Module - 2 - Building DataSet + Feature Engineering

In [ ]:
```
import os
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
%matplotlib inline
sns.set_style('darkgrid', {'axes.facecolor': '1'})
# Run Module-1 Key functions
%run Module-1-functions.py
```

In [ ]:
```
(customer_profiles_df,
 station_profiles_df,
 transactions_df)=Simulate_dataset(
     n_customers = 10000,
     n_stations = 10000,
    nb_days=150,
    start_date="2022-01-01",
    r=7)
```

In [ ]:
```
transactions_df = Simulate_frauds(customer_profiles_df,
                                  station_profiles_df,
                                  transactions_df)
```

In [ ]:
```
transactions_df
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[ ]:

	TRANSACTION_ID	Trans_DATETIME	CUSTOMER_ID	STORE_ID	Trans_AMOUNT	Trans_TIME_SECONDS	Trans_TIME_DAYS
0	0	2022-01-01 00:00:17	5820	2647	17.99	17	0
1	1	2022-01-01 00:00:17	6160	2980	44.48	17	0
2	2	2022-01-01 00:00:30	356	752	86.87	30	0
3	3	2022-01-01 00:00:31	1829	2266	16.65	31	0
4	4	2022-01-01 00:00:31	596	2344	98.33	31	0
...	...	...	...	...	...	...	...
3631920	3631920	2022-05-30 23:59:08	3779	6482	63.78	12959948	149
3631921	3631921	2022-05-30 23:59:11	2051	7172	17.76	12959951	149
3631922	3631922	2022-05-30 23:59:27	6519	3400	54.42	12959967	149
3631923	3631923	2022-05-30 23:59:39	304	338	54.93	12959979	149
3631924	3631924	2022-05-30 23:59:52	6986	8432	88.5	12959992	149

3631925 rows × 8 columns

Module - 2 - Feature Engineering

In [ ]:

```
%run Module-2-helper.py
# Below is the Module-2-helper function to build the new features
# -------------------------------------------------------------------------------------- Feature Engineering
# -------------------------------------------------------
# Binary Output : Whether a day is during weekend or during weekday
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
    weekday = tx_datetime.weekday()
    is_weekend = weekday>=5
    return int(is_weekend)

# ---------------------------------------------------
# Binary Output: Whether the transaction happens during night
def is_night(tx_datetime):
    tx_hour = tx_datetime.hour
    is_night = tx_hour<=8
    return int(is_night)

# ---------------------------------------------------
# define a function computing the average transaction amount in each window size (Customer Views)
def compute_avg_amt(C_T, window):
    for window_size in window:
        # Compute the SUM
        _SUM = C_T['Trans_AMOUNT'].rolling(str(window_size)+'d').sum()
        _WIND = C_T['Trans_AMOUNT'].rolling(str(window_size)+'d').count()
        # Compute the AVG
        _AVG = _SUM/_WIND
        # Saving
        C_T['WIND_Trans_'+str(window_size)+'DAY']=list(_WIND)
        C_T['AVG_AMOUNT_'+str(window_size)+'DAY']=list(_AVG)

def customers_features(C_T, window=[1,7,30]):
    # Order transactions chronologically
    C_T=C_T.sort_values('Trans_DATETIME')
    C_T.index=C_T.Trans_DATETIME
    compute_avg_amt(C_T, window)
    # Reindex according to transaction IDs
    C_T.index=C_T.TRANSACTION_ID
    # And return the dataframe with the new features
    return C_T

# ---------------------------------------------------
# define a function computing the average transaction amount in each window size (STORE Views)
def store_related_features(store_T, delay_period, window, feature, NB_FRAUD_DELAY, NB_Trans_DELAY):
    for window_size in window:
        NB_FRAUD=store_T['Trans_FRAUD'].rolling(str(delay_period+window_size)+'d').sum()
        NB_DELAY=store_T['Trans_FRAUD'].rolling(str(delay_period+window_size)+'d').count()
        NB_FRAUD_WINDOW=NB_FRAUD-NB_FRAUD_DELAY
        NB_Trans_WINDOW=NB_DELAY-NB_Trans_DELAY
        RISK_WINDOW=NB_FRAUD_WINDOW/NB_Trans_WINDOW
        store_T[feature+'_NB_Trans_'+str(window_size)+'DAY_WINDOW']=list(NB_Trans_WINDOW)
        store_T[feature+'_RISK_'+str(window_size)+'DAY_WINDOW']=list(RISK_WINDOW)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
def window_rolling_features(store_T, delay_period=7, window=[1,7,30], feature="STORE_ID"):
    store_T=store_T.sort_values('Trans_DATETIME')
    store_T.index=store_T.Trans_DATETIME
    NB_FRAUD_DELAY=store_T['Trans_FRAUD'].rolling(str(delay_period)+'d').sum()
    NB_Trans_DELAY=store_T['Trans_FRAUD'].rolling(str(delay_period)+'d').count()
    store_related_features(store_T, delay_period, window, feature, NB_FRAUD_DELAY, NB_Trans_DELAY)
    store_T.index=store_T.TRANSACTION_ID
    # Replace NA values with 0 (all undefined risk scores where NB_Trans_WINDOW is 0)
    store_T.fillna(0,inplace=True)
    return store_T
```

In [ ]:

