LendingClub End-to-End Project | **Part 1 – Data Wrangling**

```sql
/* Part 1: Data Wrangling for Lending Club Project
 Data set: https://www.kaggle.com/datasets/ethon0426/lending-club-20072020q1
 A raw subset for the 2018 year was created from the linked dataset and now I will clean it

 loan_data_2018_RAW.csv is imported as a flat file with Microsoft SQL Server Management Studio
    where no primary key is selected, and where NULL values are allowed in every column

Data cleaning steps:
1: Remove all duplicates
2: Standardize data
3: Manage NULL values and remove unnecessary rows/columns
*/

USE loan_data_database
--------------------------------------------------------------------------------
/*
1: Remove all duplicates
*/
--------------------------------------------------------------------------------

-- NOTE: N is written before every string to declare
--      it as nvarchar as opposed to varchar for unicode
--      this is done here with the staging table and in every procedure created further below

-- create staging table by duplicating the raw table format and copying all raw data into it
-- this way if any mistakes are made, the untouched raw data table will still exist to reference
IF NOT EXISTS (SELECT * FROM sys.objects
WHERE [object_id] = OBJECT_ID(N'[dbo].[loan_data_2018_staging]') AND type in (N'U'))
SELECT * INTO loan_data_2018_staging FROM loan_data_2018_RAW
```

```
(57171 rows affected)
```

- dbo.loan_data_2018_RAW
- dbo.loan_data_2018_staging

```sql
-- create a CTE to isolate duplicate entries, where the data in all columns is exactly the same
-- 8 duplicate rows exist
WITH cte_duplicates AS
(
    SELECT *,
    ROW_NUMBER() OVER (
        PARTITION BY id, loan_amount, term, interest_rate, installment, grade, sub_grade, employee_length, home_ownership,
            annual_income, issue_date, loan_status, purpose, purpose2, address_state, debt_to_income, total_payment ORDER BY (SELECT NULL)) AS row_num
    FROM loan_data_2018_staging
)
SELECT *
FROM cte_duplicates
WHERE row_num > 1
```

	id	loan_amount	term	interest_rate	installment	grade	sub_grade	employee_length	home_ownership	annual_income	issue_date	loan_status	purpose	purpose2	address_state	debt_to_income	total_payment	row_num
1	127935939	15750	36 months	0.200000002980232	585.330017089844	D	D4	5 years	Mortgage	83083.5234375	2018-02-18	Charged Off	Car	NULL	NC	0.200900003314018	9933.0400390625	2
2	129827583	24000	60 months	0.160099998116493	583.77001953125	C	C5	10+ years	Mortgage	100000	2018-03-18	Fully Paid	Debt	Consolidation	CA	0.320100009441376	28774.87890625	2
3	130464147	40000	36 months	0.119800001382828	1328.19995117188	B	B5	1 year	Mortgage	81000	2018-03-18	Fully Paid	Debt	Consolidation	AZ	0.318800002336502	42033.22265625	2
4	134042794	24800	60 months	0.0957999974489212	521.820007324219	B	B1	3 years	Mortgage	81323	2018-05-18	Fully Paid	Debt	Consolidation	MD	0.154100000858307	25036.634765625	2
5	139100306	15000	36 months	0.20890000462532	564.280029296875	D	D4	4 years	Rent	85000	2018-08-18	Charged Off	Other	NULL	TX	0.196099996566772	6727.83984375	2
6	140978011	25000	36 months	0.110600002110004	819.179992675781	B	B3	< 1 year	Rent	90000	2018-10-18	Current	Debt	Consolidation	WV	0.179900005459785	15610.509765625	2
7	144295663	14000	36 months	0.0755999982357025	435.880004882813	A	A3	3 years	Mortgage	85000	2018-11-18	Fully Paid	Debt	Consolidation	AK	0.121100001037121	15016.6494140625	2
8	144488901	17500	36 months	0.103299997746944	567.400024414063	B	B1	< 1 year	Rent	56786	2018-12-18	Current	Credit	Card	CA	0.172399997711182	9685.98046875	2

```sql
-- create a derived table with row_num column to count and delete duplicate rows that were isolated in the CTE above
-- 8 duplicate rows removed
DELETE dt_duplicates
FROM
(
    SELECT *, row_num = ROW_NUMBER() OVER (PARTITION BY id, loan_amount, term, interest_rate, installment,
        grade, sub_grade, employee_length, home_ownership, annual_income, issue_date, loan_status, purpose, purpose2,
        address_state, debt_to_income, total_payment ORDER BY (SELECT NULL))
    FROM loan_data_2018_staging
) AS dt_duplicates
WHERE row_num > 1
```

```
(8 rows affected)
```

```sql
-- 57171 rows in raw table, 57163 rows in staging table, 8 duplicate rows removed
SELECT *
FROM loan_data_2018_staging
```

0.187800005078316	15819.2001953125

(16.0 RTM) | -_ (60) | loan_data_database | 00:00:01 | 57,163 rows

```
68 ----------------------------------------------------------------------------------
69 /*
70 2: Standardize Data
71 */
72 ----------------------------------------------------------------------------------
73
74 -- removing any potential trailing spaces from string fields
75 /*
76 UPDATE loan_data_2018_staging2
77 SET term = TRIM(term),
78     grade = TRIM(grade),
79     ... etc ... manually list out all string columns to trim
80 */
81
82 /*
83 NOTE:
84     the above approach for trimming string fields works, but it would
85     become impractical when dealing with a table with a large number
86     of string fields, therefore I will write a script to trim all
87     string fields to solve this efficiency issue
88 */
89
90 ----------------------------------------------------------------------
```

```sql
90  ----------------------------------------------------------------------
91  /*
92      stored procedure to trim all columns in a table
93  */
94  ----------------------------------------------------------------------
95  IF EXISTS
96  (
97      SELECT type_desc, type
98      FROM sys.procedures WITH(NOLOCK)
99      WHERE NAME = 'trim_string_fields' AND type = 'P'
100 )
101 DROP PROCEDURE trim_string_fields
102 GO
103 CREATE PROCEDURE trim_string_fields @table nvarchar(max) AS DECLARE @query AS nvarchar(max)
104
105 -- trim all string values in every column and show which columns were trimmed in output
106 SET @query = STUFF((SELECT N', ' + QUOTENAME([name]) + N' = TRIM(' + QUOTENAME([name]) + N')'
107 FROM sys.columns
108 WHERE [object_id] = OBJECT_ID(@table) AND [system_type_id] IN(29,35,99,167,175,231)
109 FOR XML PATH('')),1,1,'')
110
111 SET @query = N'UPDATE ' + @table + N' SET' + @query
112 PRINT @query
113
114 EXEC(@query)
115 GO
116 ----------------------------------------------------------------------
117 -- test the script by adding trailing/leading spaces to all 'PA' state values
118 -- (1880 rows affected) trailing space address_state
119 UPDATE loan_data_2018_staging
120 SET address_state = '  PA   '
121 WHERE address_state = 'PA'
122
123 -- we can see that all 'PA' values now have the trailing/leading spaces added
124 SELECT * FROM loan_data_2018_staging
```

