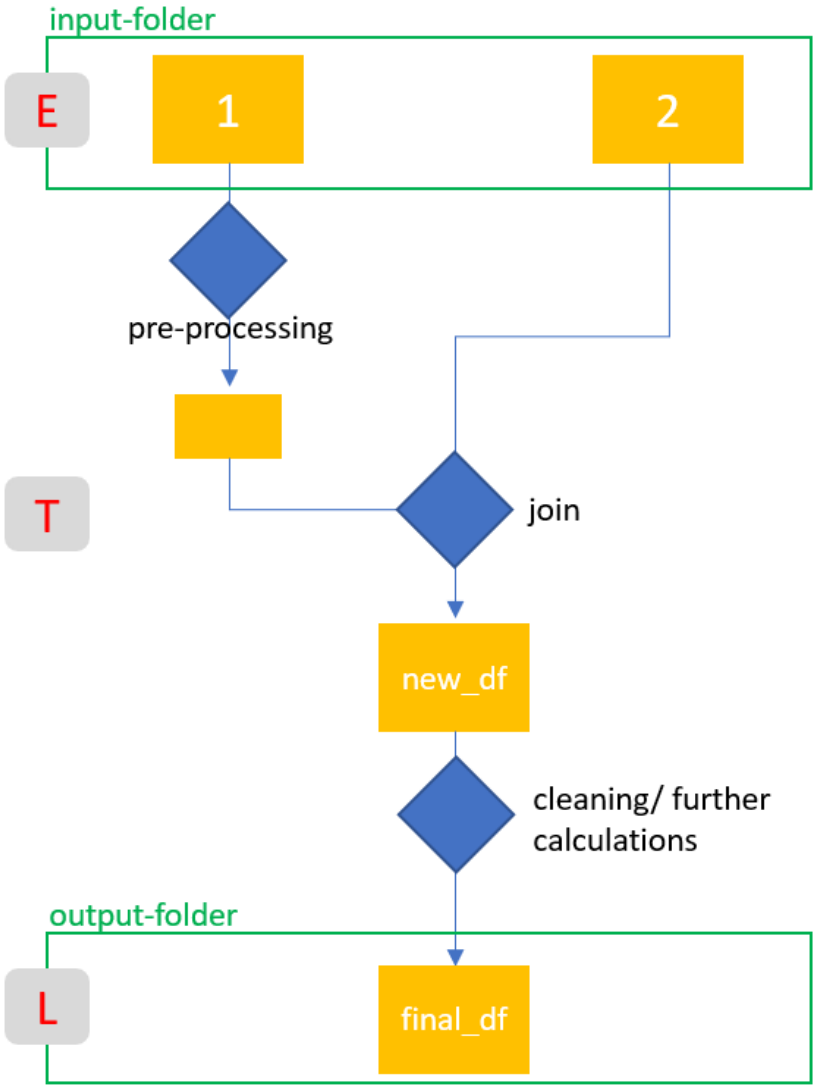
BAB IV
HASIL DAN PEMBAHASAN

4.1 Analisis Perancangan

Dalam perencanaan bisnis suatu produk atau untuk menjaga agar kualitas dari pembiayaan tetap terjaga baik dengan menggunakan *Vintage Analysis* ini dapat terlihat gambaran kualitas dari periode pembiayaan. Hasil analisis ini merupakan gambaran awal untuk melihat seberapa sehat kualitas suatu pembiayaan di sebuah Bank Syariah. Nantinya hasil ini dapat digunakan untuk melihat dari berbagai macam segmentasi sebagai penyebab kualitas suatu pembiayaan dapat menurun.



Gambar 3: Pipeline ETL Process

Untuk melakukan Analisis ini hal yang pertama dilakukan adalah mempersiapkan *tools* atau alat bantu yang diperlukan. Berikut alat bantu yang diperlukan :

1. Python

   Python sebagai bahasa pemograman yang digunakan untuk melakukan seluruh proses analitik dari awal sampai akhir.
2. Jupyter Notebook

   Jupyter Notebook digunakan sebagai kode editor.
3. Python library :
   a. Pandas
   b. Numpy
   c. Matplotlib
   d. Pyodbc
   e. Seaborn
4. Gith Bash / Command Prompt / Terminal / PowerShell

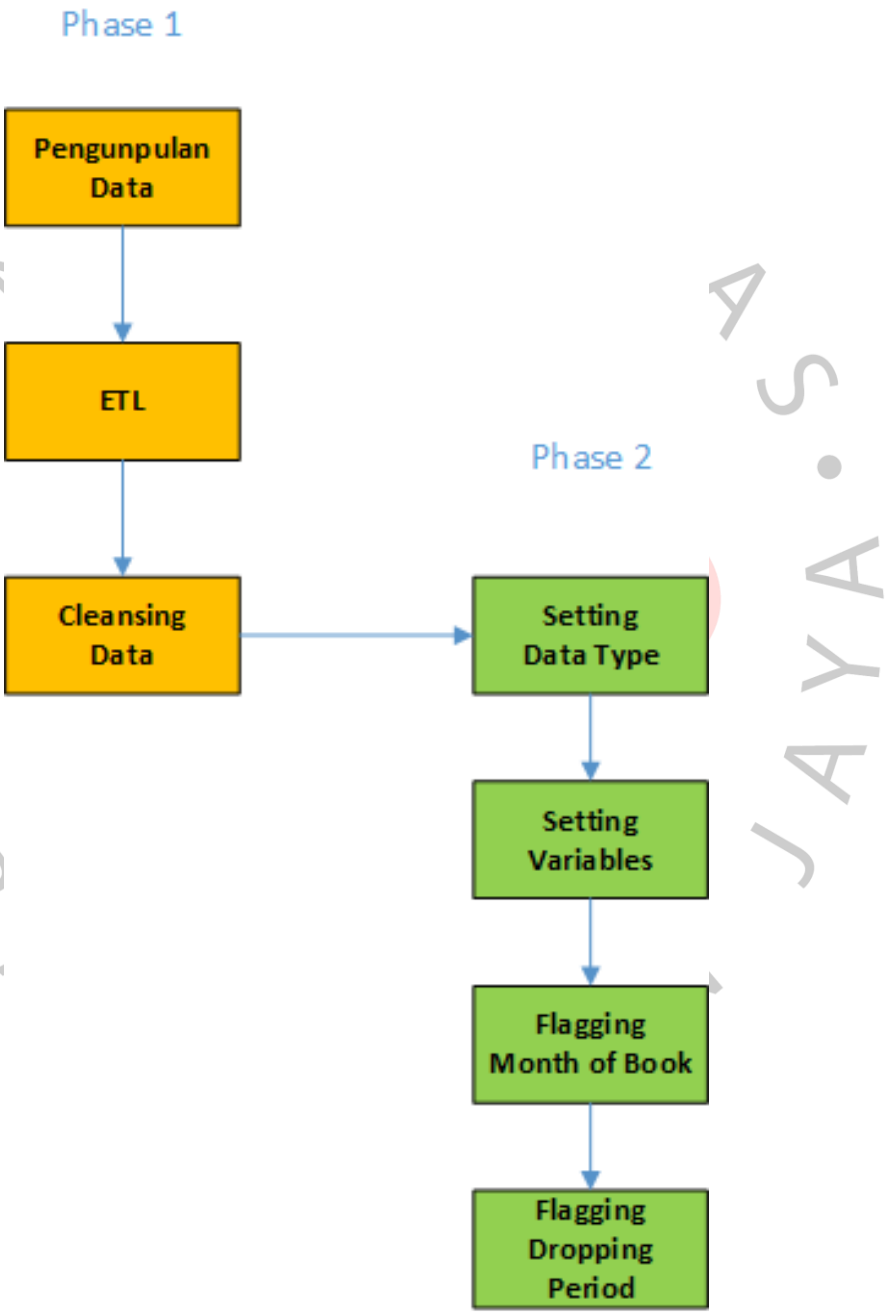
   Command Prompt atau Terminal ini digunakan untuk keperluan instalasi ataupun alat bantu berisikan instruksi terkait dengan akses kedalam *root system*.
5. SQL Server