```
transactions_df.head()
```

Out[ ]:

	TRANSACTION_ID	Trans_DATETIME	CUSTOMER_ID	STORE_ID	Trans_AMOUNT	Trans_TIME_SECONDS	Trans_TIME_DAYS	Trans_
0	0	2022-01-01 00:00:17	5820	2647	17.99	17	0	
1	1	2022-01-01 00:00:17	6160	2980	44.48	17	0	
2	2	2022-01-01 00:00:30	356	752	86.87	30	0	
3	3	2022-01-01 00:00:31	1829	2266	16.65	31	0	
4	4	2022-01-01 00:00:31	596	2344	98.33	31	0	

In [ ]:

```
transactions_df['Trans_DURING_WEEKEND']=transactions_df.Trans_DATETIME.apply(is_weekend)
transactions_df['Trans_DURING_NIGHT']=transactions_df.Trans_DATETIME.apply(is_night)
transactions_df[transactions_df.Trans_TIME_DAYS>=40]
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[ ]:

	TRANSACTION_ID	Trans_DATETIME	CUSTOMER_ID	STORE_ID	Trans_AMOUNT	Trans_TIME_SECONDS	Trans_TIME_DAYS
968873	968873	2022-02-10 00:00:23	7888	1642	23.07	3456023	40
968874	968874	2022-02-10 00:00:30	3066	6134	116.18	3456030	40
968875	968875	2022-02-10 00:01:07	7948	2572	68.97	3456067	40
968876	968876	2022-02-10 00:01:10	9791	9134	45.8	3456070	40
968877	968877	2022-02-10 00:02:25	1702	4933	80.24	3456145	40
...	...	...	...	...	...	...	...
3631920	3631920	2022-05-30 23:59:08	3779	6482	63.78	12959948	149
3631921	3631921	2022-05-30 23:59:11	2051	7172	17.76	12959951	149
3631922	3631922	2022-05-30 23:59:27	6519	3400	54.42	12959967	149
3631923	3631923	2022-05-30 23:59:39	304	338	54.93	12959979	149
3631924	3631924	2022-05-30 23:59:52	6986	8432	88.5	12959992	149

2663052 rows × 10 columns

In [ ]:

```
transactions_df=transactions_df.groupby('CUSTOMER_ID').apply(lambda x: customers_features(x, windows_size_in_(
transactions_df=transactions_df.sort_values('Trans_DATETIME').reset_index(drop=True)
transactions_df.head()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[ ]:

	TRANSACTION_ID	Trans_DATETIME	CUSTOMER_ID	STORE_ID	Trans_AMOUNT	Trans_TIME_SECONDS	Trans_TIME_DAYS	Trans_
0	0	2022-01-01 00:00:17	5820	2647	17.99	17	0	
1	1	2022-01-01 00:00:17	6160	2980	44.48	17	0	
2	2	2022-01-01 00:00:30	356	752	86.87	30	0	
3	3	2022-01-01 00:00:31	1829	2266	16.65	31	0	
4	5	2022-01-01 00:00:31	6820	8046	128.04	31	0	

In [ ]:

```
transactions_df=transactions_df.groupby('STORE_ID').apply(lambda x: window_rolling_features(x, delay_period=7
transactions_df=transactions_df.sort_values('Trans_DATETIME').reset_index(drop=True)
```

Output Data

In [ ]:

```
# Outputing the data for future anaylysis, for the model building, we are not using the entire data
# we only use part of the data for training, and a valid gap period, and then a test set
OUTPUT = "./simulated-data-transformed/"
if not os.path.exists(OUTPUT):
    os.makedirs(OUTPUT)
start_date = datetime.datetime.strptime("2022-01-01", "%Y-%m-%d")
for day in range(transactions_df.Trans_TIME_DAYS.max()+1):
    transactions_day = transactions_df[
        transactions_df.Trans_TIME_DAYS==day].sort_values('Trans_TIME_SECONDS')
    date = start_date + datetime.timedelta(days=day)
    filename_output = date.strftime("%Y-%m-%d")+'.pkl'
    transactions_day.to_pickle(OUTPUT+filename_output)
```

In [ ]:

```
transactions_df
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[ ]:

	TRANSACTION_ID	Trans_DATETIME	CUSTOMER_ID	STORE_ID	Trans_AMOUNT	Trans_TIME_SECONDS	Trans_TIME_DAYS
0	0	2022-01-01 00:00:17	5820	2647	17.99	17	0
1	1	2022-01-01 00:00:17	6160	2980	44.48	17	0
2	2	2022-01-01 00:00:30	356	752	86.87	30	0
3	5	2022-01-01 00:00:31	6820	8046	128.04	31	0
4	4	2022-01-01 00:00:31	596	2344	98.33	31	0
...	...	...	...	...	...	...	...
3631920	3631920	2022-05-30 23:59:08	3779	6482	63.78	12959948	149
3631921	3631921	2022-05-30 23:59:11	2051	7172	17.76	12959951	149
3631922	3631922	2022-05-30 23:59:27	6519	3400	54.42	12959967	149
3631923	3631923	2022-05-30 23:59:39	304	338	54.93	12959979	149
3631924	3631924	2022-05-30 23:59:52	6986	8432	88.50	12959992	149

3631925 rows × 22 columns

In [ ]:

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js