boras-auto-finance-eda

February 23, 2026

0.1 Data Cleaning and Exploratory Data Analysis

0.2 Data Overview :

---

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
data_path = r"D:/Automotive_Loans_dataset/"

files = {
    "branches": "branch_master.csv",
    "brokers" : "broker_master.csv",
    "contracts" : "contracts_master.csv",
    "dealers" : "dealer_master.csv",
    "executives" : "executive_master.csv",
    "repayment" : "repayment_master.csv"
}

monthly_dpds = {}

for name,file in files.items():
    monthly_dpds[name] = pd.read_csv(data_path + file)



branches = monthly_dpds["branches"]
brokers = monthly_dpds["brokers"]
contracts = monthly_dpds["contracts"]
dealers = monthly_dpds["dealers"]
executives = monthly_dpds["executives"]
repayment = monthly_dpds["repayment"]
```

[22]:
```
print(contracts.head(2),"\n")
print(contracts.info(),"\n")
```

```
print(contracts.shape,"\n")
```

```
      loan_id customer_name        asset_name  invoice_value  loan_amount  \
0  1000000001    Adira Loke  Mahindra Scorpio        1060000       870000
1  1000000002    Jivika Som  Mahindra Scorpio        1650000      1320000

   emi_amount  tenure_months    irr loan_issue_date loan_status  \
0       20123             60  13.73      01-05-2023           L
1       46496             36  16.14      18-11-2023           L

  reg_payment_mode psl_tag rc_verified  dealer_id  broker_id  executive_id  \
0              ECS       N           N        NaN        NaN      27030005
1              ECS       N           Y        NaN        NaN      27030006

   branch_id  manager_id
0       1006    27020412
1       1007    27020412

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9436 entries, 0 to 9435
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   loan_id           9436 non-null   int64
 1   customer_name     9152 non-null   object
 2   asset_name        9436 non-null   object
 3   invoice_value     9436 non-null   int64
 4   loan_amount       9436 non-null   int64
 5   emi_amount        9436 non-null   int64
 6   tenure_months     9436 non-null   int64
 7   irr               9436 non-null   float64
 8   loan_issue_date   9436 non-null   object
 9   loan_status       9436 non-null   object
 10  reg_payment_mode  8511 non-null   object
 11  psl_tag           8950 non-null   object
 12  rc_verified       9169 non-null   object
 13  dealer_id         3327 non-null   float64
 14  broker_id         2340 non-null   float64
 15  executive_id      9436 non-null   int64
 16  branch_id         9436 non-null   int64
 17  manager_id        9436 non-null   int64
dtypes: float64(3), int64(8), object(7)
memory usage: 1.3+ MB
None

(9436, 18)
```

[3]:
```
print(repayment.head(2),"\n")
print(repayment.info(),"\n")
print(repayment.shape,"\n")
```

```
      loan_id    due_date  emi_amount  interest_component  \
0  1000000001  2023-05-05     20123.0             9954.25
1  1000000001  2023-06-05     20123.0             9837.90

   principal_component  due_amount  amount_received  overdue_amount  dpd_days  \
0             10168.75     20123.0          20123.0             0.0         0
1             10285.10     20123.0          20123.0             0.0         0

   balance_amount  branch_id  manager_id
0       859831.25       1006    27020412
1       849546.15       1006    27020412

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113232 entries, 0 to 113231
Data columns (total 12 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   loan_id              113232 non-null  int64
 1   due_date             113232 non-null  object
 2   emi_amount           113232 non-null  float64
 3   interest_component   113232 non-null  float64
 4   principal_component  113232 non-null  float64
 5   due_amount           113232 non-null  float64
 6   amount_received      113232 non-null  float64
 7   overdue_amount       113232 non-null  float64
 8   dpd_days             113232 non-null  int64
 9   balance_amount       113232 non-null  float64
 10  branch_id            113232 non-null  int64
 11  manager_id           113232 non-null  int64
dtypes: float64(7), int64(4), object(1)
memory usage: 10.4+ MB
None

