Task 2

Predictive Analytics

1. For this task, you'll likely need to use statistical software such as R, SAS, or Python.
2. Using the same transaction dataset, identify the annual salary for each customer
3. Explore correlations between annual salary and various customer attributes (e.g. age). These attributes could be those that are readily available in the data (e.g. age) or those that you construct or derive yourself (e.g. those relating to purchasing behaviour). Visualise any interesting correlations using a scatter plot.
4. Build a simple regression model to predict the annual salary for each customer using the attributes you identified above
5. How accurate is your model? Should ANZ use it to segment customers (for whom it does not have this data) into income brackets for reporting purposes?
6. For a challenge: build a decision-tree based model to predict salary. Does it perform better? How would you accurately test the performance of this model?

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_excel("ANZ synthesised transaction dataset.xlsx")
df.head()
```

Out[2]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_descri
0	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	
1	authorized	0.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES
2	authorized	1.0	NaN	ACC-1222300524	AUD	151.23 -33.94	
3	authorized	1.0	NaN	ACC-1037050564	AUD	153.10 -27.66	SALES
4	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES

5 rows × 23 columns

In [3]:

```
df.tail()
```

Out[3]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_d
12038	authorized	0.0	NaN	ACC-3021093232	AUD	149.83 -29.47	
12039	authorized	1.0	NaN	ACC-1608363396	AUD	151.22 -33.87	S/
12040	authorized	1.0	NaN	ACC-3827517394	AUD	151.12 -33.89	
12041	authorized	1.0	NaN	ACC-2920611728	AUD	144.96 -37.76	S/
12042	authorized	1.0	NaN	ACC-1443681913	AUD	150.92 -33.77	S/

5 rows × 23 columns

In [4]:

```
df.columns
```

Out[4]:

```
Index(['status', 'card_present_flag', 'bpay_biller_code', 'account',
       'currency', 'long_lat', 'txn_description', 'merchant_id',
       'merchant_code', 'first_name', 'balance', 'date', 'gender', 'age',
       'merchant_suburb', 'merchant_state', 'extraction', 'amount',
       'transaction_id', 'country', 'customer_id', 'merchant_long_lat',
       'movement'],
      dtype='object')
```

In [5]:

```
df.shape
```

Out[5]:

```
(12043, 23)
```

In [6]:

```
df['txn_description'].value_counts()
```

Out[6]:

```
SALES-POS      3934
POS            3783
PAYMENT        2600
PAY/SALARY      883
INTER BANK      742
PHONE BANK      101
Name: txn_description, dtype: int64
```

In [7]:

```
df['movement'].value_counts()
```

Out[7]:

```
debit     11160
credit      883
Name: movement, dtype: int64
```

In [8]:

```
salary_df = df[df['txn_description'] == 'PAY/SALARY']

salary_df.head()
```

Out[8]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_descript
50	posted	NaN	0	ACC-588564840	AUD	151.27 -33.76	PAY/SALA
61	posted	NaN	0	ACC-1650504218	AUD	145.01 -37.93	PAY/SALA
64	posted	NaN	0	ACC-3326339947	AUD	151.18 -33.80	PAY/SALA
68	posted	NaN	0	ACC-3541460373	AUD	145.00 -37.83	PAY/SALA
70	posted	NaN	0	ACC-2776252858	AUD	144.95 -37.76	PAY/SALA

5 rows × 23 columns

In [9]:

```
salary_df.shape
```

Out[9]:

```
(883, 23)
```

In [10]:

```
salary_df.movement.value_counts()
```

Out[10]:

```
credit    883
Name: movement, dtype: int64
```

In [11]:

```
salary_df.iloc[1]
```

Out[11]:

```
status                                        posted
card_present_flag                                NaN
bpay_biller_code                                   0
account                               ACC-1650504218
currency                                         AUD
long_lat                               145.01 -37.93
txn_description                          PAY/SALARY
merchant_id                                      NaN
merchant_code                                      0
first_name                                  Marissa
balance                                      2040.58
date                             2018-08-01 00:00:00
gender                                             F
age                                               23
merchant_suburb                                  NaN
merchant_state                                   NaN
extraction              2018-08-01T12:00:00.000+0000
amount                                       1626.48
transaction_id      1822eb0e1bbe4c9e95ebbb0fa2cc4323
country                                    Australia
customer_id                           CUS-2500783281
merchant_long_lat                                NaN
movement                                      credit
Name: 61, dtype: object
```

In [12]:

```
salary_df['gender'] = pd.get_dummies(salary_df['gender'], drop_first=True)
```

```
C:\Users\Jesus\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Settin
gWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

In [13]:

```
salary_df['year'] = [i.year for i in salary_df['date']]
salary_df['month'] = [i.month for i in salary_df['date']]
salary_df['day'] = [i.day_name() for i in salary_df['date']]
```

```
C:\Users\Jesus\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Settin
gWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
C:\Users\Jesus\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: Settin
gWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy

C:\Users\Jesus\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: Settin
gWithCopyWarning:
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: http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing import
s until
```

In [14]:

```
salary_df.head()
```

Out[14]:

extraction	amount	transaction_id	country	customer_id	merchai
2018-08-T11:00:00.000+0000	3903.95	9ca281650e5d482d9e53f85e959baa66	Australia	CUS-1462656821	
2018-08-T12:00:00.000+0000	1626.48	1822eb0e1bbe4c9e95ebbb0fa2cc4323	Australia	CUS-2500783281	
2018-08-T12:00:00.000+0000	983.36	bd62b1799a454cedbbb56364f7c40cbf	Australia	CUS-326006476	
2018-08-T13:00:00.000+0000	1408.08	0d95c7c932bb48e5b44c2637bdd3efe9	Australia	CUS-1433879684	
2018-08-T13:00:00.000+0000	1068.04	f50ccf1195214d14a0acbfcb5a265193	Australia	CUS-4123612273	

In [15]:

```
salary_df.columns
```

Out[15]:

```
Index(['status', 'card_present_flag', 'bpay_biller_code', 'account',
       'currency', 'long_lat', 'txn_description', 'merchant_id',
       'merchant_code', 'first_name', 'balance', 'date', 'gender', 'age',
       'merchant_suburb', 'merchant_state', 'extraction', 'amount',
       'transaction_id', 'country', 'customer_id', 'merchant_long_lat',
       'movement', 'year', 'month', 'day'],
      dtype='object')
```

In [16]:

```
salary_df.iloc[1]
```

Out[16]:

```
status                                         posted
card_present_flag                                 NaN
bpay_biller_code                                    0
account                                ACC-1650504218
currency                                          AUD
long_lat                                145.01 -37.93
txn_description                            PAY/SALARY
merchant_id                                       NaN
merchant_code                                       0
first_name                                   Marissa
balance                                       2040.58
date                              2018-08-01 00:00:00
gender                                              0
age                                                23
merchant_suburb                                   NaN
merchant_state                                    NaN
extraction                   2018-08-01T12:00:00.000+0000
amount                                        1626.48
transaction_id       1822eb0e1bbe4c9e95ebbb0fa2cc4323
country                                     Australia
customer_id                            CUS-2500783281
merchant_long_lat                                 NaN
movement                                       credit
year                                             2018
month                                               8
day                                         Wednesday
Name: 61, dtype: object
```

In [17]:

```
result_df = salary_df.groupby('customer_id').mean()[['age','gender']]
result_df['annual_amount'] = salary_df.groupby('customer_id').sum()[['amount']] * 4
result_df['avg_weekly_amount'] = salary_df.groupby(['customer_id','day']).mean().reset_index().\
                                groupby('customer_id').mean()['amount']
result_df['avg_monthly_amount'] = salary_df.groupby(['customer_id','month']).sum().reset_index().\
                                groupby('customer_id').mean()['amount']
result_df['max_amount'] = salary_df[['customer_id','amount','month']].groupby(['customer_id', 'month']).sum().\
reset_index()[['customer_id','amount']].groupby('customer_id').max()

result_df.head()
```