   Sebagai server database yang menyimpan sumber data dan hasil dari pengolahan data agar dapat terstruktur dan dapat digunakan oleh banyak pengguna kedepannya.
6. Browser Firefox / Chrome

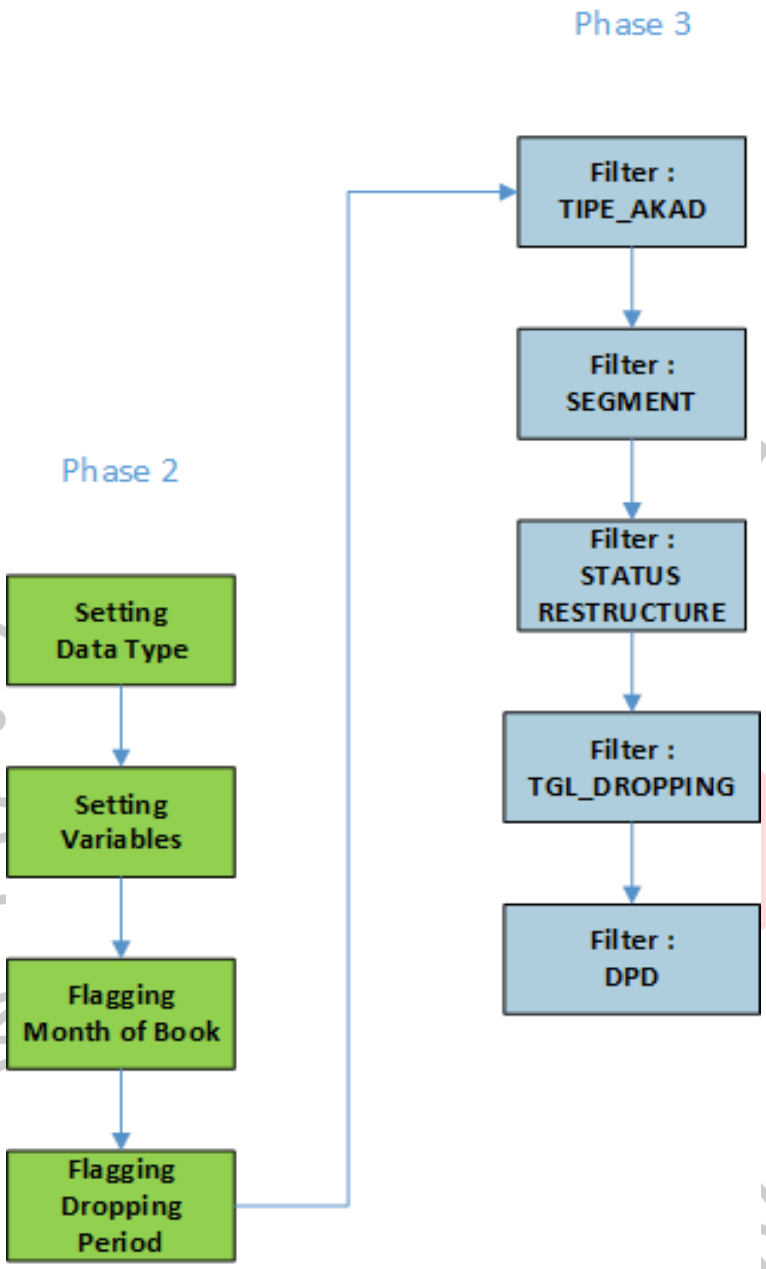
Penggunaan data pembiayaan Al-Murabahah pada Bank Syariah merupakan pembiayaan segmentasi Konsumer dengan jangka waktu atau umur pembiayaan 2 (dua) tahun sebagai minimal periode untuk dilakukannya penelitian analitik.

4.2 Perancangan Alur Proses

Pada penelitian ini agar memudahkan proses analisis data, maka dibagi menjadi beberapa fase proses dan masing-masing fase proses memiliki proses sendiri juga sehingga akan memudahkan dalam proses pengolahan data.



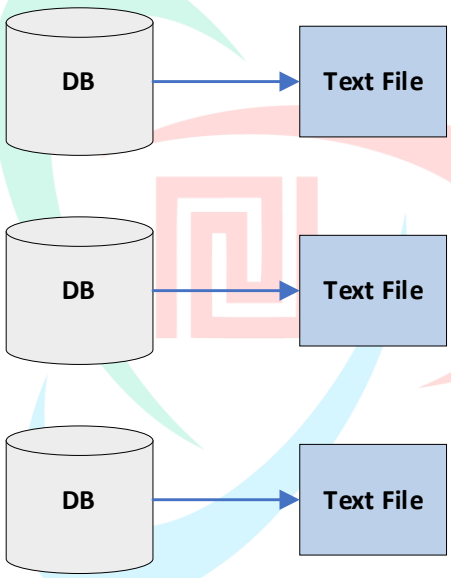
Gambar 4 : Flow Process Phase 1 & 2



Gambar 5: Flow Process Phase 2 & 3

4.2.1 Pengumpulan Data Observasi

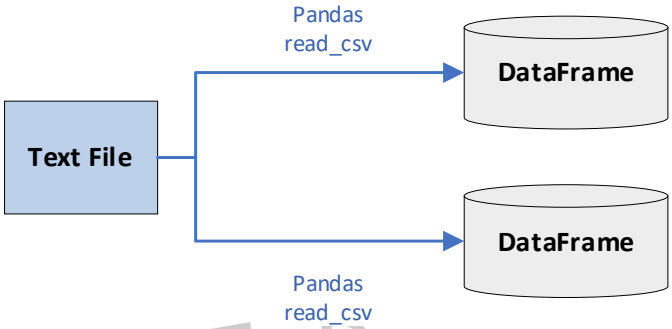
Dalam perancangan proses *Vintage Analysis* data observasi didapat dari beberapa *data warehouse* unit kerja, antara lain unit bisnis, unit *risk management*, dan unit IT Departemen. Database yang digunakan juga berbeda dari setiap unit kerja. Untuk itu data yang dibutuhkan di *export* kedalam bentuk *text file (.txt)* atau dalam bentuk *comma separated values (.csv)*. Proses selanjutnya adalah *staging* yaitu melakukan pengumpulan data yang berasal dari masing-masing unit kedalam sebuah database atau satu *folder* untuk mempermudah pengolahan data menggunakan *python*.



