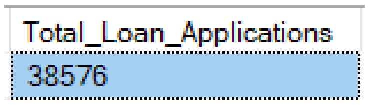
QUERY DOCUMENT

KPI

```
--- Use Bank_loan database for this query
use Bank_loan
Go
--Select data from table loan_Data
Select * from loan_data
Go
```

1 Total Loan Applications

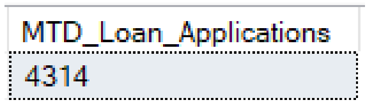
```
Select count(*) as Total_Loan_Applications from loan_data
```



Total_Loan_Applications
38576

2 MTD Loan Applications

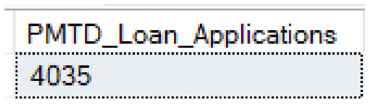
```
Select Count(*) as MTD_Loan_Applications FROM loan_data
where year(issue_date) = 2021
and month(issue_date) = 12
```



MTD_Loan_Applications
4314

3 PMTD Loan Applications

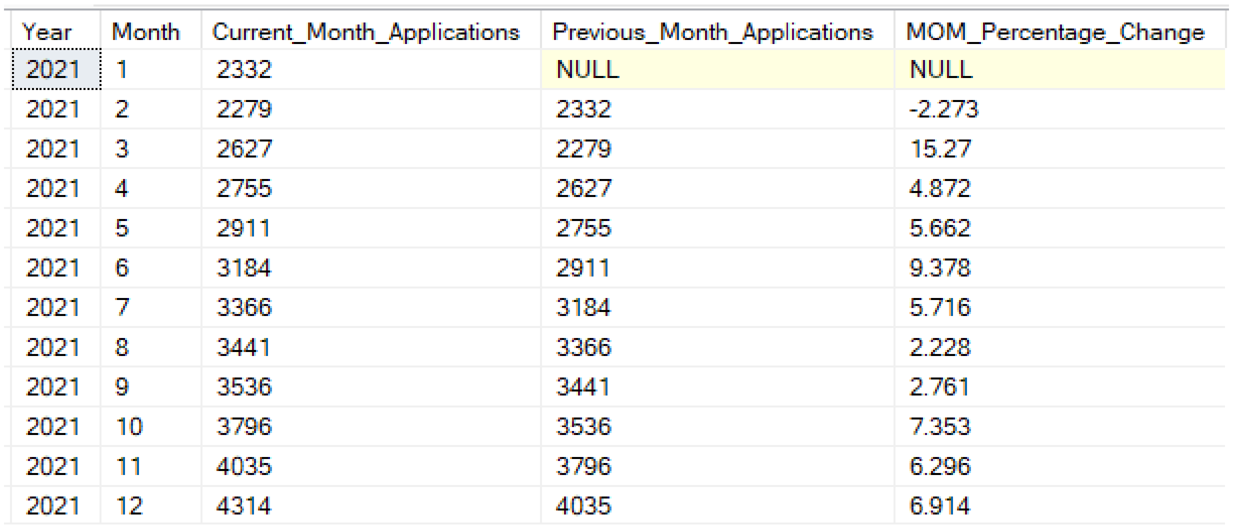
```
Select Count(*) as PMTD_Loan_Applications FROM loan_data
where year(issue_date) = 2021
and month(issue_date) = 11
```



PMTD_Loan_Applications
4035

4 MOM Loan Applications

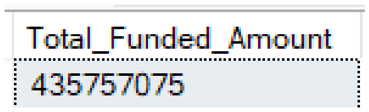
```sql
With MonthlyCounts as
(SELECT
Year(issue_date) AS Year,
Month(issue_date) AS Month,
Count(*) AS Loan_Applications
FROM loan_data
GROUP BY Year(issue_date), Month(issue_date)
),
--WITH is common table expression (CTE) stores data results temporarily

MOM AS(
SELECT
Year,
Month,
Loan_Applications as Current_Month_Applications,
LAG(Loan_Applications) OVER(ORDER BY Year, Month) as Previous_Month_Applications,
(Loan_Applications-Lag(Loan_Applications) OVER( ORDER BY Year, Month)) * 100.0/
NULLIF(LAG(Loan_Applications) OVER (ORDER BY Year, Month), 0) AS MOM_Percentage_Change
from MonthlyCounts
)
--NULLIF - Prevents division by zero when the previous month s applications are 0.
--OVER- Clause defines the order in which rows are processed.
--LAG -Retrieves the value from a previous row in the result set
-- MOM = ((MTD-PMTD)/PMTD)

Select
Year,
Month,
Current_Month_Applications,
Previous_Month_Applications,
CAST(Round(MOM_Percentage_Change,3) as float) as MOM_Percentage_Change
--Cast is used to change one data type to another e.g CAST(column_name AS data_type)
--Round (round(expression,decimal_places)
from MOM
Order by Month ASC;
```



Year	Month	Current_Month_Applications	Previous_Month_Applications	MOM_Percentage_Change
2021	1	2332	NULL	NULL
2021	2	2279	2332	-2.273
2021	3	2627	2279	15.27
2021	4	2755	2627	4.872
2021	5	2911	2755	5.662
2021	6	3184	2911	9.378
2021	7	3366	3184	5.716
2021	8	3441	3366	2.228
2021	9	3536	3441	2.761
2021	10	3796	3536	7.353
2021	11	4035	3796	6.296
2021	12	4314	4035	6.914

5 Total Funded Amount

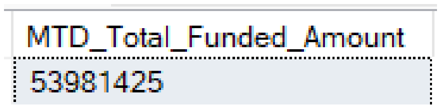
```
Select SUM(loan_amount) as Total_Funded_Amount from loan_data
```



```
Total_Funded_Amount
435757075
```

6 MTD Funded Amount

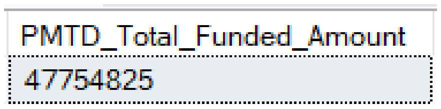
```
Select SUM(loan_amount) as MTD_Total_Funded_Amount from loan_data
where year(issue_date) = 2021
and month(issue_date) = 12
```



```
MTD_Total_Funded_Amount
53981425
```

7 PMTD Funded Amount

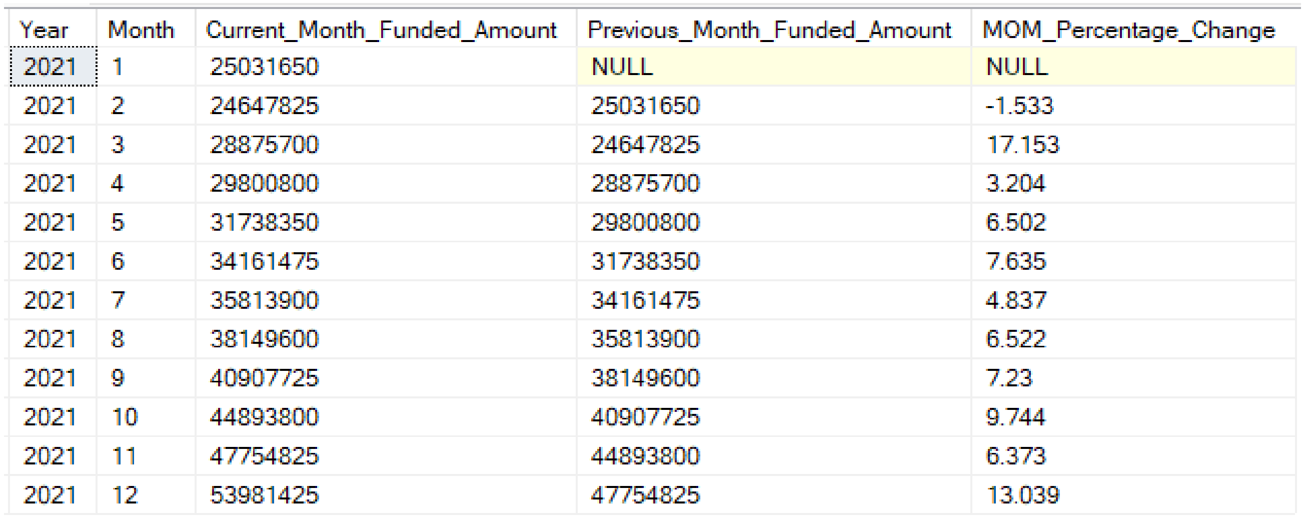
```
Select SUM(loan_amount) as PMTD_Total_Funded_Amount from loan_data
where year(issue_date) = 2021
and month(issue_date) = 11
```



```
PMTD_Total_Funded_Amount
47754825
```