Out[17]:

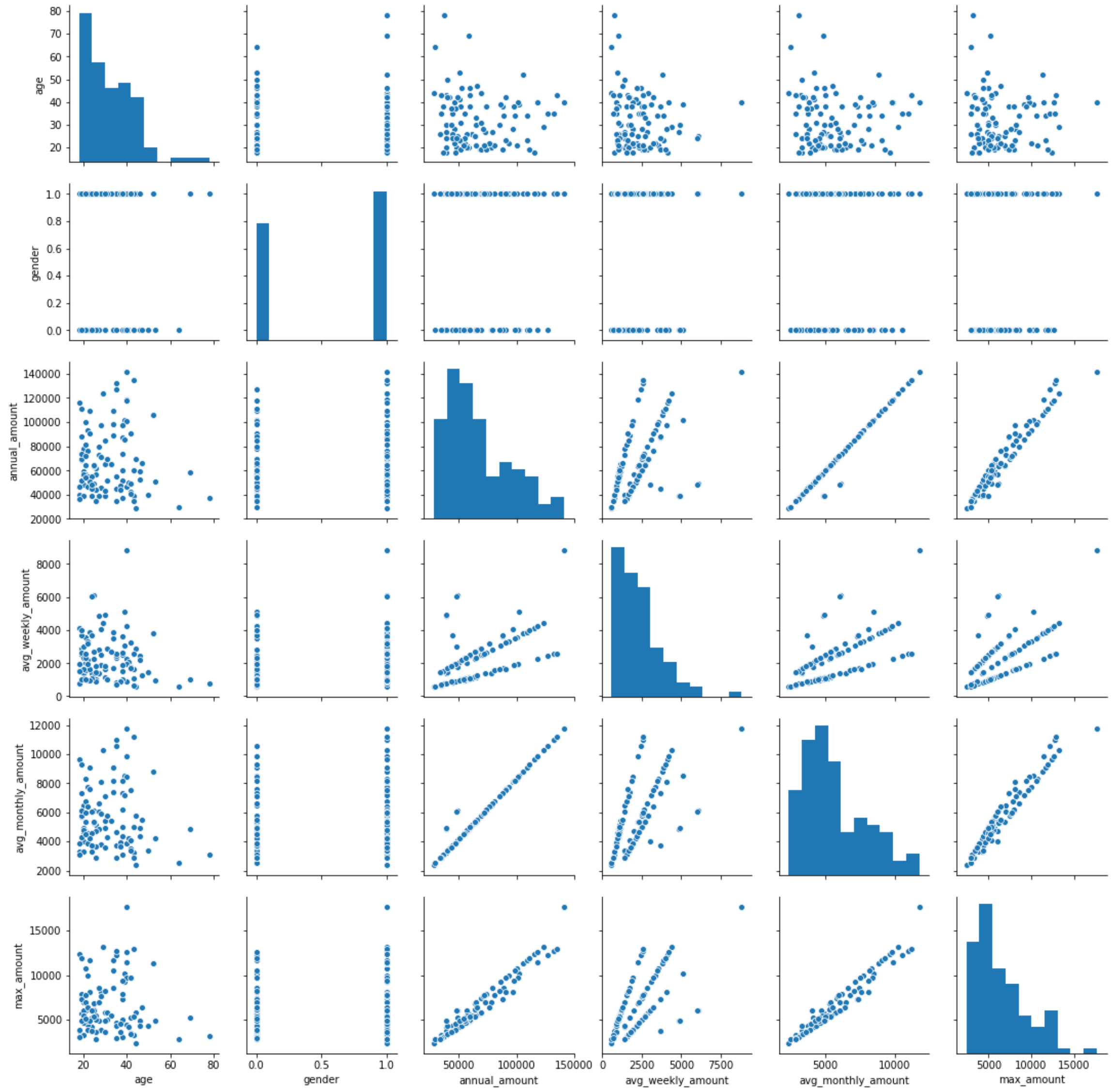
	age	gender	annual_amount	avg_weekly_amount	avg_monthly_amount	max_a
customer_id						
CUS-1005756958	53.0	0	50464.44	970.47	4205.370000	4
CUS-1117979751	21.0	1	100202.20	3578.65	8350.183333	10
CUS-1140341822	28.0	1	45996.24	1916.51	3833.020000	3
CUS-1147642491	34.0	0	88992.28	1711.39	7416.023333	8
CUS-1196156254	34.0	0	109304.44	3903.73	9108.703333	11