Gambar 6: Proses *Export* Data Menjadi *Text File*

4.2.2 *Extract Transform Load (ETL)*

Pada tahap ini yaitu melakukan proses membuat dataframe yang berasal dari *txt file* atau *csv file* menggunakan *jupyter notebook*. Jupyter Notebook merupakan aplikasi berbasis web browser untuk digunakan sebagai editor kode atau naskah perintah menjalankan suatu proses. Ekstraksi data dari file txt menjadi dataframe menggunakan Pustaka dari *python* bernama *Pandas*.



Gambar 7: Proses *Transform text file* menjadi *dataframe*

Proses membuat dataframe dengan file yang mempunyai ukuran besar agar tidak memakan waktu dan sisa memori RAM dapat menggunakan fungsi *chunksize* atau membuat parsial data dari data asli. *Chunksize* dapat diatur sesuai dengan kebutuhan, nantinya akan terbentuk banyak virtual data hasil dari *chunksize* ini. Banyak nya virtual data tergantung dari besaran *chunksize* yang dimasukan, seperti semisal besaran *chunksize* 1000 maka sistem akan membaca jumlah total baris dari file txt, lalu jumlah total baris tersebut akan dibagi dengan besaran *chunksize* yaitu 1000. Setiap virtual data akan mempunyai 1000 baris data.

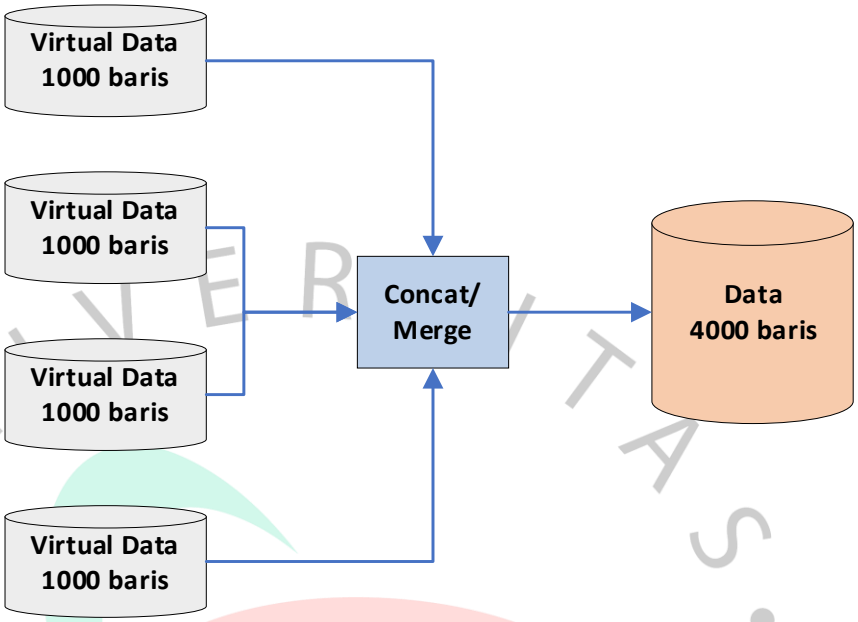
```
In [3]: df_chunk17 = pd.read_csv('df_2017_master.csv',
                                 sep=';',
                                 chunksize=100000,
                                 low_memory=False)
        df_2017_master = pd.concat(df_chunk17)
        df_2017_master = pd.DataFrame(df_2017_master)
```

Gambar 8 : Read csv File

Tabel 1 : Struktur Data Dataframe

Column Name	*Non-Null Count*	*Dtype*
PERIODDATE	1417018 non-null	Datetime64
TIPE_AKAD	1417018 non-null	Object
AREA_MR	1417018 non-null	Object
NO_BASE	1417018 non-null	Object
NO_KARTU	1417018 non-null	Object
NO_KARTU_LAMA	1417018 non-null	Object
KOLCIF	1417018 non-null	Int64
KOL	1417018 non-null	Int64
DPD	1417018 non-null	Int64
DPD_SLAP	1417018 non-null	Object
C09_SEGMENTASI	1417018 non-null	Object
SEGMENT_GROUP	1417018 non-null	Object
OS_POKOK_LCY	1417018 non-null	Float
JK_WKT_MULAI	1417018 non-null	Datetime64
JK_WKT_SELESAI	1417018 non-null	Datetime64
TGL_DROPPING	1417018 non-null	Datetime64
STATUS_RESTRUKTUR	1417018 non-null	Object

4.2.3 *Concat* atau *merge*



Gambar 9: Proses *concat/merge* data

Tahap ini adalah proses menggabungkan semua virtual data hasil *chunksize* menjadi satu data. *Concat/merge*. Data yang terbentuk akan memiliki baris sesuai dengan total baris data pada data asli.

```
In [12]: df_chunk17 = pd.read_csv('df_2017_master.csv', sep=';', chunksize=100000, usecols=kolom, low_memory=False)
         df_2017 = pd.concat(df_chunk17)
         df_2017 = pd.DataFrame(df_2017)
         df_2017.head()
```

Out[12]:

	PERIODDATE	TIPE_AKAD	AREA_MR	NO_BASE	NO_KARTU	NO_KARTU_LAMA	KOLCIF	KOL	DPD	DPD_SLAP	C09_SEGMENTASI_ACCOUI
0	2017-01-31	E-IJARAH	SUMBAGUT	000378245	230E305141250008	230E305141250008	1	1	0	1 - Current	45_CONSUMER_MUI PURPO
1	2017-01-31	E-IJARAH	JAWA BARAT	000385653	161E305142260002	161E305142260002	1	1	0	1 - Current	45_CONSUMER_MUI PURPO
2	2017-01-31	E-IJARAH	JATIM, BALI & NUSRA	003730384	731E305141270006	731E305141270006	1	1	0	1 - Current	45_CONSUMER_MUI PURPO
3	2017-01-31	E-IJARAH	SUMBAGSEL	001371325	371E305142310002	371E305142310002	1	1	0	1 - Current	45_CONSUMER_MUI PURPO
4	2017-01-31	E-IJARAH	SUMBAGSEL	003696016	373E305142230002	373E305142230002	5	1	0	1 - Current	21_RETAIL_SI

Gambar 10 : Proses *read_csv python*

4.2.4 *Cleansing Data*

Proses tahapan ini menjadi proses yang sangat penting. Dengan melakukan pembersihan data atau *Cleansing Data* akan membuat data menjadi lebih mudah dibaca, tipe data sesuai dengan tipe dari informasi kolom tersebut dan juga membuang data-data yang tidak digunakan ataupun untuk dapat menghindari adanya data ganda. Proses *cleansing data* seperti :

1. Merubah Tipe data *datetime*

   Data yang bersumber dari *raw* data seperti dari *text file* pada umumnya semua jenis tipe data akan terbaca sebagai *string* atau *text*. Untuk dapat membaca data tersebut sebagai tanggal yang akan bisa diolah dikemudian nanti, perlu melakukan perubahan tipe data. Gunakan fungsi yang tersedia didalam paket Pustaka *Pandas* yaitu fungsi *to_datetime()*. Data yang diubah antara lain :

   a. Kolom Periode Data Pelaporan

      Sebelumnya tipe data untuk Kolom Periode Data adalah *Object* atau teks *string*.

```
---  ------                  --------------  -----
0    PERIODDATE              1418574 non-null  object
```

Gambar 11 : Tipe Data Sebelum Dirubah

Kemudian merubah tipe data menjadi *datetime*

```
In [12]:  df_2017_master['PERIODDATE'] = pd.to_datetime(df_2017_master['PERIODDATE'])
```

Gambar 12 : Pandas to_datetime

Sehingga setelah dirubah maka tipe data akan berubah semula adalah *object* menjadi *datetime64*.

```
---  ------                  --------------  -----
0    PERIODDATE              1418574 non-null  datetime64[ns]
```

Gambar 13 : Tipe Data Setelah Diubah

   b. Kolom Periode Tanggal Pencairan

   c. Kolom Periode Tanggal Akhir Pembiayaan

   d. Format tanggal menjadi format ‘YYYY-MM-DD’

      Dengan menambahkan kode : format = ‘%Y-%m-%d’.

