Introduction

An important way to test the relationship between two variables Y and X is to run the model: Y = a + bX using ordinary least squares (OLS). When we run regressions, we not only estimate the parameters a and b that can then be used for predictions, we also get to understand how well the model fits (i.e., how much of the variance in Y is explained by a+bX).

This project aims to identify and demonstrate a model explaining audit fees (Y) using firm characteristics (X).

Understanding the key drivers of audit fees is critical for stakeholders like C-suite levels, auditors, financial analysts, and regulators. The project will involve multiple stages, including exploratory data analysis (EDA), feature selection, and model building. Python programming language is utilized in this project.

Data Source

Data from two separate sources is provided:

- "BANA-680 Assignment 4 OL AuditFees201019" contains audit fee information from the Audit Analytics database
- "BANA-680 Assignment 4 OL Compustat201019" contains financial characteristics of firms from the Compustat Annual Industrial file.

Read data and so some preliminary processing

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

path = "/Users/qmacstore/Downloads/[BANA 680]/_Assignment/"
file = 'BANA-680 Assignment 4 OL AuditFees201019.csv'
file1 = 'BANA-680 Assignment 4 OL Compustat201019.csv'

# Audit fees
df_au = pd.read_csv(path+file, encoding='ISO-8859-1')

# Compustat Annual Industrial
df_com = pd.read_csv(path+file1, encoding='ISO-8859-1')
```

```
/opt/anaconda3/lib/python3.8/site-packages/scipy/__init__.py:138: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detecte
d version 1.23.5)
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion} is required for this version of "
```

Getting to understand the tables

In [2]:
```
df_au.info()
df_au.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101178 entries, 0 to 101177
Data columns (total 6 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   FISCAL_YEAR        101178 non-null  object 
 1   FISCAL_YEAR_ENDED  100968 non-null  object 
 2   AUDIT_FEES         100968 non-null  float64
 3   AUDITOR_NAME       100968 non-null  object 
 4   COMPANY_FKEY       100968 non-null  float64
 5   BEST_EDGAR_TICKER  52305 non-null   object 
dtypes: float64(2), object(4)
memory usage: 4.6+ MB
```

Out[2]:

	FISCAL_YEAR	FISCAL_YEAR_ENDED	AUDIT_FEES	AUDITOR_NAME	COMPANY_FKEY	BEST_EDGAR_TICKER
0	2009	02JAN2010	643000.0	Grant Thornton LLP	20.0	NaN
1	2010	31MAY2010	1490000.0	KPMG LLP	1750.0	AIR
2	2011	31MAY2011	1275000.0	KPMG LLP	1750.0	AIR
3	2012	31MAY2012	1745640.0	KPMG LLP	1750.0	AIR
4	2013	31MAY2013	1689980.0	KPMG LLP	1750.0	AIR

In [3]:
```
df_com.info()
df_com.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115269 entries, 0 to 115268
Data columns (total 22 columns):
 #   Column    Non-Null Count   Dtype  
---  ------    --------------   -----  
 0   gvkey     115269 non-null  int64  
 1   datadate  115269 non-null  int64  
 2   fyear     115134 non-null  float64
 3   indfmt    115269 non-null  object 
 4   consol    115269 non-null  object 
 5   popsrc    115269 non-null  object 
 6   datafmt   115269 non-null  object 
 7   tic       115175 non-null  object 
 8   conm      115269 non-null  object 
 9   curcd     115134 non-null  object 
 10  act       66335 non-null   float64
 11  at        89865 non-null   float64
 12  ceq       89685 non-null   float64
 13  ebit      78880 non-null   float64
 14  ebitda    76613 non-null   float64
 15  emp       74854 non-null   float64
 16  invt      81415 non-null   float64
 17  lct       66552 non-null   float64
 18  pifo      22419 non-null   float64
 19  exchg     115169 non-null  float64
 20  costat    115269 non-null  object 
 21  fic       115269 non-null  object 
dtypes: float64(11), int64(2), object(9)
memory usage: 19.3+ MB
```

Out[3]:

	gvkey	datadate	fyear	indfmt	consol	popsrc	datafmt	tic	conm	curcd	...	ceq	ebit	ebitda	emp	invt	lct	pifo	exchg	costat	fic
0	1004	20100531	2009.0	INDL	C	D	STD	AIR	AAR CORP	USD	...	746.906	95.415	134.345	5.8	496.904	325.550	NaN	11.0	A	USA
1	1004	20110531	2010.0	INDL	C	D	STD	AIR	AAR CORP	USD	...	835.845	137.016	196.312	6.1	507.274	416.010	NaN	11.0	A	USA
2	1004	20120531	2011.0	INDL	C	D	STD	AIR	AAR CORP	USD	...	864.649	142.360	222.693	6.7	599.752	473.226	NaN	11.0	A	USA
3	1004	20130531	2012.0	INDL	C	D	STD	AIR	AAR CORP	USD	...	918.600	136.600	245.200	6.3	582.900	389.000	NaN	11.0	A	USA
4	1004	20140531	2013.0	INDL	C	D	STD	AIR	AAR CORP	USD	...	999.500	142.600	256.000	5.8	632.900	402.100	NaN	11.0	A	USA

5 rows × 22 columns

Data cleaning

In [4]:
```
# Some preprocessing steps
df_au1 = df_au[df_au['BEST_EDGAR_TICKER'].notnull()]   #remove null
df_au1['tic'] = df_au1['BEST_EDGAR_TICKER']    #new column to join
df_au1['FISCAL_YEAR'] = df_au1['FISCAL_YEAR'].astype('float64')
df_au1['fyear'] = df_au1['FISCAL_YEAR']    #new column to join

