Years of programming experience (this includes programming in a professional or student capacity, as well as any programming done on your own time)

- ○ 0 - 1
- ○ 1 - 3
- ○ 3 - 5
- ○ 5+

Years of data analysis experience. In this context, data analysis refers to the preparation, transformation, and visualization of data for purposes of summary or detailed analyses. Relevant analysis tools include the use of spreadsheet tools such as Excel, statistical software such as Stata, SPSS, shell-based data processing languages like awk, visualization software such as Tableau, database query languages such as SQL, programming languages such as R, Julia, Matlab/Octave, or the use data-centric libraries in other programming languages such as Pandas/Numpy/SciPy in Python, among others. Data in our context is focused on tabular representations (i.e. one or more tables).

- ○ 0 - 1
- ○ 1 - 3
- ○ 3 - 5
- ○ 5+

You have experience with (Mark all that apply)

☐ Python	☐ Matlab
☐ Pandas (Python library)	☐ SAS
☐ SQL	☐ Stata
☐ Julia	☐ Shell tools (e.g. sed, awk)
☐ R	

Select your experience level with the Python library Pandas

- ○ No knowledge
- ○ Fundamental awareness

- ○ Novice
- ○ Intermediate
- ○ Advanced
- ○ Expert

validation

Read the code fragment below and then answer the question regarding its intended behavior

```
import pandas as pd

df = pd.read_csv('data.csv')
df['col1'] = df['col1'].map(lambda x: x * 2)
```

The code fragment above

- ○ Removes column named col1
- ○ Scales column named col1
- ○ Creates a dictionary/map using a column named col1

Survey Introduction: Background Setting

Background
As you proceed through this survey, picture yourself in the following role. You work at a large data analytics company, and your job consists of preparing datasets for use by other analysts. Preparing a dataset in this context means taking an input dataset, in the form of tabular data (e.g. a spreadsheet) and transforming it to add additional information, normalizing values and their representation, removing or renaming columns, among others. In general, you could refer to these tasks as "data wrangling" or "data munging". For example, a typical dataset transformation might be to add a column that derives additional data from an existing column (e.g. splitting an address column into a city and state columns). As part of your job, you are often presented with new datasets, some of which may or may not have detailed data documentation. In this context, data documentation refers to a catalogue-style manual that describes the meaning of each column, the expected values, and their appropriate representation (e.g. are the values

numeric, or are they strings, do they come from a limited set of values). To carry out your data preparation tasks you typically use a standard programming language such as Python and the accompanying data analysis library ecosystem (e.g. Pandas, Numpy, Scikit-learn, Scipy, among others).

You have been asked to evaluate six sets of five code snippets, each set associated with a specific task on a single dataset. Code snippets may or may not be relevant to the task at hand. Your job will be to answer questions about these snippets, such as their relevance and ordering, given the task at hand.

In the following section we introduce the dataset you have been tasked on working with.

Dataset 1 Overview

Dataset: Loan data

Your company recently acquired a new client, a loan issuer, and they would like your help in preparing one of their key datasets: loan issuance data. The loan issuance data consists of a single comma-separated values (csv) file, which reports the individual loan and corresponding borrower details for 800,000 loans. This information is represented using 74 different columns, such as *id* (unique loan identifier), *loan_amt* (the total dollar amount that was issued), *member_id* (an id for the borrower), and so on. This dataset is particularly valuable for your client as they can use it to perform critical analyses, including predicting borrower defaults (i.e. borrowers who will fail to make payments and give up on their payment plan), borrower prepayments (i.e. borrowers who will pay off loans earlier than expected), and borrower delinquencies (i.e. borrowers who will fall behind on payments but have not yet default). For a loan issuer these analyses are critical, as they are used to monitor their current portfolio of loans and make future issuance decisions as well.

Dataset 1, Task 1

New Task

Your task is to identify the subset of loans in the dataset that are not current. In this context, not current means the borrower is not up to date on their required payments. In order to subset these loans, you can make use of the column named *loan_status.* Grouping loans into current and not-current may be useful as these borrowers may have different credit profiles and behaviors.

The general task (i.e. ignoring dataset domain) is reflective of tasks you, as a participant, might perform in your own data analyses.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Neither agree nor disagree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Dataset 1, Task 1: Treatment group

Task Reminder

Your task is to identify the subset of loans in the dataset that are not current. In this context, not current means the borrower is not up to date on their required payments. In order to subset these loans, you can make use of the column named *loan_status.* Grouping loans into current and not-current may be useful as these borrowers may have different credit profiles and behaviors.