	id	loan_amount	term	interest_rate	installment	grade	sub_grade	employee_length	home_ownership	annual_income	issue_date	loan_status	purpose	purpose2	address_state	debt_to_income	total_payment
1	138408587	10000	36 months	0.0666999965906143	307.269989013672	A	A2	6 years	Rent	28787	2018-08-18	Current	Credit	Card	PA	0.158899992704391	6141.68994140625
2	138228354	5000	36 months	0.0846000015735626	157.75	A	A5	10+ years	Mortgage	35000	2018-08-18	Fully Paid	Debt	Consolidation	WV	0.269499987363815	5341.96337890625
3	138384636	40000	60 months	0.0784000009298325	808	A	A4	3 years	Mortgage	480000	2018-08-18	Current	Small	Business	FL	0.124600000679493	16950.580078125

```sql
-- run procedure to trim all columns including our test above
EXEC trim_string_fields loan_data_2018_staging

-- after executing the trim script above, we see that it works
SELECT * FROM loan_data_2018_staging
```

	id	loan_amount	term	interest_rate	installment	grade	sub_grade	employee_length	home_ownership	annual_income	issue_date	loan_status	purpose	purpose2	address_state	debt_to_income	total_payment
1	138408587	10000	36 months	0.0666999965906143	307.269989013672	A	A2	6 years	Rent	28787	2018-08-18	Current	Credit	Card	PA	0.158899992704391	6141.68994140625
2	138228354	5000	36 months	0.0846000015735626	157.75	A	A5	10+ years	Mortgage	35000	2018-08-18	Fully Paid	Debt	Consolidation	WV	0.269499987363815	5341.96337890625
3	138384636	40000	60 months	0.0784000009298325	808	A	A4	3 years	Mortgage	480000	2018-08-18	Current	Small	Business	FL	0.124600000679493	16950.580078125

```sql
-----------------------------------------------------------------------

-- visually inspect all string field categories to check for errors/inconsistencies

-- it appears the purpose field was split into two columns
SELECT DISTINCT purpose, purpose2
FROM loan_data_2018_staging
ORDER BY purpose2
```

	purpose	purpose2
7	Wedding	NULL
8	Moving	NULL
9	Other.	NULL
10	Vacation	NULL
11	NULL	NULL
12	Small	Business
13	Credit	Card

```sql
-- by joining purpose1 with purpose2 on the id, we prove that 6 categories need to be merged, namely:
--  "Credit Card", "Debt Consolidation", "Home Improvement", "Major Purchase",
--  "Renewable Energy", and "Small Business"
SELECT DISTINCT t1.purpose, t2.purpose2
FROM loan_data_2018_staging t1
FULL OUTER JOIN loan_data_2018_staging t2
ON t1.id = t2.id

-- purpose_merged gives us the merged columns as they should be
SELECT DISTINCT purpose, purpose2, ISNULL(purpose,'') + ' ' + ISNULL(purpose2,'') AS purpose_merged
FROM loan_data_2018_staging
```

	purpose	purpose2	purpose_merged
1	Small	Business	Small Business
2	Vacation	NULL	Vacation
3	Major	Purchase	Major Purchase
4	Debt	Consolidation	Debt Consolidation
5	Renewable	Energy	Renewable Energy
6	NULL	NULL	
7	Medical	NULL	Medical
8	Wedding	NULL	Wedding
9	.Other.	NULL	.Other.
10	Credit	Card	Credit Card
11	Other	NULL	Other
12	Other.	NULL	Other.
13	Moving	NULL	Moving
14	Home	Improvement	Home Improvem...

```sql
-- create a second staging table with a consolidated purpose column (in the same position) to store the merged values
IF NOT EXISTS (SELECT * FROM sys.objects
WHERE [object_id] = OBJECT_ID(N'[dbo].[loan_data_2018_staging2]') AND type in (N'U'))
CREATE TABLE [dbo].[loan_data_2018_staging2](
    [id] [int] NULL,
    [loan_amount] [int] NULL,
    [term] [nvarchar](50) NULL,
    [interest_rate] [float] NULL,
    [installment] [float] NULL,
    [grade] [nvarchar](50) NULL,
    [sub_grade] [nvarchar](50) NULL,
    [employee_length] [nvarchar](50) NULL,
    [home_ownership] [nvarchar](50) NULL,
    [annual_income] [float] NULL,
    [issue_date] [date] NULL,
    [loan_status] [nvarchar](50) NULL,
    [purpose] [nvarchar](50) NULL,
    [address_state] [nvarchar](50) NULL,
    [debt_to_income] [float] NULL,
    [total_payment] [float] NULL,
) ON [PRIMARY]
```

```
Commands completed successfully.
```

