Problem Statement:

Part-A:

1. Given a data to be available in an OpenFDA API (https://open.fda.gov/apis/drug/label/). The End user need to gather is data.
2. Transform the data as you see it.
3. Visualize and explore the results.

Part-B:

1. Calculate the average number of ingredients/year and per delivery route.

Optional:

1. Predict the total number of ingredients for the next year.
2. Most common drug interaction

Solutions:

1. Gathering of Data from API:

   Tools Used – Talend (Open Source)

   I have created a Talend job that will hit the OpenFDA API, pull the data, and save it in the form of csv. As the API as a flexibility to have at most limit of 100, I have constructed a loop based job which will internally use skip and limit and pull the records and save it in the form of csv into a directory.

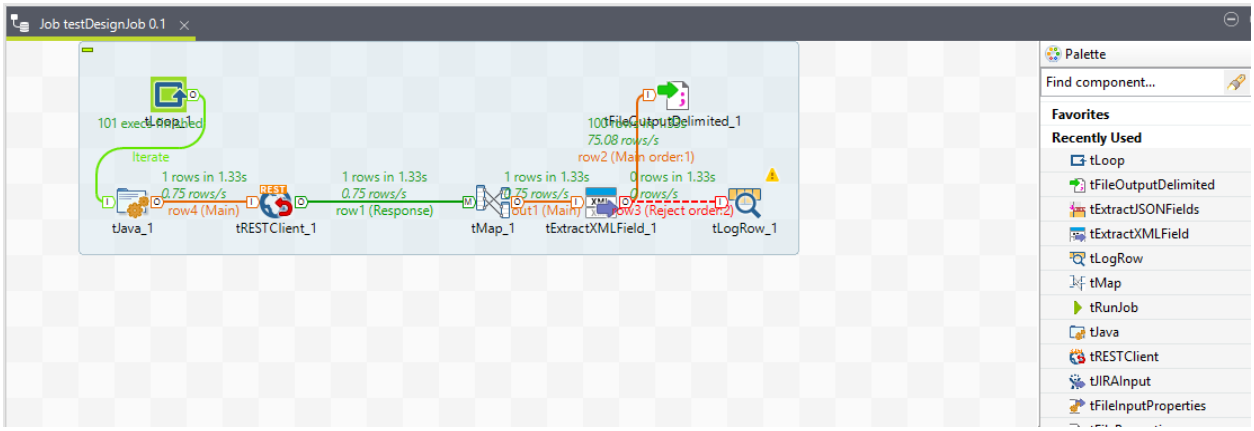
   API used -
   "https://api.fda.gov/drug/label.json?search=effective_time:[20110101+TO+20190630]&skip="+(context.initValue * 100)+"&limit=100";

   As the API response is in JSON, with Talend we can easily do the parsing and extracting the necessary fields in required fashion.

   Here, I have extracted the drug information for the period Jan-2011 to Jun-2019 in each iteration the skip will incremented and the limit is set to 100.

   When the solution moves to the production mode, this job could be configured in scheduler mode (via windows scheduler or cron job) which will pull the data at regular intervals.

Screenshots:



2. <u>Fields selection and visualization:</u>

From the API response, the assumptions made are

- Id – unique id of the product.
- effective_time – the medicine/drug created and started using on date.
- spl_product_data_elements – ingredients used, which is comma separated.
- Openfda.generic_name - the name of the drug/medicine.
- Openfda.route - the route/intake mechanism of the medicine.

Based on the above assumptions, and for the period Jan-2011 to Jun-2019 and for the first 10,000 records the expected outputs are below

In [12]:
```
groupedFrame = frame.groupby(['year','drugName'])['number_of_ingredients'].mean().reset_index(name="avg_no_of_ingredients")
groupedFrame.head()
```

Out[12]:

	year	drugName	avg_no_of_ingredients
0	2011	ABIES NIG., AESCULUS HIPP., ALOE, ALUMINA, BRY...	30.0
1	2011	ABIES NIGRA, ACONITUM NAPELLUS, ARSENICUM IODA...	12.0
2	2011	ABSINTHIUM, ACONITUM NAP., ANTIMON. TART., BEL...	29.0
3	2011	ACETAMINOPHEN	2.0
4	2011	ACETAMINOPHEN AND CODEINE PHOSPHATE	3.0

```
In [15]: groupedFrame_1 = frame.groupby(['year','route'])['number_of_ingredients'].mean().reset_index(name="avg_no_of_ingredients")
         groupedFrame_1.head()
```

Out[15]:

	year	route	avg_no_of_ingredients
0	2011	CUTANEOUS	1.250000
1	2011	DENTAL	1.166667
2	2011	INTRAMUSCULAR	1.000000
3	2011	INTRAVENOUS	1.000000
4	2011	NASAL	1.500000

