ALVIN'S ANSWER FOR PLANO'S SENIOR DATA SCIENTIST ASSESSMENT

TESTER: DWIGHT

CONTENTS

GLANCING THE COLUMNS

COLUMNS A TO G

	A	B	C	D	E	F	G
1	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate
2	1077501	1296599	5000	5000	4975	36 months	10.65
3	1077430	1314167	2500	2500	2500	60 months	15.27
4	1077175	1313524	2400	2400	2400	36 months	15.96
5	1076863	1277178	10000	10000	10000	36 months	13.49
6	1075358	1311748	3000	3000	3000	60 months	12.69
7	1075269	1311441	5000	5000	5000	36 months	7.9
8	1069639	1304742	7000	7000	7000	60 months	15.96
9	1072053	1288686	3000	3000	3000	36 months	18.64
10	1071795	1306957	5600	5600	5600	60 months	21.28

- id
- member_id
- loan_amnt
- funded_amnt
- funded_amnt_inv
- term
- int_rate

COLUMNS H TO N

	H	I	J	K	L	M	N
1	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc
2	162.87	B	B2		10+ years	RENT	24000
3	59.83	C	C4	Ryder	< 1 year	RENT	30000
4	84.33	C	C5		10+ years	RENT	12252
5	339.31	C	C1	AIR RESOURCES BOARD	10+ years	RENT	49200
6	67.79	B	B5	University Medical Group	1 year	RENT	80000
7	156.46	A	A4	Veolia Transportaton	3 years	RENT	36000
8	170.08	C	C5	Southern Star Photography	8 years	RENT	47004
9	109.43	E	E1	MKC Accounting	9 years	RENT	48000
10	152.39	F	F2		4 years	OWN	40000

- installment grade
- sub_grade
- emp_title
- emp_length
- home_ownership
- annual_inc

COLUMNS O TO T

	O	P	Q	R	S	T
1	verification_status	issue_d	loan_status	pymnt_plan	url	desc
2	Verified	Dec-2011	Fully Paid	n	https://www.lend	Borrower added on 12/22/11 > I need to upgrade my business te
3	Source Verified	Dec-2011	Charged Off	n	https://www.lend	Borrower added on 12/22/11 > I plan to use this money to financ
4	Not Verified	Dec-2011	Fully Paid	n	https://www.lendingclub.com/browse/loanDetail.action?loan_id=1077175	
5	Source Verified	Dec-2011	Fully Paid	n	https://www.lend	Borrower added on 12/21/11 > to pay for property tax (borrow fro
6	Source Verified	Dec-2011	Current	n	https://www.lend	Borrower added on 12/21/11 > I plan on combining three large in
7	Source Verified	Dec-2011	Fully Paid	n	https://www.lendingclub.com/browse/loanDetail.action?loan_id=1075269	
8	Not Verified	Dec-2011	Current	n	https://www.lend	Borrower added on 12/18/11 > I am planning on using the funds
9	Source Verified	Dec-2011	Fully Paid	n	https://www.lend	Borrower added on 12/16/11 > Downpayment for a car.
10	Source Verified	Dec-2011	Charged Off	n	https://www.lend	Borrower added on 12/21/11 > I own a small home-based judgm

- verification_status
- issue_d
- loan_status
- pymnt_plan
- url
- desc

COLUMNS U TO AC

	U	V	W	X	Y	Z	AA	AB	AC
1	purpose	title	zip_code	addr_state	dti	delinq_2yrs	earliest_cr_line	inq_last_6mths	mths_since_last_delinq
2	credit_card	Computer	860xx	AZ	27.65	0	Jan-1985	1	
3	car	bike	309xx	GA	1	0	Apr-1999	5	
4	small_busine	real estate bu	606xx	IL	8.72	0	Nov-2001	2	
5	other	personel	917xx	CA	20	0	Feb-1996	1	35
6	other	Personal	972xx	OR	17.94	0	Jan-1996	0	38
7	wedding	My wedding l	852xx	AZ	11.2	0	Nov-2004	3	
8	debt_consolid	Loan	280xx	NC	23.51	0	Jul-2005	1	
9	car	Car Downpay	900xx	CA	5.35	0	Jan-2007	2	
10	small_busine	Expand Busin	958xx	CA	5.55	0	Apr-2004	2	

- purpose
- title
- zip_code
- addr_state
- dti
- delinq_2yrs
- earliest_cr_line
- inq_last_6mths
- mths_since_last_delinq

COLUMNS AD TO AL

	AD	AE	AF	AG	AH	AI	AJ	AK	AL
1	mths_since_last_record	open_acc	pub_rec	revol_bal	revol_util	total_acc	initial_list_status	out_prncp	out_prncp_inv
2		3	0	13648	83.7	9	f	0	0
3		3	0	1687	9.4	4	f	0	0
4		2	0	2956	98.5	10	f	0	0
5		10	0	5598	21	37	f	0	0
6		15	0	27783	53.9	38	f	766.9	766.9
7		9	0	7963	28.3	12	f	0	0
8		7	0	17726	85.6	11	f	1889.15	1889.15
9		4	0	8221	87.5	4	f	0	0
10		11	0	5210	32.6	13	f	0	0

- mths_since_last_record
- open_acc
- pub_rec revol_bal
- revol_util
- total_acc
- initial_list_status out_prncp
- out_prncp_inv

COLUMNS AM TO AU

	AM	AN	AO	AP	AQ	AR	AS	AT	AU
1	total_pymnt	total_pymnt_inv	total_rec_prncp	total_rec_int	total_rec_late	recoveries	collection_recovery_fee	last_pymnt_d	last_pymnt_amnt
2	5861.071414	5831.78	5000	861.07	0	0	0	Jan-2015	171.62
3	1008.71	1008.71	456.46	435.17	0	117.08	1.11	Apr-2013	119.66
4	3003.653644	3003.65	2400	603.65	0	0	0	Jun-2014	649.91
5	12226.30221	12226.3	10000	2209.33	16.97	0	0	Jan-2015	357.48
6	3242.17	3242.17	2233.1	1009.07	0	0	0	Jan-2016	67.79
7	5631.377753	5631.38	5000	631.38	0	0	0	Jan-2015	161.03
8	8136.84	8136.84	5110.85	3025.99	0	0	0	Jan-2016	170.08
9	3938.144334	3938.14	3000	938.14	0	0	0	Jan-2015	111.34
10	646.02	646.02	162.02	294.94	0	189.06	2.09	Apr-2012	152.39

- total_pymnt
- total_pymnt_inv
- total_rec_prncp
- total_rec_int
- total_rec_late_fee
- recoveries
- collection_recovery_fee
- last_pymnt_d
- last_pymnt_amnt

COLUMNS AV TO BB

	AV	AW	AX	AY	AZ	BA	BB
1	next_pymnt_d	last_credit_pull_d	collections_12_mths_ex_med	mths_since_last_major_derog	policy_code	application_type	annual_inc_joint
2		Jan-2016	0		1	INDIVIDUAL	
3		Sep-2013	0		1	INDIVIDUAL	
4		Jan-2016	0		1	INDIVIDUAL	
5		Jan-2015	0		1	INDIVIDUAL	
6	Feb-2016	Jan-2016	0		1	INDIVIDUAL	
7		Sep-2015	0		1	INDIVIDUAL	
8	Feb-2016	Jan-2016	0		1	INDIVIDUAL	
9		Dec-2014	0		1	INDIVIDUAL	
10		Aug-2012	0		1	INDIVIDUAL	

- next_pymnt_d
- last_credit_pull_d
- collections_12_mths_ex_med
- mths_since_last_major_derog
- policy_code
- application_type
- annual_inc_joint

COLUMNS BB TO BJ

	BB	BC	BD	BE	BF	BG	BH	BI	BJ
1	annual_inc_joint	dti_joint	verification_status_joint	acc_now_delinq	tot_coll_amt	tot_cur_bal	open_acc_6m	open_il_6m	open_il_12m
2				0					
3				0					
4				0					
5				0					
6				0					
7				0					
8				0					
9				0					
10				0					

- dti_joint
- verification_status_joint
- acc_now_delinq
- tot_coll_amt
- tot_cur_bal
- open_acc_6m
- open_il_6m
- open_il_12m

COLUMNS BK TO BT

	BK	BL	BM	BN	BO	BP	BQ	BR	BS	BT	
1	open_il_24m	mths_since_rcnt_il	total_bal_il	il_util	open_rv_12m	open_rv_24m	max_bal_bc	all_util	total_rev_hi_lim	inq_fi	t
2											
3											
4											
5											
6											
7											
8											
9											
10											

- open_il_24m
- mths_since_rcnt_il
- total_bal_il
- il_util
- open_rv_12m
- open_rv_24m
- max_bal_bc
- all_util
- total_rev_hi_lim
- inq_fi

COLUMNS BU TO BV

	BU	BV
1	total_cu_tl	inq_last_12m
2		
3		
4		
5		
6		
7		
8		
9		
10		

- total_cu_tl
- inq_last_12m

PART I

(A) USE SEED (1234) AND SAMPLE 7000 ROWS FROM THE ENTIRE DATASET.

```
!pip install numpy
!pip install matplotlib
!pip install seaborn
!pip install pandas
!pip install scipy
!pip install sklearn
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
import pandas as pd
import sklearn
```

```
df = pd.read_csv('/home/dralvin/Desktop/PLANO/Plano-Data Scientist assessment/From Dwight/LendingClubLoan.csv')
```

```
df7000 = df.sample(n=7000, random_state=1234)
```

```
df7000
```