- dbo.loan_data_2018_RAW
- dbo.loan_data_2018_staging
- dbo.loan_data_2018_staging2

```sql
-- copy all needed fields from the original staging table into the second staging table
INSERT INTO loan_data_2018_staging2
SELECT id, loan_amount, term, interest_rate, installment, grade, sub_grade, employee_length, home_ownership,
            annual_income, issue_date, loan_status, ISNULL(purpose,'') + ' ' + ISNULL(purpose2,'') AS purpose,
            address_state, debt_to_income, total_payment
FROM loan_data_2018_staging

-- rerun procedure to trim all columns on the newly created loan_data_2018_staging2
-- since values in the first purpose column that didn't have any corressponding value in the
-- 2nd purpose column will be left with a trailing ' ' space
EXEC trim_string_fields loan_data_2018_staging2

-- all purpose categories have been merged into one and only 1 purpose column remains
SELECT DISTINCT purpose
FROM loan_data_2018_staging2
ORDER BY purpose

-- I will continue working with the 2nd staging table that has the most cleaned data
```

	purpose
1	
2	..Other.
3	.Other.
4	Car
5	Credit Card
6	Debt Consolidation
7	Home Improvement
8	House
9	Major Purchase
10	Medical
11	Moving
12	Other
13	Other.
14	Renewable Energy
15	Small Business
16	Vacation
17	Wedding

```sql
-- purpose has some rows where the Other category has leading/trailing punctuation '.' marks
SELECT DISTINCT purpose
FROM loan_data_2018_staging2
ORDER BY purpose

-- remove all leading/trailing '.' dots from the purpose field and rerun query above to confirm
UPDATE loan_data_2018_staging2
SET purpose = TRIM('.' FROM purpose)
```

	purpose
1	
2	Car
3	Credit Card
4	Debt Consolidation
5	Home Improvement
6	House
7	Major Purchase
8	Medical
9	Moving
10	Other
11	Renewable Energy
12	Small Business
13	Vacation
14	Wedding

```sql
SELECT DISTINCT term
FROM loan_data_2018_staging2
ORDER BY term

SELECT DISTINCT grade
FROM loan_data_2018_staging2
ORDER BY grade

SELECT DISTINCT sub_grade
FROM loan_data_2018_staging2
ORDER BY sub_grade

SELECT DISTINCT employee_length
FROM loan_data_2018_staging2
ORDER BY employee_length

-- home_ownership has 2 redundant Rent labels as 'Renter/Renting' which need to be consolidated to Rent
SELECT DISTINCT home_ownership
FROM loan_data_2018_staging2
ORDER BY home_ownership
```

	home_ownership
1	NULL
2	Mortgage
3	Own
4	Rent
5	Renter
6	Renting

```sql
-- fixed 14 rows showing as Renter or Renting and renamed to Rent, rerun query above to confirm only Rent remains
UPDATE loan_data_2018_staging2
SET home_ownership = 'Rent'
WHERE home_ownership IN ('Renter', 'Renting')
```

```
(14 rows affected)
```

	home_ownership
1	NULL
2	Mortgage
3	Own
4	Rent

```sql
SELECT DISTINCT loan_status
FROM loan_data_2018_staging2
ORDER BY loan_status

SELECT DISTINCT address_state
FROM loan_data_2018_staging2
ORDER BY address_state

-- issue dates are already standardized so the part below will be commented out
SELECT DISTINCT issue_date
FROM loan_data_2018_staging2
ORDER BY issue_date

/*
-- format all date entries to MMMM-yyyy format
SELECT DISTINCT FORMAT(issue_date, 'MMMM-yyyy') AS formatted_issue_date
FROM loan_data_2018_staging2
ORDER BY formatted_issue_date

-- update issue_date field to standardize date format
UPDATE loan_data_2018_staging2
SET issue_date = FORMAT(issue_date, 'MMMM-yyyy')
WHERE issue_date IS NOT NULL -- to ensure we don't try to format NULL values
*/
```

	issue_date
1	NULL
2	2018-01-18
3	2018-02-18
4	2018-03-18
5	2018-04-18
6	2018-05-18
7	2018-06-18
8	2018-07-18
9	2018-08-18
10	2018-09-18
11	2018-10-18
12	2018-11-18
13	2018-12-18

```sql
-- ensure there aren't any negative numerical values present; 0 found
SELECT loan_amount, interest_rate, installment, debt_to_income, total_payment
FROM loan_data_2018_staging2
WHERE loan_amount < 0 OR interest_rate < 0 OR installment < 0 OR debt_to_income < 0 OR total_payment < 0
```

loan_amount	interest_rate	installment	debt_to_income	total_payment

0 rows

```sql
/*
3: Manage NULL values and remove unnecessary rows/columns

NOTE:
    I will write a script to find any potential NULL and/or
    empty values for the same efficiency reason that I wrote
    one to trim all columns
*/

-----------------------------------------------------------------------
/*
    stored procedure to display all null values in a table
*/
-----------------------------------------------------------------------
IF EXISTS
(
    SELECT type_desc, type
    FROM sys.procedures WITH(NOLOCK)
    WHERE NAME = 'select_null_values' AND type = 'P'
)
DROP PROCEDURE select_null_values
GO
CREATE PROCEDURE select_null_values @table nvarchar(max) AS DECLARE @query AS nvarchar(max)

-- extract the table schema without extracting any data within
SET @query = N'SELECT * FROM ' + @table + N' WHERE 1 = 0'

-- check for NULL values in every column
SELECT @query += N' OR ' + QUOTENAME(name) + N' IS NULL'
FROM sys.columns
WHERE [object_id] = OBJECT_ID(@table)

EXEC(@query)
GO
```

```
292 ----------------------------------------------------------------------
293 -- run procedure to select NULL values in all columns, 3 rows with NULL data exist
294 EXEC select_null_values loan_data_2018_staging2
```

	id	loan_amount	term	interest_rate	installment	grade	sub_grade	employee_length	home_ownership	annual_income	issue_date	loan_status	purpose	address_state	debt_to_income	total_payment
1	137917888	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL		NULL	NULL	NULL
2	139305488	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL		NULL	NULL	NULL
3	131407188	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL		NULL	NULL	NULL

```
296 -- delete 3 rows with NULL data
297 -- these same 3 rows were also causing the NULL value to appear in several columns
298 DELETE
299 FROM loan_data_2018_staging2
300 WHERE id = 131407188 OR id = 137917888 OR id = 139305488
301
302 -- running one more time shows that no NULL values remain in the table
303 EXEC select_null_values loan_data_2018_staging2
```

(3 rows affected)

id	loan_amount	term	interest_rate	installment	grade	sub_grade	employee_length	home_ownership	annual_income	issue_date	loan_status	purpose	address_state	debt_to_income	total_payment

```
305 -- the annual_income column is not needed for my analysis so it must be removed
306 ALTER TABLE loan_data_2018_staging2
307 DROP COLUMN IF EXISTS annual_income
```