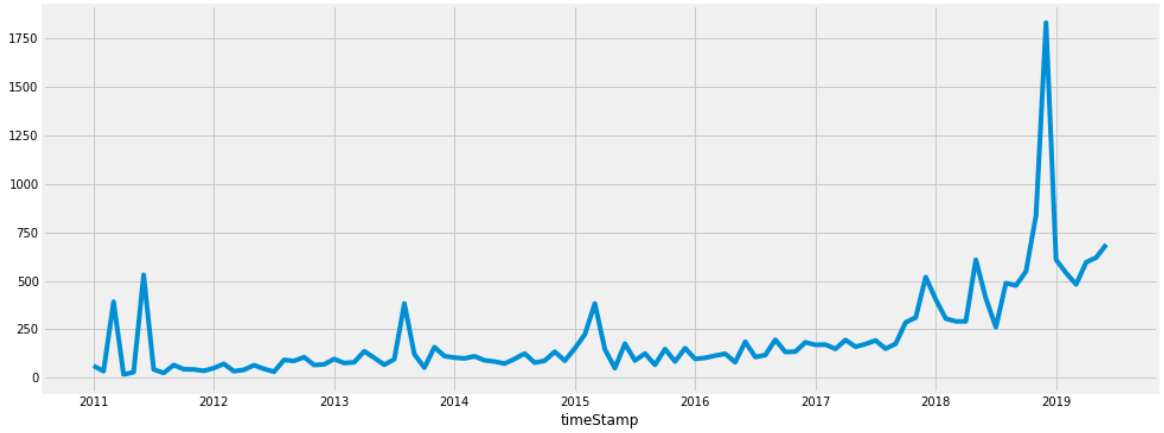
3. Time Series Analysis for the predicting the number of Ingredients for Next Year:

To predict the Average Number of Ingredients for next year, I would compare this to forecasting the future via the time-series model.

Algorithm used – ARIMA models.

Libraries used – Pandas, numpy, statsmodel.api (for ARIMA model)

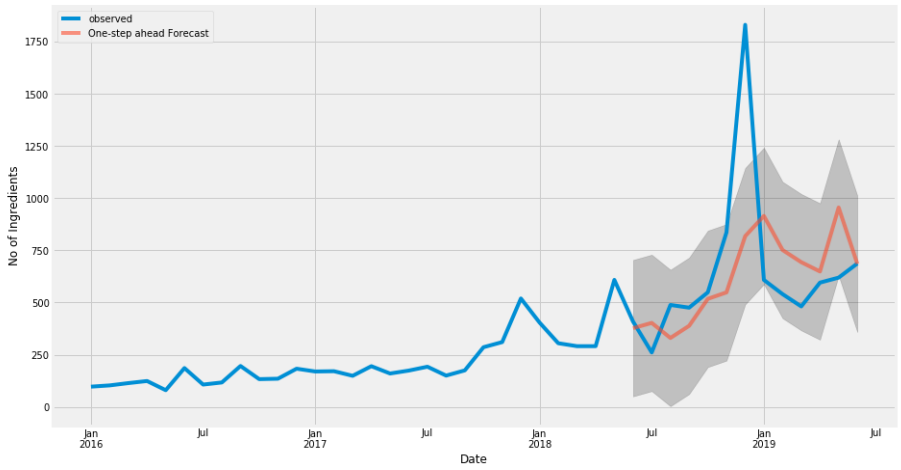
Based on the seasonality behavior of the ingredients flow month on month across the different years, we would use the previous year/month's data to predict/forecast the future.



The Above visualization shows the seasonality behavior of the number of ingredients that are in use from 2011 to 2019 (with 10k records).

Based on this, we can infer that on each year's mid there is been some decrease in the usage of ingredients except in 2014.

I have created an ARIMA model and predicted for the period Jun-2018 to 2019.



From the above prediction results, we could infer that from July'18, there is a slight decrease and then an increasing fashion of ingredients and for the Nov'18 and Dec'18 the predictions are not that accurate, however predicted that there is an increase in ingredients as well decrease in ingredients.

```
In [27]: y_forecasted = pred.predicted_mean
         print("Predictions:")
         print(y_forecasted.head())
         print("")
         print("True Value:")
         y["2018-06":][:5]

         Predictions:
         timeStamp
         2018-06-01    377.288665
         2018-07-01    402.437899
         2018-08-01    330.281195
         2018-09-01    388.456520
         2018-10-01    517.707800
         Freq: MS, dtype: float64

         True Value:

Out[27]: timeStamp
         2018-06-01    409.0
         2018-07-01    261.0
         2018-08-01    488.5
         2018-09-01    475.5
         2018-10-01    549.0
         Freq: MS, Name: avg_no_of_ingredients, dtype: float64
```

4. <u>Most Common drug interactions:</u>

I consider the most common drug interaction to be the most common drug across the period of Jan'11 to Jun'19.

```
In [47]: common_interactions = frame.groupby(['drugName'])['year'].apply(set).reset_index()
         common_interactions['count'] = common_interactions['year'].apply(lambda x: len(set(x)))
         common_interactions.sort_values('count',ascending=False).head()
```

Out[47]:

	drugName	year	count
995	CHLOROXYLENOL	{2016, 2017, 2018, 2019, 2011, 2012, 2013, 201...	9
2425	SALICYLIC ACID	{2016, 2017, 2018, 2019, 2011, 2012, 2013, 201...	9
2610	TITANIUM DIOXIDE	{2016, 2017, 2018, 2019, 2011, 2012, 2013, 201...	9
260	ALUMINUM ZIRCONIUM TETRACHLOROHYDREX GLY	{2016, 2017, 2018, 2019, 2011, 2012, 2013, 201...	9
2650	TRICLOSAN	{2016, 2017, 2018, 2019, 2011, 2012, 2013, 201...	9

5. Productionizing the model for E2E :

   For the E2E I would suggest to go with an API based model creation and model testing/predicting. Hence, I would go with “FLASK” based solution to model creation and for the prediction phase.