Data Science Project 1 - Alaukik

In [6]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pylab as plt

sns.set_style('darkgrid')

%matplotlib inline
```

Data Exploration to predict defaults on loans.

Features included:

- **loan_amnt**: The amount of money applied for
- **term**: The period over which the load should be repaid
- **annual_inc**: Annual income of the borrower
- **purpose**: The purpose of the loan, such as: credit_card, debt_consolidation, etc.
- **home_ownership**: The borrower's relationship with their primary residence
- **outcome**: The result of the loan

In [3]:
```
# Loading the data from

df = pd.read_csv('../data/loan_data_subset.csv')
```

In [5]:
```
# Information about the dataframe

df.info(null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 6 columns):
loan_amnt         20000 non-null int64
term              20000 non-null object
annual_inc        20000 non-null int64
purpose           20000 non-null object
home_ownership    20000 non-null object
outcome           20000 non-null object
dtypes: int64(2), object(4)
memory usage: 937.6+ KB
```

In [8]:
```
# 4. (1pt) Using .shape, how many rows does the dataset have?
print(f'dataframe has {df.shape[0]} rows')
```

```
dataframe has 20000 rows
```

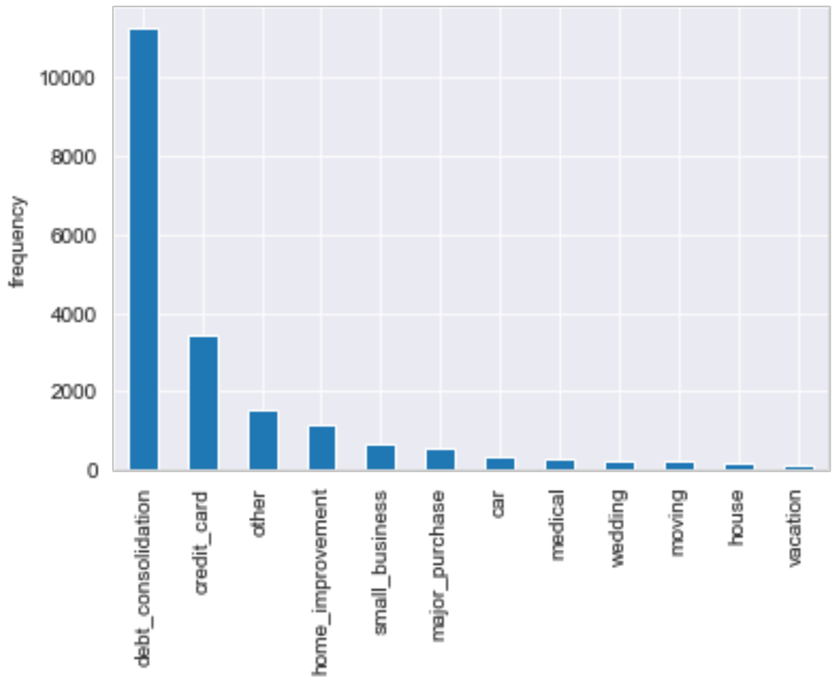
In [9]:
```
df.head()
```

Out[9]:

	loan_amnt	term	annual_inc	purpose	home_ownership	outcome
0	11000	60 months	59004	home_improvement	MORTGAGE	paid off
1	14000	36 months	120000	credit_card	RENT	default
2	10000	36 months	110000	small_business	MORTGAGE	default
3	23350	60 months	65000	debt_consolidation	MORTGAGE	default
4	12000	60 months	49000	major_purchase	MORTGAGE	paid off

In [11]:
```
# Frequencies of the values in 'purpose'
ax = df.purpose.value_counts().plot.bar()