8 MOM Funded Amount

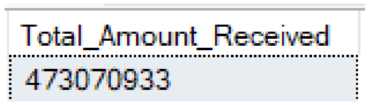
```sql
With Total_Funded_Amount as
(SELECT
Year(issue_date) AS Year,
Month(issue_date) AS Month,
SUM(loan_amount) AS Total_Funded_Amount
FROM loan_data
GROUP BY Year(issue_date), Month(issue_date)
),
--WITH is common table expression (CTE) stores data results temporarily

MOM AS(
SELECT
Year,
Month,
Total_Funded_Amount as Current_Month_Funded_Amount,
LAG(Total_Funded_Amount) OVER(ORDER BY Year, Month) as Previous_Month_Funded_Amount,
(Total_Funded_Amount-Lag(Total_Funded_Amount) OVER( ORDER BY Year, Month)) * 100.0/
NULLIF(LAG(Total_Funded_Amount) OVER (ORDER BY Year, Month), 0) AS MOM_Percentage_Change
from Total_Funded_Amount
)
--NULLIF - Prevents division by zero when the previous month s applications are 0.
--OVER- Clause defines the order in which rows are processed.
--LAG -Retrieves the value from a previous row in the result set
-- MOM = ((MTD-PMTD)/PMTD)

Select
Year,
Month,
Current_Month_Funded_Amount,
Previous_Month_Funded_Amount,
CAST(Round(MOM_Percentage_Change,3) as float) as MOM_Percentage_Change
--Cast is used to change one data type to another e.g CAST(column_name AS data_type)
--Round (round(expression,decimal_places)
FROM MOM
Order by Month ASC;
```



Year	Month	Current_Month_Funded_Amount	Previous_Month_Funded_Amount	MOM_Percentage_Change
2021	1	25031650	NULL	NULL
2021	2	24647825	25031650	-1.533
2021	3	28875700	24647825	17.153
2021	4	29800800	28875700	3.204
2021	5	31738350	29800800	6.502
2021	6	34161475	31738350	7.635
2021	7	35813900	34161475	4.837
2021	8	38149600	35813900	6.522
2021	9	40907725	38149600	7.23
2021	10	44893800	40907725	9.744
2021	11	47754825	44893800	6.373
2021	12	53981425	47754825	13.039

9 Total Amount Received

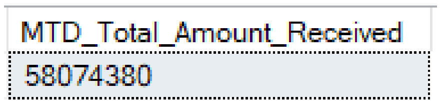
```
Select SUM(total_payment) as Total_Amount_Received from loan_data
```



```
Total_Amount_Received
473070933
```

10 MTD Total Amount Received

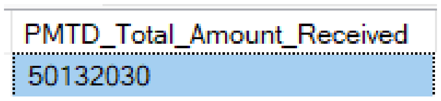
```
Select SUM(total_payment) as MTD_Total_Amount_Received from loan_data
where year(issue_date) = 2021
and month(issue_date) = 12
```



```
MTD_Total_Amount_Received
58074380
```

11 PMTD Total Amount Received

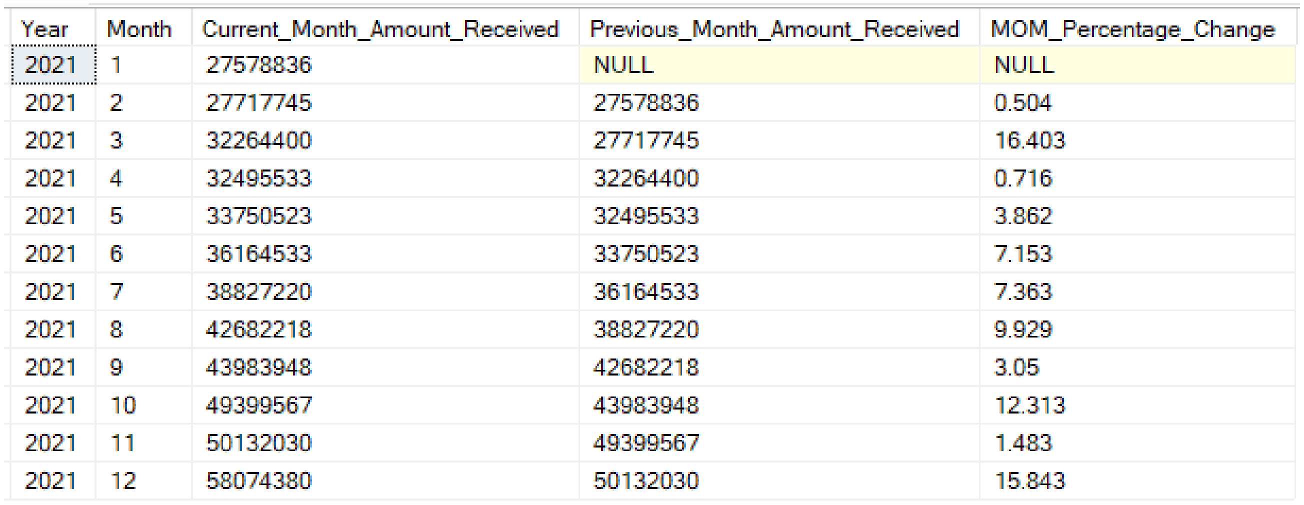
```
Select SUM(total_payment) as PMTD_Total_Amount_Received from loan_data
where year(issue_date) = 2021
and month(issue_date) = 11
```



```
PMTD_Total_Amount_Received
50132030
```

12 MOM Total Amount Received

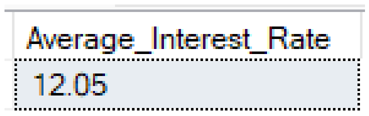
```
With Total_Amount_Received as
(SELECT
Year(issue_date) AS Year,
Month(issue_date) AS Month,
SUM(total_payment) AS Total_Amount_Received
FROM loan_data
GROUP BY Year(issue_date), Month(issue_date)
),
--WITH is common table expression (CTE) stores data results temporarily

MOM AS(
SELECT
Year,
Month,
Total_Amount_Received as Current_Month_Amount_Received,
LAG(Total_Amount_Received) OVER(ORDER BY Year, Month) as Previous_Month_Amount_Received,
(Total_Amount_Received-Lag(Total_Amount_Received) OVER( ORDER BY Year, Month)) * 100.0/
NULLIF(LAG(Total_Amount_Received) OVER (ORDER BY Year, Month), 0) AS MOM_Percentage_Change
from Total_Amount_Received
)
--NULLIF - Prevents division by zero when the previous months applications are 0.
--OVER- Clause defines the order in which rows are processed.
--LAG -Retrieves the value from a previous row in the result set
-- MOM = ((MTD-PMTD)/PMTD)

Select
Year,
Month,
Current_Month_Amount_Received,
Previous_Month_Amount_Received,
CAST(Round(MOM_Percentage_Change,3) as float) as MOM_Percentage_Change
--Cast is used to change one data type to another e.g CAST(column_name AS data_type)
--Round (round(expression,decimal_places)
FROM MOM
Order by Month ASC;
```