(113232, 12)
```

[4]:
```
print(branches.head(2),"\n")
print(branches.info(),"\n")
print(branches.shape,"\n")
```

```
   branch_id    branch_name        state  executive_id  executive_name  \
0       1001  Mumbai Branch  Maharashtra      27030000  Seher Kulkarni
1       1002    Pune Branch  Maharashtra      27030001   Armaan Bhatia
```

```
   manager_id       manager_name
0    27020412  Dharmajan Dhillon
1    27020412  Dharmajan Dhillon

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74 entries, 0 to 73
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   branch_id       74 non-null     int64
 1   branch_name     74 non-null     object
 2   state           74 non-null     object
 3   executive_id    74 non-null     int64
 4   executive_name  74 non-null     object
 5   manager_id      74 non-null     int64
 6   manager_name    74 non-null     object
dtypes: int64(3), object(4)
memory usage: 4.2+ KB
None

(74, 7)
```

[24]:
```
print(brokers.head(),"\n")
print(brokers.info(),"\n")
print(brokers.shape,"\n")
```

```
   broker_id     broker_name  manager_id       manager_name
0     600000      Aarav Sama    27020209         Badal Kala
1     600001       Ryan Gala    27020772        Vanya Sinha
2     600002  Darshit Suresh    27020666        Vanya Sinha
3     600003    Raunak Kakar    27020666          Anvi Dora
4     600004       Alia Suri    27020772  Dharmajan Dhillon

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   broker_id     200 non-null    int64
 1   broker_name   200 non-null    object
 2   manager_id    200 non-null    int64
 3   manager_name  200 non-null    object
dtypes: int64(2), object(2)
memory usage: 6.4+ KB
None

(200, 4)
```

[5]:
```
print(dealers.head(),"\n")
print(dealers.info(),"\n")
print(dealers.shape,"\n")
```

```
   dealer_id                          dealer_name  manager_id  manager_name
0     500000                        Bora Ltd Auto    27020562  Uthkarsh Roy
1     500001    Ghose, Dash and Brahmbhatt Motors    27020415  Drishya Raja
2     500002                       Vohra Ltd Cars    27020415  Uthkarsh Roy
3     500003  Srinivas, Shetty and Chaudhary Cars    27020047  Uthkarsh Roy
4     500004                       Madan LLC Cars    27020412  Purab Sachar

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   dealer_id     300 non-null    int64
 1   dealer_name   300 non-null    object
 2   manager_id    300 non-null    int64
 3   manager_name  300 non-null    object
dtypes: int64(2), object(2)
memory usage: 9.5+ KB
None

(300, 4)
```

[25]:
```
print(executives.head(),"\n")
print(executives.info(),"\n")
print(executives.shape)
```

```
   executive_id  executive_name  branch_id  manager_id
0      27030000  Seher Kulkarni       1001    27020412
1      27030001   Armaan Bhatia       1002    27020412
2      27030002   Umang Sridhar       1003    27020412
3      27030003      Anya Raval       1004    27020412
4      27030004     Dhanush Dey       1005    27020412

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74 entries, 0 to 73
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   executive_id    74 non-null     int64
 1   executive_name  74 non-null     object
 2   branch_id       74 non-null     int64
```

```
 3   manager_id      74 non-null     int64
dtypes: int64(3), object(1)
memory usage: 2.4+ KB
None

(74, 4)
```

Key Understanding:

- 1. There are 9436 contracts records in Contracts table.
- 2. There are 113232 records in Repayments table.
- 3. There are 300 records in Dealer table.
- 4. There are 200 records in Broker table.
- 5. There are 74 records in Branch table.
- 6. There are 74 records in Executive table.
- 7. It is evident from the column wise value counts that most missing values are present in the Contracts table.
- 8. We understand that Contracts table has complete customer details and repayment table contains the loan wise repayment records.

0.2.1 Finding & handling missing values.

[ ]:
```
## Missing values Assessment.

def summarize_nulls(df,name):

