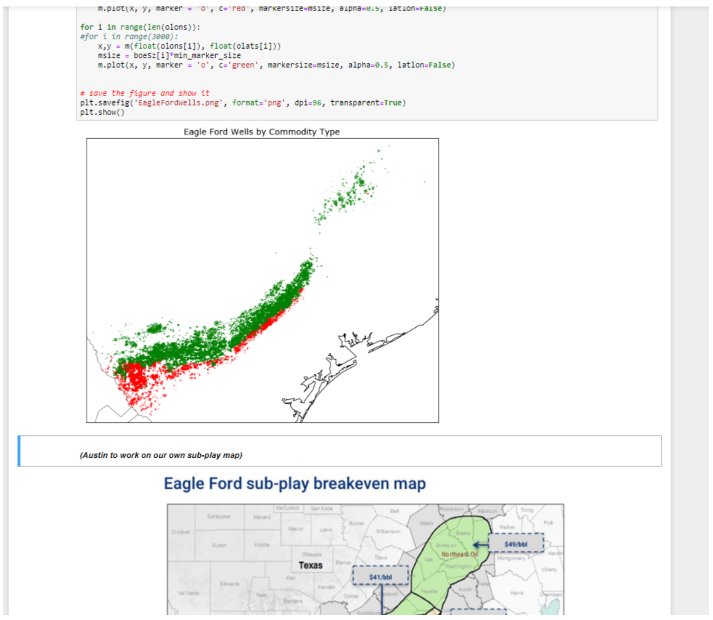
MAPS: Austin to work on our own sub-play map

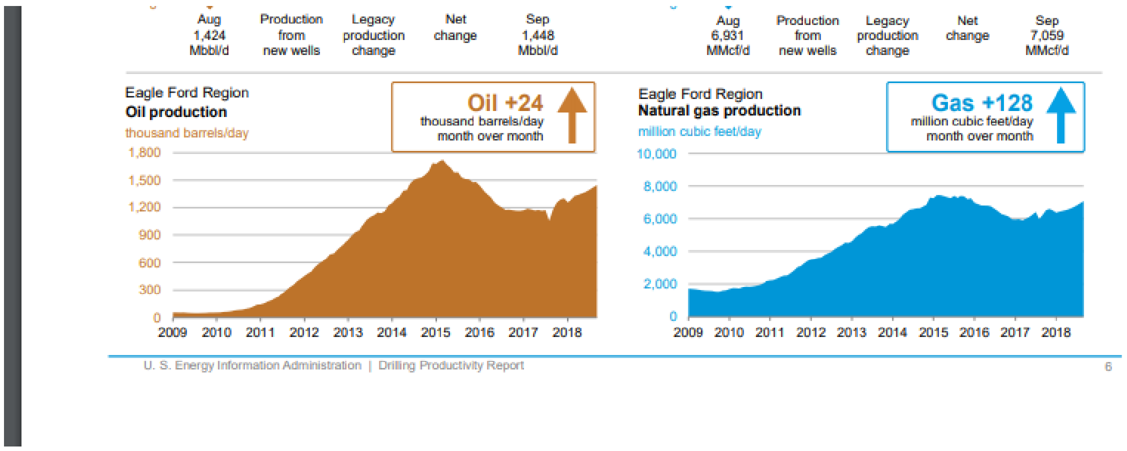


```
    m.plot(x, y, marker = 'o', c='red', markersize=msize, alpha=0.5, latlon=False)

for i in range(len(olons)):
#for i in range(3000):
    x,y = m(float(olons[i]), float(olats[i]))
    msize = boeSz[i]*min_marker_size
    m.plot(x, y, marker = 'o', c='green', markersize=msize, alpha=0.5, latlon=False)


# save the figure and show it
plt.savefig('EagleFordWells.png', format='png', dpi=96, transparent=True)
plt.show()
```