Read the following 5 code fragments and then answer the questions below.

```python
# Fragment 0
def f0(df):
    # core cleaning code
    import pandas as pd
    # df = pd.read_csv('../input/loan.csv', low_memory=False)
    df = df.loc[(df['loan_status'] != 'Current')]
    return df


# Fragment 1
def f1(df):
    # core cleaning code
    import pandas as pd
    # df = pd.read_csv('../input/loan.csv')
    df = df[((df.loan_status == 'Fully Paid') |
             (df.loan_status == 'Charged Off'))]
    return df


# Fragment 2
def f2(data):
    # core cleaning code
    import pandas as pd
    # data = pd.read_csv('../input/loan.csv', parse_dates=True)

    data = data[(data.loan_status != 'Fully Paid')]
    return data


# Fragment 3
def f3(df_loan):
    # core cleaning code
    import pandas as pd
    # df_loan = pd.read_csv('../input/loan.csv', low_memory=False)
    df_loan.loc[((
        df_loan.loan_status ==
        'Does not meet the credit policy. Status:Fully Paid'
    ), 'loan_status')] = 'NMCP Fully Paid'
    df_loan.loc[((
        df_loan.loan_status ==
        'Does not meet the credit policy. Status:Charged Off'
    ), 'loan_status')] = 'NMCP Charged Off'
    return df_loan


# Fragment 4
def f4(data):
    # core cleaning code
    import pandas as pd
    # data = pd.read_csv('../input/loan.csv')
    bad_indicators = [
        'Charged Off ', 'Default',
        'Does not meet the credit policy. Status:Charged Off',
        'In Grace Period', 'Default Receiver', 'Late (16-30 days)',
        'Late (31-120 days)'
    ]
    data.loc[(data.loan_status.isin(bad_indicators), 'bad_loan')] =
    return data
```

Which fragment snippets do you consider relevant to the task? You may leave blank if none appear relevant.

- ☐ Fragment 0
- ☐ Fragment 1
- ☐ Fragment 2
- ☐ Fragment 3
- ☐ Fragment 4

Which snippet do you consider to be most relevant to the task? (choose None, if none are relevant)

- ○ Fragment 0
- ○ Fragment 1
- ○ Fragment 2
- ○ Fragment 3
- ○ Fragment 4
- ○ None

Access to these snippets makes completing the task easier

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Neither agree nor disagree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Dataset 1, Task 1: Control group

Task Reminder
Your task is to identify the subset of loans in the dataset that are not current. In this context, not current means the borrower is not up to date on their required payments. In order to subset these loans, you can make use of the column named *loan_status*. Grouping loans into current and not-current may be useful as these borrowers may have different credit profiles and behaviors.