    """ The given function calculates total null values for each column
        of a table and provides us the summary only for columns that contain␣
 ↪null values.
    """

    print(f"{'-'*55}\nSummarized Null Value report for : {name} Table.
 ↪\n{'-'*55}")
    null_count = df.isnull().sum()
    null_percent = (null_count/len(df))*100

    summary = pd.DataFrame(
        {
            'column_name' : null_count.index,
            'null_count': null_count.values,
```

```
            'null_percent': null_percent.round(2).astype(str)+"%",
        }
    )

    if (summary['null_count'].sum() > 0 ) :

        return summary[summary['null_count']>0].reset_index( drop = True )

    else:

        print(f"No Null Values found in {name} Table.")
```

[259]:
```
print(summarize_nulls(contracts,"Contracts"))
print(summarize_nulls(repayment,"Repayment"))
```

```
-------------------------------------------------------
Summarized Null Value report for : Contracts Table.
-------------------------------------------------------
        column_name  null_count null_percent
0     customer_name         284        3.01%
1  reg_payment_mode         925         9.8%
2           psl_tag         486        5.15%
3       rc_verified         267        2.83%
4         dealer_id        6109       64.74%
5         broker_id        7096        75.2%
-------------------------------------------------------
Summarized Null Value report for : Repayment Table.
-------------------------------------------------------
No Null Values found in Repayment Table.
None
```

[8]:
```
print(summarize_nulls(branches,"Branches"))
print(summarize_nulls(brokers,"Brokers"))
print(summarize_nulls(dealers,"Dealers"))
print(summarize_nulls(executives,"Executives"))
```

```
-------------------------------------------------------
Summarized Null Value report for : Branches Table.
-------------------------------------------------------
No Null Values found in Branches Table.
None
-------------------------------------------------------
Summarized Null Value report for : Brokers Table.
-------------------------------------------------------
No Null Values found in Brokers Table.
None
-------------------------------------------------------
```

```
Summarized Null Value report for : Dealers Table.
--------------------------------------------------------
No Null Values found in Dealers Table.
None
--------------------------------------------------------
Summarized Null Value report for : Executives Table.
--------------------------------------------------------
No Null Values found in Executives Table.
None
```

0.2.2 Key Understanding:

- There are **284 customer names missing out of 9436 contract** records **but their loan_ids are present**.
- Contract records with missing customer names were retianed to preserve repayment history and avoid loss of risk signals.
- For the missing Registered Payment Mode values, PSL tag values, RC verification values, we will have to replace it with "Unknown" as they are business' regulatory categorizations, hence Mode imputation is not suitable.
- **The null values** in Dealer and Broker ID columns **will help us in Feature Engineering** to **create the channel type category**.
- **Logic:** Rows with only dealer_id value will be labelled as "Dealer", rows with only broker_id value will be labelled as "Broker" and rows with Null values in both will be labelled as "New".

0.2.3 Data Integrity Validation

[ ]:
```
# Checking duplicate loan or payment records, and how should they be handled?

duplicates_contracts = contracts[contracts.
  ↪duplicated(subset=['loan_id'],keep=False)]
print(duplicates_contracts,"\n")

duplicate_repayment = repayment[repayment.
  ↪duplicated(subset=['loan_id','due_date'],keep=False)]
print(duplicate_repayment,"\n")

## No duplicate contract records found, tables conform to expected grain.
```

```
Empty DataFrame
Columns: [loan_id, customer_name, asset_name, invoice_value, loan_amount,
emi_amount, tenure_months, irr, loan_issue_date, loan_status, reg_payment_mode,
psl_tag, rc_verified, dealer_id, broker_id, executive_id, branch_id, manager_id]
```

```
Index: []

Empty DataFrame
Columns: [loan_id, due_date, emi_amount, interest_component,
principal_component, due_amount, amount_received, overdue_amount, dpd_days,
balance_amount, branch_id, manager_id]
Index: []
```

[6]:
```
print(f"Unique Values for loan_id in repayments: {repayment['loan_id'].
 ↪nunique()}\n")
print(f"Unique Values for loan_id in contracts: {contracts['loan_id'].
 ↪nunique()}\n")

