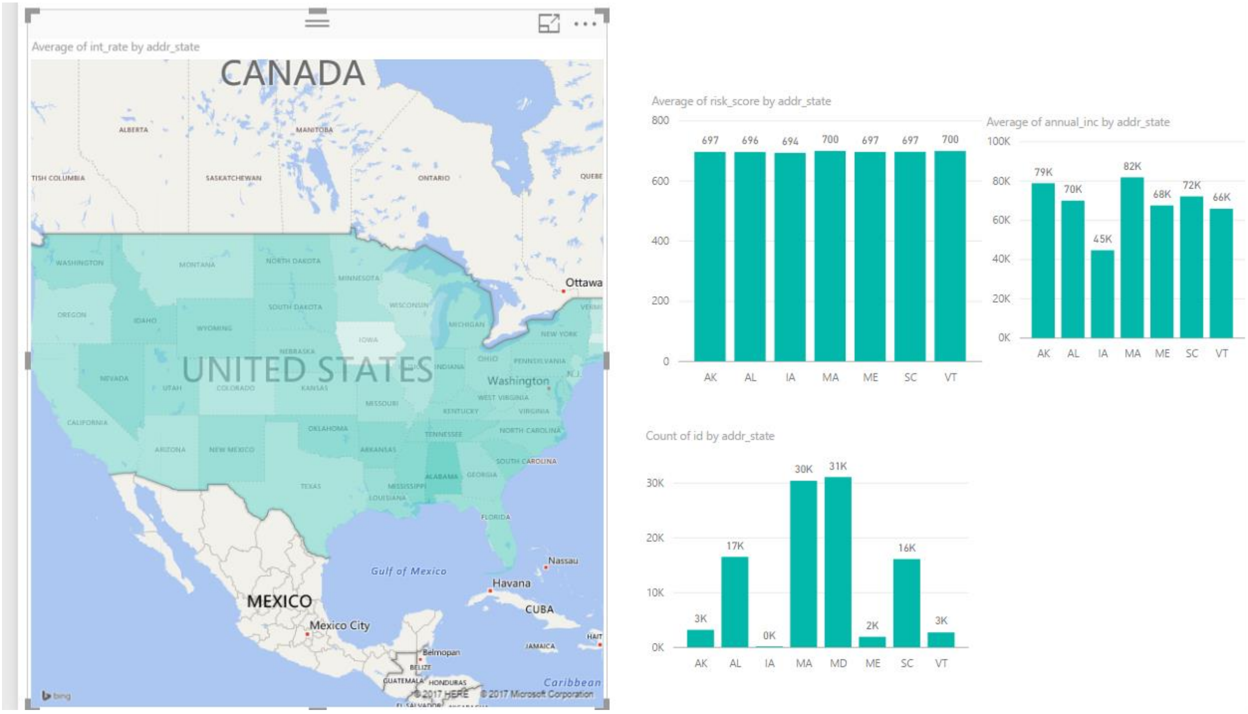
Here, lets see how interest rates and risk scores change with respect to the various loan status.

As expected, we see that the risk score is the highest for fully paid(700), current(697) and then decreases for the loans not meeting the credit policy and hence charged off (682).

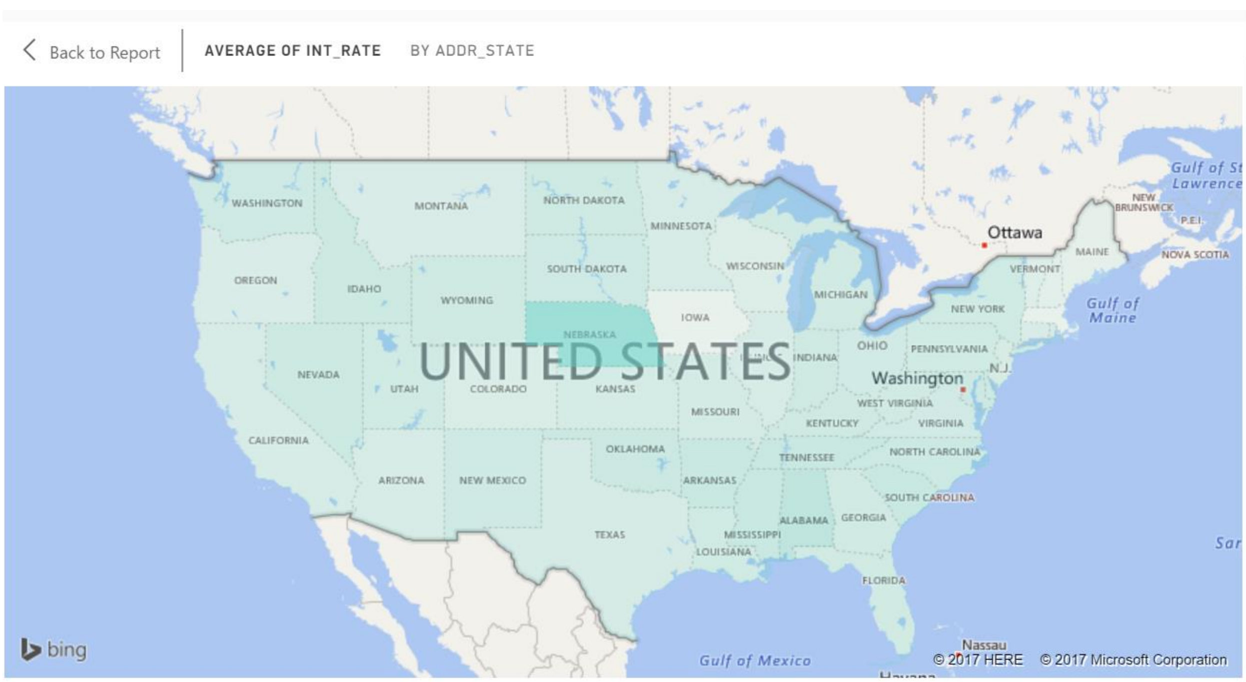
Same time, the interest rates are the lowest (13%) for fully paid and current loans. And are the highest for charged off loans and the ones that don't meet the loan policy (16). It is high for default loans as well (15%)