Read the following 5 code fragments and then answer the questions below.

```
# Fragment 0
def f0(loan):
    # core cleaning code
    import pandas as pd
    # loan = pd.read_csv('../input/loan.csv')
    del_cols = [
        'id', 'member_id', 'policy_code', 'url', 'zip_code', 'addr_s
        'pymnt_plan', 'emp_title', 'application_type', 'acc_now_deli
        'title', 'collections_12_mths_ex_med', 'collection_recovery_
    ]
    loan = loan.drop(del_cols, axis=1)
    loan = loan[(loan['loan_status'] != 'Current')]
    loan.loc[((loan['emp_length'] == '< 1 year'), 'empl_exp')] = 'in
    loan.loc[((loan['emp_length'] == '1 year'), 'empl_exp')] = 'new'
    loan.loc[((loan['emp_length'] == '2 years'), 'empl_exp')] = 'new
    loan.loc[((loan['emp_length'] == '3 years'), 'empl_exp')] = 'new

    loan.loc[((loan['emp_length'] == '4 years'), 'empl_exp')] = 'int
    loan.loc[((loan['emp_length'] == '5 years'), 'empl_exp')] = 'int
    loan.loc[((loan['emp_length'] == '6 years'), 'empl_exp')] = 'int
    loan.loc[((loan['emp_length'] == '7 years'), 'empl_exp')] = 'sea
    loan.loc[((loan['emp_length'] == '8 years'), 'empl_exp')] = 'sea
    loan.loc[((loan['emp_length'] == '9 years'), 'empl_exp')] = 'sea
    loan.loc[((loan['emp_length'] == 'n/a'), 'empl_exp')] = 'unknown
    return loan
```

```
# Fragment 1
def f1(df):
    # core cleaning code
    import pandas as pd
    # df = pd.read_csv('../input/loan.csv', low_memory=False)
    df = df.rename(
        columns={
            'loan_amnt': 'loan_amount',
            'funded_amnt': 'funded_amount',
            'funded_amnt_inv': 'investor_funds',
            'int_rate': 'interest_rate',
            'annual_inc': 'annual_income'
        }
    )
    group_dates = df.groupby(['complete_date', 'region'], as_index=F
    group_dates = group_dates.groupby(['issue_d', 'region'],
                                      as_index=False).sum()
    group_dates = group_dates.groupby(['issue_d', 'region'],
                                      as_index=False).sum()
    group_dates['loan_amount'] = (group_dates['loan_amount'] / 1000)
    by_loan_amount = df.groupby(['region', 'addr_state'],
                                as_index=False).loan_amount.sum()
    return by_loan_amount
```

```
# Fragment 2
def f2(dataset):
    # core cleaning code
    import pandas as pd
    # dataset = pd.read_csv('../input/loan.csv', low_memory=False)
    dataset = dataset.fillna(0)
    dataset['verification_status_joint'] = dataset[
        'verification_status_joint'].astype('category').cat.codes
    return dataset
```

```
# Fragment 3
def f3(dataset):
    # core cleaning code
    import pandas as pd
    # dataset = pd.read_csv('../input/loan.csv', low_memory=False)
    dataset = dataset.fillna(0)
    dataset['loan_status'] = dataset['loan_status'].astype(
        'category'
    ).cat.codes
    return dataset


# Fragment 4
def f4(df):
    # core cleaning code
    import pandas as pd
    # df = pd.read_csv('../input/loan.csv', low_memory=False)
    df = df.loc[(df['loan_status'] != 'Current')]
    return df
```

Which fragment snippets do you consider relevant to the task? You may leave blank if none appear relevant.

☐ Fragment 0
☐ Fragment 1
☐ Fragment 2
☐ Fragment 3
☐ Fragment 4

Which snippet do you consider to be most relevant to the task? (choose None, if none are relevant)

○ Fragment 0
○ Fragment 1
○ Fragment 2
○ Fragment 3
○ Fragment 4
○ None

Access to these snippets makes completing the task easier

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Neither agree nor disagree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Dataset 1, Task 2

New Task
Your task is to transform the interest rate column (*int_rate*) by rounding it to the nearest integer. This can be useful to group loans by whole-number interest rates (e.g. all loans with a 3% interest rate, without distinguishing between those loans that have a 3.1% and 3.5% rate). This may be typically done as large increases in the interest rate may reflect lower creditworthiness.

The general task (i.e. ignoring dataset domain) is reflective of tasks you, as a participant, might perform in your own data analyses.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Neither agree nor disagree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Dataset 1, Task 2: Treatment Group

Task Reminder
Your task is to transform the interest rate column (*int_rate*) by rounding it to the nearest integer. This can be useful to group loans by whole-number interest rates (e.g. all loans

with a 3% interest rate, without distinguishing between those loans that have a 3.1% and 3.5% rate). This may be typically done as large increases in the interest rate may reflect lower creditworthiness.

Read the following 5 code fragments and then answer the questions below.

```
# Fragment 0
def f0(df_loan):
    # core cleaning code
    import pandas as pd
    # df_loan = pd.read_csv('../input/loan.csv', low_memory=False)
    df_loan['int_round'] = df_loan['int_rate'].round(0).astype(int)
    return df_loan


# Fragment 1
def f1(df):
    # core cleaning code
    import pandas as pd
    # df = pd.read_csv('../input/loan.csv', low_memory=False)
    df = df.rename(
        columns={
            'loan_amnt': 'loan_amount',
            'funded_amnt': 'funded_amount',
            'funded_amnt_inv': 'investor_funds',
            'int_rate': 'interest_rate',
            'annual_inc': 'annual_income'
        }
    )
    return df


# Fragment 2
def f2(data):
    # core cleaning code
    import pandas as pd
    # data = pd.read_csv('../input/loan.csv', low_memory=False)
    data['emp_length'] = data['emp_length'].astype(int)
    return data


# Fragment 3
def f3(dataset):
    # core cleaning code
    import pandas as pd
    # dataset = pd.read_csv('../input/loan.csv', low_memory=False)
    dataset = dataset.fillna(0)
    dataset['term'] = dataset['term'].astype('category').cat.codes
    return dataset


# Fragment 4
def f4(dataset):
    # core cleaning code
    import pandas as pd
    # dataset = pd.read_csv('../input/loan.csv', low_memory=False)
    dataset = dataset.fillna(0)
    dataset['verification_status_joint'] = dataset[
        'verification_status_joint'].astype('category').cat.codes
    return dataset
```

Which fragment snippets do you consider relevant to the task? You may leave blank if none appear relevant.