# Check for duplicates
print('Duplicates in Audit fees data : ',df_au1.duplicated().sum())
print('Duplicates in Compustat data :',df_com.duplicated().sum())
```

```
Duplicates in Audit fees data :  0
Duplicates in Compustat data : 0
<ipython-input-4-c28f48e1efe8>:3: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_au1['tic'] = df_au1['BEST_EDGAR_TICKER']    #new column to join
<ipython-input-4-c28f48e1efe8>:4: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_au1['FISCAL_YEAR'] = df_au1['FISCAL_YEAR'].astype('float64')
<ipython-input-4-c28f48e1efe8>:5: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_au1['fyear'] = df_au1['FISCAL_YEAR']    #new column to join
```

In [5]:
```
### Merge 2 datasets using 'Tickers' and 'Years'

df = pd.merge(df_au1, df_com, on=['tic','fyear'], how = 'left')
df.head()
```

Out[5]:

	FISCAL_YEAR	FISCAL_YEAR_ENDED	AUDIT_FEES	AUDITOR_NAME	COMPANY_FKEY	BEST_EDGAR_TICKER	tic	fyear	gvkey	datadate	...	ceq	ebit	ebitda	emp	invt	lct	pi
0	2010.0	31MAY2010	1490000.0	KPMG LLP	1750.0	AIR	AIR	2010.0	1004.0	20110531.0	...	835.845	137.016	196.312	6.10	507.274	416.010	Na
1	2011.0	31MAY2011	1275000.0	KPMG LLP	1750.0	AIR	AIR	2011.0	1004.0	20120531.0	...	864.649	142.360	222.693	6.70	599.752	473.226	Na
2	2012.0	31MAY2012	1745640.0	KPMG LLP	1750.0	AIR	AIR	2012.0	1004.0	20130531.0	...	918.600	136.600	245.200	6.30	582.900	389.000	Na
3	2013.0	31MAY2013	1689980.0	KPMG LLP	1750.0	AIR	AIR	2013.0	1004.0	20140531.0	...	999.500	142.600	256.000	5.80	632.900	402.100	Na
4	2014.0	31MAY2014	1794370.0	KPMG LLP	1750.0	AIR	AIR	2014.0	1004.0	20150531.0	...	845.100	-8.600	83.700	4.85	566.700	412.000	11

5 rows × 28 columns

In [6]:
```
df.info()