In [18]:

```
sns.pairplot(result_df)
```

Out[18]:

```
<seaborn.axisgrid.PairGrid at 0x1ef4aecb518>
```



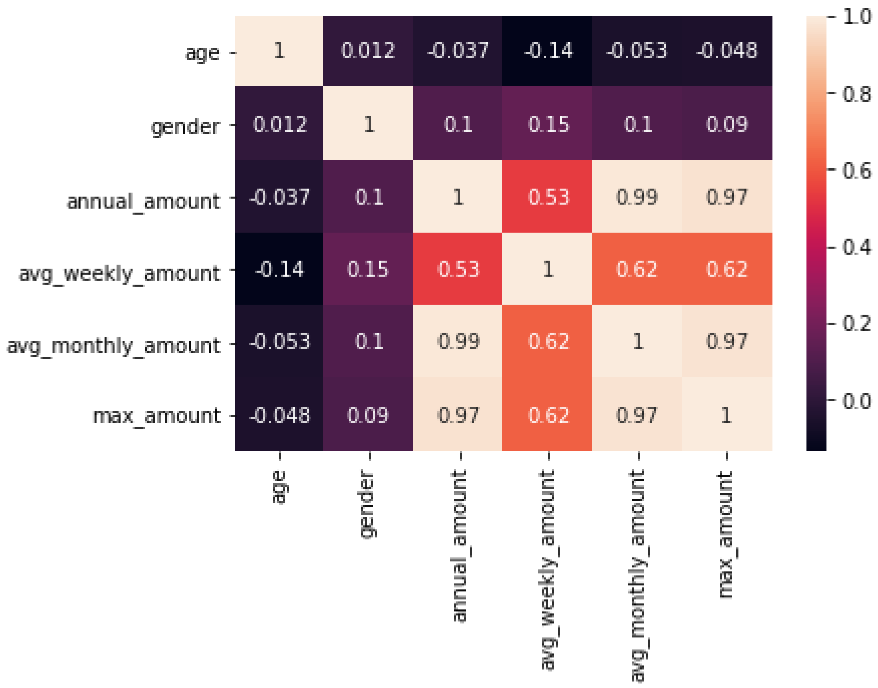
In [19]:

```
corr = result_df.corr()

sns.heatmap(corr, annot=True)
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x1ef4b9f28d0>



Linear Regression

In [32]:

```
x = result_df.drop('annual_amount', axis = 1)
y = result_df['annual_amount'].values.reshape(-1,1)
```

In [33]:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()

lr.fit(x,y)
```

Out[33]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [34]:

```
y_pred = lr.predict(x)
```

In [35]:

```
from sklearn.metrics import r2_score, mean_squared_error
```

In [36]:

```
r2_score(y, y_pred)
```

Out[36]:

```
0.9863309958711153
```

In [37]:

```
mean_squared_error(y, y_pred) ** 0.5
```

Out[37]:

```
3136.933997533792
```

Decision Tree Regressor

In [38]:

```
from sklearn.tree import DecisionTreeRegressor
```

In [39]:

```
dt_reg = DecisionTreeRegressor()
```

In [40]:

```
dt_reg.fit(x, y)
```

Out[40]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [41]:

```
y_pred = dt_reg.predict(x)
```

In [42]:

```
r2_score(y, y_pred)
```

Out[42]:

```
1.0
```

In [43]:

```
mean_squared_error(y, y_pred)
```

Out[43]:

```
0.0
```