# Labeling the y axis as 'frequency'
ax.set_ylabel('frequency')
```

Out[11]:
```
Text(0, 0.5, 'frequency')
```



In [12]:
```
# Summary statistics of the annual_inc column
df.annual_inc.describe()
# There appears to be a fairly large difference between mean and media
n
```

Out[12]:
```
count    2.000000e+04
mean     6.824335e+04
std      4.420020e+04
min      2.000000e+03
25%      4.200000e+04
50%      6.000000e+04
75%      8.200000e+04
max      1.200000e+06
Name: annual_inc, dtype: float64
```

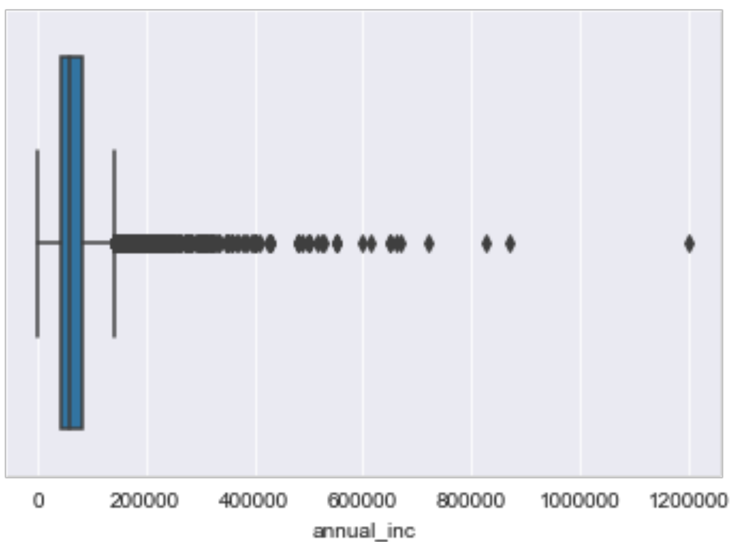
In [13]:
```
annual_inc_mean = df.annual_inc.mean()


annual_inc_median = df.annual_inc.median()


diff = annual_inc_mean - annual_inc_median
print(f'mean - median = {diff:0.2f}')
```

```
mean - median = 8243.35
```

There might a few people in the data with very high income, pulling the mean to a higher value than the median. In other words, the data might be skewed towards the right.

In [64]:
```
sns.boxplot(x = df.loc[:, 'annual_inc'])
```

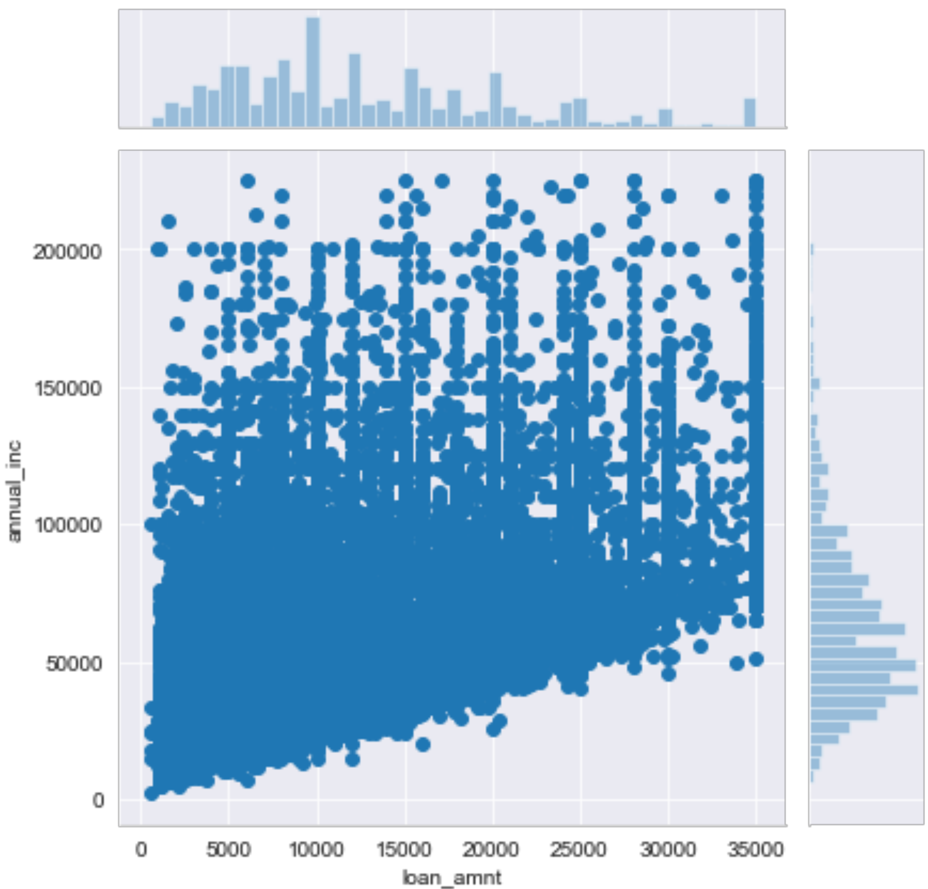
Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1b9f2490>



In [16]:
```
# It certainly looks like annual_inc is skewed to the right.