Commands completed successfully.

```sql
---------------------------------------------------------------------
/*
    stored procedure to display all empty '' values in a table
*/
---------------------------------------------------------------------
IF EXISTS
(
    SELECT type_desc, type
    FROM sys.procedures WITH(NOLOCK)
    WHERE NAME = 'select_empty_values' AND type = 'P'
)
DROP PROCEDURE select_empty_values
GO
CREATE PROCEDURE select_empty_values @table nvarchar(max) AS DECLARE @query AS nvarchar(max)

SET @query = N'SELECT * FROM ' + @table + N' WHERE 1 = 0'

-- check for empty '' values (written as '''' since escape characters are needed) in every column
SELECT @query += N' OR ' + QUOTENAME(name) + N' LIKE '''''
FROM sys.columns
WHERE [object_id] = OBJECT_ID(@table)

EXEC(@query)
GO
---------------------------------------------------------------------

-- no empty values remain in the table
EXEC select_empty_values loan_data_2018_staging2
```

id	loan_amount	term	interest_rate	installment	grade	sub_grade	employee_length	home_ownership	issue_date	loan_status	purpose	address_state	debt_to_income	total_payment

```sql
-- rename final cleaned staging table
EXEC sp_rename 'loan_data_2018_staging2', 'loan_data_2018'
```

- dbo.loan_data_2018
- dbo.loan_data_2018_RAW
- dbo.loan_data_2018_staging

```sql
-- 57160 rows with 15 columns in final cleaned table
SELECT * FROM loan_data_2018
ORDER BY id
```

	id	loan_amount	term	interest_rate	installment	grade	sub_grade	employee_length	home_ownership	issue_date	loan_status	purpose	address_state	debt_to_income	total_payment
1	118187703	40000	36 months	0.0671999976038933	1229.96997070313	A	A3	10+ years	Mortgage	2018-01-18	Current	Credit Card	MD	0.220400005578995	34424.23046875
2	119987870	11200	60 months	0.19030000269413	290.720001220703	D	D3	10+ years	Rent	2018-02-18	Fully Paid	Debt Consolidation	RI	0.174999997019768	12192.6875
3	120068032	20475	60 months	0.180600002408028	520.599975585938	D	D2	10+ years	Mortgage	2018-01-18	Current	Debt Consolidation	VA	0.18520000576973	14556.259765625
4	120640856	40000	36 months	0.0531999990344048	1204.59997558594	A	A1	1 year	Mortgage	2018-01-18	Fully Paid	Small Business	TX	0.112999998033047	41776.55859375
5	121263466	25000	36 months	0.0797000005841255	783.070007324219	A	A5	3 years	Mortgage	2018-01-18	Fully Paid	Debt Consolidation	MD	0.28099998831749	27557.921875
6	121703857	30000	60 months	0.150399997830391	714.330017089844	C	C4	10+ years	Mortgage	2018-04-18	Current	Debt Consolidation	MN	0.273299992084503	18887.169921875
7	121785745	6000	36 months	0.150499999523163	208.139999389648	C	C4	7 years	Rent	2018-01-18	Current	Other	NJ	0.149000003933907	5819.8198242187

Query executed successfully. - (16.0 RTM) -_ (60) loan_data_database 00:00:01 57,160 rows

LendingClub End-to-End Project | **Part 2 – Exploratory Data Analysis**

```sql
/* Part 2: Exploratory Data Analysis for Lending Club Project
 Data set: https://www.kaggle.com/datasets/ethon0426/lending-club-20072020q1
 A raw subset for the 2018 year was cleaned in Part 1 of the Project

 Now I will perform exploratory data analysis on the cleaned subset
 */

USE loan_data_database

-- 57160 total loan applications issued, 3253 issued in December alone, and 3606 in November
SELECT COUNT(id) AS total_loan_applications FROM loan_data_2018
SELECT COUNT(id) AS MTD_total_loan_applications FROM loan_data_2018 WHERE MONTH(issue_date) = 12
SELECT COUNT(id) AS prior_MTD_total_loan_applications FROM loan_data_2018 WHERE MONTH(issue_date) = 11
```

	total_loan_applications
1	57160

	MTD_total_loan_applications
1	3253

	prior_MTD_total_loan_applications
1	3606

```sql
-- $1,138.2M in total funded loans, $65M funded in December alone, and $74M in November
SELECT FORMAT(SUM(loan_amount), 'C', 'en-US') AS total_funded_amount FROM loan_data_2018
SELECT FORMAT(SUM(loan_amount), 'C', 'en-US') AS MTD_total_funded_amount FROM loan_data_2018 WHERE MONTH(issue_date) = 12
SELECT FORMAT(SUM(loan_amount), 'C', 'en-US') AS prior_MTD_total_funded_amount FROM loan_data_2018 WHERE MONTH(issue_date) = 11
```

	total_funded_amount
1	$1,138,202,000.00

	MTD_total_funded_amount
1	$65,045,200.00

	prior_MTD_total_funded_amount
1	$74,097,350.00

```sql
-- $840M in total payments received, $38.5M received in December alone, and $45.5M in November
-- 2018 has not seen overall profitability as the total payments received are 73.8% of the total funded loans
SELECT FORMAT(SUM(total_payment), 'C', 'en-US') AS total_amount_received FROM loan_data_2018
SELECT FORMAT(SUM(total_payment), 'C', 'en-US') AS MTD_total_amount_received FROM loan_data_2018 WHERE MONTH(issue_date) = 12
SELECT FORMAT(SUM(total_payment), 'C', 'en-US') AS prior_MTD_total_amount_received FROM loan_data_2018 WHERE MONTH(issue_date) = 11
```

	total_amount_received
1	$840,021,972.47

	MTD_total_amount_received
1	$38,506,403.32

	prior_MTD_total_amount_received
1	$45,535,225.22

```sql
-- avg interest rate of 14.75% for 2018, avg 14.33% interest rate in December alone, and 14.29% in in November
SELECT ROUND(AVG(interest_rate) * 100, 2) AS avg_interest_rate FROM loan_data_2018
SELECT ROUND(AVG(interest_rate) * 100, 2) AS MTD_avg_interest_rate FROM loan_data_2018 WHERE MONTH(issue_date) = 12
SELECT ROUND(AVG(interest_rate) * 100, 2) AS prior_MTD_avg_interest_rate FROM loan_data_2018 WHERE MONTH(issue_date) = 11
```