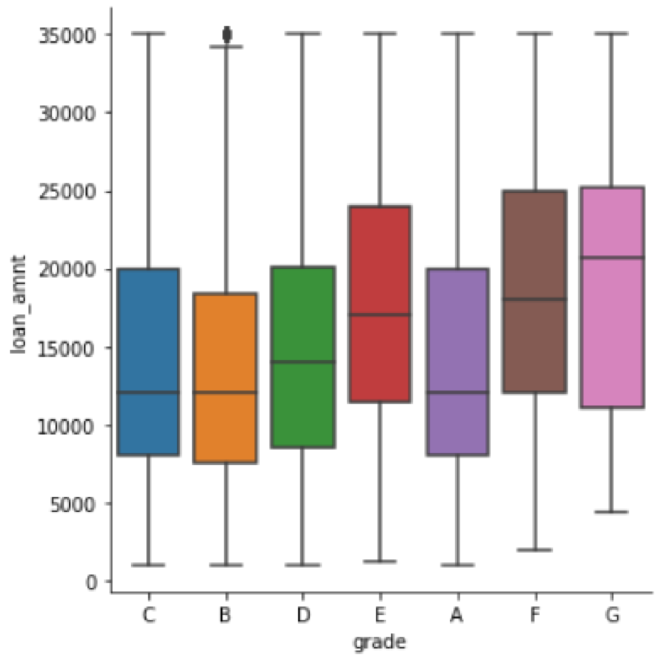
	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	total_bal_il	il_util	open_rv_12
132265	5042403	6344974	8000.0	8000.0	8000.0	36 months	14.33	274.71	C	C1	...	NaN	NaN	Na
571549	61360126	65478888	5400.0	5400.0	5400.0	36 months	9.17	172.15	B	B2	...	NaN	NaN	Na
596193	60741457	64783260	10000.0	10000.0	10000.0	36 months	17.86	360.83	D	D5	...	NaN	NaN	Na
207309	1417317	1667580	3000.0	3000.0	3000.0	36 months	15.31	104.46	C	C2	...	NaN	NaN	Na
468550	68575005	73464780	2000.0	2000.0	2000.0	36 months	13.44	67.82	C	C3	...	1954.0	97.7	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
270934	28743363	31276509	15000.0	15000.0	15000.0	60 months	9.17	312.62	B	B1	...	NaN	NaN	Na

(B) PLEASE PLOT THE FOLLOWING:

PLOT A BOXPLOT OF LOAN_AMT VS GRADE.

```
In [17]: sb.catplot(data=df7000,x="grade", y="loan_amnt", kind="box")
```

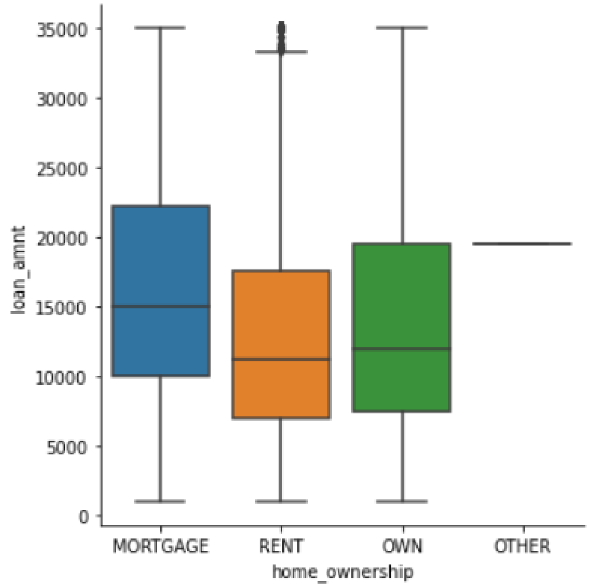
```
Out[17]: <seaborn.axisgrid.FacetGrid at 0x7fc0c0585040>
```



PLOT A BOXPLOT OF LOAN_AMT VS HOME_OWNERSHIP

```
In [22]: sb.catplot(data=df7000,x="home_ownership", y="loan_amnt", kind="box")

Out[22]: <seaborn.axisgrid.FacetGrid at 0x7fc098061ac0>
```



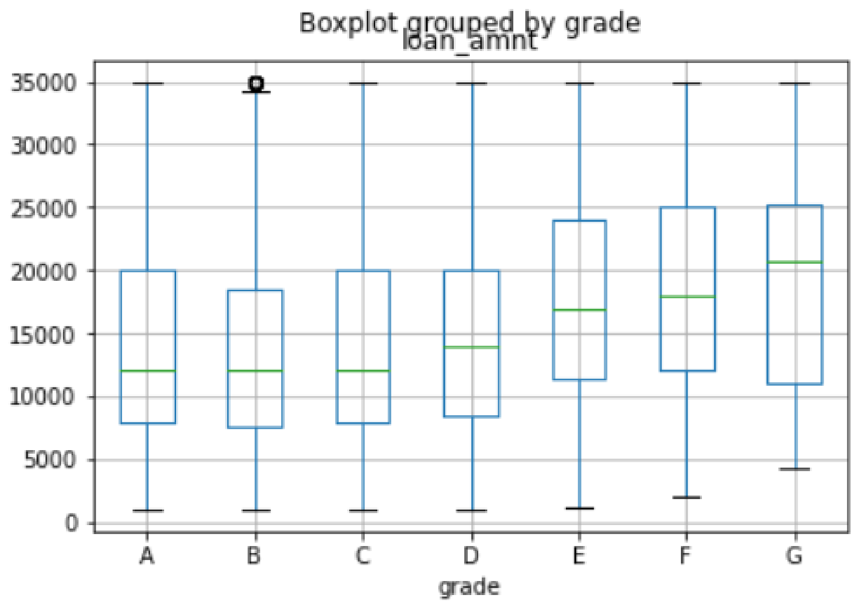
(C) A BANKER PROPOSES THAT LOAN_AMOUNT FROM GRADE A TO D HAS NO DIFFERENCE.

PLEASE PERFORM THE APPROPRIATE STATISTICAL TESTS AND INTERPRET THE RESULTS TO VALIDATE HIS ASSUMPTION.

```
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.api import anova_lm
```

```
df7000.boxplot('loan_amnt','grade')
```

```
<AxesSubplot:title={'center':'loan_amnt'}, xlabel='grade'>
```



```
model = ols('loan_amnt ~ grade', df7000).fit()
```

```
anova_lm(model)
```

	df	sum_sq	mean_sq	F	PR(>F)
grade	6.0	1.317210e+10	2.195350e+09	31.973954	3.560955e-38
Residual	6993.0	4.801433e+11	6.866056e+07	NaN	NaN

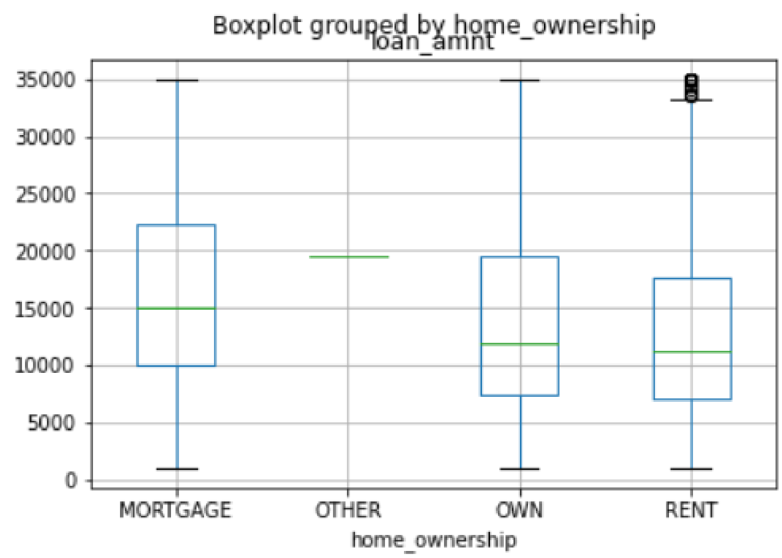
- ANOVA shows p value = 3.5×10^{-38}
- Since
    - H0: No difference between Loan Amount and Grade
    - H1: There is a difference between Loan Amount and Grade
- At a 95% confidence level, the p value shows that there IS A DIFFERENCE between Loan Amount and Grade.

(D) A BANKER PROPOSES THAT THERE IS NO STATISTICAL DIFFERENCE BETWEEN BANK LOAN GRADE A,B,C AND HOME_OWNERSHIP (MORTAGE, OWN, RENT).

PLEASE PERFORM THE APPROPRIATE STATISTICAL TESTS AND INTERPRET THE RESULTS TO VALIDATE HIS ASSUMPTION.

```
df7000.boxplot('loan_amnt','home_ownership')
```

```
<AxesSubplot:title={'center':'loan_amnt'}, xlabel='home_ownership'>
```




```
model_1 = ols('loan_amnt ~ home_ownership', df7000).fit()
```

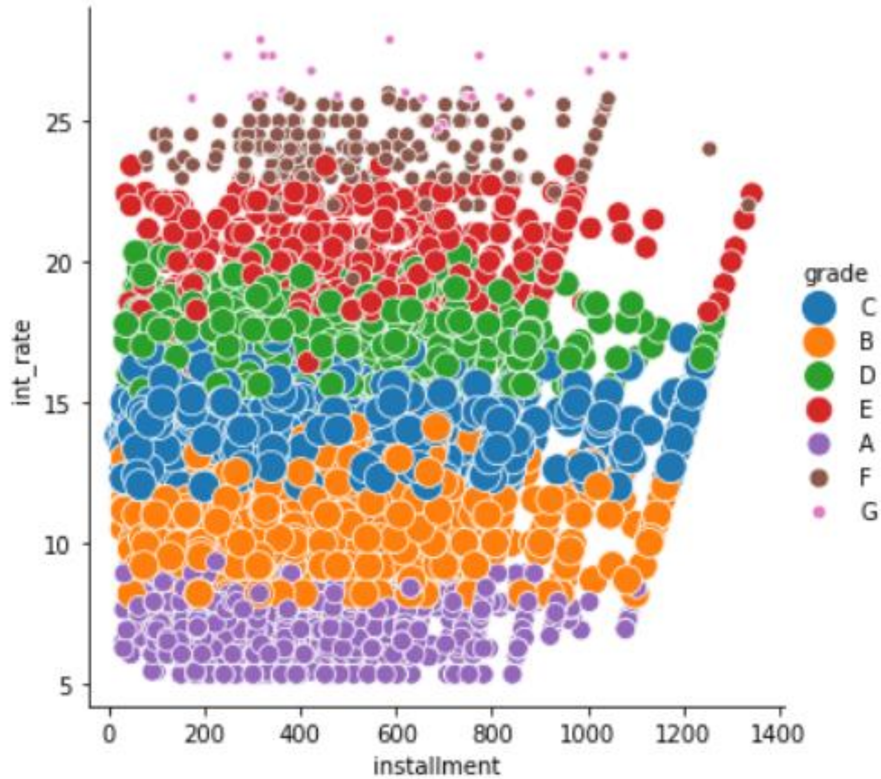
```
anova_lm(model_1)
```

	df	sum_sq	mean_sq	F	PR(>F)
home_ownership	3.0	1.850115e+10	6.167049e+09	90.866418	1.104076e-57
Residual	6996.0	4.748143e+11	6.786939e+07	NaN	NaN

- ANOVA shows p value = 1.1 $x10^{-57}$
- Since
  - H0: No difference between Loan Amount and Home Ownership
  - H1: There is a difference between Loan Amount and Home Ownership
- At a 95% confidence level, the p value shows that there IS A DIFFERENCE between Loan Amount and Home Ownership.