Keterangan : Y besar untuk merubah tahun menjadi 4 (empat) digit. Untuk bulan jika ditulis ‘m’ dengan huruf kecil maka format bulan dalam bentuk angkat 2 (dua) digit, jika ditulis ‘M’ penulisan dengan hurus besar, format bulan akan ditulis dengan nama bulan. Format tanggal dengan penulisan ‘d’ huruf kecil maka format tanggal berupa angka dengan 2 (dua) digit.

```
13  JK_WKT_MULAI            1417380 non-null  object
14  JK_WKT_SELESAI          1418560 non-null  object
15  TGL_DROPPING            1418217 non-null  object
```

Gambar 14 : Tipe Data Tanggal Mulai, Selesai, dan Dropping sebelum diubah tipe data

Proses merubah tipe data untuk kolom *JK_WKT_MULAI, JK_WKT_SELESAI, TGL_DROPPING*

```
In [15]: df_2017_master['JK_WKT_MULAI'] = pd.to_datetime(df_2017_master['JK_WKT_MULAI'],
                                                        format='%Y-%m-%d')

In [16]: df_2017_master['JK_WKT_SELESAI'] = pd.to_datetime(df_2017_master['JK_WKT_SELESAI'],
                                                        format='%Y-%m-%d')

In [17]: df_2017_master['TGL_DROPPING'] = pd.to_datetime(df_2017_master['TGL_DROPPING'],
                                                        format='%Y-%m-%d')
```

Gambar 15 : Proses Merubah Tipe Data Tanggal Mulai, Selesai, dan Tanggal *Dropping*

Hasil setelah dilakukan perubahan tipe data :

```
13  JK_WKT_MULAI            1417380 non-null  datetime64[ns]
14  JK_WKT_SELESAI          1418560 non-null  datetime64[ns]
15  TGL_DROPPING            1418217 non-null  datetime64[ns]
```

Gambar 16 : Tipe data Tanggal Mulai, Selesai, dan Tanggal *Dropping* setelah dilakukan perubahan

2. Merubah Tipe data *int* atau *integer* dan *float*

   Secara *default python* ketika membaca file jika isi dari data tersebut secara keseluruhan dalam satu kolom hanya terdapat angka bulat, maka akan dibaca menjadi *int* atau *integer*. Jika terdapat angka bilangan decimal, akan dibaca menjadi *float*. Perubahan ini penting agar isi data yang berupa angka dapat dioperasikan dengan operasi matematika. Data yang diubah antara lain:

   a. Kolom Jumlah Hari Keterlambatan atau *Days Payment Due (DPD)* dengan tipe data *int.*

b. Kolom *Outstanding* atau Jumlah Pokok Kewajiban

3. Mengganti isi data *null* dengan Nol (0)

   Untuk kolom dengan tipe data *int* atau *float* yang akan digunakan untuk dilakukannya operasi matematika, dalam *python* jika terdapat nilai *null* atau *blank* (kosong), maka proses operasi matematika tidak dapat berjalan atau tidak dapat menampilkan hasil. Agar dapat dilakukannya operasi matematika pada kolom tersebut, diperlukan perubahan nilai *null* atau *blank* menjadi Nol (0) dengan menggunakan fungsi .fillna(0). Angka Nol (0) dalam kurung adalah karakter untuk pengganti nilai yang *null* atau *blank*. Fungsi ini juga dapat digunakan bukan hanya untuk tipe data angka saja, tetapi dapat pula tipe data *string*

   a. Merubah nilai *null* pada kolom *Outstanding* :

```
In [9]: df_2017_master['OS_POKOK_LCY'] = df_2017_master['OS_POKOK_LCY'].fillna(0)
        df_2017_master['OS_POKOK_LCY'] = df_2017_master['OS_POKOK_LCY'].astype('float')
```

Gambar 17 : Fungsi *fillna(0)* pada kolom *Outstanding*

   b. Merubah nilai *nullI* pada kolom Tunggakan Pokok

```
In [9]: df_2017_master['TUNGGAKAN_POKOK'] = df_2017_master['TUNGGAKAN_POKOK'].fillna(0)
        df_2017_master['TUNGGAKAN_POKOK'] = df_2017_master['TUNGGAKAN_POKOK'].astype('float')
```

Gambar 18 : Fungsi *fillna(0)* pada kolom Tunggakan Pokok

4. Menghilangkan kolom yang tidak digunakan

   Tidak semua kolom atau informasi dibutuhkan sesuai dengan parameter yang diperlukan. Untuk membuang kolom yang tidak digunakan ada beberapa cara, antara lain :

   a. Proses seleksi kolom tertentu sesuai dengan keperluan pada saat proses *import* data yaitu dengan menambahkan fungsi *usecols* yang artinya hanya kolom-kolom yang dituliskan didalam fungsi ini yang akan dibawa pada saat proses *import* data. Penulisan nama kolom-kolom yang diperlukan dengan menggunakaan *usecols* ini bisa dilakukan langsung dalam fungsi ini ataupun merujuk kepada *list array* nama kolom.

b. Proses membuang kolom setekah proses *import* data selesai dengan menggunakan fungsi *isin()*. Penggunaan *isin* ini bisa dengan langsung menuliskan nama-nama kolom dengan format penulisan seperti penulisan *array* yaitu diawali dengan kurung siku '[' , penulisan nama kolom diawali dan diakhir dengan satu koma atas (') dan menggunakan koma untuk memisahkan satu nama kolom dengan nama kolom lainnya. Atau bisa juga dengan cara seperti point a yaitu membuat *list array* yang isi dari *array* tersebut adalah berisi nama-nama kolom yang diinginkan.