# Explore AUDIT_FEES
print(df['AUDIT_FEES'].describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59076 entries, 0 to 59075
Data columns (total 28 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   FISCAL_YEAR        59076 non-null  float64
 1   FISCAL_YEAR_ENDED  59076 non-null  object 
 2   AUDIT_FEES         59076 non-null  float64
 3   AUDITOR_NAME       59076 non-null  object 
 4   COMPANY_FKEY       59076 non-null  float64
 5   BEST_EDGAR_TICKER  59076 non-null  object 
 6   tic                59076 non-null  object 
 7   fyear              59076 non-null  float64
 8   gvkey              49278 non-null  float64
 9   datadate           49278 non-null  float64
 10  indfmt             49278 non-null  object 
 11  consol             49278 non-null  object 
 12  popsrc             49278 non-null  object 
 13  datafmt            49278 non-null  object 
 14  conm               49278 non-null  object 
 15  curcd              49278 non-null  object 
 16  act                33737 non-null  float64
 17  at                 48280 non-null  float64
 18  ceq                48188 non-null  float64
 19  ebit               41412 non-null  float64
 20  ebitda             40001 non-null  float64
 21  emp                46502 non-null  float64
 22  invt               43250 non-null  float64
 23  lct                33822 non-null  float64
 24  pifo               15402 non-null  float64
 25  exchg              49278 non-null  float64
 26  costat             49278 non-null  object 
 27  fic                49278 non-null  object 
dtypes: float64(16), object(12)
memory usage: 12.6+ MB
count    5.907600e+04
mean     2.128241e+06
std      5.975163e+06
min      0.000000e+00
25%      9.909575e+04
50%      4.985270e+05
75%      1.647162e+06
max      1.445000e+08
Name: AUDIT_FEES, dtype: float64
```

In [8]:
```
# Drop Audit fees = 0
df1 = df.loc[df['AUDIT_FEES'] != 0]

# Filter numeric values only
df_num = df1.select_dtypes(include = ['float64', 'int64'])

#df_num.head()
#df_num.info()
```

Find correlated variables

In [9]:
```
df_num_corr = df_num.corr()['AUDIT_FEES']
print(df_num_corr,'\n')

golden_features_list = df_num_corr[abs(df_num_corr) >
                                   0.5].sort_values(ascending=False)
print("There are {} strongly correlated values with AUDIT_FEES :\n{}".
      format(len(golden_features_list), golden_features_list))
```

```
FISCAL_YEAR     0.016921
AUDIT_FEES      1.000000
COMPANY_FKEY   -0.179786
fyear           0.016921
gvkey          -0.103896
datadate        0.011267
act             0.726880
at              0.697154
ceq             0.698200
ebit            0.607544
ebitda          0.650830
emp             0.428079
invt            0.361973
lct             0.698403
pifo            0.426893
exchg          -0.256033
Name: AUDIT_FEES, dtype: float64