(E) PLOT A SCATTERPLOT OF INTEREST RATE VS INSTALMENT, COLOR BY GRADE, AND WRITE DOWN ANY OBSERVATIONS THAT YOU NOTICE.

```
sb.relplot(x="installment", y="int_rate",
           hue="grade", size="grade", sizes=(20,200), data=df7000);
```

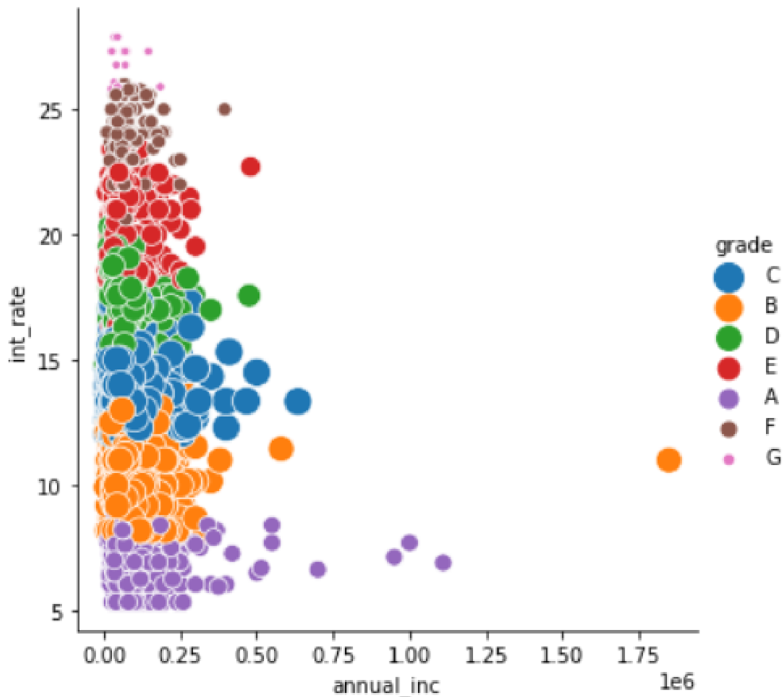


Observations:

- Grade A has lowest interest rate (between 5% to 10%),
- 2nd lowest interest rate is Grade B (around 10%)
- Grade C has the middle interest rate of around 15%
- Grade D has an interest rate of around 17 to 18 %
- Grade E has an interest rate of around 23%
- Grade F has the 2nd highest interest rate of around 25%
- Grade G has the highest interest rate of above 25%
- Grades are irrespective of installment amount, but stops at around the 1000 to 1400 range
- All grades are spread out evenly across the installment amounts (between 0 to around 1000)

(F) PLOT A SCATTERPLOT OF INTEREST RATE VS ANNUAL INCOME, COLOR BY GRADE, AND WRITE DOWN ANY OBSERVATIONS THAT YOU NOTICE

```
sb.relplot(x="annual_inc", y="int_rate",
           hue="grade", size="grade", sizes=(20,200), data=df7000);
```

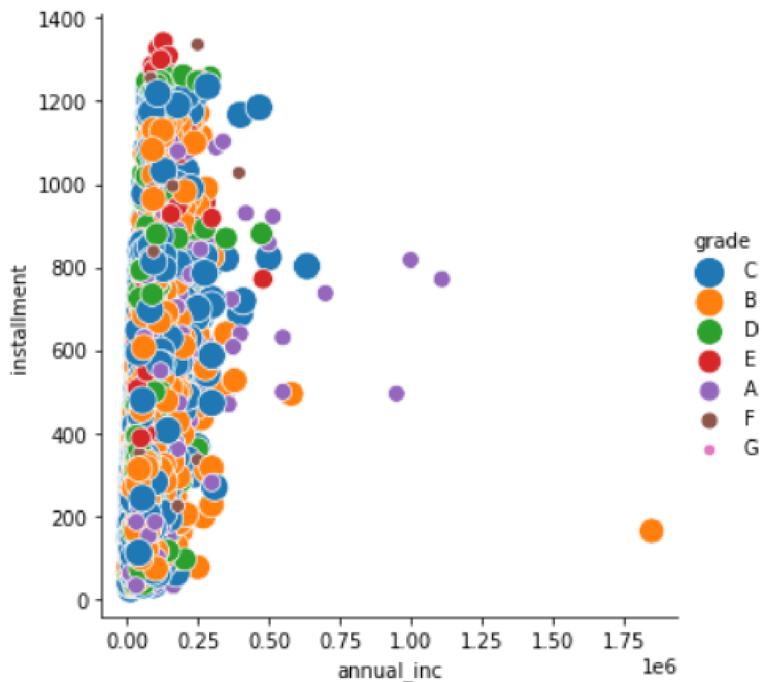


Observations:

- There is an outlier of extreme high annual income of Grade B (above 1.75E6)
- The various grades reflect the various interest rates with respective levels as described in the previous question, Part I(E) (e.g. Grade F has lowest interest rate while Grade G has highest)
- All grades tend to stop around the 0.25 to 05 (x 10^6) annual income level, with only a few Grade A's that manage to escape and reach the height of 1E6 annual income.

(G) PLOT A SCATTERPLOT OF INSTALLMENT VS ANNUAL INCOME, COLOR BY GRADE, AND WRITE DOWN ANY OBSERVATIONS THAT YOU NOTICE

```
sb.relplot(x="annual_inc", y="installment",
           hue="grade", size="grade", sizes=(20,200), data=df7000);
```



Observations:

- Once again, there's an extreme outlier of Grade B of extreme annual income (above 1.75E6).
- Grades are scattered (randomly) across the installment amounts. It appears there's no relationship between installment and grades.
- Most grades occur between the annual income of 0 and 0.25.

(H) CREATE A CORRELATION MATRIX (CORRELATION PLOT) WITH THE FOLLOWING VARIABLES:

- LOAN_AMNT,
- FUNDED_AMNT,
- FUNDED_AMNT_INV,
- INT_RATE,
- INSTALLMENT,
- ANNUAL_INC,
- DTI,
- REVOL_BAL,
- TOTAL_ACC,
- TOTAL_PYMNT,
- TOTAL_PYMNT_INV,
- TOTAL_REC_PRNCP,
- TOTAL_REC_INT,
- TOTAL_REC_LATE_FEE,
- RECOVERIES,
- COLLECTION_RECOVERY_FEE

Please remove missing values (if necessary) and write down any observations that you notice.

Observation: Many zeros in the last three columns

```
df7000_sample = df7000[['loan_amnt','funded_amnt','funded_amnt_inv','int_rate','installment','annual_inc','dti','rev
df7000_sample
```

many 0 values in the last three columns

ient	annual_inc	dti	revol_bal	total_acc	total_pymnt	total_pymnt_inv	total_rec_prncp	total_rec_int	total_rec_late_fee	recoveries	collection_recovery_fee
4.71	71000.0	4.51	7957.0	10.0	9810.345778	9810.35	8000.00	1810.35	0.0	0.0	0.0
2.15	114000.0	10.32	7963.0	36.0	348.260000	348.26	262.77	85.49	0.0	0.0	0.0
0.83	80000.0	16.74	4431.0	24.0	1072.570000	1072.57	645.50	427.07	0.0	0.0	0.0
4.46	32000.0	22.05	6591.0	29.0	3760.533786	3760.53	3000.00	760.53	0.0	0.0	0.0
7.82	47000.0	5.67	3849.0	29.0	0.000000	0.00	0.00	0.00	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
2.62	44000.0	18.99	3582.0	44.0	4370.960000	4370.96	2913.90	1457.06	0.0	0.0	0.0
9.22	56000.0	25.03	29544.0	19.0	2923.350000	2923.35	1476.81	1446.54	0.0	0.0	0.0
5.41	60000.0	24.28	17291.0	35.0	20170.641155	20170.64	18000.00	2170.64	0.0	0.0	0.0
7.80	95000.0	37.33	18381.0	28.0	761.870000	761.87	268.32	493.55	0.0	0.0	0.0
9.56	120000.0	9.63	211419.0	20.0	3291.120000	3291.12	2771.57	519.55	0.0	0.0	0.0