- ☐ Fragment 0
- ☐ Fragment 1
- ☐ Fragment 2
- ☐ Fragment 3
- ☐ Fragment 4

Which snippet do you consider to be most relevant to the task? (choose None, if none are relevant)

- ○ Fragment 0
- ○ Fragment 1
- ○ Fragment 2
- ○ Fragment 3
- ○ Fragment 4
- ○ None

Access to these snippets makes completing the task easier

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Neither agree nor disagree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Dataset 1, Task 2: Control group

Task Reminder

Your task is transform the interest rate column (*int_rate*) by rounding it to the nearest integer. This can be useful to group loans by whole-number interest rates (e.g. all loans with a 3% interest rate, without distinguishing between those loans that have a 3.1% and 3.5% rate). This may be typically done as large increases in the interest rate may reflect lower creditworthiness.

Read the following 5 code fragments and then answer the questions below.

```
# Fragment 0
```

```
def f0(loan):
    # core cleaning code
    import numpy as np
    import pandas as pd
    # loan = pd.read_csv('../input/loan.csv', low_memory=False)
    loan['title'] = np.where(loan['title'].isnull(), 0, loan['title'
    return loan


# Fragment 1
def f1(dataset):
    # core cleaning code
    import pandas as pd
    # dataset = pd.read_csv('../input/loan.csv', low_memory=False)
    dataset = dataset.fillna(0)
    dataset['verification_status'] = dataset['verification_status'].
        'category'
    ).cat.codes
    return dataset


# Fragment 2
def f2(df):
    # core cleaning code
    import pandas as pd
    # df = pd.read_csv('../input/loan.csv', low_memory=False)
    df = df.rename(
        columns={
            'loan_amnt': 'loan_amount',
            'funded_amnt': 'funded_amount',
            'funded_amnt_inv': 'investor_funds',
            'int_rate': 'interest_rate',
            'annual_inc': 'annual_income'
        }
    )
    group_dates = df.groupby(['complete_date', 'region'], as_index=F
    group_dates = group_dates.groupby(['issue_d', 'region'],
                                      as_index=False).sum()
    group_dates = group_dates.groupby(['issue_d', 'region'],
                                      as_index=False).sum()
    group_dates['loan_amount'] = (group_dates['loan_amount'] / 1000)
    by_loan_amount = df.groupby(['region', 'addr_state'],
                                as_index=False).loan_amount.sum()
    by_interest_rate = df.groupby(['region', 'addr_state'],
                                  as_index=False).interest_rate.mean
    return by_interest_rate


# Fragment 3
def f3(df):
    # core cleaning code
    import pandas as pd
    # df = pd.read_csv('../input/loan.csv', usecols=['loan_amnt', 'a
    statePop = {
        'CA': 39144818,
        'TX': 27469144,
        'FL': 20271878,
        'NY': 19795791,
        'IL': 12859995,
        'PA': 12802503,
        'OH': 11613423,
        'GA': 10214860,
        'NC': 10042802,
        'MI': 9922576,
        'NJ': 8958013,
        'VA': 8382993,
        'WA': 7170351,
        'AZ': 6828065,
        'MA': 6794422,
        'IN': 6619680,
        'TN': 6600299,
        'MO': 6083672,
        'MD': 6006401,
        'WI': 5771337,
        'MN': 5489594,
        'CO': 5456574,
        'SC': 4896146,
        'AL': 4858979,
        'LA': 4670724,
        'KY': 4425092,
        'OR': 4028977,
        'OK': 3911338,
        'CT': 3890886,
        'IA': 3123899,
        'UT': 2995919,
        'MS': 2992333,
        'AK': 2978204,
        'KS': 2911641,
        'NV': 2890845,
        'NM': 2085109,
        'NE': 1896190,
        'WV': 1844128,
        'ID': 1654930,
        'HI': 1431603,
        'NH': 1330608,
        'ME': 1329328,
        'RI': 1053298,
        'MT': 1032949,
        'DE': 945934,
        'SD': 858469,
        'ND': 756927,
        'AK': 738432,
        'DC': 672228,
        'VT': 626042,
        'WY': 586107
    }
    statePopdf = pd.DataFrame.from_dict(statePop, orient='index').re
    return statePopdf
```

```
# Fragment 4
def f4(df):
    # core cleaning code
    import pandas as pd
    from sklearn.preprocessing import LabelEncoder, OneHotEncoder
    # df = pd.read_csv('../input/loan.csv')
    df = df[((df.loan_status == 'Fully Paid') |
             (df.loan_status == 'Charged Off'))]
    df = df[(df['pymnt_plan'] == 'n')]
    df = df[(df['application_type'] == 'INDIVIDUAL')]
    df1 = df.drop(
        columns=[
            'policy_code', 'next_pymnt_d', 'out_prncp', 'out_prncp_i
            'pymnt_plan', 'initial_list_status', 'member_id', 'id',
            'application_type', 'grade', 'annual_inc_joint', 'dti_jc
        ]
    )
    df1 = df1.drop(
        columns=[
            'verification_status_joint', 'open_acc_6m', 'open_il_6m'
            'open_il_12m', 'open_il_24m', 'mths_since_rcnt_il', 'tot
            'il_util', 'open_rv_12m', 'open_rv_24m', 'max_bal_bc', '
            'inq_fi', 'total_cu_tl', 'inq_last_12m'
        ]
    )
    df1 = df1.drop(columns=['mths_since_last_major_derog'])
    lbl_enc = LabelEncoder()
    df1[(x + '_old')] = df[x]
    df1[x] = lbl_enc.fit_transform(df1[x])
    df1[(x + '_old')] = df[x]
    df1[x] = df1[x]
    df1[x] = lbl_enc.fit_transform(df1[x])
    df1.earliest_cr_line = pd.to_datetime(df1.earliest_cr_line, form
    df1['earliest_cr_line_month'] = df1.earliest_cr_line.dt.month

    return df1
```