	avg_interest_rate
1	14.75

	MTD_avg_interest_rate
1	14.33

	prior_MTD_avg_interest_rate
1	14.29

```sql
-- avg DTI of 20.97% for 2018, 20.84% avg DTI in December alone, and 20.89% in November
SELECT ROUND(AVG(debt_to_income) * 100, 2) AS avg_debt_to_income FROM loan_data_2018
SELECT ROUND(AVG(debt_to_income) * 100, 2) AS MTD_avg_debt_to_income FROM loan_data_2018 WHERE MONTH(issue_date) = 12
SELECT ROUND(AVG(debt_to_income) * 100, 2) AS prior_MTD_avg_debt_to_income FROM loan_data_2018 WHERE MONTH(issue_date) = 11
```

	avg_debt_to_income
1	20.97

	MTD_avg_debt_to_income
1	20.84

	prior_MTD_avg_debt_to_income
1	20.89

```sql
-- 12.6% of all loans issued have been charged off, where this accounts for 7200 loans and $145.3M in funds where $54.5M was paid
-- hence $90.8M total was charged off in 2018
SELECT
    (SELECT (COUNT(CASE WHEN loan_status = 'Charged Off' THEN id END) * 100.0) / COUNT(id) FROM loan_data_2018) AS bad_loan_percentage,
    COUNT(id) AS bad_loan_applications,
    FORMAT(SUM(loan_amount), 'C', 'en-US') AS bad_loan_funded_amount,
    FORMAT(SUM(total_payment), 'C', 'en-US') AS bad_loan_amount_received,
    FORMAT(SUM(loan_amount)-SUM(total_payment), 'C', 'en-US') AS total_charged_off
FROM loan_data_2018
WHERE loan_status = 'Charged Off'
```

	bad_loan_percentage	bad_loan_applications	bad_loan_funded_amount	bad_loan_amount_received	total_charged_off
1	12.555983205038	7177	$145,335,150.00	$54,515,127.14	$90,820,022.86

```sql
-- 87.4% of all loans issued have been fully paid or are current, where this accounts for 50000 loans and $992.9M in funds where $785.5M was paid
-- hence $237.1M is the approximate total outstanding loan balance for 2018 ($207.4M remaining * avg annual interest of 14.7%)
-- outstanding loan balance does not account for fees, penalities, compounding, and other hidden charges as we don't have that data in the subset
SELECT
    (SELECT (COUNT(CASE WHEN loan_status = 'Current' OR loan_status = 'Fully Paid' THEN id END) * 100.0) / COUNT(id) FROM loan_data_2018) AS good_loan_percentage,
    COUNT(id) AS good_loan_applications,
    FORMAT(SUM(loan_amount), 'C', 'en-US') AS good_loan_funded_amount,
    FORMAT(SUM(total_payment), 'C', 'en-US') AS good_loan_amount_received,
    FORMAT((SUM(loan_amount)-SUM(total_payment))*(1+AVG(interest_rate)), 'C', 'en-US') AS total_outstanding_balance
FROM loan_data_2018
WHERE loan_status = 'Current' OR loan_status = 'Fully Paid'
```

	good_loan_percentage	good_loan_applications	good_loan_funded_amount	good_loan_amount_received	total_outstanding_balance
1	87.444016794961	49983	$992,866,850.00	$785,506,845.33	$237,135,503.36

```sql
-- the average interest rate for bad loans at 17.45% is only slightly higher than the rates for good loans averaging 14.43%
-- of the 50000 loans in good standing: 17525 are fully paid and 32458 are current
-- of the loans that are current about 63.2% of the total funded amount (not including interest) has been repaid
-- all loan applicants had similar average DTI levels at around 21%
SELECT
        loan_status AS Loan_Status,
        COUNT(id) AS Total_Loan_Applications,
        FORMAT(SUM(loan_amount), 'C', 'en-US') AS Total_Funded_Amount,
        FORMAT(SUM(total_payment), 'C', 'en-US') AS Total_Amount_Received,
        AVG(interest_rate * 100) AS Avg_Interest_Rate,
        AVG(debt_to_income * 100) AS Avg_Debt_to_Income
FROM loan_data_2018
GROUP BY loan_status
ORDER BY loan_status ASC

SELECT
    loan_status AS Loan_Status,
    FORMAT(SUM(total_payment), 'C', 'en-US') AS MTD_Total_Amount_Received,
    FORMAT(SUM(loan_amount), 'C', 'en-US') AS MTD_Total_Funded_Amount
FROM loan_data_2018
WHERE MONTH(issue_date) = 12 AND YEAR(issue_date) = 2018
GROUP BY loan_status
ORDER BY loan_status ASC
```

	Loan_Status	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received	Avg_Interest_Rate	Avg_Debt_to_Income
1	Charged Off	7177	$145,335,150.00	$54,515,127.14	17.4487070358383	21.2798634478354
2	Current	32458	$668,935,375.00	$422,602,080.49	14.2068948060269	21.0670001237622
3	Fully Paid	17525	$323,931,475.00	$362,904,764.84	14.6416428452637	20.6537980119699

	Loan_Status	MTD_Total_Amount_Received	MTD_Total_Funded_Amount
1	Charged Off	$1,614,785.17	$6,184,775.00
2	Current	$21,944,025.52	$45,214,425.00
3	Fully Paid	$14,947,592.63	$13,646,000.00

```sql
-- Loans which were fully paid resulted in a profit of 12% (which is less than avg interest of 14.6%) or $39M, whereas chargeoffs were a loss of 62.5% or $90.8M
-- hence net loss was $51.8M among Fully Paid and Charged Off loans
SELECT
    loan_status AS Loan_Status,
    (SUM(total_payment) / SUM(loan_amount)-1) AS Profit_Multiple,
    FORMAT((SUM(loan_amount) * (SUM(total_payment) / SUM(loan_amount)-1)), 'C', 'en-US') AS Profit
FROM loan_data_2018 WHERE loan_status = 'Fully Paid' OR loan_status = 'Charged Off'
GROUP BY loan_status
```

	Loan_Status	Profit_Multiple	Profit
1	Fully Paid	0.120313377516796	$38,973,289.84
2	Charged Off	-0.624900602927515	($90,820,022.86)

```sql
-- the purpose of most loans was for debt consolidation (58.5% of all loans) and credit cards (21.5% of all loans)
SELECT
    purpose AS Loan_Purpose,
    COUNT(id) AS Total_Loan_Applications,
    COUNT(id) * 100.0 / (SELECT COUNT(id) FROM loan_data_2018) AS Total_Loan_Applications_Percentage,
    FORMAT(SUM(loan_amount), 'C', 'en-US') AS Total_Funded_Amount,
    FORMAT(SUM(total_payment), 'C', 'en-US') AS Total_Received_Amount
FROM loan_data_2018
-- WHERE loan_status = 'Charged Off' -- used to compare between charged off loans only and all loans to see if the distribution of loans in each category varies
GROUP BY purpose
ORDER BY Total_Loan_Applications DESC
```

