

## BA648 FINAL PROJECT

Student: Nhat My Thien Nguyen

Professor: Carol Cagle

### I. EXECUTIVE SUMMARY

Working as a business with Hardy Business Loans (HBL), I need to analyze the two portfolios come from two sources which are Beachside Lenders and WilsonandSons.

The Beachside data contains 393211 applications (observations) and The Wilsonandsons contains 294997 application. Each of application in both data has unique loan number.

I used SQL to import two dataset, then I combined and create table2 (Loan Application) which are combined from 2 dataset with corresponding variables.

#### Table2 Dictionary:

No	Loan Application Variables	Type	Description
1	member_number	Varchar	Member Number
2	loan_number	Varchar	Loan Number assigned by lender
3	net_loan_amount	Integer	Loan Amount
4	grade	Character	LC assigned loan grade (A, B, C, D, E, F, G)
5	customer_job	Varchar	Employment Title
6	length_of_employment	Varchar	Time in job
7	income	Numeric	The self-reported annual income provided by the borrower during registration.
8	loan_date	Date	Loan Date
9	loan_status	Varchar	Current status of the loan
10	title	Varchar	The loan title provided by the borrower
11	customer_zip	Varchar	Customer Zip (Region)
12	customer_state	Character	Customer State/Province

<b>13</b>	dti	Numeric	
<b>14</b>	member_date	Date	Member Date
<b>15</b>	high_fico	Integer	FICO Credit Score
<b>16</b>	delinquent	Integer	The number of months since the borrower's last delinquency.
<b>17</b>	total_payments	Numeric	Total Payments
<b>18</b>	last_date_paid	Date	Last Date Paid
<b>19</b>	last_payment	Numeric	Last Payment
<b>20</b>	last_credit_check	Varchar	The most recent month LC pulled credit for this loan
<b>21</b>	average_balance	Numeric	Average Balance
<b>22</b>	source	Varchar	Data source (Beachside or Wilsonandsons)

Then I used SQL to clean Table2 data. This step is to transform your data in preparation for analysis. I search for duplications and missing values then deal with each problem.

Duplicated member rows makes sense which means one single person did apply for multiple loan. On the contrast, duplicated loan number rows mean that person apply same application to two company Beachside and Wilsonandwilson.

After cleaning table2, I did some summary statistics, graph and charts to visualize our data.

For creating Prediction table, I create Table1 which is extract some variables from cleaned Table2, then add 3 more column, which are Down (Down Payment), Credrisk (Bad, Good, Indeterminant), and Response (Booked, Unbooked) and filled values to these new column by set some rules.

The result shows that Wilsonandsons has higher good level of customer comparing to Beachside by credit risk variables

Last but not least, I use Python to separate Table 1 data into training set and test set to prepare for further steps.

## **II. SUMMARY STATISTIC**

First I used python for general summary statistic which shows mean, standard deviation, mode, quatile for all variables.

```
In [126]: round(table2.describe(),2)
```

Out[126]:

	member_number	loan_number	net_loan_amount	income	dti	high_fico	delinquent	total_payments	last_payment	average_balance	re
count	577866.00	5.778660e+05	577866.00	577866.00	577866.00	577866.00	287796.00	577866.00	577866.00	577866.00	
mean	755559.02	8.815695e+07	15262.24	78205.86	inf	702.24	34.37	12200.36	3458.44	13456.37	
std	523009.04	3.366834e+07	9234.84	70754.70	NaN	33.10	21.93	9663.73	6115.46	16252.65	
min	5.00	5.670500e+04	1000.00	0.00	0.00	664.00	0.00	0.00	0.00	0.00	
25%	336427.00	5.901375e+07	8000.00	46000.00	0.10	674.00	16.00	4854.87	310.38	3116.00	
50%	671495.50	7.748040e+07	13200.00	65000.00	3.07	694.00	31.00	9509.14	595.21	7307.00	
75%	1020415.75	1.182269e+08	20000.00	94200.00	18.06	719.00	50.00	16978.80	3661.52	18577.00	
max	1999986.00	1.411273e+08	40000.00	10999200.00	inf	850.00	179.00	60841.20	42192.05	445856.00	