Inisiasi nama kolom yang akan dibutuhkan sebagai data penelitian :

```
In [2]: kolom = [
            'PERIODDATE',
            'TIPE_AKAD',
            'AREA_MR',
            'NO_BASE',
            'NO_KARTU',
            'NO_KARTU_LAMA',
            'KOLCIF',
            'KOL',
            'DPD',
            'DPD_SLAP',
            'C09_SEGMENTASI_ACCOUNT',
            'SEGMENT_GROUP',
            'OS_POKOK_LCY',
            'JK_WKT_MULAI',
            'JK_WKT_SELESAI',
            'TGL_DROPPING',
            'STATUS_RESTRUKTUR'
        ]
```

Gambar 19 : Inisiasi nama-nama kolom

Karena nama-nama kolom yang diperlukan sudah dibuat, selanjutnya adalah membuang nama-nama kolom di luar list yang sudah dibuat sebelumnya.

```
In [ ]: df_2017_master = df_2017_master[df_2017_master.columns[df_2017_master.columns.isin(kolom)]]
```

Gambar 20 : Membuang Nama Kolom Diluar List

5. Merubah Nama Kolom

Proses ini diperlukan dikarenakan nama-nama kolom pada data umumnya mengikuti format dari sistem atau database. Penamaan ulang ini dapat mempermudah kita kedepannya dalam mengindentifikasi suatu pemasalahan dengan nama kolom yang sesuai dan mudah diingat. Proses penamaan ulang nama kolom adalah menggunakan fungsi *rename* dan merujuk kepada kolom yang diinginkan. Seperti berikut :

a. Merubah Kolom :

df_master =

df_master.rename(columns={'OUSTATSTANDING':'OS'})

6. Pembiayaan Akad *Murabahah*

Dalam penelitian ini fokus pada Pembiayaan Syariah dengan menggunakan Akad *Murabahah*. Untuk melakukan penyaringan data terkait akad dapat dilakukan dengan cara ;

```
In [14]: # Filter Data - Tipe Akad : Murabahah
         df_2017_vin = df_2017[df_2017['TIPE_AKAD'] == 'A-MURABAHAH']
         df_2017_vin.head()
```

Gambar 21 : Filter Akad Murabahah

7. Segmentasi *Consumer*

Pada tahap ini melakukan proses penyaringan data pada kolom Segmentasi Group dengan memilih hanya Segmentasi *Consumer*.

```
In [16]: # Filter Data - Segmentasi : Consumer
         df_2017_vin = df_2017_vin[df_2017_vin['SEGMENT_GROUP'] == '5-CONSUMER']
         df_2017_vin.head()
```

Gambar 22 : Filter Segment - Consumer

8. Penyaringan Data Status Restruktur

Tahap proses ini untuk memisahkan mana saja nasabah yang baru pencairan dalam kondisi hasil restrukturisasi atau murni pencairan baru. Pencairan hasil restrukturisasi tidak dapak dikategorikan sebagai pencairan baru dikarenakan restrukturisasi adalah merupakan Langkah yang diambil oleh bank baik oleh permohonan langsung oleh nasabah ataupun inisiasi dari bank sendiri. Yang dimana historis kualitas pembiayaan sebelum restrukturisasi dipastikan adalah nasabah yang mempunyai kualitas pembiayaan bermasalah atau tidak baik.

```
In [17]:  # Filter Data - Status Restruktur : No Restruktur
          df_2017_vin = df_2017_vin[df_2017_vin['STATUS_RESTRUKTUR'] == 'NO RESTRUKTUR']
          df_2017_vin.head()
```

Gambar 23 : Filter Data No Restruktur

9. Periode Pencairan Pembiayaan Baru

Selanjutnya adalah penyaringan data berdasarkan periode pencairan yang sesuai dengan periode penelitian. Kolom yang digunakan adalah kolom keterangan Tanggal Pencairan (*Dropping*)

```
In [18]:  # Filter Data - Periode Pencairan Pembiayaan
          df_2017_vin = df_2017_vin[df_2017_vin['TGL_DROPPING'] >= '2017-01-01']
          df_2017_vin = df_2017_vin[df_2017_vin['TGL_DROPPING'] <= '2017-12-31']
          df_2017_vin.head()
```

Gambar 24 : Filter Data - Tanggal Dropping

4.2.5 Umur Pembiayaan / *Month of Book (MoB)*

Langkah selanjutnya adalah membuat kolom baru dengan nama MOB yang nanti akan diisi dengan keterangan umur pembiayaan atau *Month of Booking*. Umur pembiayaan dalam bentuk angkat bilangan bulat dengan satuan bulan dengan kata lain kolom ini menerangkan umur pembiyaan dalam bulan. Umur pembiayaan didapat dari selisih dari Tanggal Pencairan dengan Tanggal Data Pelaporan.

MOB = ((TANGGAL_DATA – TANGGAL_BUKA)/numpy.timedelta64(1, M)).

Data yang digunakan adalah data 24 bulan periode. Data setiap bulannya dibuatkan untuk kolom MOB ini sampai keseluruhan data yaitu data 24 bulan pelaporan. MOB digunakan menjadi matrix untuk menentukan kualitas disetiap bulan peaporan. Dengan MOB juga akan terlihat periode umur pembiayaan berapa suatu pembiayaan atas periode pencairan mulai terlihat memburuk. Memburuknya pembiayaan tersebut dapat disebabkan oleh buruknya kualitas dari produk ataupun operasional bank.

4.2.6 Flagging Periode Pencairan (Tanggal Dropping)

Tahapan ini membuat *flagging* untuk menentukan periode pencairan. Untuk dapat mengelompokan periode pencairan dari Tanggal Pencairan, diperlukan tambahan kolom untuk membuat *flagging* tersebut. Parameter yang digunakan adalah menentukan tanggal akhir bulan dari setiap Tanggal *Dropping*. Kolom ini yang akan menjadi *matrix* untuk pembuatan *vintage/cohort analysis*.

```
In [62]: df_2017_vin['EOM_DROPPING'] = pd.to_datetime(df_2017_vin['TGL_DROPPING'], format='%Y%m') + MonthEnd(1)
```

Gambar 25 : Flagging End of Month Tanggal *Dropping*

4.2.7 Agregasi Data

Tahap ini membuat sebuah dataframe baru hasil dari agregasi data berdasarkan beberapa kategori seperti :

DataFrame.groupby(by = None, axis = 0, level = None, as_index = True, sort = True, group_keys = True, squeeze = NoDefault.no_default, observed = False, dropna = True)

Penjelasan dari komponen kode *python* untuk melakukan agreagasi data dengan menggunakan Pustaka dari *Pandas* yaitu *groupby()* :

- ***by***

  Digunakan untuk menentukan kelompok untuk groupby. Jika by adalah suatu fungsi, itu dipanggil pada setiap nilai indeks objek.

Jika dict atau Seri dilewatkan, Seri atau *dict VALUES* akan digunakan untuk menentukan grup (nilai Seri pertama kali disejajarkan; lihat metode *.align()*). Jika daftar atau *array* dengan panjang yang sama dengan sumbu yang dipilih dilewatkan (lihat panduan grup menurut panduan pengguna), nilai akan digunakan apa adanya untuk menentukan grup. Label atau daftar label dapat diteruskan ke grup menurut kolom di dirinya sendiri. Perhatikan bahwa *tuple* ditafsirkan sebagai kunci (tunggal).