## Repayment records exist for all contracts in both tables. No orphan records␣
 ↪found.
```

```
Unique Values for loan_id in repayments: 9436

Unique Values for loan_id in contracts: 9436
```

[7]:
```
print(f"Unique Values for dealer_id in contracts: {contracts['dealer_id'].
 ↪nunique()}\n")
print(f"Unique Values for dealer_id in dealrs: {dealers['dealer_id'].
 ↪nunique()}\n")
print(f"Unique Values for broker_id in contracts: {contracts['broker_id'].
 ↪nunique()}\n")
print(f"Unique Values for broker_id in brokers: {brokers['broker_id'].
 ↪nunique()}\n")

# All registered dealers/brokers have at least one contract in the portfolio.
```

```
Unique Values for dealer_id in contracts: 300

Unique Values for dealer_id in dealrs: 300

Unique Values for broker_id in contracts: 200

Unique Values for broker_id in brokers: 200
```

0.3 ### Feature Engineering

[31]:
```
contracts.head(2) # Before
```

[31]:
```
       loan_id customer_name        asset_name  invoice_value  loan_amount  \
0  1000000001    Adira Loke  Mahindra Scorpio        1060000       870000
1  1000000002    Jivika Som  Mahindra Scorpio        1650000      1320000

   emi_amount  tenure_months    irr loan_issue_date loan_status  \
0       20123             60  13.73      01-05-2023           L
1       46496             36  16.14      18-11-2023           L

  reg_payment_mode psl_tag rc_verified  dealer_id  broker_id  executive_id  \
0              ECS       N           N        NaN        NaN      27030005
1              ECS       N           Y        NaN        NaN      27030006

   branch_id  manager_id
0       1006    27020412
1       1007    27020412
```

[3]:
```
# Created channel_type column to track contract lead source existing/channel/
 ↪broker for downstream analysis


contracts['channel_type'] = np.where(

    contracts['dealer_id'].notna()  &  contracts['broker_id'].isna(), 'Dealer',
    np.where(contracts['broker_id'].notna() & contracts['dealer_id'].isna(),␣
 ↪'Broker', 'Self')

)
```

[263]:
```
contracts.head(2) #After
```

[263]:
```
       loan_id customer_name        asset_name  invoice_value  loan_amount  \
0  1000000001    Adira Loke  Mahindra Scorpio        1060000       870000
1  1000000002    Jivika Som  Mahindra Scorpio        1650000      1320000

   emi_amount  tenure_months    irr loan_issue_date loan_status  \
0       20123             60  13.73      01-05-2023           L
1       46496             36  16.14      18-11-2023           L

  reg_payment_mode psl_tag rc_verified  dealer_id  broker_id  executive_id  \
0              ECS       N           N        NaN        NaN      27030005
1              ECS       N           Y        NaN        NaN      27030006

   branch_id  manager_id channel_type
0       1006    27020412         Self
1       1007    27020412         Self
```

[4]:
```
## Created a 'dpd_bucket' column to classify contracts into Regular/X bucket/S1/
 ↪S2/S3/NPA contracts for downstream analysis.

conditions = [

    (repayment['dpd_days'] == 0 ),
    (repayment['dpd_days'] > 0  )  &  (repayment['dpd_days'] < 30 ),
    (repayment['dpd_days'] >= 30 ) & ( repayment['dpd_days'] < 60 ),
    (repayment['dpd_days'] >= 60 ) & ( repayment['dpd_days'] < 90  ),
    (repayment['dpd_days'] >= 90)
]

classification = ['Regular', 'X bucket', 'S1', 'S2', 'NPA' ]