Year	Month	Current_Month_Amount_Received	Previous_Month_Amount_Received	MOM_Percentage_Change
2021	1	27578836	NULL	NULL
2021	2	27717745	27578836	0.504
2021	3	32264400	27717745	16.403
2021	4	32495533	32264400	0.716
2021	5	33750523	32495533	3.862
2021	6	36164533	33750523	7.153
2021	7	38827220	36164533	7.363
2021	8	42682218	38827220	9.929
2021	9	43983948	42682218	3.05
2021	10	49399567	43983948	12.313
2021	11	50132030	49399567	1.483
2021	12	58074380	50132030	15.843

13 Average Interest Rate

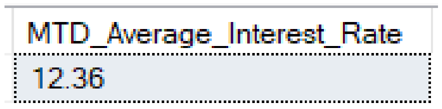
```
Select ROUND(AVG(int_rate)*100.0,2) as Average_Interest_Rate from loan_data
```



Average_Interest_Rate
12.05

14 MTD Average Interest Rate

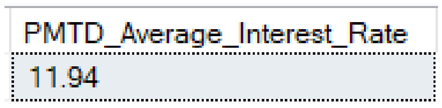
```
Select ROUND(AVG(int_rate)*100.0,2)as MTD_Average_Interest_Rate from loan_data
where year(issue_date) = 2021
and month(issue_date) = 12
```



MTD_Average_Interest_Rate
12.36

15 PMTD Average Interest Rate

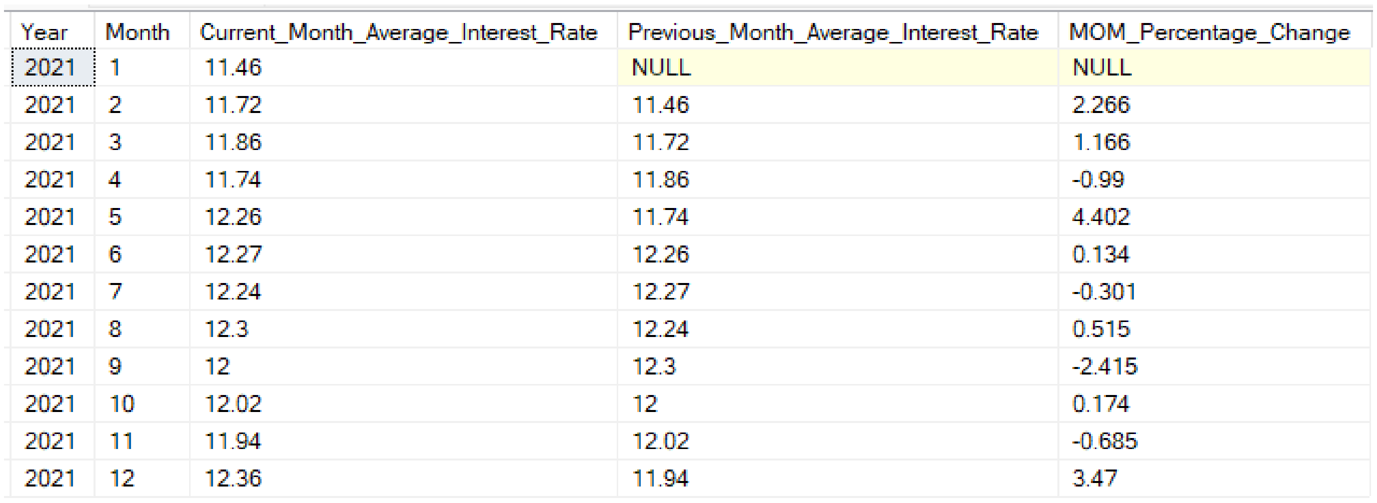
```
Select ROUND(AVG(int_rate)*100.0,2)as PMTD_Average_Interest_Rate from loan_data
where year(issue_date) = 2021
and month(issue_date) = 11
```



PMTD_Average_Interest_Rate
11.94

16 MOM Average Interest Rate

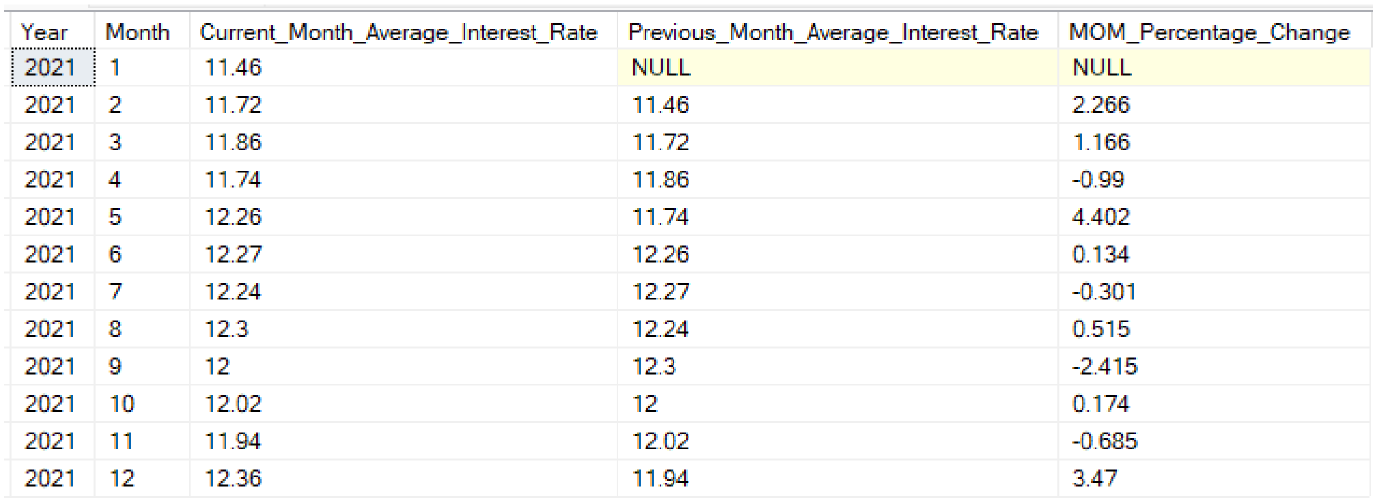
```sql
With Average_Interest_Rate as
(SELECT
Year(issue_date) AS Year,
Month(issue_date) AS Month,
AVG(int_rate)*100.0 AS Average_Interest_Rate
FROM loan_data
GROUP BY Year(issue_date), Month(issue_date)
),
--WITH is common table expression (CTE) stores data results temporarily

MOM AS(
SELECT
Year,
Month,
Average_Interest_Rate as Current_Month_Average_Interest_Rate,
LAG(Average_Interest_Rate) OVER(ORDER BY Year, Month) as Previous_Month_Average_Interest_Rate,
(Average_Interest_Rate-Lag(Average_Interest_Rate) OVER( ORDER BY Year, Month)) * 100.0/
NULLIF(LAG(Average_Interest_Rate) OVER (ORDER BY Year, Month), 0) AS MOM_Percentage_Change
from Average_Interest_Rate
)
--NULLIF - Prevents division by zero when the previous months applications are 0.
--OVER- Clause defines the order in which rows are processed.
--LAG -Retrieves the value from a previous row in the result set
-- MOM = ((MTD-PMTD)/PMTD)

Select
Year,
Month,
ROUND(Current_Month_Average_Interest_Rate,2) as Current_Month_Average_Interest_Rate,
ROUND(Previous_Month_Average_Interest_Rate, 2) as Previous_Month_Average_Interest_Rate,
CAST(Round(MOM_Percentage_Change,3) as float) as MOM_Percentage_Change
--Cast is used to change one data type to another e.g CAST(column_name AS data_type)
--Round (round(expression,decimal_places)
FROM MOM
Order by Month ASC;
```