There are 7 strongly correlated values with AUDIT_FEES :
AUDIT_FEES    1.000000
act           0.726880
lct           0.698403
ceq           0.698200
at            0.697154
ebitda        0.650830
ebit          0.607544
Name: AUDIT_FEES, dtype: float64
```

Explanation

From above results, strongly correlated values with AUDIT_FEES are (*):

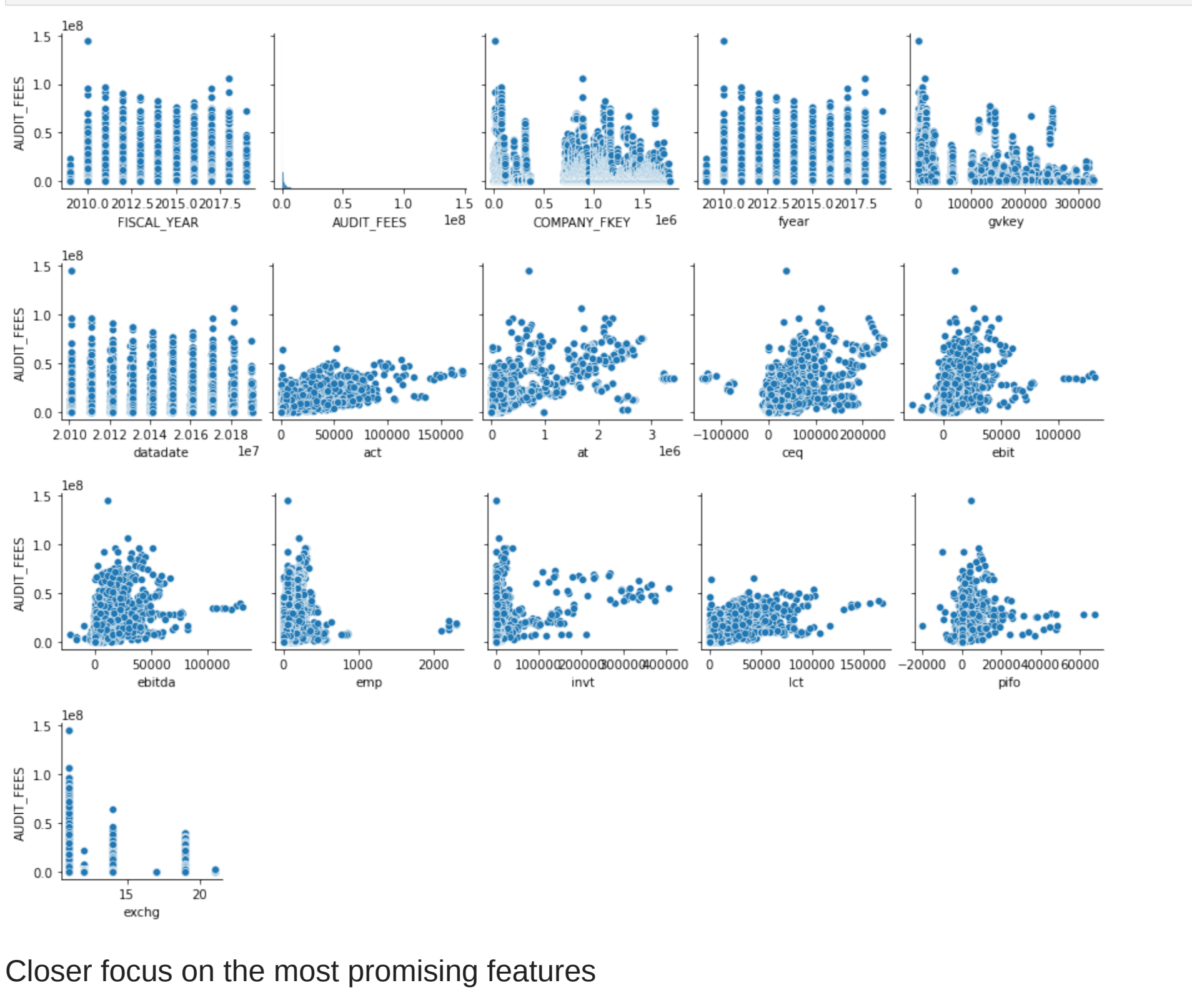
- act: Current Assets
- lct: Current Liabilities
- ceq: Common Equity
- at: Total Assets
- ebitda: Earnings Before Interest, Taxes, Depreciation, and Amortization
- ebit: Earnings Before Interest and Taxes

(*) Definitions of abbreviations are extracted from Compustat.

Below possible explanations along with evidence from several empirical research:

- Current Assets and Liabilities: Firms with large current assets (especially inventories and receivables) and current liabilities tend to have higher audit fees because these items are more difficult to verify and carry higher audit risk. (Knechel, Vanstraelen, and Zerni (2015))