- ***Axis = {0 or 'index', 1 or 'columns'}, default 0***
- ***Level = int, level name, or sequence of such, default None***
- ***As_index = bool, default True***
- ***Sort = bool, default True***
- ***Group_keys = bool, default True***
- ***Squeeze = bool, default False***
- ***Dropna = bool, default True***

  Dropna ini akan mengelimininasi atau menghapus jika dari data terdapat nilai dari kolom itu *na (not available)* atau *null.*

1. Periode Data
2. Regional (Region, Area, dan Cabang)
3. Kolektibilitas
4. MoB *(Month of Book)*

4.2.8 Initial Limit

Initial Limit merupakan jumlah agregasi pencairan baru di setiap periode pencairan. Untuk mendapatkan nilai *initial limit* di dapat dengan menjumlah total NO_KARTU dengan ketentuan nilai MOB adalah 0 (nol). Dan di lakukan *groupby* EOM_DROPPING.

Initial Dropping Period

```
In [63]: init_17 = df_2017_vin.copy()
         init_17 = init_17[init_17['MOB'] == 0]
         init_17 = init_17.groupby(['EOM_DROPPING'])['NO_KARTU'].agg(['count']).reset_index()
         init_17 = init_17.rename(columns={'count':'INIT_NOA'})
```

```
In [64]: init_17
```

Out[64]:

	EOM_DROPPING	INIT_NOA
0	2017-01-31	1158
1	2017-02-28	742
2	2017-03-31	1795
3	2017-04-30	1714
4	2017-05-31	793
5	2017-06-30	4212
6	2017-07-31	4296
7	2017-08-31	3751
8	2017-09-30	3231
9	2017-10-31	4490
10	2017-11-30	5033
11	2017-12-31	5964

Gambar 26: Initial Limit

Penjelasan :

1. Baris kode pertama : Melakukan duplikasi dataframe agar tidak merusak susunan atau original dari data yang diolah.
2. Bari kode kedua : Melakukan filter data untuk kolom MOB dengan nilai 0 (nol)
3. Baris kode ketiga : Melakukan proses agregasi dengan menggunakan kode perintah *groupby()*. Jenis agregasi yang digunakan adalah *count* atau menjumlah banyaknya NO_KARTU. Di akhir kode dilakukan untuk *reset index* agar nomor indeks yang terbentuk tidak berantakan.
4. Baris kode keempat : Melakukan penamaan ulang nama kolom hasil dari agregasi hal ini untuk memudahkan dalam proses penggabungan atau *merge join* antar dataframe nantinya.
5. Baris kode kelima : Menampilkan hasil dari proses poin 1 sampai dengan 4.

4.2.9 Data Vintage DPD 1-30

Tahapan ini dengan menggunakan *Vintage Analysis* terkait kualitas pembiayaan pada DPD (*Day Payment Due*) atau Jumlah Hari Keterlambatan. DPD akan terbentuk terhitung sejak satu hari kalender setelah tanggal jatuh tempo jika sistem tidak berhasil melakukan pendebetan sejumlah uang pada rekening pembiayaan nasabah, baik itu tidak adanya dana ataupun kurangnya dana dari jumlah nominal minimal satu kali angsuran. DPD 1-30 adalah jumlah hari keterlambatan atas pembiayaan yang sudah jatuh tempo antara 1 sampai 30 hari.

Penjelasan tahapan proses sebagai berikut :

1. Tahap awal adalah melakukan duplikasi dataframe, agar tidak merusak originalitas datanya jika pada saat proses pengolahan mengalami masalah atau ada proses yang merubah nilai dari data.
2. Tahap selanjutnya adalah menyaring data pada kolom DPD dengan rentan hari keterlambatan antara 1 sampai dengan 30 hari.
3. Tahap selanjutnya yaitu melakukan agregasi dengan menggunakan fitur *pivot_table*. Agregasi ini adalah antara kolom EOM_DROPPING dengan MOB (umur pembiayaan) yang membuat nilai perhitungan jumlah NO_KARTU.
4. Melakukan *reset_index* untuk menjadikan hasil *pivot* menjadi 1 dimensi data yang mudah dibaca.

Vintage DPD 1-30 (30DPD)

In [33]:
```
df_2017_30dpd = df_2017_vin.copy()
df_2017_30dpd = df_2017_30dpd[df_2017_30dpd['DPD'] >= 1]
df_2017_30dpd = df_2017_30dpd[df_2017_30dpd['DPD'] <= 30]
```

Vintage DPD 1-30 (30DPD) - Pivot Table

In [67]:
```
pv_2017_30dpd = df_2017_30dpd.copy()
pv_2017_30dpd = pd.pivot_table(pv_2017_30dpd,
                               index='EOM_DROPPING',
                               columns='MOB',
                               values='NO_KARTU',
                               fill_value=0,
                               aggfunc='count')\
                               .reset_index()\
                               .rename_axis(None, axis=1)
pv_2017_30dpd
```

Out[67]:

	EOM_DROPPING	0	1	2	3	4	5	6	7	8	9	10	11
0	2017-01-31	4	8	3	3	4	2	5	4	7	11	5	4
1	2017-02-28	1	3	3	3	10	7	7	6	12	5	7	0
2	2017-03-31	9	10	3	11	8	7	13	9	12	9	0	0
3	2017-04-30	6	3	16	1	3	11	2	8	8	3	0	0
4	2017-05-31	3	11	3	8	6	7	5	5	0	0	0	0
5	2017-06-30	9	2	3	11	5	9	11	0	0	0	0	0
6	2017-07-31	0	3	3	13	2	1	0	0	0	0	0	0
7	2017-08-31	2	2	10	2	6	0	0	0	0	0	0	0
8	2017-09-30	1	4	0	6	0	0	0	0	0	0	0	0
9	2017-10-31	1	4	2	0	0	0	0	0	0	0	0	0
10	2017-11-30	2	4	0	0	0	0	0	0	0	0	0	0
11	2017-12-31	4	0	0	0	0	0	0	0	0	0	0	0

Gambar 27 : Pivot Table Vintage 1-30 DPD

5. Tahap berikutnya adalah melakukan *join* atau *merge* dataframe antara hasil *vintage DPD 1-30* dengan dataframe *initial_noa.*

Vintage DPD 1-30 (30DPD) - Merge Initial_Noa

```
vin17_30DPD = pd.merge(pv_2017_30dpd, init_17, how='left', on='EOM_DROPPING')
vin17_30DPD = vin17_30DPD[['EOM_DROPPING','INIT_NOA',0,1,2,3,4,5,6,7,8,9,10,11]]
```

```
vin18_30DPD = pd.merge(pv_2018_30dpd, init_18, how='left', on='EOM_DROPPING')
vin18_30DPD = vin18_30DPD[['EOM_DROPPING','INIT_NOA',0,1,2,3,4,5,6,7,8,9,10,11]]
```

Gambar 28 : Merge Initial Noa DPD 1-30

Hasil *vintage analysis DPD 1-30*

	EOM_DROPPING	INIT_NOA	0	1	2	3	4	5	6	7	8	9	10	11
0	2017-01-31 00:00:00	1158	4	8	3	3	4	2	5	4	7	11	5	4
1	2017-02-28 00:00:00	742	1	3	3	3	10	7	7	6	12	5	7	0
2	2017-03-31 00:00:00	1795	9	10	3	11	8	7	13	9	12	9	0	0
3	2017-04-30 00:00:00	1714	6	3	16	1	3	11	2	8	8	3	0	0
4	2017-05-31 00:00:00	793	3	11	3	8	6	7	5	5	0	0	0	0
5	2017-06-30 00:00:00	4212	9	2	3	11	5	9	11	0	0	0	0	0
6	2017-07-31 00:00:00	4296	0	3	3	13	2	1	0	0	0	0	0	0
7	2017-08-31 00:00:00	3751	2	2	10	2	6	0	0	0	0	0	0	0
8	2017-09-30 00:00:00	3231	1	4	0	6	0	0	0	0	0	0	0	0
9	2017-10-31 00:00:00	4490	1	4	2	0	0	0	0	0	0	0	0	0
10	2017-11-30 00:00:00	5033	2	4	0	0	0	0	0	0	0	0	0	0
11	2017-12-31 00:00:00	5964	4	0	0	0	0	0	0	0	0	0	0	0