Which fragment snippets do you consider relevant to the task? You may leave blank if none appear relevant.

- ☐ Fragment 0
- ☐ Fragment 1
- ☐ Fragment 2
- ☐ Fragment 3
- ☐ Fragment 4

Which snippet do you consider to be most relevant to the task? (choose None, if none are relevant)

- ○ Fragment 0
- ○ Fragment 1
- ○ Fragment 2
- ○ Fragment 3
- ○ Fragment 4
- ○ None

Access to these snippets makes completing the task easier

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Neither agree nor disagree
- ○ Somewhat disagree
- ○ Disagree

- ○ Strongly disagree

Dataset 1, Task 3

New Task
Your task is to compute the issuance month and year associated with each loan. You can use the *issue_d* column, which represents the issuance date for each loan. Computing the month/year of issuance can be useful to summarize total issuance amounts per year and during a year and observe changes.

The general task (i.e. ignoring dataset domain) is reflective of tasks you, as a participant, might perform in your own data analyses.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Neither agree nor disagree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Dataset 1, Task 3: Treatment Group

Task Reminder
Your task is to compute the issuance month and year associated with each loan. You can use the *issue_d* column, which represents the issuance date for each loan. Computing the month/year of issuance can be useful to summarize total issuance amounts per year and during a year and observe changes.

Read the following 5 code fragments and then answer the questions below.

```
# Fragment 0
def f0(df_loan):
    # core cleaning code
    import pandas as pd
    # df_loan = pd.read_csv('../input/loan.csv', low_memory=False)
    df_loan['issue_d'] = pd.to_datetime(df_loan['issue_d'])
    return df_loan


# Fragment 1
def f1(data):
    # core cleaning code

    import pandas as pd
    # data = pd.read_csv('../input/loan.csv', low_memory=False)
    data.earliest_cr_line = pd.to_datetime(data.earliest_cr_line)
    return data


# Fragment 2
def f2(df_loan):
    # core cleaning code
    import pandas as pd
    # df_loan = pd.read_csv('../input/loan.csv', low_memory=False)
    (df_loan['issue_month'],
     df_loan['issue_year']) = df_loan['issue_d'].str.split('-', 1).s
    return df_loan


# Fragment 3
def f3(data):
    # core cleaning code
    import pandas as pd
    # data = pd.read_csv('../input/loan.csv')
    data['issue_dt'] = pd.to_datetime(data.issue_d)
    return data


# Fragment 4
def f4(data):
    # core cleaning code
    import pandas as pd
    # data = pd.read_csv('../input/loan.csv')
    data['issue_dt'] = pd.to_datetime(data.issue_d)
    data['month'] = data['issue_dt'].dt.month
    return data
```

Which fragment snippets do you consider relevant to the task? You may leave blank if none appear relevant.