Year	Month	Current_Month_Average_Interest_Rate	Previous_Month_Average_Interest_Rate	MOM_Percentage_Change
2021	1	11.46	NULL	NULL
2021	2	11.72	11.46	2.266
2021	3	11.86	11.72	1.166
2021	4	11.74	11.86	-0.99
2021	5	12.26	11.74	4.402
2021	6	12.27	12.26	0.134
2021	7	12.24	12.27	-0.301
2021	8	12.3	12.24	0.515
2021	9	12	12.3	-2.415
2021	10	12.02	12	0.174
2021	11	11.94	12.02	-0.685
2021	12	12.36	11.94	3.47

17 MOM Average Interest Rate

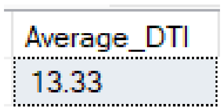
```sql
With Average_Interest_Rate as
(SELECT
Year(issue_date) AS Year,
Month(issue_date) AS Month,
AVG(int_rate)*100.0 AS Average_Interest_Rate
FROM loan_data
GROUP BY Year(issue_date), Month(issue_date)
),
--WITH is common table expression (CTE) stores data results temporarily

MOM AS(
SELECT
Year,
Month,
Average_Interest_Rate as Current_Month_Average_Interest_Rate,
LAG(Average_Interest_Rate) OVER(ORDER BY Year, Month) as Previous_Month_Average_Interest_Rate,
(Average_Interest_Rate-Lag(Average_Interest_Rate) OVER( ORDER BY Year, Month)) * 100.0/
NULLIF(LAG(Average_Interest_Rate) OVER (ORDER BY Year, Month), 0) AS MOM_Percentage_Change
from Average_Interest_Rate
)
--NULLIF - Prevents division by zero when the previous month s applications are 0.
--OVER- Clause defines the order in which rows are processed.
--LAG -Retrieves the value from a previous row in the result set
-- MOM = ((MTD-PMTD)/PMTD)

Select
Year,
Month,
ROUND(Current_Month_Average_Interest_Rate,2) as Current_Month_Average_Interest_Rate,
ROUND(Previous_Month_Average_Interest_Rate, 2) as Previous_Month_Average_Interest_Rate,
CAST(Round(MOM_Percentage_Change,3) as float) as MOM_Percentage_Change
--Cast is used to change one data type to another e.g CAST(column_name AS data_type)
--Round (round(expression,decimal_places)
FROM MOM
Order by Month ASC;
```



Year	Month	Current_Month_Average_Interest_Rate	Previous_Month_Average_Interest_Rate	MOM_Percentage_Change
2021	1	11.46	NULL	NULL
2021	2	11.72	11.46	2.266
2021	3	11.86	11.72	1.166
2021	4	11.74	11.86	-0.99
2021	5	12.26	11.74	4.402
2021	6	12.27	12.26	0.134
2021	7	12.24	12.27	-0.301
2021	8	12.3	12.24	0.515
2021	9	12	12.3	-2.415
2021	10	12.02	12	0.174
2021	11	11.94	12.02	-0.685
2021	12	12.36	11.94	3.47

18 Average Debit-to-Income Ratio (DTI)

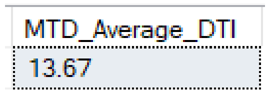
```
Select ROUND(AVG(dti)*100.0,2) as Average_DTI from loan_data
```



Average_DTI
13.33

19 MTD Average Debit-to-Income Ratio (DTI)

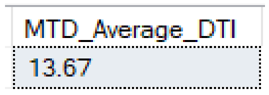
```
--Calculate Average Interest Rate for December Month
Select ROUND(AVG(dti)*100.0,2)as MTD_Average_DTI from loan_data
where year(issue_date) = 2021
and month(issue_date) = 12
```



MTD_Average_DTI
13.67

20 PMTD Average Debit-to-Income Ratio (DTI)

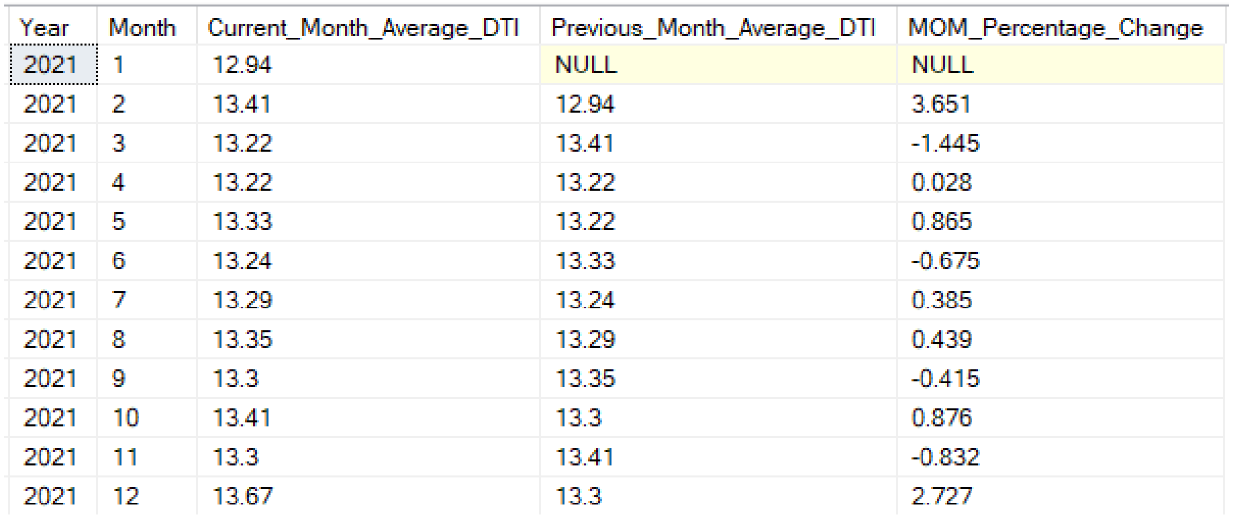
```
--Calculate Average Interest Rate for November Month
Select ROUND(AVG(dti)*100.0,2)as PMTD_Average_DTI from loan_data
where year(issue_date) = 2021
and month(issue_date) = 11
```



MTD_Average_DTI
13.67

21 MOM Average Debit-to-Income Ratio (DTI)

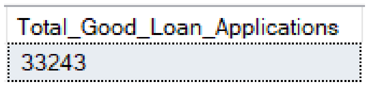
```sql
--Calculate MOM Average DTI
With Average_DTI as
(SELECT
Year(issue_date) AS Year,
Month(issue_date) AS Month,
AVG(dti)*100.0 AS Average_DTI
FROM loan_data
GROUP BY Year(issue_date), Month(issue_date)
),
--WITH is common table expression (CTE) stores data results temporarily

MOM AS(
SELECT
Year,
Month,
Average_DTI as Current_Month_Average_DTI,
LAG(Average_DTI) OVER(ORDER BY Year, Month) as Previous_Month_Average_DTI,
(Average_DTI-Lag(Average_DTI) OVER( ORDER BY Year, Month)) * 100.0/
NULLIF(LAG(Average_DTI) OVER (ORDER BY Year, Month), 0) AS MOM_Percentage_Change
from Average_DTI
)
--NULLIF - Prevents division by zero when the previous months applications are 0.
--OVER- Clause defines the order in which rows are processed.
--LAG -Retrieves the value from a previous row in the result set
-- MOM = ((MTD-PMTD)/PMTD)

Select
Year,
Month,
ROUND(Current_Month_Average_DTI,2) as Current_Month_Average_DTI,
ROUND(Previous_Month_Average_DTI, 2) as Previous_Month_Average_DTI,
CAST(Round(MOM_Percentage_Change,3) as float) as MOM_Percentage_Change
--Cast is used to change one data type to another e.g CAST(column_name AS data_type)
--Round (round(expression,decimal_places)
FROM MOM
Order by Month ASC;
```



