

```
In [1]: from functools import reduce
import json
import numpy as np
import pandas as pd

pd.set_option("display.max_columns", 50)
pd.set_option("display.max_rows", 100)
pd.options.display.float_format = "{:,.2f}".format
```

Import and format the data

Concatenate the data into a single file.

```
In [3]: !sh process_eeoc.sh
```

```
process_eeoc.sh: 3: process_eeoc.sh: cannot create data/raw/charges_11_17.txt: Directory nonexistent
process_eeoc.sh: 4: process_eeoc.sh: cannot create data/raw/charges_11_17.txt: Directory nonexistent
process_eeoc.sh: 5: process_eeoc.sh: cannot create data/raw/charges_11_17.txt: Directory nonexistent
process_eeoc.sh: 6: process_eeoc.sh: cannot create data/raw/charges_11_17.txt: Directory nonexistent
process_eeoc.sh: 7: process_eeoc.sh: cannot create data/raw/charges_11_17.txt: Directory nonexistent
process_eeoc.sh: 8: process_eeoc.sh: cannot create data/raw/charges_11_17.txt: Directory nonexistent
process_eeoc.sh: 9: process_eeoc.sh: cannot create data/raw/charges_11_17.txt: Directory nonexistent
process_eeoc.sh: 10: process_eeoc.sh: cannot create data/raw/charges_11_17.txt: Directory nonexistent
```

Import charge data for fiscal years 2011-2017.

```
In [2]: charges = pd.read_csv("data/raw/charges_11_17.txt", sep="\t", skiprows=1,
                               dtype={1: str},
                               names=["fiscal_year", "charge_num", "state", "num_employees_c",
                                      "ode",
                                      "num_employees", "naics_code", "naics_desc", "type_cod",
                                      "e",
                                      "type", "birth_date", "sex", "date_received", "date_cl",
                                      "osed",
                                      "closure_code", "closure_action", "monetary_benefits",
                                      "statute_code",
                                      "statute", "basis_code", "basis", "issue_code", "issu",
                                      "e",
                                      "court_filing_date", "civil_action_num", "court", "res",
                                      "olution_date",
                                      "case_type", "litigation_monetary_benefits"])
charges.info()
```

```

-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-2-2cda4ee70352> in <module>
      7                                     "statute", "basis_code", "basis", "issue_co
de", "issue",
      8                                     "court_filing_date", "civil_action_num", "c
ourt", "resolution_date",
----> 9                                     "case_type", "litigation_monetary_benefit
s"]
     10 charges.info()

/opt/conda/lib/python3.7/site-packages/pandas/io/parsers.py in parser_f(filepath
_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, prefix,
mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinit
ialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, ve
rbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date
_parser, dayfirst, iterator, chunksize, compression, thousands, decimal, lineter
minator, quotechar, quoting, doublequote, escapechar, comment, encoding, dialec
t, tupleize_cols, error_bad_lines, warn_bad_lines, delim_whitespace, low_memory,
memory_map, float_precision)
     700             skip_blank_lines=skip_blank_lines)
     701
--> 702         return _read(filepath_or_buffer, kwds)
     703
     704     parser_f.__name__ = name

/opt/conda/lib/python3.7/site-packages/pandas/io/parsers.py in _read(filepath_or
_buffer, kwds)
     427
     428     # Create the parser.
--> 429     parser = TextFileReader(filepath_or_buffer, **kwds)
     430
     431     if chunksize or iterator:

/opt/conda/lib/python3.7/site-packages/pandas/io/parsers.py in __init__(self, f,
engine, **kwds)
     893         self.options['has_index_names'] = kwds['has_index_names']
     894
--> 895         self._make_engine(self.engine)
     896
     897     def close(self):

/opt/conda/lib/python3.7/site-packages/pandas/io/parsers.py in _make_engine(sel
f, engine)
    1120     def _make_engine(self, engine='c'):
    1121         if engine == 'c':
-> 1122             self._engine = CParserWrapper(self.f, **self.options)
    1123         else:
    1124             if engine == 'python':

/opt/conda/lib/python3.7/site-packages/pandas/io/parsers.py in __init__(self, sr
c, **kwds)
    1851         kwds['usecols'] = self.usecols
    1852
-> 1853         self._reader = parsers.TextReader(src, **kwds)
    1854         self.unnamed_cols = self._reader.unnamed_cols
    1855

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source
()

FileNotFoundError: [Errno 2] File b'data/raw/charges_11_17.txt' does not exist:

```

Convert the date columns.

```
In [4]: charges["birth_date"] = pd.to_datetime(charges["birth_date"], errors="coerce", format="%m/%d/%Y")
charges["date_received"] = pd.to_datetime(charges["date_received"], errors="coerce", format="%m/%d/%Y")
charges["date_closed"] = pd.to_datetime(charges["date_closed"], errors="coerce", format="%m/%d/%Y")
charges["court_filing_date"] = pd.to_datetime(charges["court_filing_date"], errors="coerce", format="%m/%d/%Y")
charges["resolution_date"] = pd.to_datetime(charges["resolution_date"], errors="coerce", format="%m/%d/%Y")
charges.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3443509 entries, 0 to 3443508
Data columns (total 28 columns):
fiscal_year                object
charge_num                object
state                     object
num_employees_code        object
num_employees              object
naics_code                 float64
naics_desc                 object
type_code                  object
type                       object
birth_date                 datetime64[ns]
sex                       object
date_received              datetime64[ns]
date_closed                datetime64[ns]
closure_code               object
closure_action             object
monetary_benefits          float64
statute_code               object
statute                    object
basis_code                 object
basis                      object
issue_code                 object
issue                      object
court_filing_date          datetime64[ns]
civil_action_num           object
court                      object
resolution_date            datetime64[ns]
case_type                  object
litigation_monetary_benefits float64
dtypes: datetime64[ns](5), float64(3), object(20)
memory usage: 735.6+ MB
```