repayment['dpd_bucket'] = np.select(conditions,classification,default=␣
 ↪'Unknown')
```

[5]:
```
repayment.head(2)
```

[5]:

	loan_id	due_date	emi_amount	interest_component	principal_component	due_amount	amount_received	overdue_amount	dpd_days	balance_amount	branch_id	manager_id	dpd_bucket
0	1000000001	2023-05-05	20123.0	9954.25	10168.75	20123.0	20123.0	0.0	0	859831.25	1006	27020412	Regular
1	1000000001	2023-06-05	20123.0	9837.90	10285.10	20123.0	20123.0	0.0	0	849546.15	1006	27020412	Regular

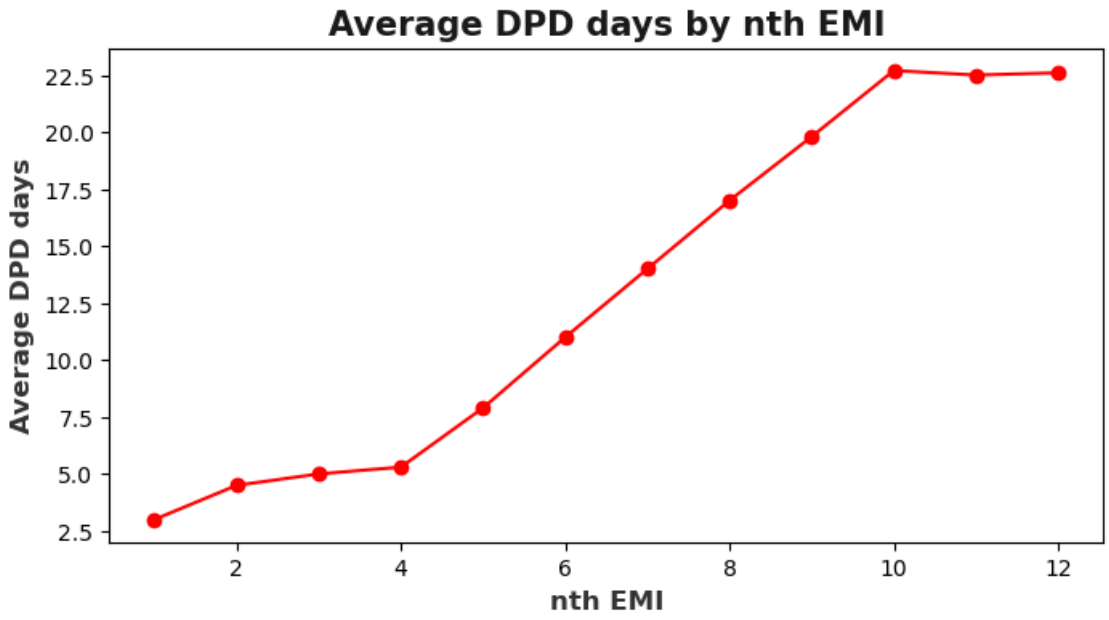
[6]:
```
# Payment Delay Patterns by EMI Sequence
# Checking if later EMIs show higher delinquency than early the EMIs.

max_date = repayment['due_date'].max()
snapshot = repayment[repayment['due_date'] <= max_date]
snapshot['emi_sequence'] = snapshot.groupby('loan_id').cumcount()+1
# Added serial numbers to each of the contract's EMI.
snapshot.head()

sequence_dpd = snapshot.groupby('emi_sequence')['dpd_days'].
 ↪agg(['mean','count']).round(1)
# Averages and count of DPD days for all nth EMIs .
print("Average DPD days by EMI Sequence:\n")
print(sequence_dpd)
```

```
Average DPD days by EMI Sequence:
```

emi_sequence	mean	count
1	3.0	9436
2	4.5	9436
3	5.0	9436
4	5.3	9436
5	7.9	9436
6	11.0	9436
7	14.0	9436
8	17.0	9436
9	19.8	9436
10	22.7	9436
11	22.5	9436
12	22.6	9436

[271]:
```
plt.figure(figsize=(8,4))
plt.plot(sequence_dpd.index,sequence_dpd['mean'],marker='o',color = 'red')
plt.title("Average DPD days by nth EMI",fontsize = 15, fontweight = "bold",␣
 ↪alpha = 0.9)
plt.xlabel("nth EMI",fontsize = 12, fontweight = "bold",alpha = 0.8)
plt.ylabel("Average DPD days",fontsize = 12, fontweight = "bold",alpha = 0.8)
```

[271]:
```
Text(0, 0.5, 'Average DPD days')
```



[8]:
```
## Checking if there is correlation between loan tenure and DPD buckets␣
 ↪(Regular/S1/S2/NPA).