Gambar 29 : Heatmap Vintage DPD 1-30 Tahun 2017

	EOM_DROPPING	INIT_NOA	0	1	2	3	4	5	6	7	8	9	10	11
0	2018-01-31 00:00:00	6246	2	5	8	33	135	115	28	11	11	7	6	4
1	2018-02-28 00:00:00	1502	3	3	9	167	112	20	4	4	6	10	6	0
2	2018-03-31 00:00:00	4059	0	16	239	297	31	19	19	6	8	3	0	0
3	2018-04-30 00:00:00	2421	0	49	59	31	3	2	1	4	1	0	0	0
4	2018-05-31 00:00:00	3128	0	12	11	2	4	1	6	0	0	0	0	0
5	2018-06-30 00:00:00	144	2	0	2	5	3	3	2	1	0	0	0	0
6	2018-07-31 00:00:00	67	5	1	3	2	1	4	0	0	0	0	0	0
7	2018-08-31 00:00:00	63	1	0	2	2	1	0	0	0	0	0	0	0
8	2018-09-30 00:00:00	42	4	1	1	0	1	0	0	0	0	0	0	0
9	2018-10-31 00:00:00	47	0	1	2	0	0	0	0	0	0	0	0	0
10	2018-11-30 00:00:00	64	0	1	0	0	0	0	0	0	0	0	0	0

Gambar 30 : Heatmap Vintage DPD 1-30 Tahun 2018

Dari hasil dapat dilihat bahwa *Trend* dengan warna biru tua menandakan bahwa populasi nasabah yang jatuh kualitasnya cukup besar di periode Bulan Maret 2017 dengan umur pembiayaan berusia 2 sampai 3 bulan. Dengan hasil ini dapat dipastikan bahwasanya harus dilakukan Analisa mendalam terkait penyebab tingginya nasabah yang jatuh ke DPD 1 – 30 hari pada MOB atau umur pembiayaan antara 2 – 4 bulan pertama sejak bulan pencairan pembiayaan.

4.2.10 Data Vintage DPD 31-60

DPD 31-60 adalah jumlah hari keterlambatan atas pembiayaan yang sudah jatuh tempo antara 31 sampai 60 hari.

Tahapan proses sebagai berikut :

1. Duplikasi dataframe terhadap dataframe *master* atau utama.
2. Tahap selanjutnya adalah menyaring data pada kolom DPD dengan rentan hari keterlambatan antara 31 sampai dengan 60 hari.

3. Melakukan *pivot table* dengan menggunakan *library* dari pandas yaitu pandas.pivot_table(namaDataframe, index=[], columns=[], values=[], fill_value=0, aggfunc='count')

Vintage DPD 31-60 (60DPD)

In [24]:
```
df_2017_60dpd = df_2017_vin.copy()
df_2017_60dpd = df_2017_60dpd[df_2017_60dpd['DPD'] >= 31]
df_2017_60dpd = df_2017_60dpd[df_2017_60dpd['DPD'] <= 60]
```

Vintage 31-60 (60DPD) : Pivot Table

In [25]:
```
pv_2017_60dpd = df_2017_60dpd.copy()
pv_2017_60dpd = pd.pivot_table(pv_2017_60dpd,
                               index='EOM_DROPPING',
                               columns='MOB',
                               values='NO_KARTU',
                               fill_value=0,
                               aggfunc='count').reset_index().rename_axis(None, axis=1)
pv_2017_60dpd
```

Out[25]:

	EOM_DROPPING	0	1	2	3	4	5	6	7	8	9	10
0	2017-01-31	1	2	1	1	2	4	2	1	1	0	0
1	2017-02-28	0	0	0	1	2	4	1	0	1	3	2
2	2017-03-31	2	1	2	1	1	5	1	1	1	2	1
3	2017-04-30	0	0	0	3	1	2	2	1	2	1	0
4	2017-05-31	0	0	3	1	1	4	3	2	0	0	0
5	2017-06-30	2	0	0	1	3	4	1	0	0	0	0
6	2017-07-31	0	0	1	1	0	0	0	0	0	0	0
7	2017-08-31	1	0	1	0	0	0	0	0	0	0	0
8	2017-09-30	0	0	1	0	0	0	0	0	0	0	0
9	2017-10-31	0	1	1	0	0	0	0	0	0	0	0

Gambar 31 : Pivot Table DPD 31-60

4. Tahap berikutnya melakukan *merge* atau *join dataframe* antara hasil dari *vintage* DPD 31-60 dengan *dataframe* init_noa.

Vintage DPD 31-60 (60DPD) - Merge Initial_Noa

```
vin_60DPD = pd.merge(pv_2017_60dpd, init_17, how='left', on='EOM_DROPPING')
vin_60DPD = vin_60DPD[['EOM_DROPPING','INIT_NOA',0,1,2,3,4,5,6,7,8,9,10]]
```

```
vin18_60DPD = pd.merge(pv_2018_60dpd, init_18, how='left', on='EOM_DROPPING')
vin18_60DPD = vin18_60DPD[['EOM_DROPPING','INIT_NOA',0,1,2,3,4,5,6,7,8,9,10]]
```

Gambar 32 : Merge Initial Noa 31-60 DPD

Hasil keluaran (*output*) :

	EOM_DROPPING	INIT_NOA	0	1	2	3	4	5	6	7	8	9	10
0	2017-01-31 00:00:00	1158	1	2	1	1	2	4	2	1	1	0	0
1	2017-02-28 00:00:00	742	0	0	0	1	2	4	1	0	1	3	2
2	2017-03-31 00:00:00	1795	2	1	2	1	1	5	1	1	1	2	1
3	2017-04-30 00:00:00	1714	0	0	0	3	1	2	2	1	2	1	0
4	2017-05-31 00:00:00	793	0	0	3	1	1	4	3	2	0	0	0
5	2017-06-30 00:00:00	4212	2	0	0	1	3	4	1	0	0	0	0
6	2017-07-31 00:00:00	4296	0	0	1	1	0	0	0	0	0	0	0
7	2017-08-31 00:00:00	3751	1	0	1	0	0	0	0	0	0	0	0
8	2017-09-30 00:00:00	3231	0	0	1	0	0	0	0	0	0	0	0
9	2017-10-31 00:00:00	4490	0	1	1	0	0	0	0	0	0	0	0