Year	Month	Current_Month_Average_DTI	Previous_Month_Average_DTI	MOM_Percentage_Change
2021	1	12.94	NULL	NULL
2021	2	13.41	12.94	3.651
2021	3	13.22	13.41	-1.445
2021	4	13.22	13.22	0.028
2021	5	13.33	13.22	0.865
2021	6	13.24	13.33	-0.675
2021	7	13.29	13.24	0.385
2021	8	13.35	13.29	0.439
2021	9	13.3	13.35	-0.415
2021	10	13.41	13.3	0.876
2021	11	13.3	13.41	-0.832
2021	12	13.67	13.3	2.727

22 Good Loan Applications

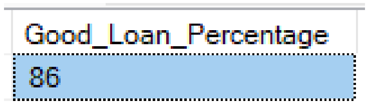
```
--Good loan - means fully paid and current
--Bad loan - means charged off

---Classification as Good loan , Bad loan
WITH Loan_Classification AS (SELECT
  id,
  loan_amount,
  loan_status,
  CASE
    WHEN loan_status IN ('Charged␣Off') THEN 'Bad␣Loan'
    WHEN loan_status IN ('Fully␣Paid', 'Current') THEN 'Good␣Loan'
  END AS Loan_Classification
FROM loan_data
)
--Calculate Total Good Loan Applications
Select count(*) as Total_Good_Loan_Applications from Loan_Classification
```



```
Total_Good_Loan_Applications
33243
```

23 Good Loan Applications Percentage

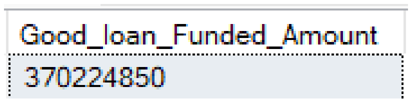
```
--Calculate Total Good Loan Applications Percentage
Select
(COUNT(CASE WHEN loan_status ='Fully␣Paid' or loan_status ='Current' Then id End)*100)/Count(
    id) AS Good_Loan_Percentage
```



```
Good_Loan_Percentage
86
```

24 Good Loan Funded Amount

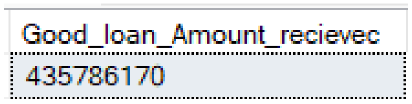
```
SELECT(
SUM(CASE WHEN loan_status = 'Fully␣Paid' or loan_status = 'Current' THEN loan_amount END)) AS
    Good_loan_Funded_Amount
FROM loan_data
Go
```



```
Good_loan_Funded_Amount
370224850
```

25 Good Loan Amount Recieved

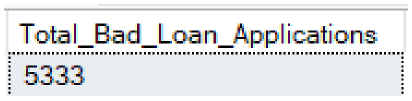
```
SELECT(
SUM(CASE WHEN loan_status = 'Fully␣Paid' or loan_status = 'Current' THEN total_payment END))
    AS Good_loan_Amount_recievec
FROM loan_data
Go
```



```
Good_loan_Amount_recievec
435786170
```

26 Bad Loan Applications

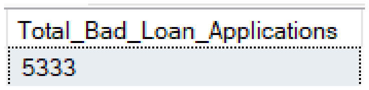
```
--Good loan - means fully paid and current
--Bad loan - means charged off

---Classification as Good loan , Bad loan
WITH Loan_Classification AS (SELECT
  id,
  loan_amount,
  loan_status,
  CASE
    WHEN loan_status IN ('Charged␣Off') THEN 'Bad␣Loan'
    WHEN loan_status IN ('Fully␣Paid', 'Current') THEN 'Good␣Loan'
  END AS Loan_Classification
FROM loan_data
)
--Calculate Total Bad Loan Applications
Select count(*) as Total_Bad_Loan_Applications from Loan_Classification
Where Loan_Classification = 'Bad␣Loan'
```



```
Total_Bad_Loan_Applications
5333
```

27 Bad Loan Applications Percentage

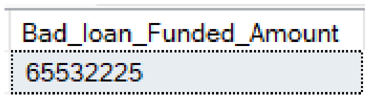
```
--Calculate Total Bad Loan Applications Percentage
Select
(COUNT(CASE WHEN loan_status ='Charged␣Off' Then id End)*100)/Count(id) AS Bad_Loan_Percentage
FROM loan_data
```



```
Total_Bad_Loan_Applications
5333
```

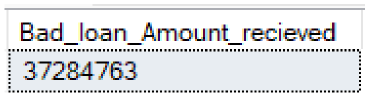
28 Bad Loan Funded Amount

```
--Calculate Total Bad Loan Applications Percentage
Select
(COUNT(CASE WHEN loan_status ='Charged␣Off' Then id End)*100)/Count(id) AS Bad_Loan_Percentage
FROM loan_data
```


Bad_loan_Funded_Amount
65532225

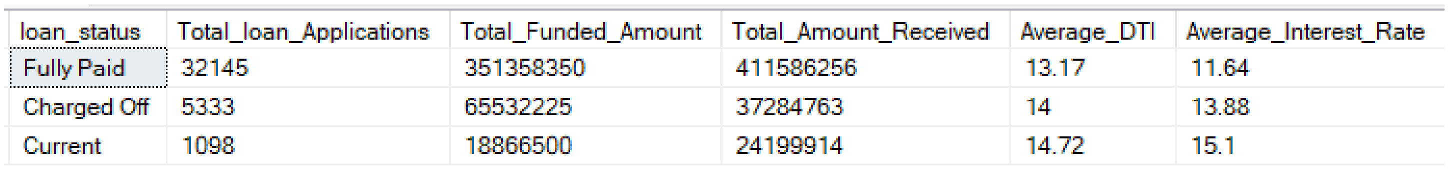
29 Bad Loan Amount Recieved

```
--Calculate Bad Loan Total Amount recieved
SELECT(
SUM(CASE WHEN loan_status = 'Charged␣Off' THEN total_payment END)) AS Bad_loan_Amount_recievec
FROM loan_data
```