☐ Fragment 0
☐ Fragment 1
☐ Fragment 2
☐ Fragment 3
☐ Fragment 4

Which snippet do you consider to be most relevant to the task? (choose None, if none are relevant)

○ Fragment 0
○ Fragment 1
○ Fragment 2
○ Fragment 3
○ Fragment 4
○ None

Access to these snippets makes completing the task easier

○ Strongly agree
○ Agree
○ Somewhat agree
○ Neither agree nor disagree

- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Dataset 1, Task 3: Control group

Task Reminder
Your task is to compute the issuance month and year associated with each loan. You can use the *issue_d* column, which represents the issuance date for each loan. Computing the month/year of issuance can be useful to summarize total issuance amounts per year and during a year and observe changes.

Read the following 5 code fragments and then answer the questions below.

```
# Fragment 0
def f0(dataset):
    # core cleaning code
    import pandas as pd
    # dataset = pd.read_csv('../input/loan.csv', low_memory=False)
    dataset = dataset.fillna(0)
    dataset['term'] = dataset['term'].astype('category').cat.codes
    return dataset
```

```
# Fragment 1
def f1(df):
    # core cleaning code
    import pandas as pd
    # df = pd.read_csv('../input/loan.csv')
    df = df[((df.loan_status == 'Fully Paid') |
             (df.loan_status == 'Charged Off'))]
    df = df[(df['pymnt_plan'] == 'n')]
    df = df[(df['application_type'] == 'INDIVIDUAL')]
    df1 = df.drop(
        columns=[
            'policy_code', 'next_pymnt_d', 'out_prncp', 'out_prncp_i
            'pymnt_plan', 'initial_list_status', 'member_id', 'id',
            'application_type', 'grade', 'annual_inc_joint', 'dti_jc
        ]
    )
    emp_lengths = []
    df1.emp_length = emp_lengths
    return df1


# Fragment 2
def f2(df):
    # core cleaning code
    import pandas as pd
    # df = pd.read_csv('../input/loan.csv', low_memory=False)
    df = df.rename(
        columns={
            'loan_amnt': 'loan_amount',
            'funded_amnt': 'funded_amount',
            'funded_amnt_inv': 'investor_funds',
            'int_rate': 'interest_rate',
            'annual_inc': 'annual_income'
        }
    )
    group_dates = df.groupby(['complete_date', 'region'], as_index=F
    group_dates = group_dates.groupby(['issue_d', 'region'],
                                      as_index=False).sum()
    return group_dates


# Fragment 3
def f3(dataset):
    # core cleaning code
    import numpy as np
    import pandas as pd
    # dataset = pd.read_csv('../input/loan.csv', low_memory=False)
    dataset = dataset.fillna(0)
    dataset['loan_status'] = dataset['loan_status'].astype(
        'category'
    ).cat.codes
    non_numerics = [
        x for x in dataset.columns if (
            not ((dataset[x].dtype == np.float64) or
                 (dataset[x].dtype == np.int8) or
                 (dataset[x].dtype == np.int64))
        )
    ]
    df = dataset
    return df


# Fragment 4
def f4(data):
    # core cleaning code
    import numpy as np
    import pandas as pd
    # data = pd.read_csv('../input/loan.csv')
    data_1 = pd.DataFrame(data)
    category_one_data = data_1[(data_1.loan_status == 'Fully Paid')]
    category_two_data = data_1[(data_1.loan_status == 'Charged Off')
    new_data = np.vstack((category_one_data, category_two_data))
    new_data = new_data[(slice(None, None, None), slice(2, (-30), No
    new_data_df = pd.DataFrame(new_data)
    title = new_data_df[19]
    title = pd.DataFrame(title)
    title.columns = ['Title']
    return title
```

Which fragment snippets do you consider relevant to the task? You may leave blank if none appear relevant.

- ☐ Fragment 0
- ☐ Fragment 1
- ☐ Fragment 2
- ☐ Fragment 3
- ☐ Fragment 4

Which snippet do you consider to be most relevant to the task? (choose None, if none are relevant)

- ○ Fragment 0
- ○ Fragment 1
- ○ Fragment 2
- ○ Fragment 3
- ○ Fragment 4
- ○ None

Access to these snippets makes completing the task easier

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Neither agree nor disagree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

end_block

Thank you for your participation. At the end of the survey, you will be forwarded to Prolific.