As results, BAD applications was much more than GOOD application in both beachside and wilsonandsons.

```
SELECT source, credrisk, COUNT (credrisk)
FROM table1
GROUP BY source,credrisk;
```

	source text	credrisk text	count bigint
1	beachside	BAD	196945
2	beachside	GOOD	85924
3	wilsonandsons	BAD	205199
4	wilsonandsons	GOOD	89798

Thus, Unbooked Response is more than Book Response respectively.

```
SELECT source, response, COUNT (response)
FROM table1
GROUP BY source,response;
```

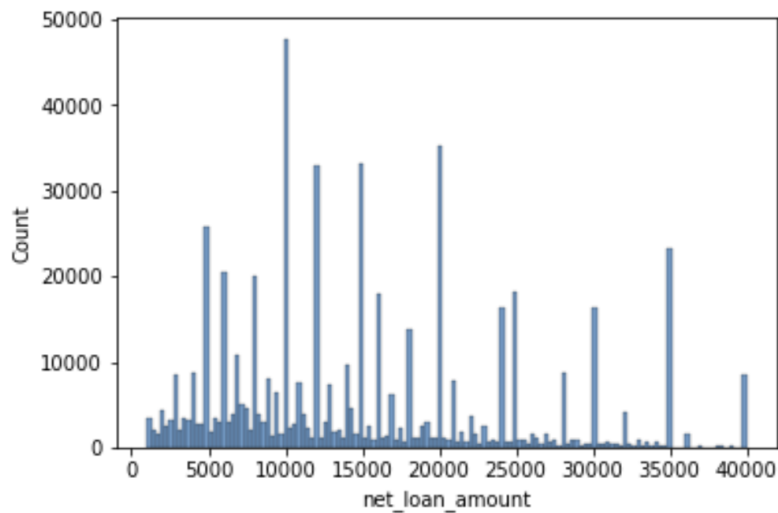
	source text	response text	count bigint
1	beachside	Booked	85924
2	beachside	Unbooked	196945
3	wilsonandsons	Booked	89798
4	wilsonandsons	Unbooked	205199

### III. GRAPHS AND CHARTS

I used python to do some graphs and charts to visualize our data.

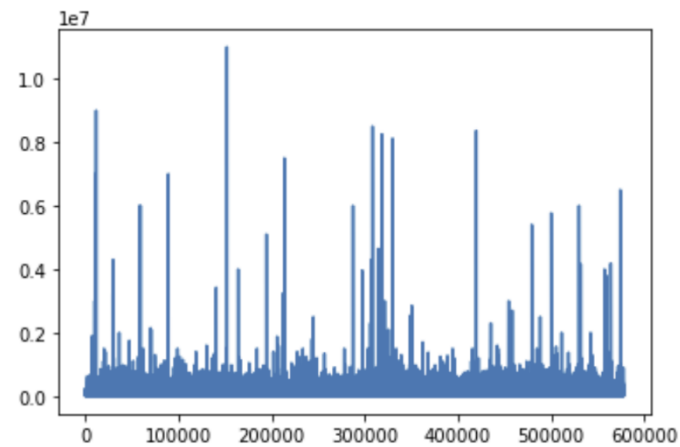
```
In [127]: ▶ seaborn.histplot(data=table2,x='net_loan_amount')
```

```
Out[127]: <AxesSubplot:xlabel='net_loan_amount', ylabel='Count'>
```