print(f"{"-"*65} \nColumns of repyment table: \n{"-"*65}\n {repayment.
 ↪columns}\n")
print(f"{"-"*65} \nLast date in the repayment table is '{repayment['due_date'].
 ↪max()}'\n{"-"*65}")

repayment['due_date'] = pd.to_datetime(repayment['due_date'])
```

```
------------------------------------------------------------------
Columns of repyment table:
------------------------------------------------------------------
 Index(['loan_id', 'due_date', 'emi_amount', 'interest_component',
       'principal_component', 'due_amount', 'amount_received',
       'overdue_amount', 'dpd_days', 'balance_amount', 'branch_id',
       'manager_id', 'dpd_bucket'],
      dtype='object')

------------------------------------------------------------------
Last date in the repayment table is '2024-11-15'
------------------------------------------------------------------
```

[9]:
```
november_2024 = repayment[

    (repayment['due_date'].dt.year == 2024) &
    (repayment['due_date'].dt.month == 11)

]][['loan_id','dpd_bucket']]

print(f"Total contracts as on November 2024: {november_2024.shape}\n")

# We have total 839 loan contracts as on November 2024.

corr_monthly_dpd = contracts[['loan_id','tenure_months']].merge(
    november_2024, on = 'loan_id', how = 'inner'
    )

corr_monthly_dpd.head()
```

```
Total contracts as on November 2024: (839, 2)
```

[9]:

	loan_id	tenure_months	dpd_bucket
0	1000000007	36	Regular
1	1000000016	60	Regular
2	1000000019	60	Regular

```
3  1000000024              36    Regular
4  1000000030              60    Regular
```

[ ]:
```
# Numeric bucket for correlation.

dpd_bucket_map = {'Regular':0,'S1':1,'S2':2,'NPA': 3}

corr_monthly_dpd['dpd_bucket_num'] = corr_monthly_dpd['dpd_bucket'].
 ↪map(dpd_bucket_map)
corr_monthly_dpd.head()
```

[ ]:
```
      loan_id  tenure_months dpd_bucket  dpd_bucket_num
0  1000000007              36    Regular               0
1  1000000016              60    Regular               0
2  1000000019              60    Regular               0
3  1000000024              36    Regular               0
4  1000000030              60    Regular               0
```

[300]:
```
correlation = corr_monthly_dpd[['tenure_months','dpd_bucket_num']].corr(method ␣
 ↪= 'spearman')

print(f"{'-'*45}\nCorrelation between Tenure and DPD buckets: \n{'-'*45}")
print(correlation,"\n")

rho = correlation.loc['tenure_months','dpd_bucket_num']
print(f"{'-'*45}\nSpearman   = {rho:.3f} \n{'-'*45}")
```

```
---------------------------------------------
Correlation between Tenure and DPD buckets:
---------------------------------------------
                tenure_months  dpd_bucket_num
tenure_months        1.000000        0.005343
dpd_bucket_num       0.005343        1.000000

---------------------------------------------
Spearman   = 0.005
---------------------------------------------
```

[ ]:
```
# Monthly Average DPD days trend by year:

monthly_dpd = repayment.groupby([
    repayment['due_date'].dt.year.rename('year'),
    repayment['due_date'].dt.month.rename('month')
])['dpd_days'].mean().reset_index(name='avg_dpd')