We output the sample to csv….and check for any NaNs…

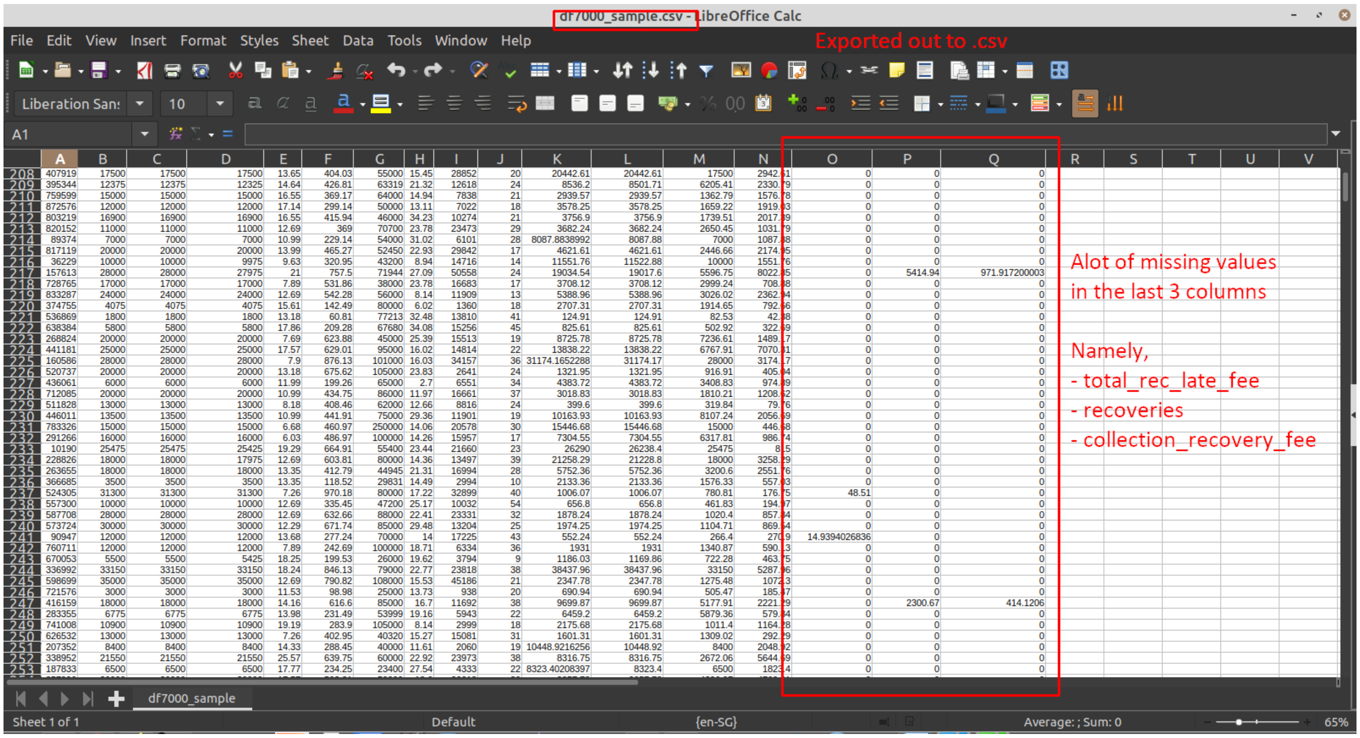
```
df7000_sample.to_csv('df7000_sample.csv')
```

```
df7000_sample.isna().any()
```

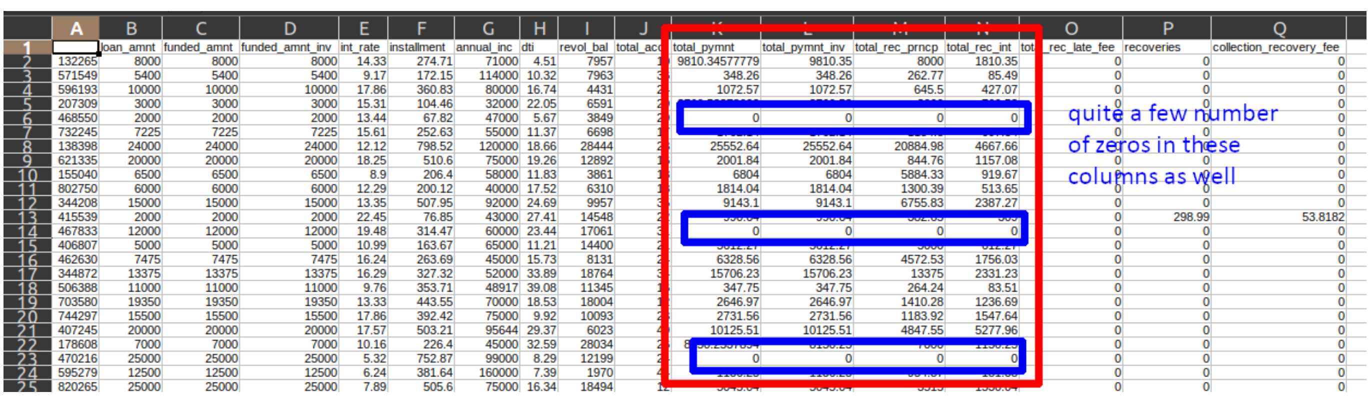
```
loan_amnt                  False
funded_amnt                False
funded_amnt_inv            False
int_rate                   False
installment                False
annual_inc                 False
dti                        False
revol_bal                  False
total_acc                  False
total_pymnt                False
total_pymnt_inv            False
total_rec_prncp            False
total_rec_int              False
total_rec_late_fee         False
recoveries                 False
collection_recovery_fee    False
dtype: bool
```

Apparently, there are no NaNs….

Observation: Many zeros in the last three columns



Observation: Some zeros remain hidden but recurring in these few columns…



We replace all zeros with NaNs…

```
df7000_cleansed = df7000_sample.replace(0,np.NaN)
df7000_cleansed
```

ient	annual_inc	dti	revol_bal	total_acc	total_pymnt	total_pymnt_inv	total_rec_prncp	total_rec_int	total_rec_late_fee	recoveries	collection_recovery_fee
4.71	71000.0	4.51	7957.0	10.0	9810.345778	9810.35	8000.00	1810.35	NaN	NaN	NaN
2.15	114000.0	10.32	7963.0	36.0	348.260000	348.26	262.77	85.49	NaN	NaN	NaN
0.83	80000.0	16.74	4431.0	24.0	1072.570000	1072.57	645.50	427.07	NaN	NaN	NaN
4.46	32000.0	22.05	6591.0	29.0	3760.533786	3760.53	3000.00	760.53	NaN	NaN	NaN
7.82	47000.0	5.67	3849.0	29.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
2.62	44000.0	18.99	3582.0	44.0	4370.960000	4370.96	2913.90	1457.06	NaN	NaN	NaN
9.22	56000.0	25.03	29544.0	19.0	2923.350000	2923.35	1476.81	1446.54	NaN	NaN	NaN
5.41	60000.0	24.28	17291.0	35.0	20170.641155	20170.64	18000.00	2170.64	NaN	NaN	NaN
7.80	95000.0	37.33	18381.0	28.0	761.870000	761.87	268.32	493.55	NaN	NaN	NaN
9.56	120000.0	9.63	211419.0	20.0	3291.120000	3291.12	2771.57	519.55	NaN	NaN	NaN

We drop all rows with NaNs….

```
df7000_cleansed_dropna = df7000_cleansed.dropna()
df7000_cleansed_dropna
```

	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	dti	revol_bal	total_acc	total_pymnt	total_pymnt_inv	total_rec_prncp
21255	10000.0	10000.0	9975.0	15.57	240.91	72000.0	15.45	16533.0	21.0	3167.86	3159.95	1286.45
51499	20500.0	20500.0	20500.0	24.50	595.71	86300.0	30.04	28344.0	28.0	12217.79	12217.79	3068.29
377203	2000.0	2000.0	2000.0	23.43	77.87	50000.0	6.60	2349.0	6.0	965.48	965.48	330.11
58723	20000.0	20000.0	20000.0	19.20	521.02	100000.0	31.67	19264.0	31.0	10903.63	10903.63	3391.73
430564	1600.0	1600.0	1600.0	18.92	58.59	20000.0	15.79	10264.0	19.0	735.31	735.31	327.41
284529	5000.0	5000.0	5000.0	14.99	173.31	30000.0	4.68	3134.0	7.0	1041.21	1041.21	173.70
137063	16950.0	16950.0	16950.0	17.77	428.31	60000.0	28.02	32741.0	30.0	7030.35	7030.35	2099.67
83449	23675.0	23675.0	23675.0	25.57	702.83	100000.0	32.32	24277.0	45.0	8816.29	8816.29	1674.61
222813	25000.0	25000.0	24975.0	10.74	815.40	75000.0	18.97	25935.0	30.0	25738.98	25713.34	19993.37
156090	9800.0	9800.0	9800.0	11.14	321.49	100000.0	9.79	9595.0	26.0	6669.99	6669.99	5040.09
200284	13250.0	13250.0	13250.0	17.77	477.50	45000.0	21.52	2087.0	22.0	4019.60	4019.60	1464.15
366892	25375.0	25375.0	25375.0	15.61	611.83	53000.0	31.09	26218.0	40.0	9312.38	9312.38	2353.77
155928	14750.0	14750.0	14750.0	14.33	506.49	55000.0	14.68	23037.0	18.0	7578.51	7578.51	4230.78
125172	22800.0	22800.0	22800.0	23.76	652.74	95000.0	19.15	14382.0	48.0	12061.47	12061.47	3204.92

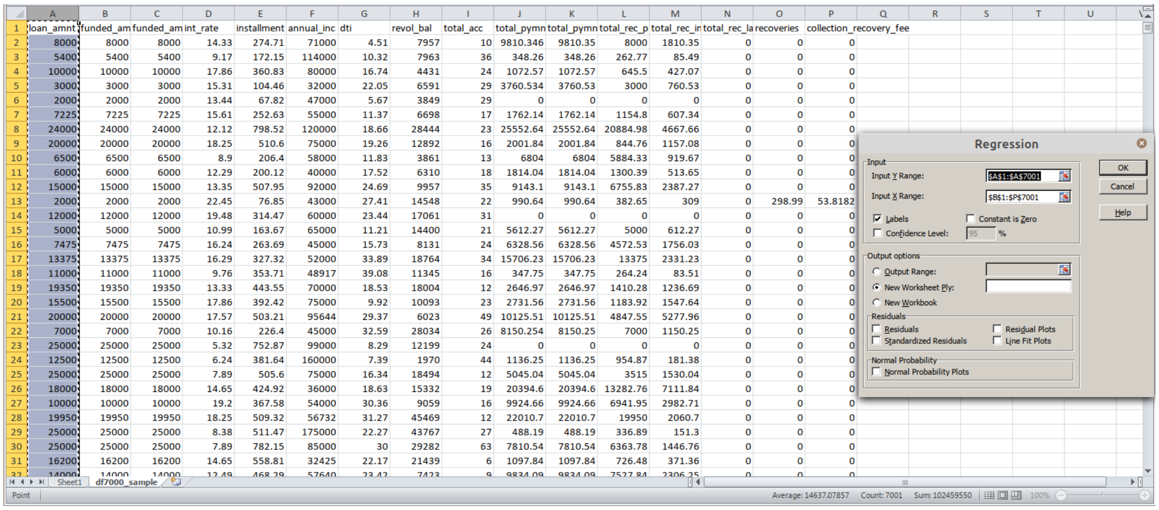
Finally, we obtain the cleansed Correlation table….

```
df7000_corr = df7000_cleansed_dropna.corr()
df7000_corr
```