Gambar 33 : Heatmap Vintage DPD 31-60 Tahun 2017

	EOM_DROPPING	INIT_NOA	0	1	2	3	4	5	6	7	8	9	10
0	2018-01-31 00:00:00	6246	0	0	3	0	14	162	112	23	2	1	3
1	2018-02-28 00:00:00	1502	0	1	0	6	159	111	23	12	0	0	4
2	2018-03-31 00:00:00	4059	2	0	3	260	186	85	15	2	2	3	2
3	2018-04-30 00:00:00	2421	0	0	42	57	16	1	2	0	2	0	0
4	2018-05-31 00:00:00	3128	0	2	10	6	3	2	1	3	0	0	0
5	2018-06-30 00:00:00	144	0	0	0	0	3	1	0	0	0	0	0
6	2018-07-31 00:00:00	67	0	2	0	2	1	0	0	0	0	0	0
7	2018-08-31 00:00:00	63	0	0	1	1	2	0	0	0	0	0	0
8	2018-09-30 00:00:00	42	1	0	1	1	0	0	0	0	0	0	0
9	2018-10-31 00:00:00	47	2	1	1	0	0	0	0	0	0	0	0

Gambar 34 : Heatmap Vintage DPD 31-60 Tahun 2018

4.2.11 Data Vintage DPD 61-90

DPD 61-90 adalah jumlah hari keterlambatan atas pembiayaan yang sudah jatuh tempo antara 61 sampai 90 hari.

Tahapan proses sebagai berikut :

1. Duplikasi dataframe master
2. Tahap selanjutnya adalah menyaring data pada kolom DPD dengan rentan hari keterlambatan antara 61 sampai dengan 90 hari.
3. Melakukan *pivot table* dengan menggunakan *library* dari pandas yaitu pandas.pivot_table(namaDataframe, index=[], columns=[], values=[], fill_value=0, aggfunc='count')
4. Tahap selanjutnya yaitu melakukan agregasi dengan menggunakan fitur *pivot_table*. Agregasi ini adalah antara kolom EOM_DROPPING dengan MOB (umur pembiayaan) yang membuat nilai perhitungan jumlah NO_KARTU.
5. Melakukan *reset_index* untuk menjadikan hasil *pivot* menjadi 1 dimensi data yang mudah dibaca.

Kode *python* :

Vintage 61-90 (90DPD)

In [29]:
```
df_2017_90dpd = df_2017_vin.copy()
df_2017_90dpd = df_2017_90dpd[df_2017_90dpd['DPD'] >= 61]
df_2017_90dpd = df_2017_90dpd[df_2017_90dpd['DPD'] <= 90]
```

Vintage 61-90 (90DPD) : Pivot Table

In [30]:
```
pv_2017_90dpd = df_2017_90dpd.copy()
pv_2017_90dpd = pd.pivot_table(pv_2017_90dpd,
                               index='EOM_DROPPING',
                               columns='MOB',
                               values='NO_KARTU',
                               fill_value=0,
                               aggfunc='count').reset_index().rename_axis(None, axis=1)
pv_2017_90dpd
```

Out[30]:

	EOM_DROPPING	0	1	2	3	4	5	6	7	8	9	10
0	2017-01-31	0	0	1	0	0	0	2	0	0	1	1
1	2017-02-28	0	0	0	0	0	1	1	1	0	1	2
2	2017-03-31	0	1	1	3	1	2	3	3	2	3	0
3	2017-04-30	0	0	0	0	2	1	0	1	1	0	0
4	2017-05-31	0	0	0	1	0	0	1	2	0	0	0
5	2017-06-30	0	1	0	0	0	1	2	0	0	0	0
6	2017-07-31	0	0	0	1	1	0	0	0	0	0	0
7	2017-08-31	0	1	0	0	0	0	0	0	0	0	0
8	2017-09-30	17	0	0	0	0	0	0	0	0	0	0
9	2017-10-31	2	0	0	0	0	0	0	0	0	0	0

Gambar 35 : Pivot Table DPD 61-90

6. Selanjutnya melakukan *join* atau *merge* dataframe *initial_noa* dengan hasil *Vintage DPD 61-90.*

Vintage DPD 61-90 (90DPD) - Merge Initial_Noa

```
vin_90DPD = pd.merge(pv_2017_90dpd, init_17, how='left', on='EOM_DROPPING')
vin_90DPD = vin_90DPD[['EOM_DROPPING','INIT_NOA',0,1,2,3,4,5,6,7,8,9,10]]
```

```
vin18_90DPD = pd.merge(pv_2018_90dpd, init_18, how='left', on='EOM_DROPPING')
vin18_90DPD = vin18_90DPD[['EOM_DROPPING','INIT_NOA',0,1,2,3,4,5,6,7,8,9,10]]
```

Gambar 36 : Merge Initial Noa DPD 61-90

Vintage Analysis 90 DPD :

	EOM_DROPPING	INIT_NOA	0	1	2	3	4	5	6	7	8	9	10
0	2017-01-31 00:00:00	1158	0	0	1	0	0	0	2	0	0	1	1
1	2017-02-28 00:00:00	742	0	0	0	0	0	1	1	1	0	1	2
2	2017-03-31 00:00:00	1795	0	1	1	3	1	2	3	3	2	3	0
3	2017-04-30 00:00:00	1714	0	0	0	0	2	1	0	1	1	0	0
4	2017-05-31 00:00:00	793	0	0	0	1	0	0	1	2	0	0	0
5	2017-06-30 00:00:00	4212	0	1	0	0	0	1	2	0	0	0	0
6	2017-07-31 00:00:00	4296	0	0	0	1	1	0	0	0	0	0	0
7	2017-08-31 00:00:00	3751	0	1	0	0	0	0	0	0	0	0	0
8	2017-09-30 00:00:00	3231	17	0	0	0	0	0	0	0	0	0	0
9	2017-10-31 00:00:00	4490	2	0	0	0	0	0	0	0	0	0	0

Gambar 37: Heatmap Vintage DPD 90 Tahun 2017

Hasil dari *Vintage Analysis 90DPD* ini dapat dilihat pada periode pencairan Bulan Maret menujukan adanya penurunan kualitas pembiayaan dengan ditandainya sudah ada beberapa nasabah yang turun ke kolektibilitas 2 DPD 61-90.

	EOM_DROPPING	INIT_NOA	0	1	2	3	4	5	6	7	8	9	10
0	2018-01-31 00:00:00	6246	0	1	0	3	0	14	155	107	19	2	1
1	2018-02-28 00:00:00	1502	0	0	1	0	0	161	104	10	3	1	1
2	2018-03-31 00:00:00	4059	0	1	1	1	233	211	95	1	0	1	0
3	2018-04-30 00:00:00	2421	0	0	0	34	47	27	0	1	1	0	0
4	2018-05-31 00:00:00	3128	0	0	0	12	7	0	2	0	0	0	0
5	2018-06-30 00:00:00	144	0	0	0	0	0	3	1	0	0	0	0
6	2018-07-31 00:00:00	67	0	0	1	0	0	0	0	0	0	0	0
7	2018-08-31 00:00:00	63	0	0	0	1	1	0	0	0	0	0	0
8	2018-10-31 00:00:00	47	0	1	0	0	0	0	0	0	0	0	0
9	2018-12-31 00:00:00	43	1	0	0	0	0	0	0	0	0	0	0

Gambar 38 : Heatmap Vintage DPD 61-90 Tahun 2018