Import charge data for fiscal year 2010.

```
In [5]: charges_10 = pd.read_csv("data/complaints_10.txt", sep="\t", skiprows=1,
                                dtype={0: str},
                                names=["charge_num", "state", "num_employees_code",
                                        "num_employees", "naics_code", "naics_desc", "type_
code",
                                        "type", "birth_date", "sex", "date_received", "date
_fepa_sent_to_eEOC",
                                        "date_closed", "closure_code", "closure_action", "m
onetary_benefits",
                                        "statute_code", "statute", "basis_code", "basis", "
issue_code", "issue",
                                        "court_filing_date", "civil_action_num", "court", "
resolution_date",
                                        "litigation_monetary_benefits", "case_type"])
charges_10.info()
```

/home/jyerardi/anaconda3/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2698: DtypeWarning: Columns (22,23,24,25,26) have mixed types. Specify dtype option on import or set low_memory=False.

interactivity=interactivity, compiler=compiler, result=result)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 343863 entries, 0 to 343862
Data columns (total 28 columns):
charge_num          343863 non-null object
state              343801 non-null object
num_employees_code  328035 non-null object
num_employees       328035 non-null object
naics_code         187222 non-null float64
naics_desc         185282 non-null object
type_code          343829 non-null object
type              343829 non-null object
birth_date         310000 non-null object
sex               343453 non-null object
date_received      343863 non-null object
date_fepa_sent_to_eEOC 20188 non-null object
date_closed        230050 non-null object
closure_code       230050 non-null object
closure_action     230050 non-null object
monetary_benefits  37964 non-null object
statute_code       343863 non-null object
statute           343367 non-null object
basis_code        342598 non-null object
basis            343859 non-null object
issue_code        343863 non-null object
issue            343863 non-null object
court_filing_date  366 non-null object
civil_action_num   355 non-null object
court            355 non-null object
resolution_date    41 non-null object
litigation_monetary_benefits 11 non-null object
case_type         894 non-null object
dtypes: float64(1), object(27)
memory usage: 73.5+ MB
```

Convert the date columns.

```
In [6]: charges_10["birth_date"] = pd.to_datetime(charges_10["birth_date"], errors="coerce", format="%m/%d/%y")
charges_10["date_received"] = pd.to_datetime(charges_10["date_received"], errors="coerce", format="%m/%d/%y")
charges_10["date_closed"] = pd.to_datetime(charges_10["date_closed"], errors="coerce", format="%m/%d/%y")
charges_10["court_filing_date"] = pd.to_datetime(charges_10["court_filing_date"], errors="coerce", format="%m/%d/%y")
charges_10["resolution_date"] = pd.to_datetime(charges_10["resolution_date"], errors="coerce", format="%m/%d/%y")
charges_10.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 343863 entries, 0 to 343862
Data columns (total 28 columns):
charge_num          343863 non-null object
state              343801 non-null object
num_employees_code  328035 non-null object
num_employees       328035 non-null object
naics_code         187222 non-null float64
naics_desc         185282 non-null object
type_code          343829 non-null object
type               343829 non-null object
birth_date         310000 non-null datetime64[ns]
sex                343453 non-null object
date_received      343863 non-null datetime64[ns]
date_fepa_sent_to_eec 20188 non-null object
date_closed        230050 non-null datetime64[ns]
closure_code       230050 non-null object
closure_action     230050 non-null object
monetary_benefits  37964 non-null object
statute_code       343863 non-null object
statute            343367 non-null object
basis_code         342598 non-null object
basis              343859 non-null object
issue_code         343863 non-null object
issue              343863 non-null object
court_filing_date  366 non-null datetime64[ns]
civil_action_num   355 non-null object
court              355 non-null object
resolution_date    41 non-null datetime64[ns]
litigation_monetary_benefits 11 non-null object
case_type          894 non-null object
dtypes: datetime64[ns](5), float64(1), object(22)
memory usage: 73.5+ MB
```

The columns in the 2010 charge data differ from those of the 2011 through 2017 data. We need to delete, add, rename and reorder the columns before concatenating the data.

```
In [7]: charges_10.drop("date_fepa_sent_to_eec", axis=1, inplace=True)
charges_10["fiscal_year"] = "FY2010"
charges_10 = charges_10[["fiscal_year", "charge_num", "state", "num_employees_code", "num_employees", "naics_code", "naics_desc", "type_code", "type", "birth_date", "sex", "date_received", "date_closed", "closure_code", "closure_action", "monetary_benefits", "statute_code", "statute", "basis_code", "basis", "issue_code", "issue", "court_filing_date", "civil_action_num", "court", "resolution_date", "case_type", "litigation_monetary_benefits"]]
charges_10.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 343863 entries, 0 to 343862
Data columns (total 28 columns):
fiscal_year      343863 non-null object
charge_num      343863 non-null object
state           343801 non-null object
num_employees_code  328035 non-null object
num_employees    328035 non-null object
naics_code      187222 non-null float64
naics_desc      185282 non-null object
type_code       343829 non-null object
type            343829 non-null object
birth_date      310000 non-null datetime64[ns]
sex             343453 non-null object
date_received   343863 non-null datetime64[ns]
date_closed     230050 non-null datetime64[ns]
closure_code    230050 non-null object
closure_action  230050 non-null object
monetary_benefits  37964 non-null object
statute_code    343863 non-null object
statute         343367 non-null object
basis_code      342598 non-null object
basis           343859 non-null object
issue_code      343863 non-null object
issue           343863 non-null object
court_filing_date  366 non-null datetime64[ns]
civil_action_num  355 non-null object
court           355 non-null object
resolution_date  41 non-null datetime64[ns]
case_type       894 non-null object
litigation_monetary_benefits  11 non-null object
dtypes: datetime64[ns](5), float64(1), object(22)
memory usage: 73.5+ MB
```

Concatenate the two sets of charge data.