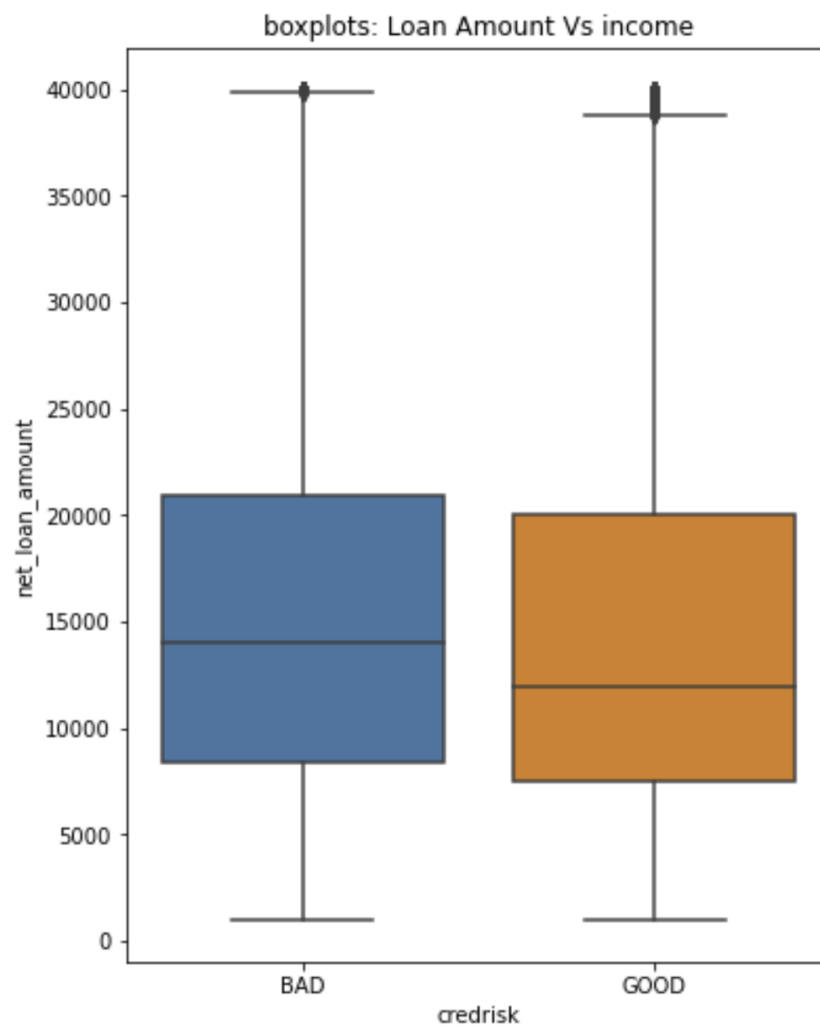


```
In [128]: ▶ table2['income'].plot()
```

```
Out[128]: <AxesSubplot:>
```