annual_inc_99 = np.percentile(df.loc[:, 'annual_inc'], 99)
print(f'99th percentile of annual_inc: {annual_inc_99:0.2f}')

y = df.loc[df['annual_inc'] < annual_inc_99 ]

sns.jointplot(x = 'loan_amnt', y = 'annual_inc', data= y )
```

```
99th percentile of annual_inc: 225010.00
```

Out[16]: `<seaborn.axisgrid.JointGrid at 0x102cbe290>`

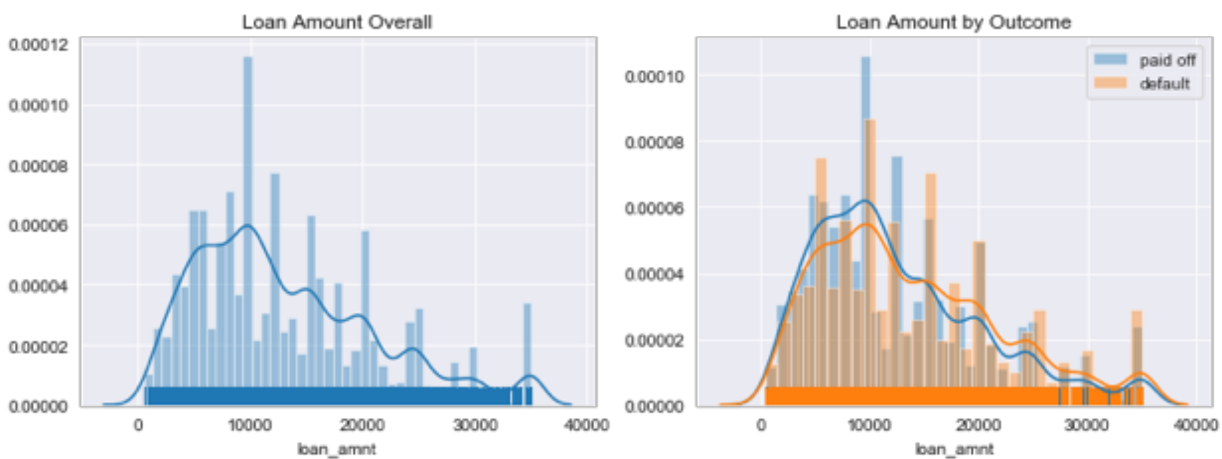


In [96]:
```
amnt = df.loc[(df.purpose == 'debt_consolidation') & (df.annual_inc <
annual_inc_99)].loan_amnt.mean()
print(f'mean loan amount for debt consolidation for most annual income
s: {amnt:0.2f}')
```

```
mean loan amount for debt consolidation for most annual incomes: 141
66.53
```

In [19]:
```
# Predicting loan outcome using plots

fig,ax = plt.subplots(1,2,figsize =(12,4))


sns.distplot(df.loan_amnt, rug = True, ax = ax[0]);

ax[0].set_title('Loan Amount Overall')

sns.distplot(df.loc[df.outcome == 'paid off'].loan_amnt,rug = True, ax
= ax[1], label = 'paid off');

sns.distplot(df.loc[df.outcome == 'default'].loan_amnt,rug = True, ax
= ax[1], label = 'default');

ax[1].set_title('Loan Amount by Outcome')