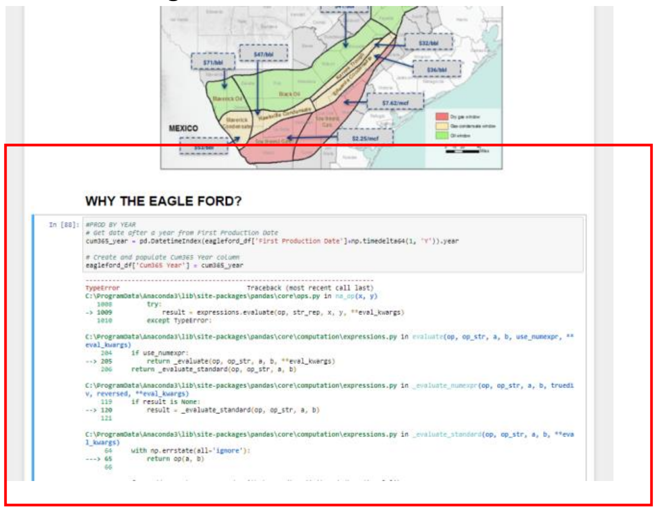
(Austin to work on our own sub-play map)

Eagle Ford sub-play breakeven map

WHY THE EAGLE FORD: David, we can put your PRODUCTION BY YEAR chart there. Currently, its breaking in the master because a TypeError (`TypeError: must be str, not numpy.timedelta64`). A good graph to compare yours to is below. It dropped in 2015 due to the price of crashing.



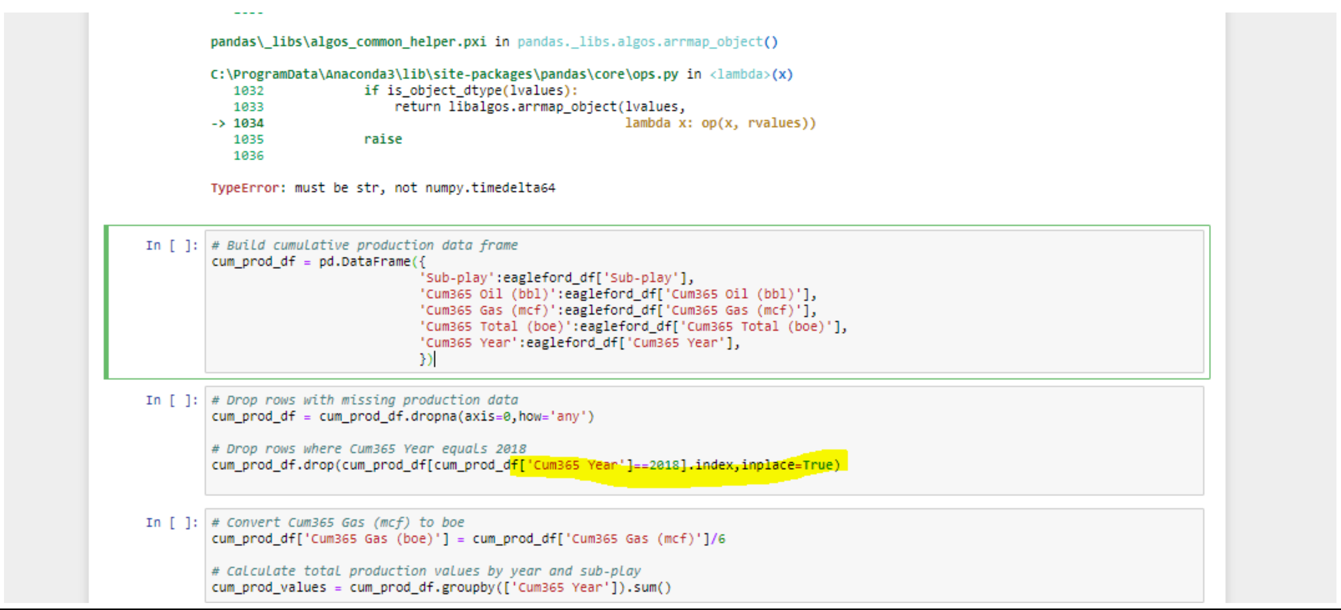
I've pasted your code in, it just needs to be tweaked. I would take your date back to 2017 maybe. Production in only current through June 2018. Our newest wells came on in October 2017 (9 months).