- Total Assets and Firm Size: Larger firms generally incur higher audit fees due to more complex audits, which require more time and resources to assess. (Simunic, D. A. (1980))
- EBITDA and EBIT: Higher levels of EBIT and EBITDA typically correlate with higher audit fees, as they are the indicators of larger, more complex operations that require more auditing effort. (Basioudis, I. G., & Francis, J. R. (2007))

In [10]:
```
# Plot variables
for i in range(0, len(df_num.columns), 5):
    sns.pairplot(data=df_num,
                 x_vars=df_num.columns[i:i+5],
                 y_vars=['AUDIT_FEES'])
plt.show()
```



Closer focus on the most promising features

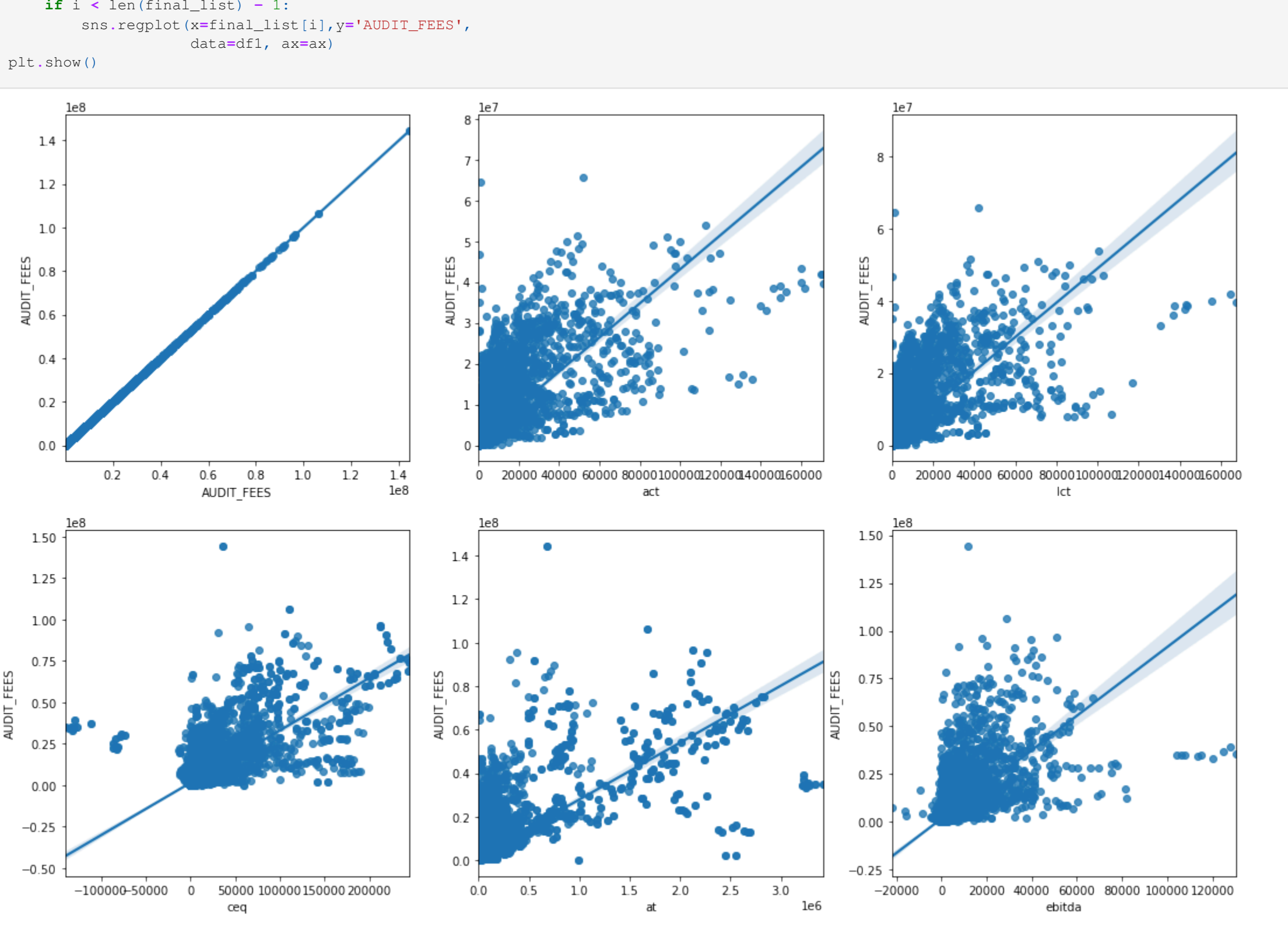
Focus on variables with the highest correlation with AUDIT_FEES.

In [11]:
```
final_list = golden_features_list.index.tolist()
#final_list.append('AUDIT_FEES')
final_list
```

Out[11]: ['AUDIT_FEES', 'act', 'lct', 'ceq', 'at', 'ebitda', 'ebit']

In [12]:
```
fig, ax = plt.subplots(round(len(final_list) / 3), 3,
                       figsize = (18, 12))