	Loan_Purpose	Total_Loan_Applications	Total_Loan_Applications_Percentage	Total_Funded_Amount	Total_Received_Amount
1	Debt Consolidation	33435	58.493701889433	$705,984,675.00	$520,128,743.04
2	Credit Card	12309	21.534289713086	$243,634,925.00	$179,159,762.15
3	Other	3887	6.800209937018	$57,036,725.00	$43,271,613.84
4	Home Improvement	3292	5.759272218334	$64,872,775.00	$48,595,017.44
5	Major Purchase	1062	1.857942617214	$20,045,100.00	$14,592,753.09
6	Medical	686	1.200139958012	$8,738,600.00	$6,503,100.16
7	House	606	1.060181945416	$11,288,125.00	$8,363,314.25
8	Moving	576	1.007697690692	$5,744,950.00	$4,215,035.76
9	Small Business	550	0.962211336599	$12,800,325.00	$9,179,222.55
10	Car	365	0.638558432470	$4,554,350.00	$3,395,482.21
11	Vacation	328	0.573827851644	$2,703,775.00	$1,973,778.91
12	Renewable Energy	63	0.110216934919	$782,775.00	$636,401.81
13	Wedding	1	0.001749475157	$14,900.00	$7,747.28

```sql
-- 40% of all loans issued were to those who had 10+ years employment history
SELECT
    employee_length AS Employee_Length,
    COUNT(id) AS Total_Loan_Applications,
    COUNT(id) * 100.0 / (SELECT COUNT(id) FROM loan_data_2018) AS Total_Loan_Applications_Percentage,
    FORMAT(SUM(loan_amount), 'C', 'en-US') AS Total_Funded_Amount,
    FORMAT(SUM(total_payment), 'C', 'en-US') AS Total_Received_Amount
FROM loan_data_2018
-- WHERE loan_status = 'Charged Off'
GROUP BY employee_length
ORDER BY Total_Loan_Applications DESC
```

	Employee_Length	Total_Loan_Applications	Total_Loan_Applications_Percentage	Total_Funded_Amount	Total_Received_Amount
1	10+ years	23090	40.395381385584	$480,092,350.00	$354,889,446.00
2	2 years	5136	8.985304408677	$96,791,650.00	$72,241,496.57
3	3 years	5025	8.791112666200	$97,214,050.00	$71,642,651.47
4	4 years	3885	6.796710986703	$73,316,900.00	$54,383,099.65
5	< 1 year	3830	6.700489853044	$77,442,200.00	$55,684,570.52
6	5 years	3793	6.635759272218	$73,374,325.00	$54,298,067.56
7	1 year	3402	5.951714485654	$63,480,500.00	$46,044,174.18
8	6 years	2869	5.019244226731	$56,186,300.00	$41,518,071.22
9	7 years	2303	4.029041287613	$44,750,625.00	$33,425,779.28
10	8 years	2143	3.749125262421	$42,138,975.00	$31,304,296.69
11	9 years	1684	2.946116165150	$33,414,125.00	$24,590,319.34

```sql
-- 58% of loans issued to those who have a mortgage on their house, 32% to renters, and 10% to those who own their house outright
SELECT
    home_ownership AS Home_Ownership,
    COUNT(id) AS Total_Loan_Applications,
    COUNT(id) * 100.0 / (SELECT COUNT(id) FROM loan_data_2018) AS Total_Loan_Applications_Percentage,
    FORMAT(SUM(loan_amount), 'C', 'en-US') AS Total_Funded_Amount,
    FORMAT(SUM(total_payment), 'C', 'en-US') AS Total_Received_Amount
FROM loan_data_2018
-- WHERE loan_status = 'Charged Off'
GROUP BY home_ownership
ORDER BY Total_Loan_Applications DESC
```

	Home_Ownership	Total_Loan_Applications	Total_Loan_Applications_Percentage	Total_Funded_Amount	Total_Received_Amount
1	Mortgage	33169	58.028341497550	$701,010,800.00	$519,683,997.03
2	Rent	18274	31.969909027291	$327,711,775.00	$237,404,542.66
3	Own	5717	10.001749475157	$109,479,425.00	$82,933,432.78

```sql
-- CA is the top state with issued loans at 12.3% of all loans, followed by TX at 8.7%, and FL and NY tied for third at 6.8%
-- IA is the only state with no loans issued
SELECT
    address_state AS State,
    COUNT(id) AS Total_Loan_Applications,
    COUNT(id) * 100.0 / (SELECT COUNT(id) FROM loan_data_2018) AS Total_Loan_Applications_Percentage,
    FORMAT(SUM(loan_amount), 'C', 'en-US') AS Total_Funded_Amount,
    FORMAT(SUM(total_payment), 'C', 'en-US') AS Total_Received_Amount
--  (SUM(loan_amount)-SUM(total_payment))/SUM(loan_amount) AS funding_gap_percent
FROM loan_data_2018
-- WHERE loan_status = 'Charged Off'
GROUP BY address_state
ORDER BY Total_Loan_Applications DESC
```