Bad_loan_Amount_recieved
37284763

30 Loan Status Grid View

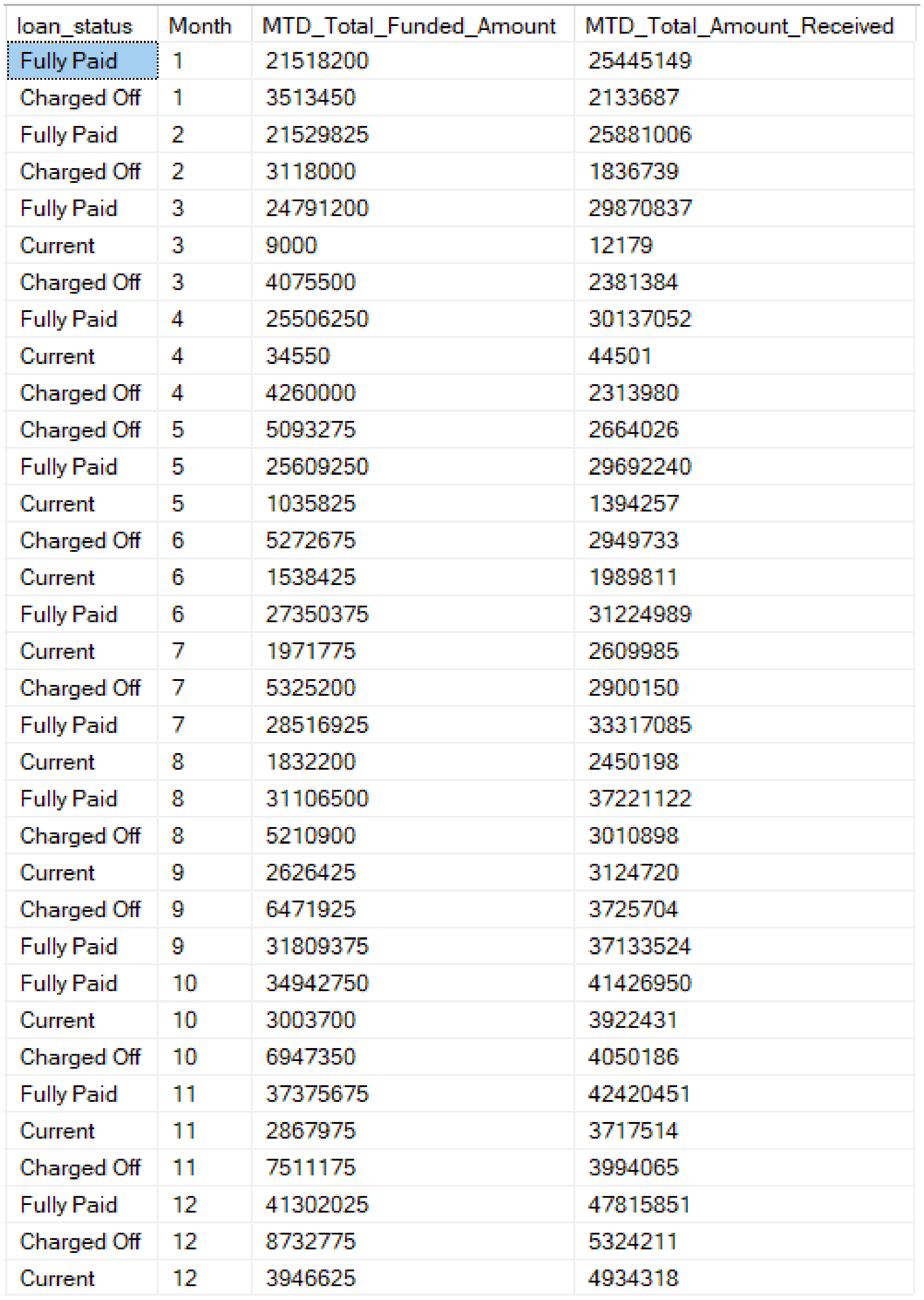
```
SELECT
loan_status,
count(id) as Total_loan_Applications,
SUM(loan_amount) AS Total_Funded_Amount,
SUM(total_payment) AS Total_Amount_Received,
ROUND(AVG(dti)*100.0,2) as Average_DTI ,
ROUND(AVG(int_rate)*100.0,2) as Average_Interest_Rate
from loan_data
GROUP BY loan_status
```



loan_status	Total_loan_Applications	Total_Funded_Amount	Total_Amount_Received	Average_DTI	Average_Interest_Rate
Fully Paid	32145	351358350	411586256	13.17	11.64
Charged Off	5333	65532225	37284763	14	13.88
Current	1098	18866500	24199914	14.72	15.1

31 MTD Funded Amount and Amount Recieved based on Loan Status

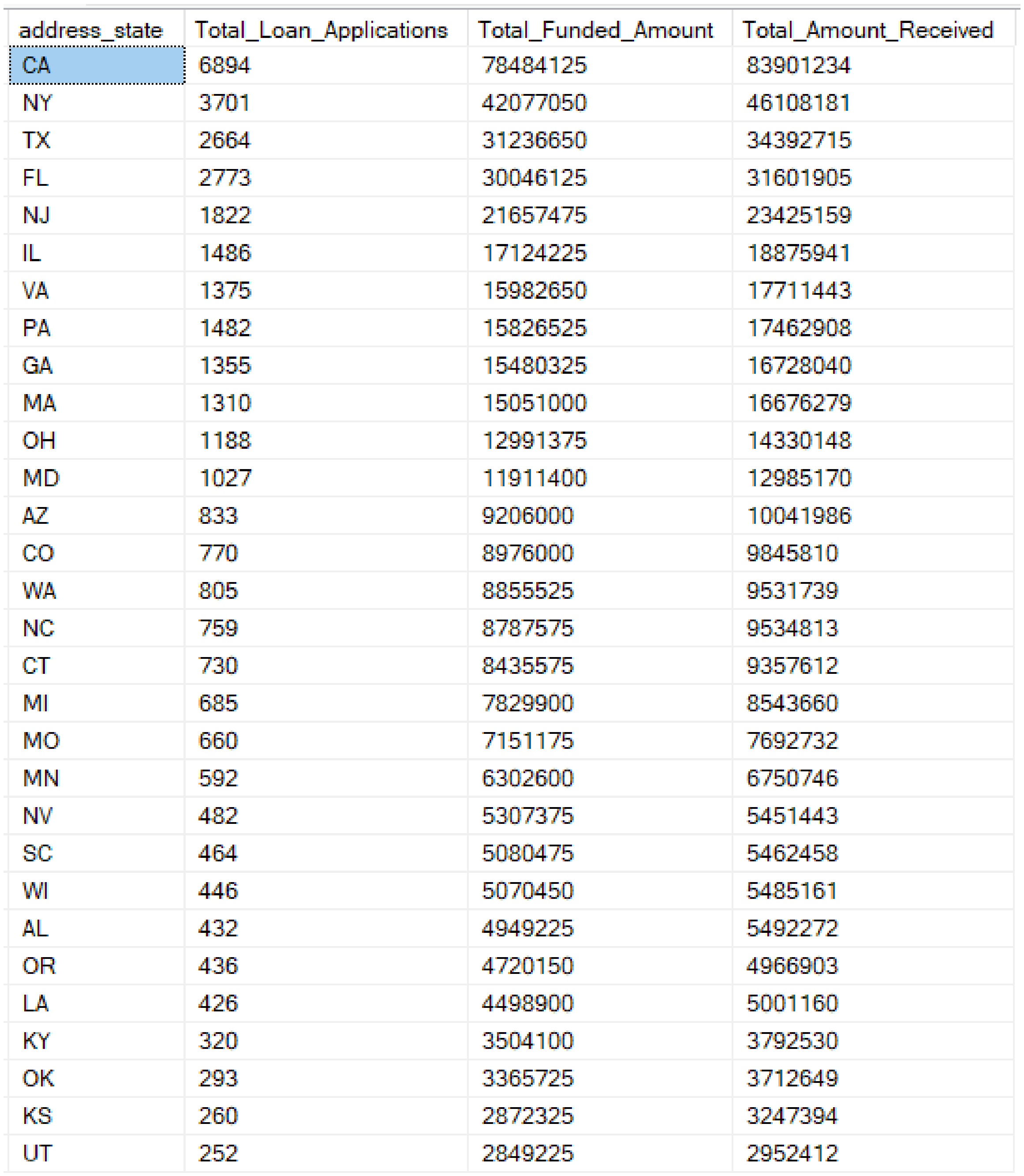
```
--Calculate MTD Funded Amount and Amount recieved
SELECT
  loan_status,
  MONTH(issue_date) AS Month,
  SUM(loan_amount) AS MTD_Total_Funded_Amount,
  SUM(total_payment) AS MTD_Total_Amount_Received
FROM loan_data
GROUP BY loan_status, MONTH(issue_date)
ORDER BY Month ASC;
```