```
In [8]: charges = pd.concat([charges, charges_10], ignore_index=True)
charges.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3787372 entries, 0 to 3787371
Data columns (total 28 columns):
fiscal_year                object
charge_num                object
state                     object
num_employees_code        object
num_employees             object
naics_code                float64
naics_desc                object
type_code                object
type                     object
birth_date                datetime64[ns]
sex                       object
date_received             datetime64[ns]
date_closed               datetime64[ns]
closure_code              object
closure_action            object
monetary_benefits         object
statute_code              object
statute                   object
basis_code                object
basis                     object
issue_code                object
issue                     object
court_filing_date         datetime64[ns]
civil_action_num          object
court                     object
resolution_date           datetime64[ns]
case_type                 object
litigation_monetary_benefits object
dtypes: datetime64[ns](5), float64(1), object(22)
memory usage: 809.1+ MB
```

Convert the columns to their proper data types.


```

In [9]: charges = charges.astype(dtype={"fiscal_year": "category", "charge_num": str, "state": "category",
                                         "num_employees_code": "category", "num_employees": "category",
                                         "naics_code": "category", "naics_desc": "category",
                                         "type_code": "category", "type": "category", "sex": "category",
                                         "closure_code": "category", "closure_action": "category",
                                         "statute_code": "category", "statute": "category",
                                         "basis_code": "category", "basis": "category", "issue_code": "category",
                                         "issue": "category", "court": "category", "case_type": "category"})
charges["monetary_benefits"] = pd.to_numeric(charges["monetary_benefits"], errors="coerce", downcast="float")
charges["litigation_monetary_benefits"] = pd.to_numeric(charges["litigation_monetary_benefits"], errors="coerce", downcast="float")
charges.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3787372 entries, 0 to 3787371
Data columns (total 28 columns):
fiscal_year                category
charge_num                 object
state                     category
num_employees_code         category
num_employees              category
naics_code                 category
naics_desc                 category
type_code                 category
type                      category
birth_date                datetime64[ns]
sex                       category
date_received              datetime64[ns]
date_closed                datetime64[ns]
closure_code               category
closure_action             category
monetary_benefits          float32
statute_code               category
statute                   category
basis_code                 category
basis                     category
issue_code                 category
issue                     category
court_filing_date          datetime64[ns]
civil_action_num           object
court                     category
resolution_date            datetime64[ns]
case_type                  category
litigation_monetary_benefits float32
dtypes: category(19), datetime64[ns](5), float32(2), object(2)
memory usage: 307.1+ MB

```

Create a grouped basis column.

```

In [10]: charges["grouped_basis"] = np.where(charges["basis"].str.contains("age", case=False), "Age",
                                             np.where(charges["basis"].str.contains("color", case=False), "Color",
                                             np.where(charges["basis"].isin(["Alcoholism", "Allergies", "Alzheimers",
                                             "Asthma", "Autism", "Blood
                                             (Other)",
                                             "Brain/Head Impairment", "
                                             Brain/Head Injury (Traumatic)", "
                                             Cancer", "Cerebral Palsy", "Chemical Sensitivity",
                                             "Cumulative Trauma Disorder", "Cystic Fibrosis",
                                             "Depression", "Diabetes",
                                             "Disfigurement",
                                             "Drug Addiction", "Dwarfism", "Epilepsy",
                                             "Gastrointestinal", "HIV",
                                             "Handicap (Not ADA)",
                                             "Hearing Impairment", "Heart/Cardiovascular",
                                             "Intellectual Disability",
                                             "Kidney Impairment",
                                             "Learning Disability", "Manic Depression (Bi-polar)",
                                             "Missing Digits/Limbs", "Multiple Sclerosis",
                                             "Nonparalytic Orthopedic Impairment",
                                             "Orthopedic/Structural Back Impairment",
                                             "Other Anxiety Disorder",
                                             "Other Disability",
                                             "Other Neurological", "Other Psychiatric Disorders",
                                             "Other Pulmo/Respiratory",
                                             "Paralysis",
                                             "Post-Traumatic Stress Disorder", "Record Of Disability",
                                             "Regarded As Disabled", "Schizophrenia",
                                             "Speech Impairment", "Tuberculosis", "Vision Impairment"]), "Disability/Medical",
                                             np.where(charges["basis"].str.contains("equal pay", case=False), "Equal Pay",
                                             np.where(charges["basis"].str.contains("genetic", case=False), "Genetics",
                                             np.where(charges["basis"].str.contains("ancestry", case=False) | charges["basis"].str.contains("national origin", case=False), "Ancestry/National Origin",
                                             np.where(charges["basis"].isin(["Conviction Record", "Marital Status", "Other",
                                             "Relationship/Assn.", "Unassigned"]), "Other",
                                             np.where(charges["basis"].str.contains("race", case=False), "Race",
                                             np.where(charges["basis"].str.contains("retaliation", case=False), "Retaliation",
                                             np.where(charges["basis"].str.contains("religion", case=False), "Religion",
                                             np.where(charges["basis"].str.contains("sex", case=False), "Sex",
                                             "")))))))))

```

```
Out[10]: grouped_basis
Age                384691
Ancestry/National Origin  241419
Color              73329
Disability/Medical  713753
Equal Pay          12963
Genetics           4378
Other              58394
Race               645117
Religion           78711
Retaliation        955242
Sex                619375
Name: charge_num, dtype: int64
```

How many alleged violations are we dealing with?

```
In [11]: charges["charge_num"].count()
```

```
Out[11]: 3787372
```

And how many cases are we dealing with?

```
In [12]: charges["charge_num"].nunique()
```

```
Out[12]: 1056978
```

How many of these cases occurred in each year?

```
In [13]: all_cases_by_year = charges.groupby("fiscal_year")["charge_num"].nunique().reset_in
dex()
all_cases_by_year.rename(columns={"charge_num": "all_cases"}, inplace=True)
all_cases_by_year
```

```
Out[13]:
```

	fiscal_year	all_cases
0	FY2010	104514
1	FY2011	146123
2	FY2012	146294
3	FY2013	139852
4	FY2014	134106
5	FY2015	130871
6	FY2016	132519
7	FY2017	122824

How does this break down by basis?