WHY THE EAGLE FORD?

```
In [88]: #PROD BY YEAR
         # Get date after a year from First Production Date
         cum365_year = pd.DatetimeIndex(eagleford_df['First Production Date']+np.timedelta64(1, 'Y')).year

         # Create and populate Cum365 Year column
         eagleford_df['Cum365 Year'] = cum365_year
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops.py in na_op(x, y)
   1008         try:
-> 1009             result = expressions.evaluate(op, str_rep, x, y, **eval_kwargs)
   1010         except TypeError:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\computation\expressions.py in evaluate(op, op_str, a, b, use_numexpr, **
eval_kwargs)
    204     if use_numexpr:
--> 205         return _evaluate(op, op_str, a, b, **eval_kwargs)
    206     return _evaluate_standard(op, op_str, a, b)

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\computation\expressions.py in _evaluate_numexpr(op, op_str, a, b, truedi
v, reversed, **eval_kwargs)
    119     if result is None:
--> 120         result = _evaluate_standard(op, op_str, a, b)
    121

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\computation\expressions.py in _evaluate_standard(op, op_str, a, b, **eva
l_kwargs)
     64     with np.errstate(all='ignore'):
---> 65         return op(a, b)
     66
```



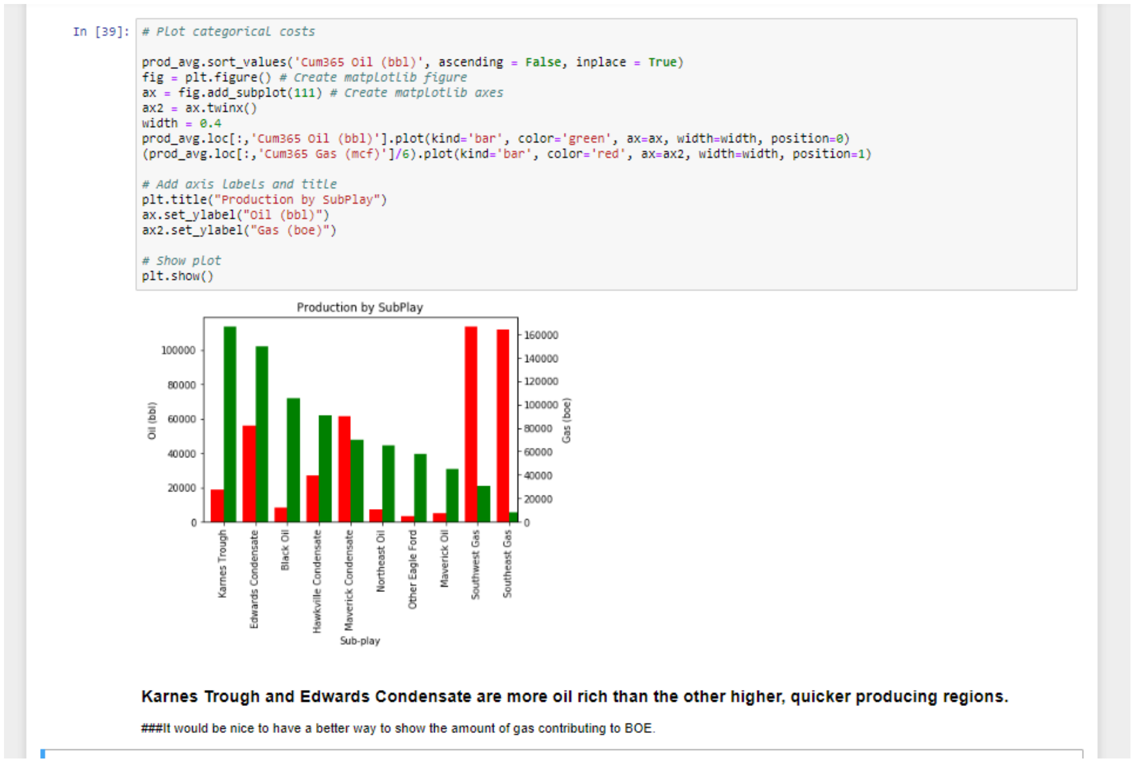
```
    ----
pandas\_libs\algos_common_helper.pxi in pandas._libs.algos.arrmap_object()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\ops.py in <lambda>(x)
   1032             if is_object_dtype(lvalues):
   1033                 return libalgos.arrmap_object(lvalues,
-> 1034                                               lambda x: op(x, rvalues))
   1035             raise
   1036

TypeError: must be str, not numpy.timedelta64
```

```
In [ ]: # Build cumulative production data frame
        cum_prod_df = pd.DataFrame({
                                    'Sub-play':eagleford_df['Sub-play'],
                                    'Cum365 Oil (bbl)':eagleford_df['Cum365 Oil (bbl)'],
                                    'Cum365 Gas (mcf)':eagleford_df['Cum365 Gas (mcf)'],
                                    'Cum365 Total (boe)':eagleford_df['Cum365 Total (boe)'],
                                    'Cum365 Year':eagleford_df['Cum365 Year'],
                                    })
```

```
In [ ]: # Drop rows with missing production data
        cum_prod_df = cum_prod_df.dropna(axis=0,how='any')

        # Drop rows where Cum365 Year equals 2018
        cum_prod_df.drop(cum_prod_df[cum_prod_df['Cum365 Year']==2018].index,inplace=True)
```

```
In [ ]: # Convert Cum365 Gas (mcf) to boe
        cum_prod_df['Cum365 Gas (boe)'] = cum_prod_df['Cum365 Gas (mcf)']/6

        # Calculate total production values by year and sub-play
        cum_prod_values = cum_prod_df.groupby(['Cum365 Year']).sum()
```