monthly_dpd['month_label'] = monthly_dpd['month'].map({
```

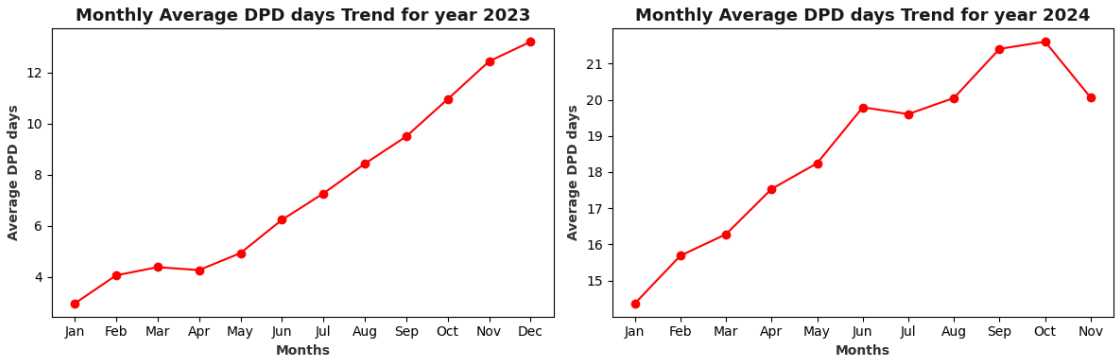
```
    1:'Jan', 2:'Feb', 3:'Mar', 4:'Apr', 5:'May', 6:'Jun',
    7:'Jul', 8:'Aug', 9:'Sep', 10:'Oct', 11:'Nov', 12:'Dec'
})

print(monthly_dpd)
```

```
    year  month    avg_dpd month_label
0   2023      1   2.951432         Jan
1   2023      2   4.055118         Feb
2   2023      3   4.375267         Mar
3   2023      4   4.259674         Apr
4   2023      5   4.927866         May
5   2023      6   6.230134         Jun
6   2023      7   7.263330         Jul
7   2023      8   8.428300         Aug
8   2023      9   9.497890         Sep
9   2023     10  10.957650         Oct
10  2023     11  12.440386         Nov
11  2023     12  13.210047         Dec
12  2024      1  14.362331         Jan
13  2024      2  15.686299         Feb
14  2024      3  16.279791         Mar
15  2024      4  17.532097         Apr
16  2024      5  18.246126         May
17  2024      6  19.792241         Jun
18  2024      7  19.604793         Jul
19  2024      8  20.051315         Aug
20  2024      9  21.410146         Sep
21  2024     10  21.608392         Oct
22  2024     11  20.059595         Nov
```

[186]:
```
for_2023 =␣
  ↪monthly_dpd[monthly_dpd['year']==2023][['month','month_label','avg_dpd']]
for_2024 =␣
  ↪monthly_dpd[monthly_dpd['year']==2024][['month','month_label','avg_dpd']]
```

[219]:
```
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.plot(for_2023['month_label'],for_2023['avg_dpd'],marker = 'o', color =␣
  ↪'red')
plt.xlabel('Months',fontweight = 'bold', alpha = 0.8)
plt.ylabel('Average DPD days',fontweight = 'bold', alpha = 0.8)
plt.title('Monthly Average DPD days Trend for year 2023',fontsize =␣
  ↪13,fontweight = 'bold', alpha = 0.9)

plt.subplot(1,2,2)
```

```
plt.plot(for_2024['month_label'],for_2024['avg_dpd'],marker = 'o', color =␣
 ↪'red')
plt.xlabel('Months',fontweight = 'bold', alpha = 0.8)
plt.ylabel('Average DPD days',fontweight = 'bold', alpha = 0.8)
plt.title('Monthly Average DPD days Trend for year 2024',fontsize =␣
 ↪13,fontweight = 'bold', alpha = 0.9)

plt.tight_layout()
plt.show()
```



[ ]:
```
## Loading DataFrames into MYSQL.

from sqlalchemy import create_engine

engine =  create_engine(
    "mysql+pymysql://root:MYsql%4060323@localhost:3306/automotive_loans"
)
```

[18]:
```
tables = {

    "branches"      : branches,
    "brokers"       : brokers,
    "contracts"     : contracts,
    "dealers"       : dealers,
    "executives"    : executives,
    "repayment"     : repayment
}

for table_name, df in tables.items():

    df.to_sql(
        name = table_name,
        con = engine,
```

```
        if_exists = "replace",
        index = False
    )
    print(f"{table_name} loaded.\n")
```

```
branches loaded.

brokers loaded.

contracts loaded.

dealers loaded.

executives loaded.

repayment loaded.
```