	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	dti	revol_bal	total_acc	total_pymnt	total_pym
loan_amnt	1.000000	0.999999	0.999985	-0.061145	0.998904	0.764515	-0.530270	0.348262	0.094901	0.771711	0.7
funded_amnt	0.999999	1.000000	0.999988	-0.061596	0.998861	0.764869	-0.529727	0.349003	0.095610	0.771107	0.7
funded_amnt_inv	0.999985	0.999988	1.000000	-0.061446	0.998703	0.764890	-0.526681	0.350447	0.098260	0.769737	0.7
int_rate	-0.061145	-0.061596	-0.061446	1.000000	-0.054312	-0.551307	0.177736	-0.607457	-0.548164	0.251453	0.2
installment	0.998904	0.998861	0.998703	-0.054312	1.000000	0.758484	-0.553290	0.331655	0.071630	0.798131	0.8
annual_inc	0.764515	0.764869	0.764890	-0.551307	0.758484	1.000000	-0.603222	0.750859	0.331123	0.422593	0.4
dti	-0.530270	-0.529727	-0.526681	0.177736	-0.553290	-0.603222	1.000000	-0.146383	0.363969	-0.629156	-0.6
revol_bal	0.348262	0.349003	0.350447	-0.607457	0.331655	0.750859	-0.146383	1.000000	0.413004	-0.033934	-0.0
total_acc	0.094901	0.095610	0.098260	-0.548164	0.071630	0.331123	0.363969	0.413004	1.000000	-0.253638	-0.2
total_pymnt	0.771711	0.771107	0.769737	0.251453	0.798131	0.422593	-0.629156	-0.033934	-0.253638	1.000000	0.9
total_pymnt_inv	0.774318	0.773717	0.772372	0.250898	0.800550	0.424948	-0.627645	-0.031251	-0.250581	0.999983	1.0
total_rec_prncp	0.761142	0.760556	0.759133	0.203135	0.788806	0.440739	-0.632405	-0.004923	-0.223570	0.997409	0.9
total_rec_int	0.816820	0.816326	0.815459	0.401820	0.834652	0.391596	-0.539709	-0.054305	-0.270567	0.964447	0.9
total_rec_late_fee	-0.517890	-0.517694	-0.520908	0.002456	-0.510066	-0.396261	0.079187	-0.291422	-0.281572	-0.384731	-0.3

(I) CREATE A REGRESSION MODEL WITH THE ABOVE MENTIONED VARIABLES TO PREDICT LOAN_AMT USING THE OTHER VARIABLES.

INTERPRET THE REGRESSION RESULTS AND WRITE DOWN ANY OBSERVATIONS THAT YOU NOTICE.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	loan_amnt	funded_am	funded_am	int_rate	installment	annual_inc	dti	revol_bal	total_acc	total_pymn	total_pymn	total_rec_p	total_rec_in	total_rec_la	recoveries	collection_recovery_fee	
2	8000	8000	8000	14.33	274.71	71000	4.51	7957	10	9810.346	9810.35	8000	1810.35	0	0	0	
3	5400	5400	5400	9.17	172.15	114000	10.32	7963	36	348.26	348.26	262.77	85.49	0	0	0	
4	10000	10000	10000	17.86	360.83	80000	16.74	4431	24	1072.57	1072.57	645.5	427.07	0	0	0	
5	3000	3000	3000	15.31	104.46	32000	22.05	6591	29	3760.534	3760.53	3000	760.53	0	0	0	
6	2000	2000	2000	13.44	67.82	47000	5.67	3849	29	0	0	0	0	0	0	0	
7	7225	7225	7225	15.61	252.63	55000	11.37	6698	17	1762.14	1762.14	1154.8	607.34	0	0	0	
8	24000	24000	24000	12.12	798.52	120000	18.66	28444	23	25552.64	25552.64	20884.98	4667.66	0	0	0	
9	20000	20000	20000	18.25	510.6	75000	19.26	12892	16	2001.84	2001.84	844.76	1157.08	0	0	0	
10	6500	6500	6500	8.9	206.4	58000	11.83	3861	13	6804	6804	5884.33	919.67	0	0	0	
11	6000	6000	6000	12.29	200.12	40000	17.52	6310	18	1814.04	1814.04	1300.39	513.65	0	0	0	
12	15000	15000	15000	13.35	507.95	92000	24.69	9957	35	9143.1	9143.1	6755.83	2387.27	0	0	0	
13	2000	2000	2000	22.45	76.85	43000	27.41	14548	22	990.64	990.64	382.65	309	0	298.99	53.8182	
14	12000	12000	12000	19.48	314.47	60000	23.44	17061	31	0	0	0	0	0	0	0	
15	5000	5000	5000	10.99	163.67	65000	11.21	14400	21	5612.27	5612.27	5000	612.27	0	0	0	
16	7475	7475	7475	16.24	263.69	45000	15.73	8131	24	6328.56	6328.56	4572.53	1756.03	0	0	0	
17	13375	13375	13375	16.29	327.32	52000	33.89	18764	34	15706.23	15706.23	13375	2331.23	0	0	0	
18	11000	11000	11000	9.76	353.71	48917	39.08	11345	16	347.75	347.75	264.24	83.51	0	0	0	
19	19350	19350	19350	13.33	443.55	70000	18.53	18004	12	2646.97	2646.97	1410.28	1236.69	0	0	0	
20	15500	15500	15500	17.86	392.42	75000	9.92	10093	23	2731.56	2731.56	1183.92	1547.64	0	0	0	
21	20000	20000	20000	17.57	503.21	95644	29.37	6023	49	10125.51	10125.51	4847.55	5277.96	0	0	0	
22	7000	7000	7000	10.16	226.4	45000	32.59	28034	26	8150.254	8150.25	7000	1150.25	0	0	0	
23	25000	25000	25000	5.32	752.87	99000	8.29	12199	24	0	0	0	0	0	0	0	
24	12500	12500	12500	6.24	381.64	160000	7.39	1970	44	1136.25	1136.25	954.87	181.38	0	0	0	
25	25000	25000	25000	7.89	505.6	75000	16.34	18494	12	5045.04	5045.04	3515	1530.04	0	0	0	
26	18000	18000	18000	14.65	424.92	36000	18.63	15332	19	20394.6	20394.6	13282.76	7111.84	0	0	0	
27	10000	10000	10000	19.2	367.58	54000	30.36	9059	16	9924.66	9924.66	6941.95	2982.71	0	0	0	
28	19950	19950	19950	18.25	509.32	56732	31.27	45469	12	22010.7	22010.7	19950	2060.7	0	0	0	
29	25000	25000	25000	8.38	511.47	175000	22.27	43767	27	488.19	488.19	336.89	151.3	0	0	0	
30	25000	25000	25000	7.89	782.15	85000	30	29282	63	7810.54	7810.54	6363.78	1446.76	0	0	0	
31	16200	16200	16200	14.65	558.81	32425	22.17	21439	6	1097.84	1097.84	726.48	371.36	0	0	0	

- Multiple Linear Regression analysis using Excel (since its 7000 rows and can be handled by Excel)

Regression Statistics	
Multiple R	0.999463939
R Square	0.998928165
Adjusted R Square	0.998925863
Standard Error	275.153005
Observations	7000

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	15	4.92787[illegible]11	32852444891	433929.4991	0
Residual	6984	5287[illegible]886.3	75709.17616		
Total	6999	4[illegible]3315E+11			

	Coefficients	*tandard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	31.0628896	15.72892714	1.97488928	0.048320037	0.2294153	61.8963639	0.2294153	61.8963639
funded_amnt	1.065175158	0.010407478	102.3471011	0	1.044773342	1.085576975	1.044773342	1.085576975
funded_amnt_inv	-0.062418992	0.010290978	-6.065408943	1.38488E-09	-0.082592436	-0.042245549	-0.082592436	-0.042245549
int_rate	-1.541093997	0.895600616	-1.720737983	0.085342673	-3.29674321	0.214555217	-3.29674321	0.214555217
installment	-0.141618658	0.044799628	-3.161157023	0.001578157	-0.229439535	-0.053797781	-0.229439535	-0.053797781
annual_inc	-7.27936E-05	7.21342E-05	-1.009141042	0.312941958	-0.000214199	6.86114E-05	-0.000214199	6.86114E-05
dti	-0.207471241	0.442674821	-0.468676399	0.639315587	-1.075248337	0.660305854	-1.075248337	0.660305854
revol_bal	2.62233E-05	9.44908E-05	0.277521746	0.781387725	-0.000159007	0.000211454	-0.000159007	0.000211454
total_acc	0.223379695	0.302770778	0.737784858	0.460669984	-0.370142985	0.816902375	-0.370142985	0.816902375
total_pymnt	5606.610206	3529.23984	1.588616943	0.112192166	-1311.771762	12524.99217	-1311.771762	12524.99217
total_pymnt_inv	0.05125343	0.011086344	4.623113686	3.84819E-06	0.029520828	0.072986032	0.029520828	0.072986032
total_rec_prncp	-5606.659948	3529.239755	-1.588631076	0.112188974	-12525.04175	1311.721853	-12525.04175	1311.721853
total_rec_int	-5606.65426	3529.239777	-1.588629454	0.11218934	-12525.0361	1311.727584	-12525.0361	1311.727584
total_rec_late_fee	-5607.33203	3529.221737	-1.588829619	0.112144129	-12525.67851	1311.014452	-12525.67851	1311.014452
recoveries	-5606.651252	3529.239742	-1.588628618	0.112189529	-12525.03303	1311.730524	-12525.03303	1311.730524
collection_recovery_fe	0.078838904	0.066730718	1.18144845	0.237464849	-0.051973571	0.209651379	-0.051973571	0.209651379

Sheet1 df7000_sample

The Multiple Regression Model is"

- Loan Amount = 31.06 + (1.06*funded_amnt) – (0.06*funded_amnt_invt) – (1.54*int_rate)…..
- R2 and Adjusted R2 values are 99.9% fitting, which means that the MR fit is perfect.
- The Significance F (Global Test P Value) is 0 (<5% alpha).
- Thus, we accept H1 that the equation is important and at least one of the variables is significant.