for i, ax in enumerate(fig.axes):
    if i < len(final_list) - 1:
        sns.regplot(x=final_list[i],y='AUDIT_FEES',
                    data=df1, ax=ax)
plt.show()
```



Evaluating the model

In [13]:
```
import statsmodels.formula.api as sm
result = sm.ols(formula="AUDIT_FEES ~ act + lct + ceq + at + ebitda + ebit",
                data=df1).fit()

result.summary()
```

Out[13]:

OLS Regression Results

Dep. Variable:	AUDIT_FEES	R-squared:	0.595
Model:	OLS	Adj. R-squared:	0.595
Method:	Least Squares	F-statistic:	8058.
Date:	Sun, 17 Nov 2024	Prob (F-statistic):	0.00
Time:	13:29:55	Log-Likelihood:	-5.3369e+05
No. Observations:	32877	AIC:	1.067e+06
Df Residuals:	32870	BIC:	1.067e+06
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.226e+06	1.55e+04	79.028	0.000	1.2e+06	1.26e+06
act	283.4637	6.108	46.411	0.000	271.492	295.435
lct	-138.3435	7.943	-17.418	0.000	-153.911	-122.776
ceq	-191.5131	4.617	-41.479	0.000	-200.563	-182.463
at	183.1326	2.775	65.985	0.000	177.693	188.572
ebitda	-480.3293	29.040	-16.540	0.000	-537.248	-423.411
ebit	473.5070	29.538	16.030	0.000	415.611	531.403

Omnibus:	23122.702	Durbin-Watson:	0.436
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2991107.369
Skew:	2.522	Prob(JB):	0.00
Kurtosis:	49.455	Cond. No.	3.03e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.03e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Model Analysis

- P-value of all variables are all less than 0.05, indicating observed resesults are statistically significant.
- R-squared is 0.595, meaning that the data relatively fits the regression model.

Conclusion

By using EDA and expert-driven approach, best sets of variables (Current Assets, Current Liabilities, Common Equity, Total Assets, Earnings Before Interest, Taxes, Depreciation, and Amortization, Earnings Before Interest and Taxes) have been identified to construct a model explaining audit fees.

Further investigations may include:

- It could be used Big4/non-Big4 flag to evaluate further the impacts using logistics regression. (Big 4 Audit firms are Deloitte, KPMG, EY, and PWC). Generally, Big 4 firms tend to charge higher audit fees than lower tiers audit firms due to their greater perceived credibility and expertise. (Hay, Knechel, & Wong (2006))
- It could be improved by exploring exponential (non-linear) effect.