ax[1].legend()
```

Out[19]: <matplotlib.legend.Legend at 0x1a140e92d0>



Hypothesis Testing with an A/B tests

Suppose we work at a large company that is developing online data science tools.

Currently the tool has interface type A but we'd like to know if using interface tool B might be more efficient. To measure this, we'll look at length of active work on a project (aka project length).

We'll perform an A/B test where half of the projects will use interface A and half will use interface B.

In [24]:
```
df_project = pd.read_csv('../data/project_lengths.csv')
df_project.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
lengths_A    1000 non-null float64
lengths_B    1000 non-null float64
dtypes: float64(2)
memory usage: 15.8 KB
```

In [25]:
```
# Calculating the difference in mean project length between interface A and B (mean B - mean A)
# Printing the result with 2 significant digits
a_lengths = df_project.iloc[:, 0].mean()
b_lengths = df_project.iloc[:, 1].mean()

observed_mean_diff = a_lengths - b_lengths
print(f'observed difference: {observed_mean_diff:0.2f}')
```

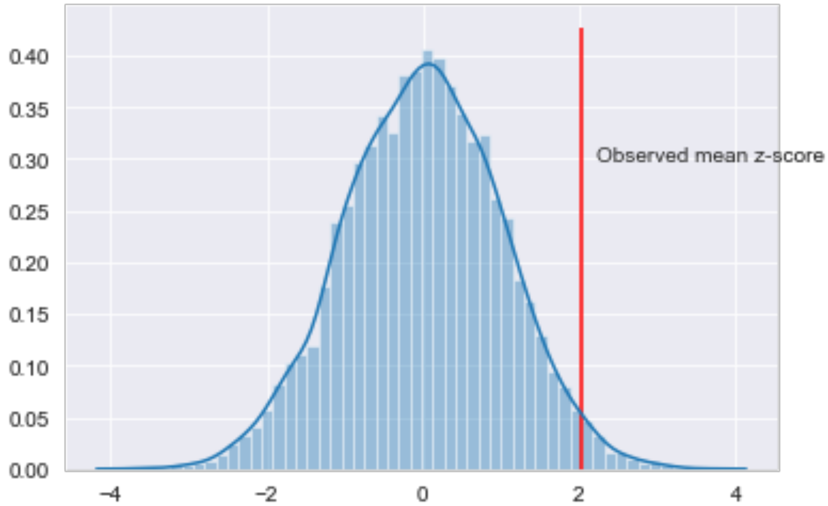
```
observed difference: 1.58
```

In [26]:
```
# We'll perform a permutation test to see how significant this result is
#by generating 10000 random permutation samples of mean difference

rand_mean_diffs = []
n_samples = 10000
combined_times = np.concatenate([df_project.lengths_A.values, df_project.lengths_B.values])


n_A = sum(df_project.lengths_A.notnull()) # number of observations for page A






for i in range(n_samples):

    rand_perm = np.random.permutation(combined_times)

    rand_mean_A = rand_perm[:n_A].mean()

    rand_mean_B = rand_perm[n_A:].mean()

    diff = rand_mean_B - rand_mean_A

    rand_mean_diffs.append(diff)

print(len(rand_mean_diffs))
```

```
10000
```

In [27]:
```
# Transform all values to their z-score


mean_rand_mean_diffs = np.mean(rand_mean_diffs)

std_rand_mean_diffs = np.std(rand_mean_diffs)


rand_mean_diffs_zscore = (rand_mean_diffs - mean_rand_mean_diffs)/ std_rand_mean_diffs

observed_mean_diff_zscore = (observed_mean_diff - mean_rand_mean_diffs)/ std_rand_mean_diffs
```

In [28]:
```
ax = sns.distplot(rand_mean_diffs_zscore)

ax.vlines(observed_mean_diff_zscore, *ax.get_ylim(),color='r')
ax.text(observed_mean_diff_zscore + 0.2,ax.get_ylim()[1]/1.5,'Observed
mean z-score');
```



In [32]:
```
# the plot seems to indicate a likely difference in scores
# Lets calculate a two-tailed p_value (to three significant digits)

gt = np.abs(np.array(rand_mean_diffs)) >= np.abs(observed_mean_diff)
num_gt = sum(gt)
p_value = num_gt / len(rand_mean_diffs)
print(f'p_value: {p_value:0.3f}')
```

```
p_value: 0.044
```

Out[32]:
```
437
```