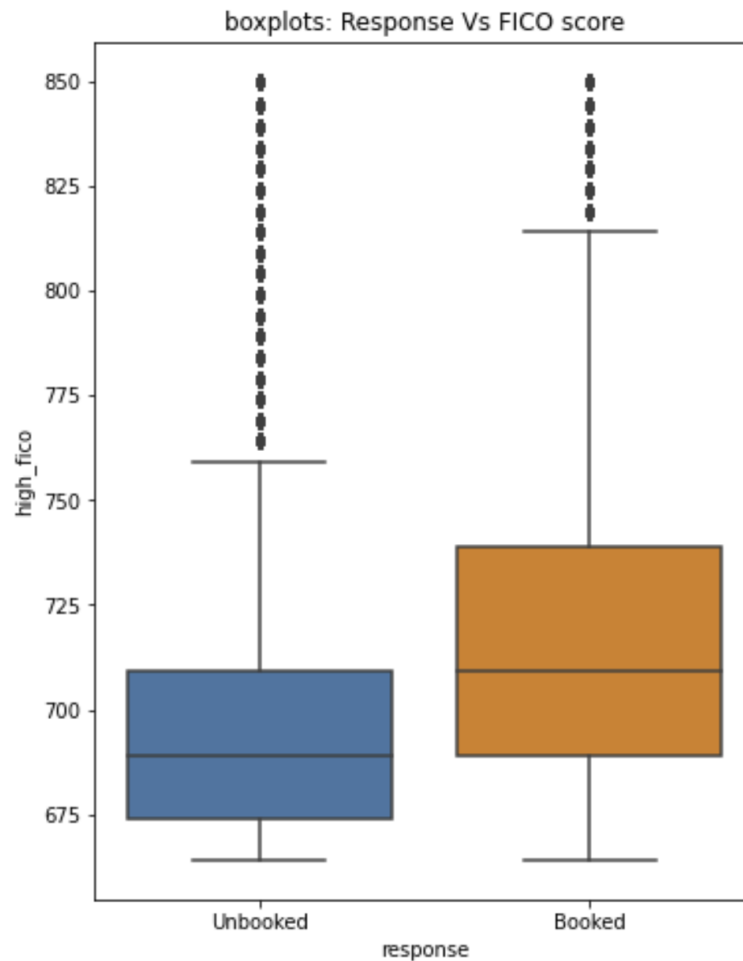


```
In [17]: ▶ plt.figure(figsize=(6,8))  
plt.title('boxplots: Loan Amount Vs income')  
sns.boxplot(data=table1,y='net_loan_amount',x='credrisk')  
plt.show()
```



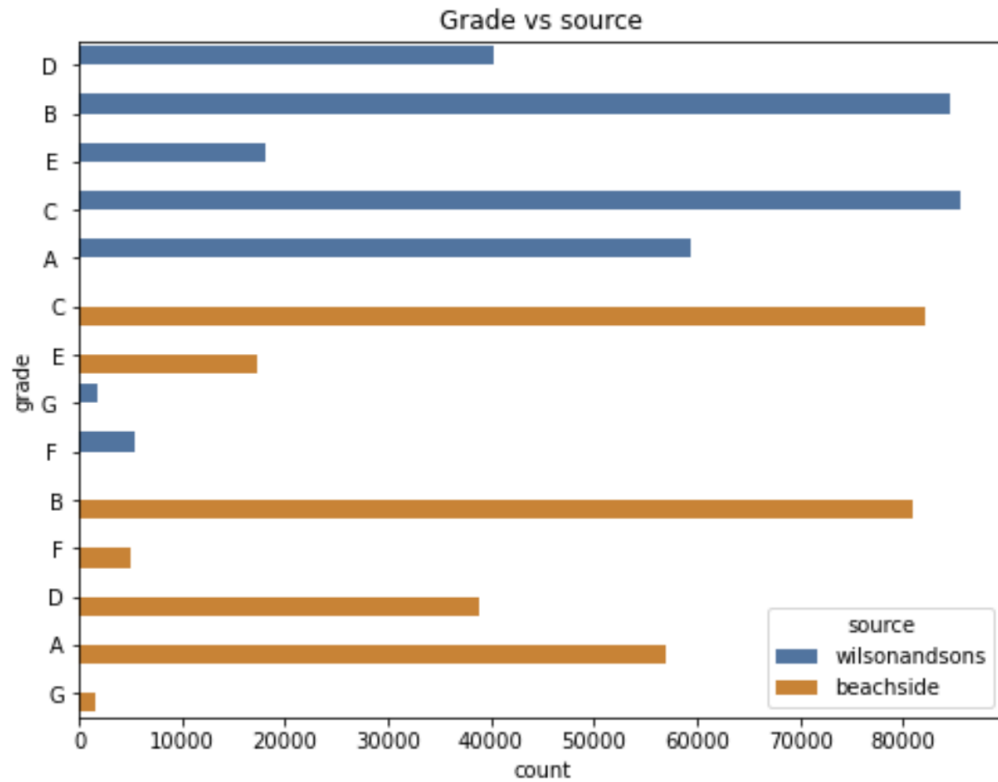
We can see that the Credrisk assigned as “Good” has lower Loan Amount than that as “Bad”.

```
In [27]: ▶ plt.figure(figsize=(6,8))  
plt.title('boxplots: Response Vs FICO score')  
sns.boxplot(data=table1,y='high_fico',x='response')  
plt.show()
```