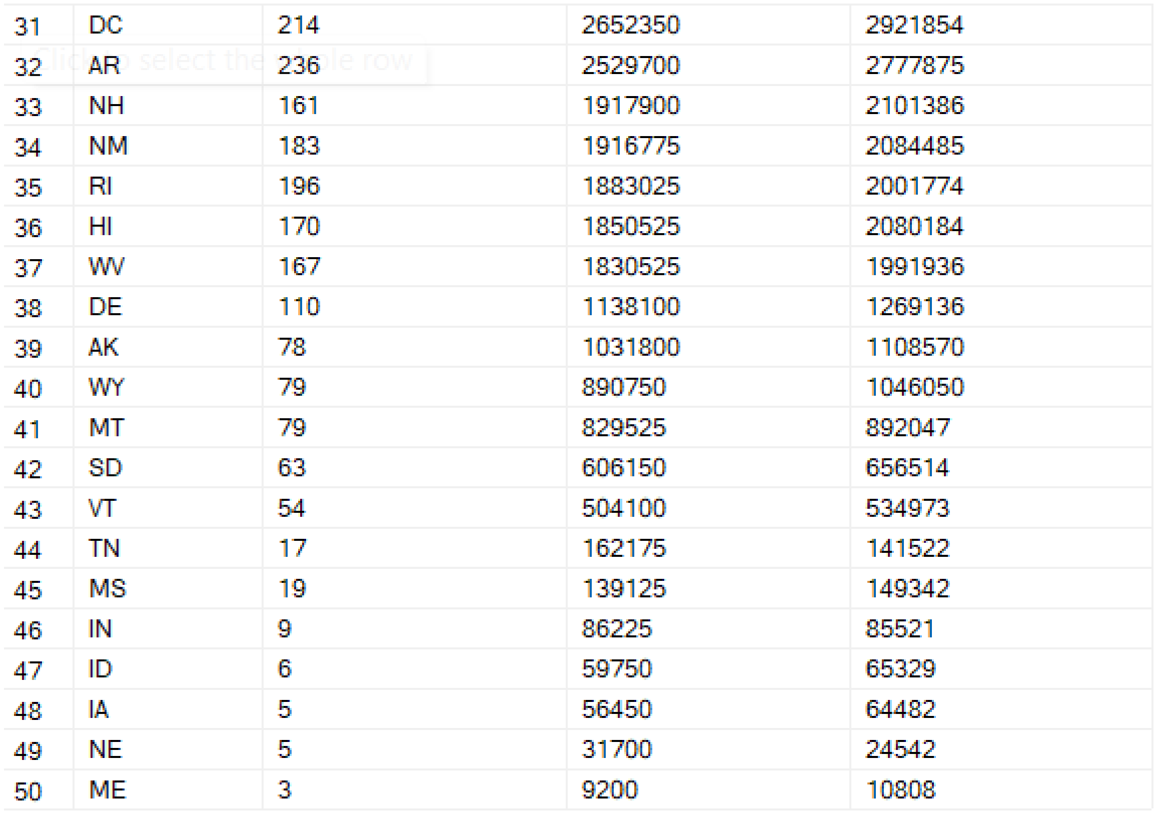
loan_status	Month	MTD_Total_Funded_Amount	MTD_Total_Amount_Received
Fully Paid	1	21518200	25445149
Charged Off	1	3513450	2133687
Fully Paid	2	21529825	25881006
Charged Off	2	3118000	1836739
Fully Paid	3	24791200	29870837
Current	3	9000	12179
Charged Off	3	4075500	2381384
Fully Paid	4	25506250	30137052
Current	4	34550	44501
Charged Off	4	4260000	2313980
Charged Off	5	5093275	2664026
Fully Paid	5	25609250	29692240
Current	5	1035825	1394257
Charged Off	6	5272675	2949733
Current	6	1538425	1989811
Fully Paid	6	27350375	31224989
Current	7	1971775	2609985
Charged Off	7	5325200	2900150
Fully Paid	7	28516925	33317085
Current	8	1832200	2450198
Fully Paid	8	31106500	37221122
Charged Off	8	5210900	3010898
Current	9	2626425	3124720
Charged Off	9	6471925	3725704
Fully Paid	9	31809375	37133524
Fully Paid	10	34942750	41426950
Current	10	3003700	3922431
Charged Off	10	6947350	4050186
Fully Paid	11	37375675	42420451
Current	11	2867975	3717514
Charged Off	11	7511175	3994065
Fully Paid	12	41302025	47815851
Charged Off	12	8732775	5324211
Current	12	3946625	4934318

32 Regional Analysis by State

```
SELECT
  address_state,
  COUNT(id) AS Total_Loan_Applications,
  SUM(loan_amount) AS Total_Funded_Amount,
  SUM(total_payment) AS Total_Amount_Received
FROM loan_data
GROUP BY address_state
ORDER BY SUM(loan_amount) DESC;
```



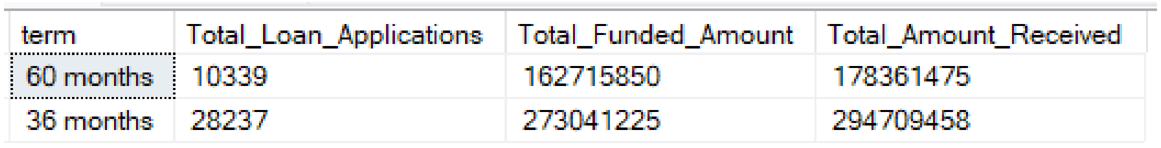
address_state	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
CA	6894	78484125	83901234
NY	3701	42077050	46108181
TX	2664	31236650	34392715
FL	2773	30046125	31601905
NJ	1822	21657475	23425159
IL	1486	17124225	18875941
VA	1375	15982650	17711443
PA	1482	15826525	17462908
GA	1355	15480325	16728040
MA	1310	15051000	16676279
OH	1188	12991375	14330148
MD	1027	11911400	12985170
AZ	833	9206000	10041986
CO	770	8976000	9845810
WA	805	8855525	9531739
NC	759	8787575	9534813
CT	730	8435575	9357612
MI	685	7829900	8543660
MO	660	7151175	7692732
MN	592	6302600	6750746
NV	482	5307375	5451443
SC	464	5080475	5462458
WI	446	5070450	5485161
AL	432	4949225	5492272
OR	436	4720150	4966903
LA	426	4498900	5001160
KY	320	3504100	3792530
OK	293	3365725	3712649
KS	260	2872325	3247394
UT	252	2849225	2952412



31	DC	214	2652350	2921854
32	AR	236	2529700	2777875
33	NH	161	1917900	2101386
34	NM	183	1916775	2084485
35	RI	196	1883025	2001774
36	HI	170	1850525	2080184
37	WV	167	1830525	1991936
38	DE	110	1138100	1269136
39	AK	78	1031800	1108570
40	WY	79	890750	1046050
41	MT	79	829525	892047
42	SD	63	606150	656514
43	VT	54	504100	534973
44	TN	17	162175	141522
45	MS	19	139125	149342
46	IN	9	86225	85521
47	ID	6	59750	65329
48	IA	5	56450	64482
49	NE	5	31700	24542
50	ME	3	9200	10808