```
In [14]: all_cases_by_basis = charges.groupby(["basis"])["charge_num"].nunique().reset_index()
all_cases_by_basis.rename(columns={"charge_num": "all_cases"}, inplace=True)
all_cases_by_basis.sort_values("all_cases", ascending=False).head()
```

Out[14]:

	basis	all_cases
74	Retaliation	412515
60	Race-Black/African American	279225
0	Age	231539
76	Sex-Female	203779
50	Other Disability	109168

How does this break down by grouped basis?

```
In [15]: all_cases_by_grouped_basis = charges.groupby(["grouped_basis"])["charge_num"].nunique().reset_index()
all_cases_by_grouped_basis.rename(columns={"charge_num": "all_cases"}, inplace=True)
all_cases_by_grouped_basis.sort_values("all_cases", ascending=False).head()
```

Out[15]:

	grouped_basis	all_cases
9	Retaliation	412515
7	Race	350112
10	Sex	309303
3	Disability/Medical	290739
0	Age	231544

How many closed cases are we dealing with?

```
In [16]: closed_charges = charges[charges["closure_action"].notnull()]
closed_cases = charges[closed_charges["closure_action"].notnull()]
closed_cases["charge_num"].nunique()
```

Out[16]: 928767

And how many of these cases were essentially dismissed outright?

```
In [17]: closed_cases[closed_cases["closure_action"] == "No Cause Finding Issued"]["charge_num"].nunique()
```

Out[17]: 603606

```
In [18]: closed_cases[closed_cases["closure_action"] == "No Cause Finding Issued"]["charge_num"].nunique() / closed_cases["charge_num"].nunique()
```

Out[18]: 0.6499003517566838

How many of these occurred in each year?

```
In [19]: closed_cases_by_year = closed_cases.groupby("fiscal_year")["charge_num"].nunique().reset_index()
closed_cases_by_year.rename(columns={"charge_num": "closed_cases"}, inplace=True)
closed_cases_by_year
```

Out[19]:

	fiscal_year	closed_cases
0	FY2010	72894
1	FY2011	145116
2	FY2012	142801
3	FY2013	137169
4	FY2014	128130
5	FY2015	111254
6	FY2016	117586
7	FY2017	73818

How does this break down by basis?

```
In [20]: closed_cases_by_basis = closed_cases.groupby(["basis"])["charge_num"].nunique().reset_index()
closed_cases_by_basis.rename(columns={"charge_num": "closed_cases"}, inplace=True)
closed_cases_by_basis.sort_values("closed_cases", ascending=False).head()
```

Out[20]:

	basis	closed_cases
74	Retaliation	357958
60	Race-Black/African American	245966
0	Age	205352
76	Sex-Female	175415
50	Other Disability	94349

How does this break down by grouped basis?

```
In [21]: closed_cases_by_grouped_basis = closed_cases.groupby(["grouped_basis"])["charge_num"].nunique().reset_index()
closed_cases_by_grouped_basis.rename(columns={"charge_num": "closed_cases"}, inplace=True)
closed_cases_by_grouped_basis.sort_values("closed_cases", ascending=False).head()
```

Out[21]:

	grouped_basis	closed_cases
9	Retaliation	357958
7	Race	309201
10	Sex	268660
3	Disability/Medical	252599
0	Age	205355

In how many cases did the EEOC find merit to the complaint?

```
In [22]: meritorious_outcomes = ["Case Settled By Legal Unit", "Conciliation Failure", "Successful Conciliation"]
meritorious_cases = closed_cases[closed_cases["closure_action"].isin(meritorious_outcomes)]
meritorious_cases["charge_num"].nunique()
```

Out[22]: 17253

```
In [23]: meritorious_cases["charge_num"].nunique() / closed_cases["charge_num"].nunique()
```

Out[23]: 0.01857624140392585

Not every case deemed meritorious by the agencies resulted in a worker receiving some form of relief. In how many such cases did a worker not receive relief?

```
In [24]: meritorious_cases[meritorious_cases["closure_action"] == "Conciliation Failure"]["charge_num"].nunique() / meritorious_cases["charge_num"].nunique()
```

Out[24]: 0.5332985567727352

How many meritorious cases occurred in each year?

```
In [25]: meritorious_cases_by_year = meritorious_cases.groupby("fiscal_year")["charge_num"].nunique().reset_index()
meritorious_cases_by_year.rename(columns={"charge_num": "meritorious_cases"}, inplace=True)
meritorious_cases_by_year
```

Out[25]:

	fiscal_year	meritorious_cases
0	FY2010	885
1	FY2011	4369
2	FY2012	3411
3	FY2013	2888
4	FY2014	2414
5	FY2015	1478
6	FY2016	1482
7	FY2017	327

How does this break down by basis?

```
In [26]: meritorious_cases_by_basis = meritorious_cases.groupby(["basis"])["charge_num"].nunique().reset_index()
meritorious_cases_by_basis.rename(columns={"charge_num": "meritorious_cases"}, inplace=True)
meritorious_cases_by_basis.sort_values("meritorious_cases", ascending=False).head()
```

Out [26]:

	basis	meritorious_cases
74	Retaliation	6700
76	Sex-Female	3840
60	Race-Black/African American	3095
0	Age	2989
50	Other Disability	2188

How does this break down by grouped basis?

```
In [27]: meritorious_cases_by_grouped_basis = meritorious_cases.groupby(["grouped_basis"])["charge_num"].nunique().reset_index()
meritorious_cases_by_grouped_basis.rename(columns={"charge_num": "meritorious_cases"}, inplace=True)
meritorious_cases_by_grouped_basis.sort_values("meritorious_cases", ascending=False).head()
```

Out [27]:

	grouped_basis	meritorious_cases
9	Retaliation	6700
3	Disability/Medical	5908
10	Sex	5473
7	Race	3735
0	Age	2989

And in how many cases did the EEOC grant some form of relief (including non-monetary relief) to the complainant?