The boxplot shows that the booked application has much more higher FICO score than the unbooked application.

```
n [37]: ▶ plt.figure(figsize=(8,6))
plt.title('Grade vs source')
sns.countplot(y='grade',hue='source',data =table1)
plt.show()
```



The plot shows frequency of Grade between wilsonandsons and beachside. Overall, class of grade in two source is approximately to each other.

#### IV. DATA CLEANING JOURNAL and SQL CODE

```
SELECT * FROM Table2;
```

	member_number character varying (10)	loan_number character varying (10)	net_loan_amount integer	grade character	customer_job character varying (50)	length_of_employment character varying (15)	income numeric	l c
1	1592618	68407277	3600	C	leadman	10+ years	55000	D
2	1945270	68341763	20000	B	truck driver	10+ years	63000	D
3	672413	66310712	35000	C	Information Systems Officer	10+ years	110000	D
4	1263724	68356421	22400	C	Executive Director	6 years	95000	D
5	661182	68566886	29900	C	Corporate Insurance	10+ years	65000	D
6	678965	68607141	17600	A	Network Security Specialist III	< 1 year	88000	D
7	1582780	68526883	15000	A	DEPUTY SHERIFF	10+ years	44000	D
8	1767504	68533595	12000	B	Public Affairs Specialist	1 year	98000	D
9	1693178	68426865	20000	C	GM	6 years	110000	D
10	1487057	68476676	20000	D	Clinical Intake	5 years	70000	D
11	363557	68527009	20000	B	Manager	5 years	75000	D
12	1009061	68446771	7200	D	Bank Officer Senior Custome...	< 1 year	55000	D
13	459530	68416916	28000	D	Engineer	10+ years	75000	D
14	149715	68597047	19000	C	Application Analyst	9 years	75000	D
15	1351159	68617057	15000	B	Senior Buyer	10+ years	65000	D

SELECT COUNT (\*) FROM Table2;

	count bigint
1	688208

--CHECKING AND DELETE DUPLICATED OBSERVATIONS

SELECT Member\_Number, COUNT(\*)

FROM Table2

GROUP BY Member\_Number

HAVING COUNT(\*) > 1

	member_number character varying (10)	count bigint
1	208465	2
2	934171	2
3	1897934	2
4	88490	2
5	891242	2
6	567560	2
7	664339	2
8	279454	2
9	1345617	2
10	427552	4
11	615375	2

--> There are 112991 duplicated member number, which means there are many people who did more than 1 application. This duplicated is reasonable so we can keep it as it is.



```
SELECT Loan_Number, COUNT(*)
FROM Table2
GROUP BY Loan_Number
HAVING COUNT(*) > 1;
```

	loan_number character varying (10)	count bigint
1	65772403	2
2	63254340	2
3	68585734	2
4	67417378	2
5	65604509	2
6	63179585	2
7	67115561	2
8	67346412	2
9	65352556	2

-->There are 110342 duplicated loan number, which means there are 110342 people who did the applications in both companies. These duplications matter so I will delete duplicated rows.

```
DELETE FROM Table2 a USING Table2 b
WHERE a.loan_number=b.loan_number AND a.record_number > b.record_number;
```

```
SELECT Loan_Number, COUNT(*)
FROM Table2
GROUP BY Loan_Number
HAVING COUNT(*) > 1;
```

	loan_number character varying (10)	count bigint

## --CHECKING FOR MISSING VALUES

-- Member\_Number:

```
SELECT COUNT (*) Member_Number
FROM table2
WHERE Member_Number IS NULL;
```

	member_number bigint
1	0

-->There is no missing value in Member Number

-- Loan\_Number:

```
SELECT COUNT (*) Loan_Number
```

```
FROM table2
```

```
WHERE loan_number IS NULL;
```