USING THE REGRESSION RESULTS, PERFORM FEATURE SELECTION ON THE DATASET AND SELECT THE USEFUL VARIABLES

Variables	P-Value	alpha
funded_amnt	0	
funded_amnt_inv	1.38E-09	< 5%
int_rate	0.085343	
installment	0.001578	< 5%
annual_inc	0.312942	
dti	0.639316	
revol_bal	0.781388	
total_acc	0.46067	
total_pymnt	0.112192	
total_pymnt_inv	3.85E-06	< 5%
total_rec_prncp	0.112189	
total_rec_int	0.112189	
total_rec_late_fee	0.112144	
recoveries	0.11219	
collection_recovery_fee	0.237465	
Alpha = 5%		

- The only important variables are:
    - Funded_amnt_inv
    - Installment
    - Total_pymnt_inv

SUBSET THE DATASET TO ONLY INCLUDE THESE SELECTED USEFUL VARIABLES.

```
df7000_useful = df7000[['funded_amnt_inv','installment', 'total_pymnt_inv']]
df7000_useful
```

	funded_amnt_inv	installment	total_pymnt_inv
132265	8000.0	274.71	9810.35
571549	5400.0	172.15	348.26
596193	10000.0	360.83	1072.57
207309	3000.0	104.46	3760.53
468550	2000.0	67.82	0.00
...	...	...	...
270934	15000.0	312.62	4370.96
700482	20600.0	479.22	2923.35
66692	18000.0	606.41	20170.64
494175	29725.0	837.80	761.87
705024	18000.0	549.56	3291.12

7000 rows × 3 columns

(J) PERFORM A RANDOM FOREST AND USE THE SUBSET TO PREDICT LOAN_AMT USING THE OTHER VARIABLES

SUBSET THE DATASET INTO TRAINING (70%) AND TESTING (30%)

INTERPRET THE REGRESSION RESULTS USING THE APPROPRIATE METRICS AND PLOTS.

Random Forest

In [9]:
```
from sklearn.model_selection import train_test_split

X=df7000_sample[['funded_amnt','funded_amnt_inv','int_rate','installment','annual_inc','dti',
                 'revol_bal','total_acc','total_pymnt','total_pymnt_inv','total_rec_prncp','total_rec_int',
                 'total_rec_late_fee','recoveries','collection_recovery_fee']]

y=df7000_sample['loan_amnt']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [14]:
```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
```


In [14]:
```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
```


In [15]:
```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.7323809523809524

Current Accuracy at 73%
(using df7000_sample)

PERFORM A VARIABLE IMPORTANCE PLOT.

```
## Finding Important Features
```

```
In [20]: clf.feature_importances_
```

```
Out[20]: array([0.26195562, 0.24052854, 0.04588335, 0.10833669, 0.04210399,
                0.03816555, 0.04134246, 0.03512872, 0.04101208, 0.04142776,
                0.05661056, 0.0414628 , 0.00131821, 0.00244981, 0.00227387])
```

```
In [32]: dataindex = pd.read_csv('/home/dralvin/Desktop/PLANO/Plano-Data Scientist assessment/df7000_sample_index.csv',
                                 header = None)
         dataindex
```

Out[32]:

	0
0	funded_amnt
1	funded_amnt_inv
2	int_rate
3	installment
4	annual_inc
5	dti

```
In [33]: import pandas as pd
         feature_imp = pd.Series(clf.feature_importances_,index=dataindex).sort_values(ascending=False)
         feature_imp
```

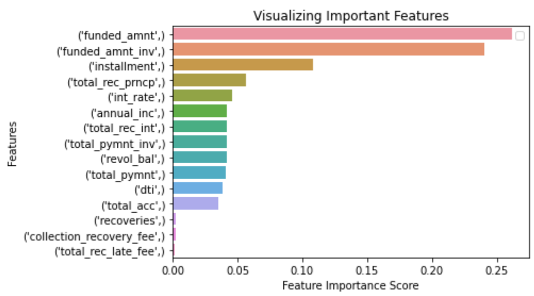
```
Out[33]: (funded_amnt,)                0.261956
         (funded_amnt_inv,)            0.240529
         (installment,)                0.108337
         (total_rec_prncp,)            0.056611
         (int_rate,)                   0.045883
         (annual_inc,)                 0.042104
         (total_rec_int,)              0.041463
         (total_pymnt_inv,)            0.041428
         (revol_bal,)                  0.041342
         (total_pymnt,)                0.041012
         (dti,)                        0.038166
         (total_acc,)                  0.035129
         (recoveries,)                 0.002450
         (collection_recovery_fee,)    0.002274
         (total_rec_late_fee,)         0.001318
         dtype: float64
```

```
In [42]: import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)

# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored w
hen legend() is called with no argument.
```



The lowest 3 features of importance are:

- (recoveries,) 0.002450
- (collection_recovery_fee,) 0.002274
- (total_rec_late_fee,) 0.001318

Thus, we drop them and re-run the accuracy test…

```
In [43]: # Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into features and labels
X=df7000_sample[['funded_amnt','funded_amnt_inv','int_rate','installment','annual_inc',
                 'dti','revol_bal','total_acc','total_pymnt','total_pymnt_inv','total_rec_prncp','total_rec_int']]

y=df7000_sample['loan_amnt']

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.70, random_state=5) # 70% training and 30% test
```

WRITE DOWN ANY OBSERVATIONS THAT YOU NOTICE FROM THE RESULTS.

```
In [44]: from sklearn.ensemble import RandomForestClassifier

         #Create a Gaussian Classifier
         clf=RandomForestClassifier(n_estimators=100)

         #Train the model using the training sets y_pred=clf.predict(X_test)
         clf.fit(X_train,y_train)

         # prediction on test set
         y_pred=clf.predict(X_test)

         #Import scikit-learn metrics module for accuracy calculation
         from sklearn import metrics
         # Model Accuracy, how often is the classifier correct?
         print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

         Accuracy: 0.7155102040816327
```

Dropped to 71.55% accuracy after dropping the non-essential features

Strangely, even after dropping off the non-essential features, the accuracy dipped slightly to 71.55%.

This means that we shouldn't drop off any more features but leave it as is.

PART 2

You are given a large dataset LendingClubLoan.csv, the predictor column is loan_status.

You are required to create a model that predicts if a new customer will default on his loan ("Charged Off") or will pay up fully ("Fully Paid"). The bank will prioritize customers that can fully service their loan. This type of customer analytics enables bank to identify customers who can pay up their loans.

You are required to do the following:

1. Preprocess (data wrangling) the dataset to improve the quality of the dataset

2. Conduct a feature selection

3. Split the data into training set (75%) and testing set (25%)

4. Create a model to predict and classify the customers as described above.

Please aim to achieve at least a 70% classification accuracy, as well as clearly label your steps and stages.

STEP 1: DATA WRANGLING

IMPORTING ALL LIBRARIES AND READING DATAFRAME

Importing and Viewing

```
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

# Pandas options
pd.set_option('display.max_colwidth', 1000, 'display.max_rows', None, 'display.max_columns', None)

# Plotting options
%matplotlib inline
mpl.style.use('ggplot')
sns.set(style='whitegrid')
```

```
loans = pd.read_csv('/home/dralvin/Desktop/PLANO/Plano-Data Scientist assessment/From Dwight/LendingClubLoan.csv')
```

```
/home/dralvin/.local/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3444: DtypeWarning: Columns (19,5
5) have mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Checking the Dataframe

```
loans.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 887379 entries, 0 to 887378
Data columns (total 74 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   id                           887379 non-null  int64
 1   member_id                    887379 non-null  int64
 2   loan_amnt                    887379 non-null  float64
 3   funded_amnt                  887379 non-null  float64
 4   funded_amnt_inv              887379 non-null  float64
 5   term                         887379 non-null  object
 6   int_rate                     887379 non-null  float64
 7   installment                  887379 non-null  float64
 8   grade                        887379 non-null  object
 9   sub_grade                    887379 non-null  object
 10  emp_title                    835917 non-null  object
 11  emp_length                   842554 non-null  object
 12  home_ownership               887379 non-null  object
 13  annual_inc                   887375 non-null  float64
 14  verification_status          887379 non-null  object
 15  issue_d                      887379 non-null  object
 16  loan_status                  887379 non-null  object
 17  pymnt_plan                   887379 non-null  object
 18  url                          887379 non-null  object
 19  desc                         126028 non-null  object
 20  purpose                      887379 non-null  object
 21  title                        887227 non-null  object
 22  zip_code                     887379 non-null  object
 23  addr_state                   887379 non-null  object
 24  dti                          887379 non-null  float64
 25  delinq_2yrs                  887350 non-null  float64
 26  earliest_cr_line             887350 non-null  object
 27  inq_last_6mths               887350 non-null  float64
 28  mths_since_last_delinq       433067 non-null  float64
 29  mths_since_last_record       137053 non-null  float64
 30  open_acc                     887350 non-null  float64
 31  pub_rec                      887350 non-null  float64
 32  revol_bal                    887379 non-null  float64
 33  revol_util                   886877 non-null  float64
 34  total_acc                    887350 non-null  float64
 35  initial_list_status          887379 non-null  object
 36  out_prncp                    887379 non-null  float64
 37  out_prncp_inv                887379 non-null  float64
 38  total_pymnt                  887379 non-null  float64
 39  total_pymnt_inv              887379 non-null  float64
 40  total_rec_prncp              887379 non-null  float64
 41  total_rec_int                887379 non-null  float64
 42  total_rec_late_fee           887379 non-null  float64
 43  recoveries                   887379 non-null  float64
 44  collection_recovery_fee      887379 non-null  float64
 45  last_pymnt_d                 869720 non-null  object
 46  last_pymnt_amnt              887379 non-null  float64
 47  next_pymnt_d                 634408 non-null  object
 48  last_credit_pull_d           887326 non-null  object
 49  collections_12_mths_ex_med   887234 non-null  float64
 50  mths_since_last_major_derog  221703 non-null  float64
 51  policy_code                  887379 non-null  float64
 52  application_type             887379 non-null  object
 53  annual_inc_joint             511 non-null     float64
 54  dti_joint                    509 non-null     float64
 55  verification_status_joint    511 non-null     object
 56  acc_now_delinq               887350 non-null  float64
 57  tot_coll_amt                 817103 non-null  float64
 58  tot_cur_bal                  817103 non-null  float64
 59  open_acc_6m                  21372 non-null   float64
 60  open_il_6m                   21372 non-null   float64
 61  open_il_12m                  21372 non-null   float64
 62  open_il_24m                  21372 non-null   float64
 63  mths_since_rcnt_il           20810 non-null   float64
 64  total_bal_il                 21372 non-null   float64
 65  il_util                      18617 non-null   float64
 66  open_rv_12m                  21372 non-null   float64
 67  open_rv_24m                  21372 non-null   float64
 68  max_bal_bc                   21372 non-null   float64
 69  all_util                     21372 non-null   float64
 70  total_rev_hi_lim             817103 non-null  float64
 71  inq_fi                       21372 non-null   float64
 72  total_cu_tl                  21372 non-null   float64
 73  inq_last_12m                 21372 non-null   float64
dtypes: float64(49), int64(2), object(23)
memory usage: 501.0+ MB
```

- Total 74 columns and 88,7378 rows

Glancing at 3 Sample Rows of DAta