```
In [28]: relief_outcomes = ["Case Settled By Legal Unit", "Settlement With Benefits", "Successful Conciliation", "Withdrawal With Benefits"]
relief_cases = closed_cases[closed_cases["closure_action"].isin(relief_outcomes)]
relief_cases["charge_num"].nunique()
```

Out [28]: 164376

```
In [29]: relief_cases["charge_num"].nunique() / closed_cases["charge_num"].nunique()
```

Out [29]: 0.1769830323428804

How many of these cases occurred in each year?

```
In [30]: relief_cases_by_year = relief_cases.groupby("fiscal_year")["charge_num"].nunique().reset_index()
relief_cases_by_year.rename(columns={"charge_num": "relief_cases"}, inplace=True)
relief_cases_by_year
```

Out[30]:

	fiscal_year	relief_cases
0	FY2010	13372
1	FY2011	26591
2	FY2012	24993
3	FY2013	24071
4	FY2014	22570
5	FY2015	20639
6	FY2016	19850
7	FY2017	12290

How does this break down by basis?

```
In [31]: relief_cases_by_basis = relief_cases.groupby("basis")["charge_num"].nunique().reset_index()
relief_cases_by_basis.rename(columns={"charge_num": "relief_cases"}, inplace=True)
relief_cases_by_basis.sort_values("relief_cases", ascending=False).head()
```

Out[31]:

	basis	relief_cases
74	Retaliation	61607
60	Race-Black/African American	37244
76	Sex-Female	34860
0	Age	33799
50	Other Disability	19613

How does this break down by grouped basis?

```
In [32]: relief_cases_by_grouped_basis = relief_cases.groupby("grouped_basis")["charge_num"].nunique().reset_index()
relief_cases_by_grouped_basis.rename(columns={"charge_num": "relief_cases"}, inplace=True)
relief_cases_by_grouped_basis.sort_values("relief_cases", ascending=False).head()
```

Out[32]:

	grouped_basis	relief_cases
9	Retaliation	61607
3	Disability/Medical	52120
10	Sex	51930
7	Race	45802
0	Age	33799

And in how many cases did a worker see any monetary benefits?

```
In [33]: monetary_benefits_cases = closed_cases[closed_charges["monetary_benefits"] > 0]
         monetary_benefits_cases["charge_num"].nunique()
```

Out[33]: 112169

```
In [34]: monetary_benefits_cases["charge_num"].nunique() / closed_cases["charge_num"].nunique()
```

Out[34]: 0.12077194818506687

How much money did they get?

```
In [35]: monetary_benefits_cases.groupby(["charge_num"])["monetary_benefits"].mean().sum()
```

Out[35]: 2353018000.0

How many of these cases occurred in each year?

```
In [36]: monetary_benefits_cases_by_year = monetary_benefits_cases.groupby("fiscal_year")["charge_num"].nunique().reset_index()
         monetary_benefits_cases_by_year.rename(columns={"charge_num": "monetary_benefits_cases"}, inplace=True)
         monetary_benefits_cases_by_year
```

Out[36]:

	fiscal_year	monetary_benefits_cases
0	FY2010	1008
1	FY2011	19414
2	FY2012	18254
3	FY2013	17451
4	FY2014	16440
5	FY2015	15316
6	FY2016	14880
7	FY2017	9406

How does this break down by basis?

```
In [37]: monetary_benefits_cases_by_basis = monetary_benefits_cases.groupby("basis")["charge_num"].nunique().reset_index()
monetary_benefits_cases_by_basis.rename(columns={"charge_num": "monetary_benefits_cases"}, inplace=True)
monetary_benefits_cases_by_basis.sort_values("monetary_benefits_cases", ascending=False).head()
```

Out [37]:

	basis	monetary_benefits_cases
74	Retaliation	44295
60	Race-Black/African American	26505
76	Sex-Female	24603
0	Age	22661
50	Other Disability	12464

How does this break down by grouped basis?

```
In [38]: monetary_benefits_cases_by_grouped_basis = monetary_benefits_cases.groupby("grouped_basis")["charge_num"].nunique().reset_index()
monetary_benefits_cases_by_grouped_basis.rename(columns={"charge_num": "monetary_benefits_cases"}, inplace=True)
monetary_benefits_cases_by_grouped_basis.sort_values("monetary_benefits_cases", ascending=False).head()
```

Out [38]:

	grouped_basis	monetary_benefits_cases
9	Retaliation	44295
10	Sex	36422
3	Disability/Medical	35613
7	Race	32311
0	Age	22661

How many cases alleged some form of racial discrimination?

```
In [39]: race_discrimination = charges[charges["grouped_basis"] == "Race"]
race_discrimination["charge_num"].nunique()
```

Out [39]: 350112

```
In [40]: race_discrimination["charge_num"].nunique() / charges["charge_num"].nunique()
```

Out [40]: 0.331238682356681

How many cases alleged racial discrimination against African-Americans?

```
In [41]: aa_discrimination = charges[charges["basis"] == "Race-Black/African American"]
aa_discrimination["charge_num"].nunique()
```

Out [41]: 279225

```
In [42]: aa_discrimination["charge_num"].nunique() / charges["charge_num"].nunique()
```

```
Out[42]: 0.26417295345787706
```

Combine the data by year.

```
In [43]: cases_by_year_dfs = [all_cases_by_year, closed_cases_by_year, meritorious_cases_by_year, relief_cases_by_year, monetary_benefits_cases_by_year]
cases_by_year = reduce(lambda left, right: pd.merge(left, right, on="fiscal_year"),
cases_by_year_dfs)
cases_by_year
```

```
Out[43]:
```

	fiscal_year	all_cases	closed_cases	meritorious_cases	relief_cases	monetary_benefits_cases
0	FY2010	104514	72894	885	13372	1008
1	FY2011	146123	145116	4369	26591	19414
2	FY2012	146294	142801	3411	24993	18254
3	FY2013	139852	137169	2888	24071	17451
4	FY2014	134106	128130	2414	22570	16440
5	FY2015	130871	111254	1478	20639	15316
6	FY2016	132519	117586	1482	19850	14880
7	FY2017	122824	73818	327	12290	9406

Calculate the proportion of all cases that fall into each category each fiscal year.