	State	Total_Loan_Applications	Total_Loan_Applications_Percentage	Total_Funded_Amount	Total_Received_Amount
1	CA	7019	12.279566130160	$144,476,925.00	$108,563,639.50
2	TX	4952	8.663400979706	$101,243,725.00	$74,891,556.69
3	FL	3888	6.801959412176	$74,955,000.00	$54,571,346.45
4	NY	3855	6.744226731980	$78,296,475.00	$56,146,690.01
5	IL	2340	4.093771868439	$47,692,500.00	$35,019,564.38
6	GA	1971	3.448215535339	$40,388,550.00	$29,330,374.74
7	NJ	1914	3.348495451364	$40,430,125.00	$29,565,126.81
8	PA	1880	3.289013296011	$37,095,650.00	$27,351,635.75
9	OH	1867	3.266270118964	$34,986,050.00	$25,638,098.68
10	VA	1643	2.874387683694	$34,449,250.00	$25,448,940.21
11	MD	1596	2.792162351294	$33,735,000.00	$24,555,303.83
12	NC	1565	2.737928621413	$30,631,725.00	$22,310,330.67
13	MI	1513	2.646955913226	$28,954,675.00	$21,700,257.83
14	CO	1325	2.318054583624	$26,519,700.00	$20,679,953.29
15	AZ	1307	2.286564030790	$24,787,075.00	$18,711,776.14
16	WA	1302	2.277816655003	$25,682,000.00	$20,660,972.06
17	MA	1250	2.186843946815	$25,649,025.00	$19,020,403.61
18	MN	1067	1.866689993002	$20,988,375.00	$15,058,203.79
19	IN	1061	1.856193142057	$20,235,750.00	$14,975,923.76
20	TN	1039	1.817704688593	$19,823,200.00	$14,535,993.42
21	CT	1002	1.752974107767	$20,218,175.00	$14,474,533.16
22	MO	994	1.738978306508	$18,902,050.00	$13,687,993.75
23	WI	897	1.569279216235	$16,740,875.00	$12,487,145.20
24	NV	843	1.474807557732	$16,392,300.00	$12,707,247.53
25	SC	772	1.350594821553	$15,247,475.00	$11,409,347.64

26	AL	685	1.198390482855	$13,081,950.00	$9,229,371.83
27	LA	683	1.194891532540	$13,388,925.00	$9,291,666.26
28	OR	677	1.184394681595	$12,877,600.00	$9,979,152.08
29	KY	572	1.000699790062	$10,124,050.00	$7,249,637.86
30	OK	558	0.976207137858	$11,302,425.00	$7,939,040.52
31	KS	491	0.858992302309	$9,582,250.00	$6,769,880.18
32	AR	436	0.762771168649	$8,294,200.00	$5,805,124.18
33	UT	421	0.736529041287	$8,169,750.00	$6,567,368.40
34	MS	383	0.670048985304	$7,525,050.00	$5,280,051.31
35	NE	364	0.636808957312	$6,379,200.00	$4,731,400.58
36	WV	354	0.619314205738	$6,966,250.00	$5,214,387.47
37	NH	314	0.549335199440	$6,543,350.00	$4,949,197.73
38	NM	284	0.496850944716	$5,402,100.00	$3,879,498.56
39	HI	268	0.468859342197	$5,563,800.00	$3,996,057.31
40	RI	250	0.437368789363	$4,773,250.00	$3,406,322.21
41	ID	224	0.391882435269	$4,073,150.00	$3,229,931.03
42	ME	220	0.384884534639	$4,063,450.00	$3,111,372.98
43	DE	182	0.318404478656	$3,583,525.00	$2,517,650.85
44	WY	156	0.272918124562	$2,936,600.00	$2,272,245.45
45	AK	154	0.269419174247	$3,361,600.00	$2,485,352.51
46	MT	147	0.257172848145	$2,649,375.00	$1,969,317.32
47	VT	146	0.255423372988	$2,458,775.00	$1,788,511.05
48	ND	134	0.234429671098	$2,752,450.00	$2,083,537.06
49	SD	110	0.192442267319	$1,918,900.00	$1,384,207.37
50	DC	85	0.148705388383	$1,908,375.00	$1,389,331.45

```sql
-- May is the month with most loans issued at 10% of all loans, while December is the month with the least at 5.7%
-- Loan applications trended up from Jan. through May, then trended down from May through the rest of the year
SELECT
    MONTH(issue_date) AS Mnth,
    DATENAME(MONTH, issue_date) AS 'Month',
    COUNT(id) AS Total_Loan_Applications,
    COUNT(id) * 100.0 / (SELECT COUNT(id) FROM loan_data_2018) AS Total_Loan_Applications_Percentage,
    FORMAT(SUM(loan_amount), 'C', 'en-US') AS Total_Funded_Amount,
    FORMAT(SUM(total_payment), 'C', 'en-US') AS Total_Received_Amount
FROM loan_data_2018
-- WHERE loan_status = 'Charged Off'
GROUP BY MONTH(issue_date), DATENAME(MONTH, issue_date)
ORDER BY Mnth
```

	Mnth	Month	Total_Loan_Applications	Total_Loan_Applications_Percentage	Total_Funded_Amount	Total_Received_Amount
1	1	January	4565	7.986354093771	$92,398,725.00	$79,264,216.92
2	2	February	4326	7.568229531140	$86,815,275.00	$72,694,970.79
3	3	March	5333	9.329951014695	$106,329,750.00	$86,589,651.71
4	4	April	5492	9.608117564730	$108,591,650.00	$85,519,103.56
5	5	May	5699	9.970258922323	$111,167,375.00	$84,516,780.22
6	6	June	4865	8.511196641007	$94,388,200.00	$71,063,507.64
7	7	July	5307	9.284464660601	$104,000,650.00	$76,874,141.60
8	8	August	5470	9.569629111266	$107,553,425.00	$76,170,421.09
9	9	September	4614	8.072078376487	$92,962,300.00	$62,523,015.26
10	10	October	4630	8.100069979006	$94,852,100.00	$60,764,535.13
11	11	November	3606	6.308607417774	$74,097,350.00	$45,535,225.22
12	12	December	3253	5.691042687193	$65,045,200.00	$38,506,403.32

```sql
-- 58% of loans issued for a 36 month term while 42% issued for a 60 month term
SELECT
    term AS Term,
    COUNT(id) AS Total_Loan_Applications,
    COUNT(id) * 100.0 / (SELECT COUNT(id) FROM loan_data_2018) AS Total_Loan_Applications_Percentage,
    FORMAT(SUM(loan_amount), 'C', 'en-US') AS Total_Funded_Amount,
    FORMAT(SUM(total_payment), 'C', 'en-US') AS Total_Received_Amount
FROM loan_data_2018
-- WHERE loan_status = 'Charged Off'
GROUP BY Term
ORDER BY Term
```