PRODUCTION: It would be nice to find a graph the shows the amount of oil and gas contributing to boe. The graph below is not quite the same scale. Gas is considered an economic equivalent once divided by 6, so we can normalize that value.

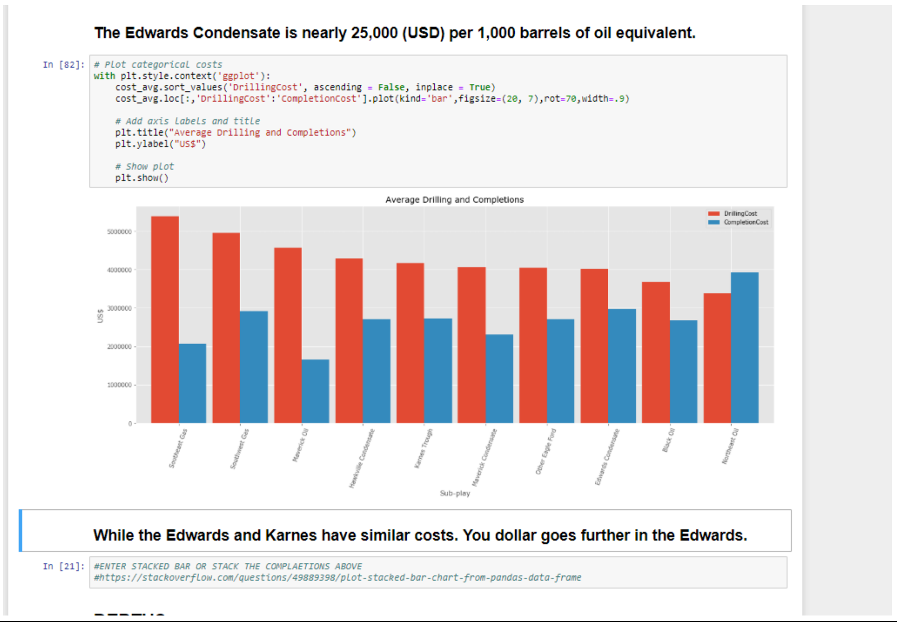


```
In [39]: # Plot categorical costs

         prod_avg.sort_values('Cum365 Oil (bbl)', ascending = False, inplace = True)
         fig = plt.figure() # Create matplotlib figure
         ax = fig.add_subplot(111) # Create matplotlib axes
         ax2 = ax.twinx()
         width = 0.4
         prod_avg.loc[:,'Cum365 Oil (bbl)'].plot(kind='bar', color='green', ax=ax, width=width, position=0)
         (prod_avg.loc[:,'Cum365 Gas (mcf)']/6).plot(kind='bar', color='red', ax=ax2, width=width, position=1)

         # Add axis labels and title
         plt.title("Production by SubPlay")
         ax.set_ylabel("Oil (bbl)")
         ax2.set_ylabel("Gas (boe)")

         # Show plot
         plt.show()
```

Karnes Trough and Edwards Condensate are more oil rich than the other higher, quicker producing regions.