```
In [44]: cases_by_year["pct_all_cases_closed"] = cases_by_year["closed_cases"] / cases_by_year["all_cases"]
cases_by_year["pct_closed_cases_meritorious"] = cases_by_year["meritorious_cases"] / cases_by_year["closed_cases"]
cases_by_year["pct_closed_cases_relief"] = cases_by_year["relief_cases"] / cases_by_year["closed_cases"]
cases_by_year["pct_closed_cases_monetary_benefits"] = cases_by_year["monetary_benefits_cases"] / cases_by_year["closed_cases"]
cases_by_year = cases_by_year[["fiscal_year", "all_cases", "closed_cases", "pct_all_cases_closed", "meritorious_cases", "pct_closed_cases_meritorious", "relief_cases", "pct_closed_cases_relief", "monetary_benefits_cases", "pct_closed_cases_monetary_benefits"]]
cases_by_year
```

```
Out[44]:
```

	fiscal_year	all_cases	closed_cases	pct_all_cases_closed	meritorious_cases	pct_closed_cases_meritorious
0	FY2010	104514	72894	0.70	885	0.01
1	FY2011	146123	145116	0.99	4369	0.03
2	FY2012	146294	142801	0.98	3411	0.02
3	FY2013	139852	137169	0.98	2888	0.02
4	FY2014	134106	128130	0.96	2414	0.02
5	FY2015	130871	111254	0.85	1478	0.01
6	FY2016	132519	117586	0.89	1482	0.01
7	FY2017	122824	73818	0.60	327	0.00

Combine the data by basis.

```
In [45]: cases_by_basis_dfs = [all_cases_by_basis, closed_cases_by_basis, meritorious_cases_
by_basis, relief_cases_by_basis, monetary_benefits_cases_by_basis]
cases_by_basis = reduce(lambda left, right: pd.merge(left, right, on="basis"), case
s_by_basis_dfs)
cases_by_basis.sort_values("all_cases", ascending=False).head()
```

Out [45]:

	basis	all_cases	closed_cases	meritorious_cases	relief_cases	monetary_benefits_cases
74	Retaliation	412515	357958	6700	61607	44295
60	Race-Black/African American	279225	245966	3095	37244	26505
0	Age	231539	205352	2989	33799	22661
76	Sex-Female	203779	175415	3840	34860	24603
50	Other Disability	109168	94349	2188	19613	12464

Calculate the proportion of all cases that fall into each category by basis.

```
In [46]: cases_by_basis["pct_all_cases_closed"] = cases_by_basis["closed_cases"] / cases_by_
basis["all_cases"]
cases_by_basis["pct_closed_cases_meritorious"] = cases_by_basis["meritorious_case
s"] / cases_by_basis["closed_cases"]
cases_by_basis["pct_closed_cases_relief"] = cases_by_basis["relief_cases"] / cases_
by_basis["closed_cases"]
cases_by_basis["pct_closed_cases_monetary_benefits"] = cases_by_basis["monetary_ben
efits_cases"] / cases_by_basis["closed_cases"]
cases_by_basis = cases_by_basis[["basis", "all_cases", "closed_cases", "pct_all_cas
es_closed", "meritorious_cases", "pct_closed_cases_meritorious", "relief_cases", "p
ct_closed_cases_relief", "monetary_benefits_cases", "pct_closed_cases_monetary_bene
fits"]]
cases_by_basis.sort_values("all_cases", ascending=False).head()
```

Out [46]:

	basis	all_cases	closed_cases	pct_all_cases_closed	meritorious_cases	pct_closed_cases_meritoriou
74	Retaliation	412515	357958	0.87	6700	0.0
60	Race-Black/African American	279225	245966	0.88	3095	0.0
0	Age	231539	205352	0.89	2989	0.0
76	Sex-Female	203779	175415	0.86	3840	0.0
50	Other Disability	109168	94349	0.86	2188	0.0

Combine the data by grouped basis.

```
In [47]: cases_by_grouped_basis_dfs = [all_cases_by_grouped_basis, closed_cases_by_grouped_b
asis, meritorious_cases_by_grouped_basis, relief_cases_by_grouped_basis, monetary_b
enefits_cases_by_grouped_basis]
cases_by_grouped_basis = reduce(lambda left, right: pd.merge(left, right, on="group
ed_basis"), cases_by_grouped_basis_dfs)
cases_by_grouped_basis.sort_values("all_cases", ascending=False).head()
```

Out [47]:

	grouped_basis	all_cases	closed_cases	meritorious_cases	relief_cases	monetary_benefits_cases
9	Retaliation	412515	357958	6700	61607	44295
7	Race	350112	309201	3735	45802	32311
10	Sex	309303	268660	5473	51930	36422
3	Disability/Medical	290739	252599	5908	52120	35613
0	Age	231544	205355	2989	33799	22661

Calculate the proportion of all cases that fall into each category by grouped basis.

```
In [48]: cases_by_grouped_basis["pct_all_cases_closed"] = cases_by_grouped_basis["closed_cas
es"] / cases_by_grouped_basis["all_cases"]
cases_by_grouped_basis["pct_closed_cases_meritorious"] = cases_by_grouped_basis["me
ritorious_cases"] / cases_by_grouped_basis["closed_cases"]
cases_by_grouped_basis["pct_closed_cases_relief"] = cases_by_grouped_basis["relief_
cases"] / cases_by_grouped_basis["closed_cases"]
cases_by_grouped_basis["pct_closed_cases_monetary_benefits"] = cases_by_grouped_bas
is["monetary_benefits_cases"] / cases_by_grouped_basis["closed_cases"]
cases_by_grouped_basis = cases_by_grouped_basis[["grouped_basis", "all_cases", "clo
sed_cases", "pct_all_cases_closed", "meritorious_cases", "pct_closed_cases_meritori
ous", "relief_cases", "pct_closed_cases_relief", "monetary_benefits_cases", "pct_cl
osed_cases_monetary_benefits"]]
cases_by_grouped_basis.sort_values("all_cases", ascending=False).head()
```

Out [48]:

	grouped_basis	all_cases	closed_cases	pct_all_cases_closed	meritorious_cases	pct_closed_cases_merit
9	Retaliation	412515	357958	0.87	6700	
7	Race	350112	309201	0.88	3735	
10	Sex	309303	268660	0.87	5473	
3	Disability/Medical	290739	252599	0.87	5908	
0	Age	231544	205355	0.89	2989	