-->There is no missing value in Loan Number

-- Net\_Loan\_Amount:

```
SELECT COUNT (*) Net_Loan_Amount
```

```
FROM table2
```

```
WHERE Net_Loan_Amount IS NULL;
```

-->There is no missing value in Net\_Loan\_Amount

-- Grade:

```
SELECT COUNT (*) grade
```

```
FROM table2
```

```
WHERE grade IS NULL;
```

-->There is no missing value in Grade

```
SELECT COUNT (*) Customer_job
```

```
FROM table2
```

```
WHERE Customer_job IS NULL;
```

-->There are 40605 missing values. I replace missing values by "Not-specified"

```
UPDATE table2
```

```
SET Customer_job='Not-specified'
```

```
WHERE Customer_job IS NULL;
```

	customer_job bigint	
1	40605	

-- Length\_of\_employment:

```
SELECT COUNT (*) length_of_employment
FROM table2
WHERE length_of_employment IS NULL;
```

	length_of_employment	
	bigint	
1		38658

-->There are 38658 missing values. Since this is categorical variable so I will replace missing values by mode.

```
UPDATE table2
SET length_of_employment=(SELECT mode() WITHIN GROUP (ORDER BY
length_of_employment)
FROM table2)
WHERE length_of_employment IS NULL;
```

-- Loan\_date:

```
SELECT COUNT (*) loan_date
FROM table2
WHERE loan_date IS NULL;
```

-->There is no missing value in Loan\_Date

-- Loan\_status:

```
SELECT COUNT (*) loan_status
FROM table2
WHERE loan_status IS NULL;
```

-->There is no missing value in Loan\_Status

-- Title:

```
SELECT COUNT (*) Title
FROM table2
WHERE Title IS NULL;
```

	title	
	bigint	
1		6708

-->There are 6708 missing value in Title. Since this variable is categorical so I will replace missing value by mode.

```
UPDATE table2
SET title=(SELECT mode() WITHIN GROUP (ORDER BY title)
FROM table2)
WHERE title IS NULL;
```

-- Customer\_zip:

```
SELECT COUNT (*) customer_zip
FROM table2
WHERE customer_zip IS NULL;
```

-->There is no missing value in Customer Zip

-- Customer\_state:

```
SELECT COUNT (*) customer_state
FROM table2
WHERE customer_state IS NULL;
```

-->There is no missing value in Customer State

--- DTI:

```
SELECT COUNT (*) dti
FROM table2
WHERE dti IS NULL;
```

	dti bigint 
1	295238

-->There are 295238 missing values in dti, most of them are from wilsonandsons dataset. I assume and replace missing values by formula : (last\_payment x 12)/ income.

```
UPDATE table2
SET dti = CASE
WHEN income > 0 THEN last_payment*12/income
ELSE dti
END;
```

--- High Fico:

```
SELECT COUNT (*) high_fico
```

```
FROM table2
```

```
WHERE high_fico IS NULL;
```


-->There is no missing value in High\_Fico

-- Delinquent:

```
SELECT COUNT (*) delinquent
```

```
FROM table2
```

```
WHERE delinquent IS NULL;
```

Data Output		Expla
	<b>delinquent</b> bigint	
1	290070	

-->There are 290070 missing value in delinquent. These missing values make senses because that implies borrowers do not have delinquent payment. Thus it doesnot need to filled.