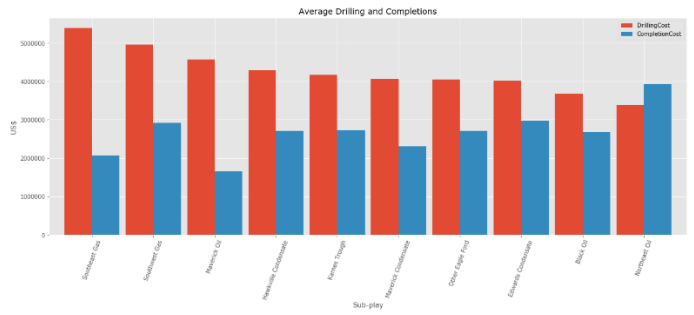
###It would be nice to have a better way to show the amount of gas contributing to BOE.

AVERAGE DRILLING AND COMPLETIONS: Maybe we limit this to only Sub-plays of interest: Karnes, Edwards, and maybe one other if there is an argument.



The Edwards Condensate is nearly 25,000 (USD) per 1,000 barrels of oil equivalent.

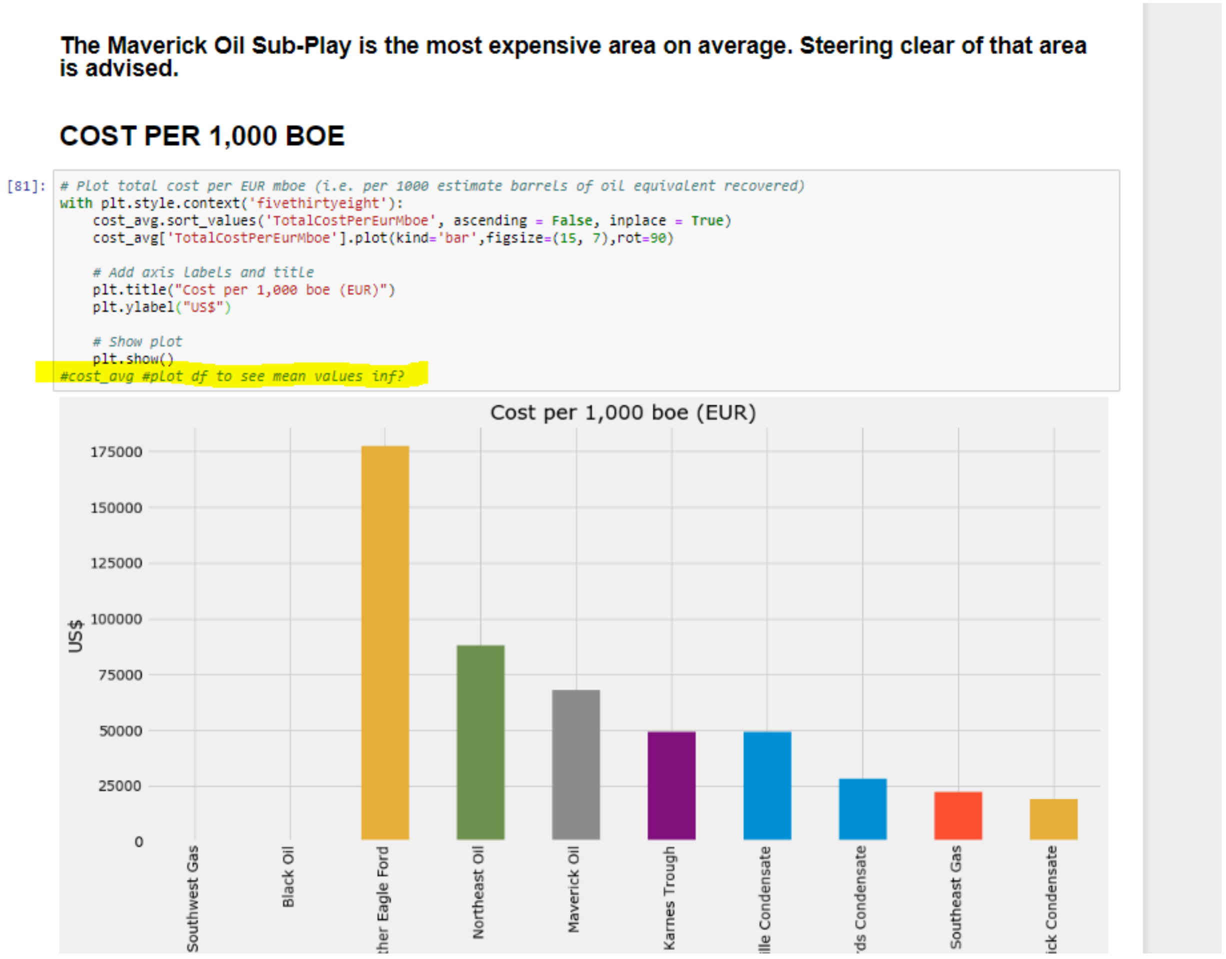
```
In [82]: # Plot categorical costs
         with plt.style.context('ggplot'):
             cost_avg.sort_values('DrillingCost', ascending = False, inplace = True)
             cost_avg.loc[:,'DrillingCost':'CompletionCost'].plot(kind='bar',figsize=(20, 7),rot=70,width=.9)

             # Add axis labels and title
             plt.title("Average Drilling and Completions")
             plt.ylabel("US$")

             # Show plot
             plt.show()
```