	Term	Total_Loan_Applications	Total_Loan_Applications_Percentage	Total_Funded_Amount	Total_Received_Amount
1	36 months	33139	57.975857242827	$564,465,475.00	$466,477,998.39
2	60 months	24021	42.024142757172	$573,736,525.00	$373,543,974.08

```sql
/*
-- 62% of all loans issued were to C - G graded applicants which represent moderate through very high risk loans (and therefore carry higher average interest rates)
    since all applicants in the dataset have manageable DTI levels, these C-G grades have average through very low credit scores,
    may have a history of missed payments, and potentially carry significant risk factors
-- 38% of all loans issued were to A and B graded applicants having high to good credit scores with overall positive credit history representing low risk loans
-- While the average interest rate only jumps by about 4% from A to B grades, the interest rate more than doubles from A to C grades, and more than doubles from C to G grades
-- Only 2.3% of all A and B graded loans issued were charged off (about 18% of all bad loans),
    meaning 82% of all charge offs came from C - G graded loans, hence there is a strong negative correlation between grade and bad loans issued
*/
SELECT
    grade AS Grade,
    COUNT(id) AS Total_Loan_Applications,
    COUNT(id) * 100.0 / (SELECT COUNT(id) FROM loan_data_2018) AS Total_Loan_Applications_Percentage,
    AVG(interest_rate) AS Average_Interest_Rate,
    AVG(debt_to_income) AS Average_DTI
FROM loan_data_2018
-- WHERE loan_status = 'Charged Off'
GROUP BY grade
ORDER BY grade ASC
```

	Grade	Total_Loan_Applications	Total_Loan_Applications_Percentage	Average_Interest_Rate	Average_DTI
1	A	9015	15.771518544436	0.0714684080747816	0.19682044370665
2	B	12636	22.106368089573	0.109755286315103	0.205070995602753
3	C	17611	30.810006997900	0.14739977402695	0.211282925462931
4	D	13245	23.171798460461	0.195359275817714	0.216341978155721
5	E	3792	6.634009797060	0.253063528786732	0.220699235227132
6	F	717	1.254373687893	0.294253139607245	0.225109763090165
7	G	144	0.251924422673	0.308143063965771	0.239577082296213

```sql
SELECT
    grade AS Grade,
    COUNT(id) AS Total_Loan_Applications,
    COUNT(id) * 100.0 / (SELECT COUNT(id) FROM loan_data_2018) AS Total_Loan_Applications_Percentage,
    AVG(interest_rate) AS Average_Interest_Rate,
    AVG(debt_to_income) AS Average_DTI
FROM loan_data_2018
WHERE loan_status = 'Charged Off'
GROUP BY grade
ORDER BY grade ASC
```

	Grade	Total_Loan_Applications	Total_Loan_Applications_Percentage	Average_Interest_Rate	Average_DTI
1	A	332	0.580825752274	0.0734542168744178	0.194529819470572
2	B	1007	1.761721483554	0.110860774472963	0.204482919380232
3	C	2274	3.978306508047	0.148142261301203	0.211964819567211
4	D	2412	4.219734079776	0.197066459675037	0.215559079665474
5	E	900	1.574527641707	0.254597000430028	0.220034888941381
6	F	204	0.356892932120	0.294812746784266	0.222345588144426
7	G	48	0.083974807557	0.308150008320808	0.238149997778237

```sql
/*
Overall Findings:
-   Even in the best case scenario where the roughly 237.1M outstanding loan balance had been paid off for current loans by the end of 2018,
        2018 would've still seen a loss of (1-((total_amount_received + total_outstanding_balance) / total_funded_amount))*100 = 3.86%
            indicating that far too many loans were charged off at 12.6% of all loans issued
-   The distribution of all loans issued by fields like purpose, employee length, home ownership, apps by state, apps by month, and apps by term
        remain consistent regardless of the loan status, hence there is no significant correlation between loan status and these other fields
-   As the loan grade increases (A being the highest) the likelihood of a charge off significantly decreases (especially after increasing from C to B),
        hence there is a strong negative correlation between grade and bad loans issued
-   The first half of the year is the most opportune for capturing as much business as possible
*/

SELECT
    (1 - ((SUM(total_payment) + ((SUM(loan_amount) - SUM(total_payment)) * (1 + AVG(interest_rate))))
        / SUM(loan_amount))) * 100 AS best_case_scenario_net_profit_percent
FROM loan_data_2018
```

	best_case_scenario_net_profit_percent
1	-3.86339933778701

```sql
-- contingency table (cross-tabulation) of the categorical loan_status and grade variables to analyze correlation
-- 4th column added to view Bad_Good Ratio of loans per grade for easy interpretation
-- as the Bad_Good_Ratio increases the loan grade decreases (G being the lowest)
    -- overall risk profile of the corressponding loans increases
SELECT
    COALESCE(bad_loans.grade, good_loans.grade) AS Grade,
    COALESCE(bad_loans.Total_Bad_Loans, 0) AS Total_Bad_Loans,
    COALESCE(good_loans.Total_Good_Loans, 0) AS Total_Good_Loans,
    CASE
        WHEN COALESCE(good_loans.Total_Good_Loans, 0) = 0 THEN NULL
        ELSE CAST(COALESCE(bad_loans.Total_Bad_Loans, 0) AS FLOAT) /
             CAST(COALESCE(good_loans.Total_Good_Loans, 1) AS FLOAT)
    END AS Bad_Good_Ratio
FROM
    (SELECT
        grade AS grade,
        COUNT(id) AS Total_Bad_Loans
     FROM loan_data_2018
     WHERE loan_status = 'Charged Off'
     GROUP BY grade) AS bad_loans
FULL OUTER JOIN
    (SELECT
        grade AS grade,
        COUNT(id) AS Total_Good_Loans
     FROM loan_data_2018
     WHERE loan_status = 'Current' OR loan_status = 'Fully Paid'
     GROUP BY grade) AS good_loans
ON
    bad_loans.grade = good_loans.grade
ORDER BY
    Grade ASC
```

	Grade	Total_Bad_Loans	Total_Good_Loans	Bad_Good_Ratio
1	A	332	8683	0.0382356328457906
2	B	1007	11629	0.0865938601771434
3	C	2274	15337	0.148268892221425
4	D	2412	10833	0.222653004707837
5	E	900	2892	0.311203319502075
6	F	204	513	0.39766081871345
7	G	48	96	0.5

LendingClub End-to-End Project | **Part 3 - Tableau Dashboard**

https://public.tableau.com/app/profile/alexander.j.porter/viz/2018LendingClubDashboard/Summary

---

LendingClub End-to-End Project | **Part 4 - Presentation**

https://github.com/Alexander-J-Porter/LendingClub-Loan-End-to-End-Project-using-Excel-SQL-Tableau/blob/main/LendingClub_Loan_Presentation.pdf