-- total\_payments:

```
SELECT COUNT (*) total_payments
```

```
FROM table2
```

```
WHERE total_payments IS NULL;
```

-->There is no missing value in total\_payments

-- Last\_date\_paid:

```
SELECT COUNT (*) Last_date_paid
```

```
FROM table2
```

```
WHERE Last_date_paid IS NULL;
```

-->There is no missing value in last\_date\_paid

-- Last\_payment:

```
SELECT COUNT (*) last_payment
```

```
FROM table2
```

```
WHERE last_payment IS NULL;
```

-->There is no missing value in last\_payment

-- last\_credit\_check:

```
SELECT COUNT (*) last_credit_check
```

```
FROM table2
```

```
WHERE last_credit_check IS NULL;
```

-->There is no missing value last\_credit\_check

-- CREATE PREDICTIVE TABLE (TABLE1)

```
CREATE TABLE table1 AS
```

```
SELECT
```

```
Member_Number,
```

```
Member_Date,
```

```
Loan_Number,
```

```
Loan_Date,
```

```
Loan_Status,
```

```
Customer_Job,
```

```
Length_of_Employment,
```

```
Income,
```

```
dti,
```

```
Customer_State,
```

```
Customer_Zip,
```

```
Net_Loan_Amount,
```

```
" AS Down,
```

```
" AS CredRisk,
```

```
Source,
```

```
High_FICO,
```

```
Grade,
```

```
delinquent,
```

```
" AS Response
```

```
FROM table2;
```

```
SELECT*FROM Table1;
```

	member_number character varying (10)	member_date character varying (10)	loan_number character varying (10)	loan_date character varying	loan_status character varying	customer_job character varying (50)	length charac
1	723567	Nov-09	68356922	Dec-15	Fully Paid	Patrol Officer	2 year:
2	654395	Feb-02	68536675	Dec-15	Fully Paid	EVS Attendant	< 1 ye
3	63755	Dec-08	68132490	Dec-15	Fully Paid	Manager	4 year:
4	497368	Jan-79	68212247	Dec-15	Fully Paid	[null]	[null]
5	29306	Oct-84	67859130	Dec-15	Charged Off	[null]	[null]
6	958027	Jan-77	67398758	Dec-15	Fully Paid	MD	10+ ye
7	720832	Sep-01	67398721	Dec-15	Charged Off	Sales	2 year:
8	939923	Jul-92	67346005	Dec-15	Fully Paid	[null]	[null]
9	1281888	May-06	66985475	Dec-15	Fully Paid	Anesthesia tech	< 1 ye
10	1313971	Apr-98	66085056	Dec-15	Fully Paid	[null]	10+ ye
11	1794033	Jan-07	66561734	Dec-15	Current	[null]	[null]
12	706085	Nov-96	65905989	Dec-15	Current	Owner	2 year:
13	1592618	Aug-03	68407777	Dec-15	Fully Paid	leadman	10+ ye

-- FILL VALUES TO "Down" column, assume that down payment = 15% of loan amount.

UPDATE Table1

SET Down = net\_loan\_amount\*15/100;

	net_loan_amount integer	down text
1	8000	1200
2	7300	1095
3	15825	2373
4	8000	1200
5	5000	750
6	8800	1320
7	7000	1170

-- FILL VALUE TO CredRisk Column following rules:

- Loans graded below "C" are "Bad"
- Months since last delinquent is less than 3 are "Bad"
- Debt to income ratio greater than 30% is "Bad"
- Loans with a status of "Charged Off" or "Default are "Bad"
- All other loans are "Good"

UPDATE Table1

SET Credrisk = CASE

WHEN dti > 0.3 THEN 'BAD'

```
WHEN grade <> 'A' AND grade <> 'B' THEN 'BAD'
```

```
WHEN delinquent <> Null AND delinquent < 3 THEN 'BAD'
```

```
WHEN loan_status = 'Charged Off' OR loan_status= 'Default' THEN 'BAD'
```

```
ELSE 'GOOD' END;
```

-- FILL VALUE TO Response Column following rules:

- “Credrisk” is Good then Response is “Booked”
- “Credrisk” is Bad then Response is “Unbooked”
- “Credrisk” is Indeterminant and FICO score above 700 then Response is “Booked”
- “Apart” from those, Response is “Unbook”

```
UPDATE table1
```

```
SET Response = CASE
```

```
WHEN Credrisk = 'GOOD' THEN 'Booked'
```

```
WHEN Credrisk = 'BAD' THEN 'Unbooked'
```

```
WHEN Credrisk = 'Indeterminant' AND High_FICO >= 700 THEN 'Booked'
```

```
ELSE 'Unbooked'
```

```
END;
```

## V. TEST AND TRAINING DATA

For this process, I create test and training data by using Python. My training set contains 80% of the record whereas test set contain 20% of the record. After I cleaned and combined two data and created new variables, I create the predictive file which is not include the ‘Unbooked’ data.



```
In [6]: data_train,data_test=train_test_split(data,test_size=0.2)
```

```
In [7]: data_train
```

Out[7]:

	member_number	loan_number	net_loan_amount	grade	customer_job	length_of_employment	income	loan_date	loan_status	
197450	516812	75201401	35000	C	Dispatcher/Superintendent	10+ years	100000.0	Apr-16	Current	consoli
491286	884416	136593515	15000	C	Program Manager	10+ years	160000.0	Jul-18	Current	consoli
451933	677965	80629526	3000	D	CST	1 year	22000.0	May-16	Fully Paid	M exp
361751	833577	45524305	14000	A	Doctor	< 1 year	180000.0	Apr-15	Fully Paid	consoli
196227	748485	76383363	15000	A	Driver	5 years	78000.0	Apr-16	Current	improv
...	...	...	...	...	...	...	...	...	...	...
145240	1735274	126477986	3500	B	registered nurse	10+ years	115000.0	Jan-18	Current	consoli
401849	586774	128119586	22000	B	Practice Administrator	10+ years	54000.0	Feb-18	Current	Cred refin
101082	1675147	41410355	10000	B	owner	10+ years	90000.0	Mar-15	Fully Paid	consoli
40933	1842879	60236527	5500	A	Line Cook	< 1 year	30000.0	Sep-15	Charged Off	consoli
360817	905062	46014190	18000	C	Solid waste operator 3	10+ years	44524.0	Apr-15	Charged Off	consoli

462292 rows × 23 columns

```
In [8]: data_test
```

Out[8]:

	member_number	loan_number	net_loan_amount	grade	customer_job	length_of_employment	income	loan_date	loan_status	title	...
393896	280531	129550268	15000	C	Loan Processing	10+ years	43000.0	Mar-18	Current	Other	...
102308	978169	41082088	8400	B	Academic System and Data Specialist	8 years	45000.0	Feb-15	Fully Paid	Debt consolidation	...
61848	537878	55249949	16200	D	Admin Asst	2 years	45000.0	Jul-15	Fully Paid	Debt consolidation	...
38420	1455764	60920878	10000	D	Accounting Manager	6 years	33000.0	Sep-15	Charged Off	Debt consolidation	...
568051	482049	127600644	7975	A	Not-specified	10+ years	40000.0	Jan-18	Current	Credit card refinancing	...
...	...	...	...	...	...	...	...	...	...	...	...
494838	455700	136398973	7000	D	Client Support Professional	3 years	26500.0	Jul-18	Current	Credit card refinancing	...
356849	993778	46662315	5000	B	Employment Counselor	5 years	31000.0	Apr-15	Fully Paid	Credit card refinancing	...
204949	1576964	140823387	35000	D	Contract Consultant	< 1 year	85000.0	Sep-18	Current	Debt consolidation	...
487858	734068	137543210	28000	D	Administrative Assistant	< 1 year	18000.0	Jul-18	Current	Debt consolidation	...
553267	1478183	81443809	14000	C	Not-specified	10+ years	45000.0	Jun-16	Current	Credit card refinancing	...

115574 rows × 23 columns