While the Edwards and Karnes have similar costs. You dollar goes further in the Edwards.

```
In [21]: #ENTER STACKED BAR OR STACK THE COMPLAETIONS ABOVE
         #https://stackoverflow.com/questions/49889398/plot-stacked-bar-chart-from-pandas-data-frame
```

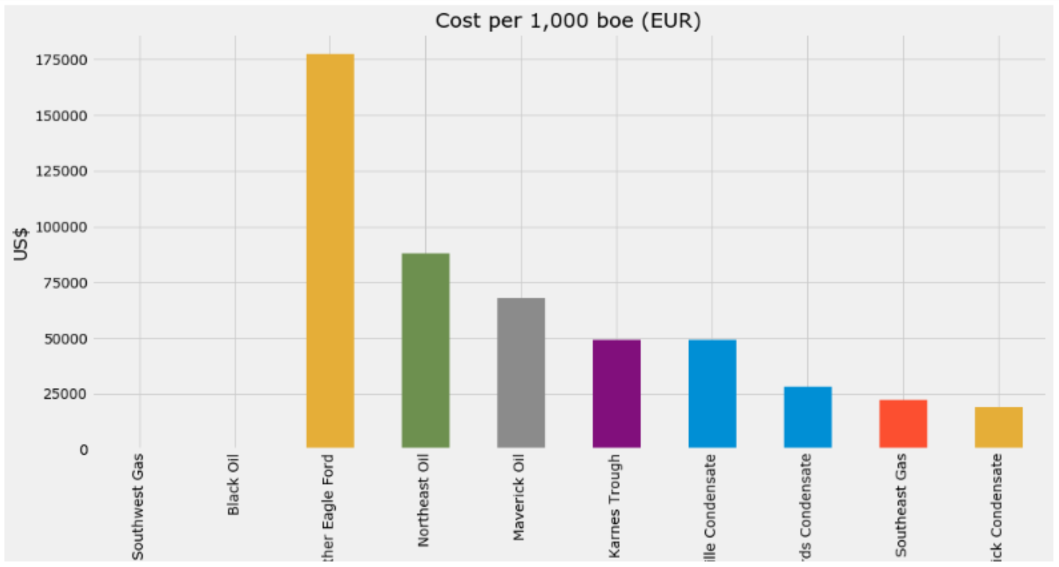
COST PER 1,000 BOE: the mean values are returning an ***inf*** value.



The Maverick Oil Sub-Play is the most expensive area on average. Steering clear of that area is advised.

COST PER 1,000 BOE

```
[81]: # Plot total cost per EUR mboe (i.e. per 1000 estimate barrels of oil equivalent recovered)
      with plt.style.context('fivethirtyeight'):
          cost_avg.sort_values('TotalCostPerEurMboe', ascending = False, inplace = True)
          cost_avg['TotalCostPerEurMboe'].plot(kind='bar',figsize=(15, 7),rot=90)

          # Add axis labels and title
          plt.title("Cost per 1,000 boe (EUR)")
          plt.ylabel("US$")

          # Show plot
          plt.show()
      #cost_avg #plot df to see mean values inf?
```


Cost per 1,000 boe (EUR)
US$
Southwest Gas
Black Oil
ther Eagle Ford
Northeast Oil
Maverick Oil
Karnes Trough
ille Condensate
ds Condensate
Southeast Gas
ick Condensate