```
loans.sample(3)
```

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_owne
323312	24785264	27228246	16850.0	16850.0	16850.0	60 months	12.99	383.31	C	C1	Curriculum Assistant	3 years	
375081	18014033	20166738	19250.0	19250.0	19100.0	36 months	9.17	613.67	B	B1	Project Engineer	10+ years	F
821251	42484425	45451191	19000.0	19000.0	19000.0	36 months	6.68	583.89	A	A3	Financial Advisor	7 years	MORTG

MISSING FRACTIONS

```
missing_fractions = loans.isnull().mean().sort_values(ascending=False)
```

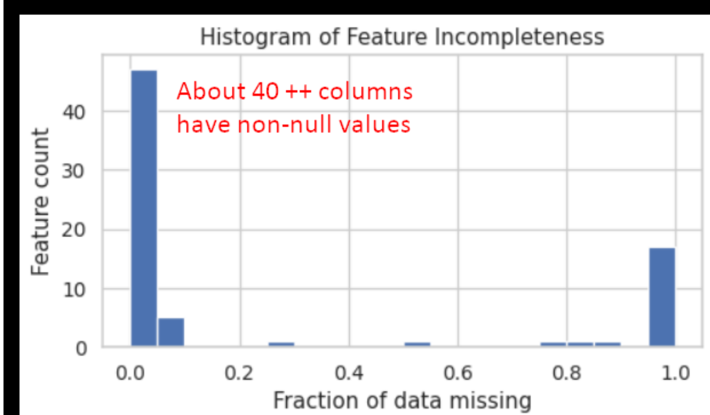
```
missing_fractions.head(50)
```

```
dti_joint                         0.999426
annual_inc_joint                  0.999424
verification_status_joint         0.999424
il_util                           0.979020
mths_since_rcnt_il                0.976549
open_acc_6m                       0.975916
open_il_6m                        0.975916
open_il_12m                       0.975916
open_il_24m                       0.975916
total_bal_il                      0.975916
inq_last_12m                      0.975916
open_rv_12m                       0.975916
open_rv_24m                       0.975916
max_bal_bc                        0.975916
all_util                          0.975916
inq_fi                            0.975916
total_cu_tl                       0.975916
desc                              0.857977
mths_since_last_record            0.845553
```

- We see many columns having very high null value rates.
- Example dti_joint = 0.999 means 99.9% of the column is filled with null values.

```
plt.figure(figsize=(6,3), dpi=90)
missing_fractions.plot.hist(bins=20)
plt.title('Histogram of Feature Incompleteness')
plt.xlabel('Fraction of data missing')
plt.ylabel('Feature count')
```

```
Text(0, 0.5, 'Feature count')
```



about 19 columns have
almost 100% null values

Drop features with more than 30% of their data missing.

```
In [8]: drop_list = sorted(list(missing_fractions[missing_fractions > 0.3].index))
        print(drop_list)

        ['all_util', 'annual_inc_joint', 'desc', 'dti_joint', 'il_util', 'inq_fi
        ', 'inq_last_12m', 'max_bal_bc', 'mths_since_last_delinq', 'mths_since_l
        ast_major_derog', 'mths_since_last_record', 'mths_since_rcnt_il', 'open_
        acc_6m', 'open_il_12m', 'open_il_24m', 'open_il_6m', 'open_rv_12m', 'ope
        n_rv_24m', 'total_bal_il', 'total_cu_tl', 'verification_status_joint']

In [9]: len(drop_list)

Out[9]: 21
```

- We will drop off 21 columns because they have too many NaNs.

```
In [9]: loans.drop(labels=drop_list, axis=1, inplace=True)

In [10]: loans.shape

Out[10]: (887379, 53)

In [11]: print(sorted(loans.columns))

         ['acc_now_delinq', 'addr_state', 'annual_inc', 'application_type', 'coll
         ection_recovery_fee', 'collections_12_mths_ex_med', 'delinq_2yrs', 'dti
         ', 'earliest_cr_line', 'emp_length', 'emp_title', 'funded_amnt', 'funded
         _amnt_inv', 'grade', 'home_ownership', 'id', 'initial_list_status', 'inq
         _last_6mths', 'installment', 'int_rate', 'issue_d', 'last_credit_pull_d
         ', 'last_pymnt_amnt', 'last_pymnt_d', 'loan_amnt', 'loan_status', 'membe
         r_id', 'next_pymnt_d', 'open_acc', 'out_prncp', 'out_prncp_inv', 'policy
         _code', 'pub_rec', 'purpose', 'pymnt_plan', 'recoveries', 'revol_bal', '
         revol_util', 'sub_grade', 'term', 'title', 'tot_coll_amt', 'tot_cur_bal
         ', 'total_acc', 'total_pymnt', 'total_pymnt_inv', 'total_rec_int', 'tota
         l_rec_late_fee', 'total_rec_prncp', 'total_rev_hi_lim', 'url', 'verifica
         tion_status', 'zip_code']
```

- We are left with 53 columns.
- But according to : https://www.kaggle.com/pileatedperch/predicting-charge-off-from-initial-listing-data#8.-Model-Training-and-Testing
- They have already identified which columns to drop, and which to keep (with reference to their financial data dictionary).
- The final columns which we will keep are:
- keep_list = ['addr_state', 'application_type', 'dti', 'earliest_cr_line', 'emp_length', 'home_ownership', 'initial_list_status', 'installment', 'int_rate', 'issue_d', 'loan_amnt', 'loan_status', 'mort_acc', 'open_acc', 'pub_rec', 'pub_rec_bankruptcies', 'purpose', 'revol_util', 'sub_grade', 'term', 'total_acc', 'verification_status']

```
In [15]: keep_list = ['addr_state', 'application_type', 'dti', 'earliest_cr_line',

In [16]: len(keep_list)
Out[16]: 22
```

```
In [18]: len(drop_list)
Out[18]: 33

In [19]: loans.drop(labels=drop_list, axis=1, inplace=True)
```

- We will try keeping 22 columns, and drop off 33 columns.

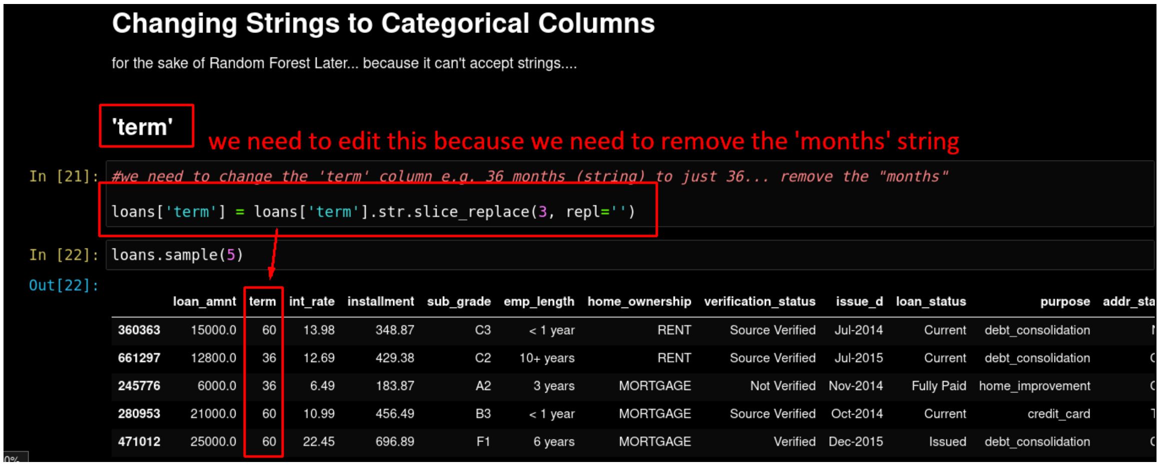
```
In [48]: loans.shape
Out[48]: (887379, 20)

In [49]: print(list(loans.columns))
         ['loan_amnt', 'term', 'int_rate', 'installment', 'sub_grade', 'emp_lengt
         h', 'home_ownership', 'verification_status', 'issue_d', 'loan_status', '
         purpose', 'addr_state', 'dti', 'earliest_cr_line', 'open_acc', 'pub_rec
         ', 'revol_util', 'total_acc', 'initial_list_status', 'application_type']
```

- We will end up with only 20 columns because of the “missing fractions > 30% NaNs” carried out in the previous section.

STEP 2: FEATURE SELECTION

'TERM'



Changing Strings to Categorical Columns

for the sake of Random Forest Later... because it can't accept strings....

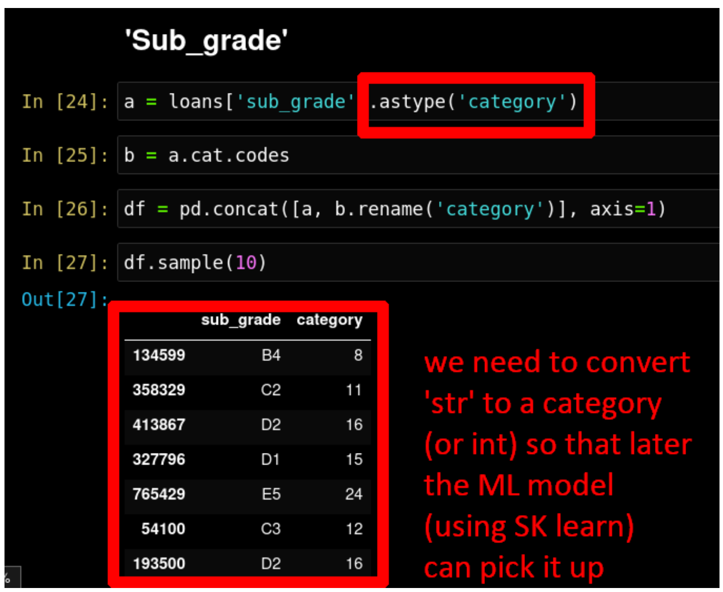
'term' we need to edit this because we need to remove the 'months' string