```
In [49]: charges.groupby("charge_num")["date_received"].nunique().reset_index().sort_values
("date_received", ascending=False).head(1)
```

/home/jyerardi/anaconda3/lib/python3.6/site-packages/pandas/core/groupby/groupby.py:3764: FutureWarning: In the future, NAT != NAT will be True rather than False.

```
inc = np.r_[1, val[1:] != val[:-1]]
```

Out [49]:

	charge_num	date_received
888641	5688973	2

```
In [50]: charges[charges["charge_num"] == "5688973"]
```

```
Out [50]:
```

	fiscal_year	charge_num	state	num_employees_code	num_employees	naics_code	naics_desc	type
582651	FY2016	5688973	CA	N	Under 15 Employees	nan	NaN	
1154228	FY2015	5688973	CA	N	Under 15 Employees	nan	NaN	

What sorts of discriminatory bases were most likely to result in an outcome where the EEOC found merit in the complaint, the complainant got some form of relief and in which the complainant saw any monetary benefits and how does that compare with the overall number of those violations (minimum 100 closed cases)?

```
In [51]: cases_by_basis[cases_by_basis["closed_cases"] >= 100].sort_values("pct_closed_cases_meritorious", ascending=False).head()
```

```
Out [51]:
```

	basis	all_cases	closed_cases	pct_all_cases_closed	meritorious_cases	pct_closed_cases_meritorious
28	Genetic Information Discrimination	1853	1530	0.83	101	
67	Religion-7th Day Adventist	1205	1029	0.85	62	
77	Sex-Gender Identity/Transgender	2082	1639	0.79	78	
48	Other	22499	19740	0.88	928	
21	Drug Addiction	1365	1223	0.90	54	

```
In [52]: cases_by_basis[cases_by_basis["closed_cases"] >= 100].sort_values("pct_closed_cases_relief", ascending=False).head()
```

```
Out [52]:
```

	basis	all_cases	closed_cases	pct_all_cases_closed	meritorious_cases	pct_closed_cases_meritorious
11	Cancer	9749	8317	0.85	284	0.034
79	Sex-Pregnancy	41019	35676	0.87	944	0.026
67	Religion-7th Day Adventist	1205	1029	0.85	62	0.060
16	Cumulative Trauma Disorder	378	331	0.88	7	0.021
12	Cerebral Palsy	807	701	0.87	25	0.036

```
In [53]: cases_by_basis[cases_by_basis["closed_cases"] >= 100].sort_values("pct_closed_cases_monetary_benefits", ascending=False).head()
```

Out [53]:

	basis	all_cases	closed_cases	pct_all_cases_closed	meritorious_cases	pct_closed_cases_meritorious
79	Sex-Pregnancy	41019	35676	0.87	944	0.03
11	Cancer	9749	8317	0.85	284	0.03
65	Regarded As Disabled	30638	26776	0.87	923	0.03
35	Kidney Impairment	2181	1872	0.86	56	0.03
30	HIV	2138	1800	0.84	77	0.04

What sorts of discriminatory grouped bases were most likely to result in an outcome where the EEOC found merit in the complaint, the complainant got some form of relief and in which the complainant saw any monetary benefits and how does that compare with the overall number of those violations (minimum 100 closed cases)?

```
In [54]: cases_by_grouped_basis[cases_by_grouped_basis["closed_cases"] >= 100].sort_values("pct_closed_cases_meritorious", ascending=False).head()
```

Out [54]:

	grouped_basis	all_cases	closed_cases	pct_all_cases_closed	meritorious_cases	pct_closed_cases_meritorious
5	Genetics	2199	1832	0.83	112	
6	Other	27597	24134	0.87	974	
4	Equal Pay	8248	6866	0.83	272	
3	Disability/Medical	290739	252599	0.87	5908	
8	Religion	40293	35436	0.88	735	

```
In [55]: cases_by_grouped_basis[cases_by_grouped_basis["closed_cases"] >= 100].sort_values("pct_closed_cases_relief", ascending=False).head()
```

Out [55]:

	grouped_basis	all_cases	closed_cases	pct_all_cases_closed	meritorious_cases	pct_closed_cases_meritorious
6	Other	27597	24134	0.87	974	
3	Disability/Medical	290739	252599	0.87	5908	
10	Sex	309303	268660	0.87	5473	
4	Equal Pay	8248	6866	0.83	272	
9	Retaliation	412515	357958	0.87	6700	

```
In [56]: cases_by_grouped_basis[cases_by_grouped_basis["closed_cases"] >= 100].sort_values("
pct_closed_cases_monetary_benefits", ascending=False).head()
```

Out[56]:

	grouped_basis	all_cases	closed_cases	pct_all_cases_closed	meritorious_cases	pct_closed_cases_merit
4	Equal Pay	8248	6866	0.83	272	
3	Disability/Medical	290739	252599	0.87	5908	
10	Sex	309303	268660	0.87	5473	
9	Retaliation	412515	357958	0.87	6700	
0	Age	231544	205355	0.89	2989	

How many cases alleged some form of racial discrimination?

```
In [57]: race_discrimination = charges[charges["grouped_basis"] == "Race"]
race_discrimination["charge_num"].nunique()
```

Out[57]: 350112

```
In [58]: race_discrimination["charge_num"].nunique() / charges["charge_num"].nunique()
```

Out[58]: 0.331238682356681

What proportion of closed cases alleging some form of racial discrimination resulted in the complainant getting some form of relief?