STATE-WISE ANALYSIS - DASHBOARD



Lets deep-dive into interest rates by state:

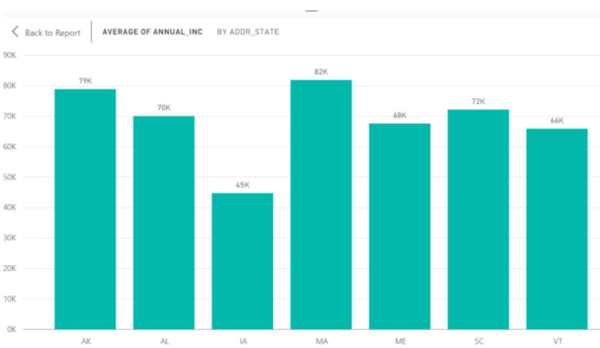


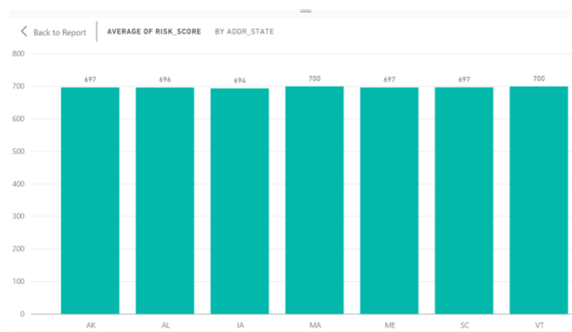
We observe that the top 3 states with the highest interest rates are: Alaska (AK)- 13.5%, Alabama(AL) – 13.6% and SC (South Carolina) – 13.4%

An interesting observation is that IA has a very small interest rate – 12. 63% .

Bottom three are MA, ME and VT.

The average income and average risk score for each of these states confirm the trend observed





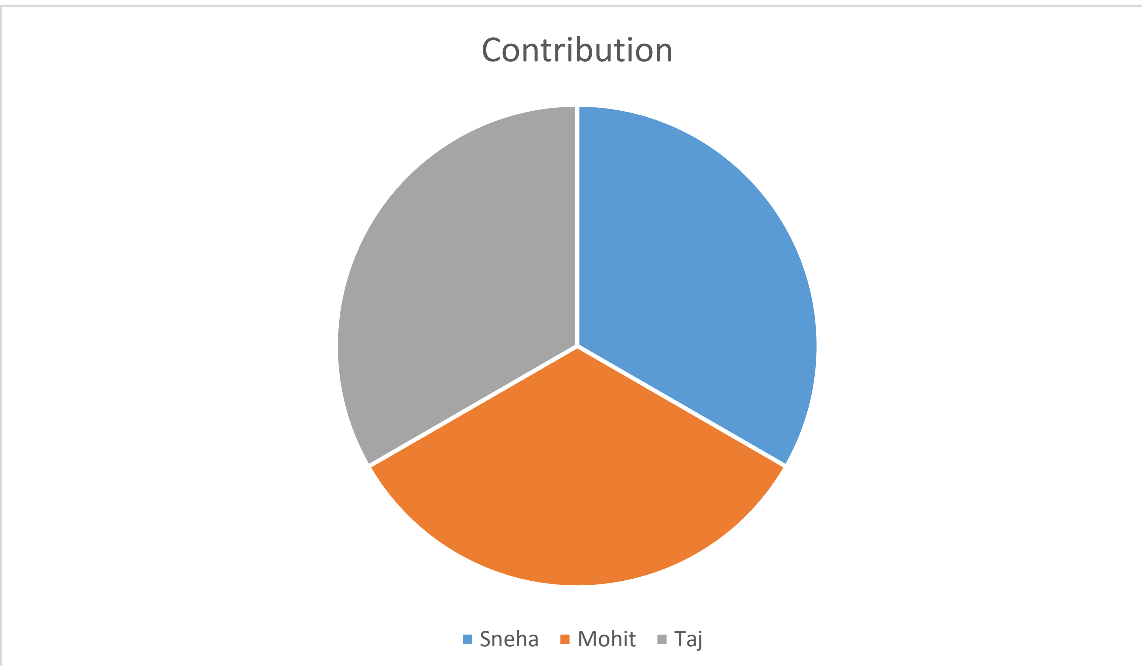
Why are interest rates in Alabama higher than interest rates around San Francisco?

Company executives, online lending experts, and the Lending Club website all confirm that interest rates are a function of loan grades. The logic is simple enough. The borrowers who are assigned A and B loan grades tend to have the healthiest credit metrics and represent the lowest lending risk. Grades C and D, down to G, tend to have progressively lower credit scores. Once a loan grade is assigned to a loan application, the corresponding interest rate can simply be obtained from a table. Consequently, if interest rates are higher in Alabama, as shown in the hot spot map above, it is fair to assume it is because the loan grades assigned in those regions reflect riskier loans.

> *A risky borrower in San Francisco should be just as risky in Mobile.*

TEAM CONTRIBUTION:

Name	Work done
Mohit Mittal	Docker Image Build, Luigi Pipline Construction, Data Clean, Report Making
Taj Poovaiah	Loan Decline Data- Cleaning, Jupyter notebook Charts and Analysis, Report Making
Sneha Ravikumar	Scraping, Cleaning loan Data, Data Analysis and Summry on Power BI, Report Making



END OF REPORT