```
In [21]: #we need to change the 'term' column e.g. 36 months (string) to just 36... remove the "months"
         loans['term'] = loans['term'].str.slice_replace(3, repl='')

In [22]: loans.sample(5)
Out[22]:
```

	loan_amnt	term	int_rate	installment	sub_grade	emp_length	home_ownership	verification_status	issue_d	loan_status	purpose	addr_sta
360363	15000.0	60	13.98	348.87	C3	< 1 year	RENT	Source Verified	Jul-2014	Current	debt_consolidation	N
661297	12800.0	36	12.69	429.38	C2	10+ years	RENT	Source Verified	Jul-2015	Current	debt_consolidation	C
245776	6000.0	36	6.49	183.87	A2	3 years	MORTGAGE	Not Verified	Nov-2014	Fully Paid	home_improvement	C
280953	21000.0	60	10.99	456.49	B3	< 1 year	MORTGAGE	Source Verified	Oct-2014	Current	credit_card	T
471012	25000.0	60	22.45	696.89	F1	6 years	MORTGAGE	Verified	Dec-2015	Issued	debt_consolidation	C

'SUB GRADE'



'Sub_grade'

```
In [24]: a = loans['sub_grade'].astype('category')

In [25]: b = a.cat.codes

In [26]: df = pd.concat([a, b.rename('category')], axis=1)

In [27]: df.sample(10)
Out[27]:
```

	sub_grade	category
134599	B4	8
358329	C2	11
413867	D2	16
327796	D1	15
765429	E5	24
54100	C3	12
193500	D2	16

we need to convert 'str' to a category (or int) so that later the ML model (using SK learn) can pick it up

```
In [28]: loans['sub_grade'] = loans['sub_grade'].astype('category')
         loans['sub_grade'] = loans['sub_grade'].cat.codes

In [29]: loans.sample(5)
```

Out[29]:

	loan_amnt	term	int_rate	installment	sub_grade	mp_length	home_ownership	verification_status	issue_d	loan_status	purpose	addr_stat
589715	7500.0	36	9.17	239.10	6	NaN	MORTGAGE	Verified	Oct-2015	Current	debt_consolidation	I
521930	11600.0	60	12.59	261.51	11	10+ years	RENT	Not Verified	Nov-2015	Current	car	A
429378	15000.0	60	15.31	359.30	13	2 years	RENT	Source Verified	Mar-2014	Current	credit_card	W
45297	2500.0	36	10.99	81.84	6	5 years	RENT	Not Verified	Dec-2013	Current	other	N
675902	3200.0	36	10.99	104.75	8	5 years	MORTGAGE	Not Verified	Jul-2015	Fully Paid	other	P

'EMP_LENGTH'

emp_length

```
In [31]: a = loans['emp_length'].astype('category')
         b = a.cat.codes
         df = pd.concat([a, b.rename('category')], axis=1)
         df.sample(10)
```

Out[31]:

	emp_length	category
56208	5 years	5
629929	10+ years	1
388462	10+ years	1
595153	5 years	5
408474	1 year	0
177039	4 years	4
467672	10+ years	1
117714	4 years	4
501912	10+ years	1

'HOME_OWNERSHIP'

home_ownership

```
In [34]: a = loans['home_ownership'].astype('category')
         b = a.cat.codes
         df = pd.concat([a, b.rename('category')], axis=1)
         df.sample(10)
```

Out[34]:

	home_ownership	category
54169	MORTGAGE	1
708371	MORTGAGE	1
302427	MORTGAGE	1
422677	MORTGAGE	1
141408	MORTGAGE	1
57251	MORTGAGE	1
98961	MORTGAGE	1

'VERIFICATION_STATUS'

verification_status

In [37]:
```
a = loans['verification_status'].astype('category')
b = a.cat.codes
df = pd.concat([a, b.rename('category')], axis=1)
df.sample(10)
```

Out[37]:

	verification_status	category
809635	Not Verified	0
781458	Not Verified	0
287143	Not Verified	0
808663	Verified	2
216453	Source Verified	1
717321	Source Verified	1
658497	Verified	2

- We repeat this process for every column.. no need to display all of them here…they are within the .ipynb file

'EARLIEST_CR_LINE'

earliest_cr_line

In [52]:
```
a = loans['earliest_cr_line'].astype('category')
b = a.cat.codes
df = pd.concat([a, b.rename('category')], axis=1)
df.sample(10)
```

Out[52]:

	earliest_cr_line	category
704260	Jun-2005	398
828690	Apr-2001	43
660816	Sep-1991	675
38680	Aug-1996	97
783847	Aug-2000	101
556078	Dec-1997	156

STEP 3: CHECKING OUT THE LOAN STATUS COLUMN

```
In [50]: loans['loan_status'].value_counts(dropna=False)

Out[50]: Current                                                 601779
         Fully Paid                                              207723
         Charged Off                                              45248
         Late (31-120 days)                                       11591
         Issued                                                    8460
         In Grace Period                                           6253
         Late (16-30 days)                                         2357
         Does not meet the credit policy. Status:Fully Paid        1988
         Default                                                   1219
         Does not meet the credit policy. Status:Charged Off        761
         Name: loan_status, dtype: int64
```

```
loans = loans.loc[loans['loan_status'].isin(['Fully Paid', 'Charged Off'])]

loans.shape

(252971, 20)

loans['loan_status'].value_counts(dropna=False)

Fully Paid     207723
Charged Off     45248
Name: loan_status, dtype: int64

loans['loan_status'].value_counts(normalize=True, dropna=False)

Fully Paid     0.821134
Charged Off    0.178866
Name: loan_status, dtype: float64

loans['charged_off'] = (loans['loan_status'] == 'Charged Off').apply(np.uint8)
loans.drop('loan_status', axis=1, inplace=True)
```

- We want to drop off all other categories of the 'loan_status' column and just take into account 'Fully Paid' vs 'Charged Off'
- And we create a new column called 'charged_off' that is binary.

```
In [63]: loans['charged_off'] = (loans['loan_status'] == 'Charged Off').apply(np.uint8)
         loans.drop('loan_status', axis=1, inplace=True)

In [64]: loans.sample(5)

Out[64]:
```

we create a new column called charged off using 'loan_status' column

erification_status	issue_d	purpose	addr_state	dti	earliest_cr_line	open_acc	pub_rec	revol_util	total_acc	initial_list_status	application_type	charged_off
1	Nov-2014	2	9	25.67	628	8.0	0.0	65.6	26.0	1	0	0
2	Jul-2012	2	35	25.20	680	12.0	0.0	73.9	18.0	0	0	1
0	Jan-2013	1	30	13.46	37	15.0	1.0	42.8	32.0	0	0	0
1	Mar-2015	2	4	3.23	680	11.0	1.0	37.1	20.0	1	0	1
0	Apr-2014	2	17	18.41	277	10.0	0.0	0.0	44.0	0	0	0

```
In [65]: loans.drop('issue_d', axis=1, inplace=True)

In [66]: loans.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 252971 entries, 0 to 887371
Data columns (total 19 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   loan_amnt            252971 non-null  float64
 1   term                 252971 non-null  object
 2   int_rate             252971 non-null  float64
 3   installment          252971 non-null  float64
 4   sub_grade            252971 non-null  int8
 5   emp_length           252971 non-null  int8
 6   home_ownership       252971 non-null  int8
 7   verification_status  252971 non-null  int8
 8   purpose              252971 non-null  int8
 9   addr_state           252971 non-null  int8
 10  dti                  252971 non-null  float64
 11  earliest_cr_line     252971 non-null  int16
 12  open_acc             252971 non-null  float64
 13  pub_rec              252971 non-null  float64
 14  revol_util           252772 non-null  float64
 15  total_acc            252971 non-null  float64
 16  initial_list_status  252971 non-null  int8
 17  application_type     252971 non-null  int8
 18  charged_off          252971 non-null  int8
dtypes: float64(8), int16(1), int8(9), object(1)
memory usage: 22.0+ MB
```

- We are finally left with these features:
  `['loan_amnt', 'term', 'int_rate', 'installment', 'sub_grade', 'emp_length', 'home_ownership', 'verification_status', 'purpose', 'addr_state', 'dti', 'earliest_cr_line', 'open_acc', 'pub_rec', 'revol_util', 'total_acc', 'initial_list_status', 'application_type', 'charged_off']`

- We remove the 'issue_d' and 'loan status' columns because we don't need them anymore.

STEP 2: TRAIN TEST SPLIT

Train Test Split

```
In [68]: from sklearn.model_selection import train_test_split
         X=loans[['loan_amnt', 'term', 'int_rate', 'installment', 'sub_grade', 'emp_length', 'home_ownership', 'verification_s
         y=loans['charged_off']

         # Split dataset into training set and test set
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% trai
```

- We did a simple train test split….

STEP 3: RANDOM FOREST PREDICTION

Random Forest

```
In [69]: from sklearn.ensemble import RandomForestClassifier

         #Create a Gaussian Classifier
         clf=RandomForestClassifier(n_estimators=100)

         #Train the model using the training sets y_pred=clf.predict(X_test)
         clf.fit(X_train,y_train)

         y_pred=clf.predict(X_test)

In [70]: #Import scikit-learn metrics module for accuracy calculation
         from sklearn import metrics

         # Model Accuracy, how often is the classifier correct?
         print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

         Accuracy: 0.8218916354820007
```

- We trained the model using Random Forest (it went smoothly but took a long time due to large amount of dataset).
- The accuracy was 82%.