```
In [59]: race_discrimination_relief_cases = closed_cases[(closed_cases["grouped_basis"] == "
Race") & (closed_cases["closure_action"].isin(relief_outcomes))]
race_discrimination_relief_cases["charge_num"].nunique()
```

Out[59]: 45802

```
In [60]: race_discrimination_relief_cases["charge_num"].nunique() / closed_cases[closed_case
s["grouped_basis"] == "Race"]["charge_num"].nunique()
```

Out[60]: 0.14813018069152428

What proportion of closed cases not alleging some form of racial discrimination resulted in the complainant getting some form of relief?

Create a list of charge numbers attached to cases in which some form of racial discrimination was alleged.

```
In [61]: race_discrimination_charge_nums = race_discrimination["charge_num"].drop_duplicates
().tolist()
```

Calculate the number of relief cases where racial discrimination was not alleged.


```
In [62]: non_race_discrimination_relief_cases = closed_cases[(~closed_cases["charge_num"].isin(
race_discrimination_charge_nums)) & (closed_cases["closure_action"].isin(relief_outcomes))]
non_race_discrimination_relief_cases["charge_num"].nunique()
```

Out[62]: 118574

Calculate the number of all closed cases where racial discrimination was not alleged.

```
In [63]: non_race_discrimination_closed_cases = closed_cases[~closed_cases["charge_num"].isin(
race_discrimination_charge_nums)]
non_race_discrimination_closed_cases["charge_num"].nunique()
```

Out[63]: 619565

```
In [64]: non_race_discrimination_relief_cases["charge_num"].nunique() / non_race_discriminat
ion_closed_cases["charge_num"].nunique()
```

Out[64]: 0.19138266364303988

How many cases alleged racial discrimination against African-Americans?

```
In [65]: aa_discrimination = charges[charges["basis"] == "Race-Black/African American"]
aa_discrimination["charge_num"].nunique()
```

Out[65]: 279225

```
In [66]: aa_discrimination["charge_num"].nunique() / charges["charge_num"].nunique()
```

Out[66]: 0.26417295345787706

What does this data look like when grouped by basis and closure action?

```
In [67]: cases_by_basis_and_closure_action = closed_cases.groupby(["basis", "closure_actio
n"])["charge_num"].nunique().reset_index()
cases_by_basis_and_closure_action.rename(columns={"charge_num": "cases"}, inplace=True)
cases_by_basis_and_closure_action.head()
```

Out[67]:

	basis	closure_action	cases
0	Age	ADEA Sect. 7(D) Closure	1062
1	Age	Administrative Closure	1897
2	Age	CP Failed To Cooperate	369
3	Age	CP Failed To Respond To 30-Day Letter	270
4	Age	CP Filed Suit	1833

```
In [68]: cases_by_basis_and_closure_action = pd.pivot_table(cases_by_basis_and_closure_action, index=["basis"], columns=["closure_action"])
cases_by_basis_and_closure_action.fillna(0, inplace=True)
cases_by_basis_and_closure_action.head()
```

Out [68]:

cases										
	closure_action	ADEA Sect. 7(D) Closure	Administrative Closure	CP Failed To Cooperate	CP Failed To Respond To 30-Day Letter	CP Filed Suit	CP Refused Full Relief	CP Withdrawal - No Ben.	Case Settled By Legal Unit	Closed Due To Court Decision
basis										
	Age	1,062.00	1,897.00	369.00	270.00	1,833.00	16.00	6,415.00	24.00	114.00
	Age (Non-Adea)	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Alcoholism	0.00	14.00	8.00	2.00	16.00	0.00	68.00	0.00	0.00
	Allergies	0.00	12.00	4.00	3.00	3.00	0.00	30.00	0.00	1.00
	Alzheimers	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

```
In [69]: pct_case_by_basis_and_closure_action = cases_by_basis_and_closure_action.apply(lambda x: x/x.sum(), axis=1)
pct_case_by_basis_and_closure_action.rename(lambda x: x + "_pct", axis=1, inplace=True)
```

```
In [70]: pct_case_by_basis_and_closure_action.columns = ["_".join(column).replace(" ", "_").lower().strip("_") for column in cases_by_basis_and_closure_action.columns.values]
pct_case_by_basis_and_closure_action.reset_index(inplace=True)
```

```
In [71]: cases_by_basis_and_closure_action.columns = ["_".join(column).replace(" ", "_").lower().strip("_") for column in cases_by_basis_and_closure_action.columns.values]
cases_by_basis_and_closure_action.reset_index(inplace=True)
```

```
In [72]: cases_by_basis_and_closure_action = cases_by_basis_and_closure_action.merge(pct_case_by_basis_and_closure_action, on="basis", suffixes=["", "_pct"])
cases_by_basis_and_closure_action.head()
```

Out [72]:

	basis	cases_adea sect. 7(d) closure	cases_administrative closure	cases_cp failed to cooperate	cases_cp failed to respond to 30-day letter	cases_cp filed suit	cases_cp refused full relief	cases_cp withdrawal - no ben.	cases_settled by legal unit
0	Age	1,062.00	1,897.00	369.00	270.00	1,833.00	16.00	6,415.00	
1	Age (Non-Adea)	0.00	1.00	0.00	0.00	0.00	0.00	0.00	
2	Alcoholism	0.00	14.00	8.00	2.00	16.00	0.00	68.00	
3	Allergies	0.00	12.00	4.00	3.00	3.00	0.00	30.00	
4	Alzheimers	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

Export the tables.

```
In [73]: cases_by_year.to_csv("data/cases_by_year.csv", index=False)
        cases_by_basis.to_csv("data/cases_by_basis.csv", index=False)
        cases_by_grouped_basis.to_csv("data/cases_by_grouped_basis.csv", index=False)
        cases_by_basis_and_closure_action.to_csv("data/cases_by_basis_and_closure_action.csv", index=False)
```