33 Loan Term Analysis

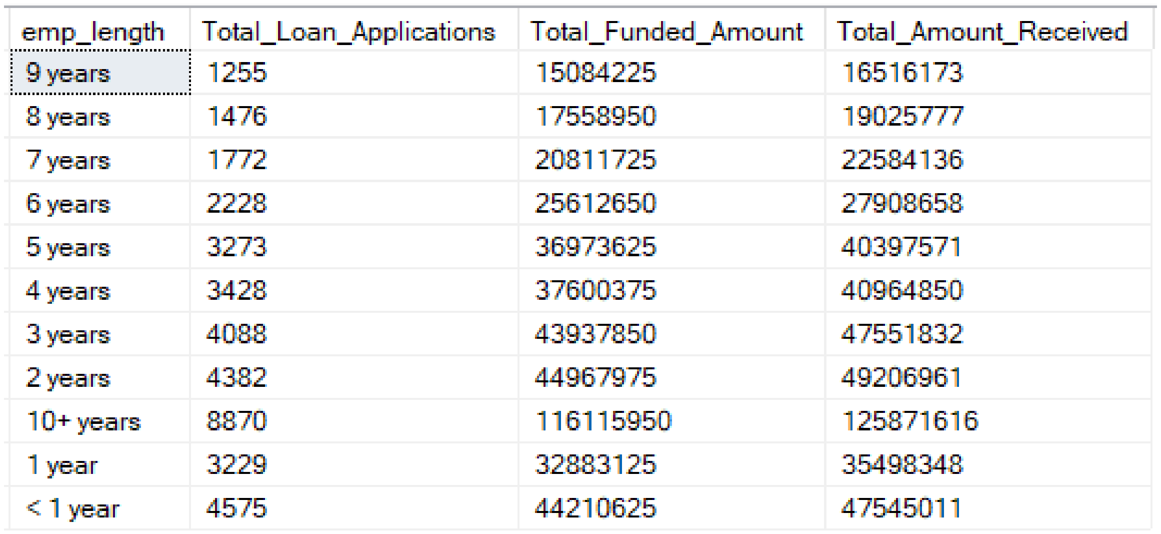
```sql
SELECT
  term,
  COUNT(id) AS Total_Loan_Applications,
  SUM(loan_amount) AS Total_Funded_Amount,
  SUM(total_payment) AS Total_Amount_Received
FROM loan_data
GROUP BY term
ORDER BY term DESC;
```



term	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
60 months	10339	162715850	178361475
36 months	28237	273041225	294709458

34 Employee length Analysis

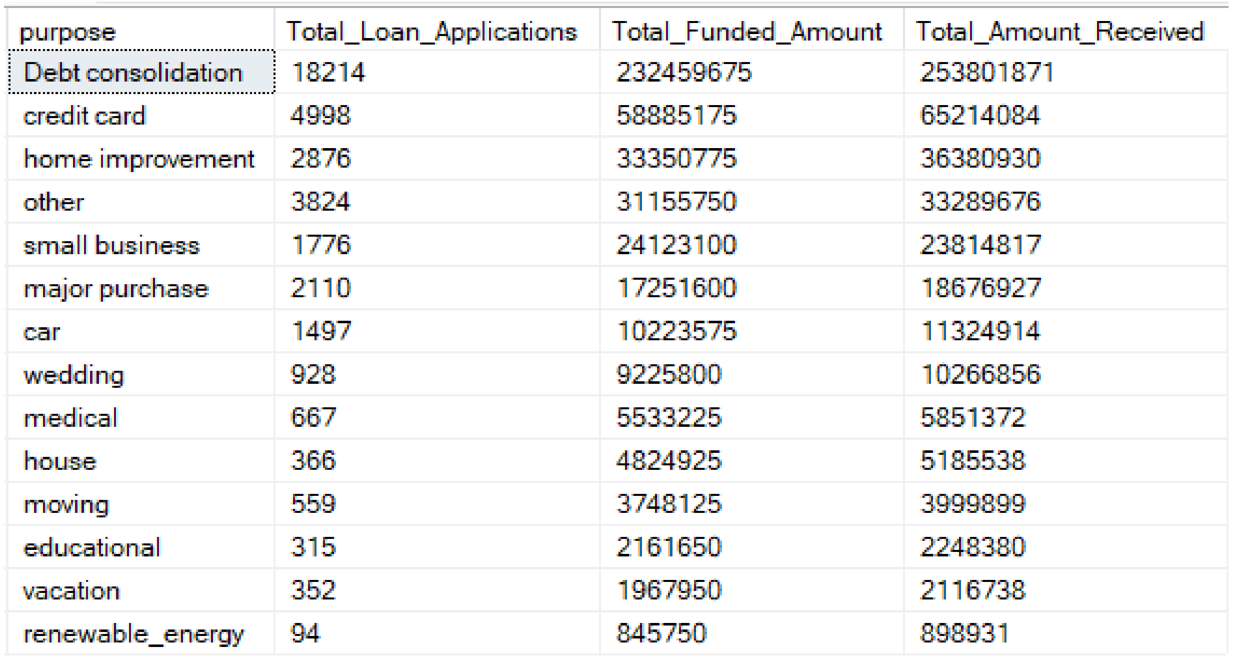
```sql
SELECT
  emp_length,
  COUNT(id) AS Total_Loan_Applications,
  SUM(loan_amount) AS Total_Funded_Amount,
  SUM(total_payment) AS Total_Amount_Received
FROM loan_data
GROUP BY emp_length
ORDER BY emp_length DESC;
```



emp_length	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
9 years	1255	15084225	16516173
8 years	1476	17558950	19025777
7 years	1772	20811725	22584136
6 years	2228	25612650	27908658
5 years	3273	36973625	40397571
4 years	3428	37600375	40964850
3 years	4088	43937850	47551832
2 years	4382	44967975	49206961
10+ years	8870	116115950	125871616
1 year	3229	32883125	35498348
< 1 year	4575	44210625	47545011

35 Loan Purpose Breakdown

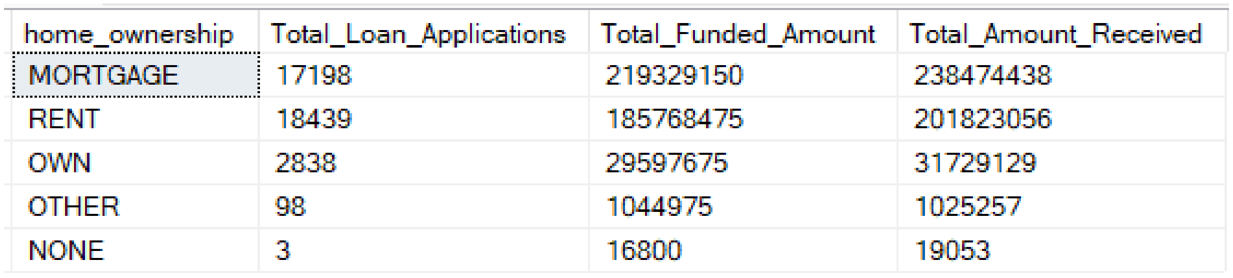
```sql
SELECT
  purpose,
  COUNT(id) AS Total_Loan_Applications,
  SUM(loan_amount) AS Total_Funded_Amount,
  SUM(total_payment) AS Total_Amount_Received
FROM loan_data
GROUP BY purpose
ORDER BY SUM(loan_amount) DESC;
```



purpose	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
Debt consolidation	18214	232459675	253801871
credit card	4998	58885175	65214084
home improvement	2876	33350775	36380930
other	3824	31155750	33289676
small business	1776	24123100	23814817
major purchase	2110	17251600	18676927
car	1497	10223575	11324914
wedding	928	9225800	10266856
medical	667	5533225	5851372
house	366	4824925	5185538
moving	559	3748125	3999899
educational	315	2161650	2248380
vacation	352	1967950	2116738
renewable_energy	94	845750	898931

36 Home Ownership Analysis

```sql
SELECT
  home_ownership,
  COUNT(id) AS Total_Loan_Applications,
  SUM(loan_amount) AS Total_Funded_Amount,
  SUM(total_payment) AS Total_Amount_Received
FROM loan_data
GROUP BY home_ownership
ORDER BY SUM(loan_amount) DESC;
```



home_ownership	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
MORTGAGE	17198	219329150	238474438
RENT	18439	185768475	201823056
OWN	2838	29597675	31729129
OTHER	98	1044975	1025257
NONE	3	16800	19053