## Mock Interview Python Screening test

```
In [1]: import pandas as pd
        import matplotlib as mpl
        import matplotlib.pyplot as plt
        dataframe = pd.read_csv("adult_census_data.csv")
```

```
In [ ]:
```

**Q1. After importing the adult_census_data.csv file, please filter this to include only the following criteria:**

- State-Gov
- Bachelors
- Never-Married
- Adm-Clerical
- Not-in-familiy
- White
- Male
- United States
- <=50K

**Feel free to any method to complete this tasks. However, we recommend you use either list filtering [], or .loc to complete this task.**

**Put your code below**

```
In [2]: df= dataframe.filter([' State-gov', ' Bachelors', ' Never-married', ' Adm-clerical', ' Not-in-family', ' White', ' Male', ' United-States', ' <=50K'])
        print(df)
```

```
               State-gov   Bachelors        Never-married  \
0       Self-emp-not-inc   Bachelors   Married-civ-spouse
1                Private     HS-grad             Divorced
2                Private        11th   Married-civ-spouse
3                Private   Bachelors   Married-civ-spouse
4                Private     Masters   Married-civ-spouse
...                  ...         ...                  ...
32555            Private  Assoc-acdm   Married-civ-spouse
32556            Private     HS-grad   Married-civ-spouse
32557            Private     HS-grad              Widowed
32558            Private     HS-grad        Never-married
32559       Self-emp-inc     HS-grad   Married-civ-spouse

             Adm-clerical   Not-in-family   White     Male   United-States  \
0         Exec-managerial         Husband   White     Male   United-States
1       Handlers-cleaners   Not-in-family   White     Male   United-States
2       Handlers-cleaners         Husband   Black     Male   United-States
3          Prof-specialty            Wife   Black   Female            Cuba
4         Exec-managerial            Wife   White   Female   United-States
...                   ...             ...     ...      ...             ...
32555        Tech-support            Wife   White   Female   United-States
32556   Machine-op-inspct         Husband   White     Male   United-States
32557        Adm-clerical       Unmarried   White   Female   United-States
32558        Adm-clerical       Own-child   White     Male   United-States
32559     Exec-managerial            Wife   White   Female   United-States

         <=50K
0        <=50K
1        <=50K
2        <=50K
3        <=50K
4        <=50K
...        ...
32555    <=50K
32556     >50K
32557    <=50K
32558    <=50K
32559     >50K

[32560 rows x 9 columns]
```

**Currently, the dataframe you are using has the following column names:**

[' State-gov', ' Bachelors', ' Never-married', ' Adm-clerical', ' Not-in-family', ' White', ' Male', ' United-States', ' <=50K']

**Q2. Please re-name all the newly filtered columns in the pandas DataFrame to the following:**

Employment Type, Degree Status, Marriage-Status, Job-Role, Family-Role, Ethnicity, Gender, Country, Earnings

E.g. State-Gov becomes Employment Type, Bachelors becomes Degree Status, etc.

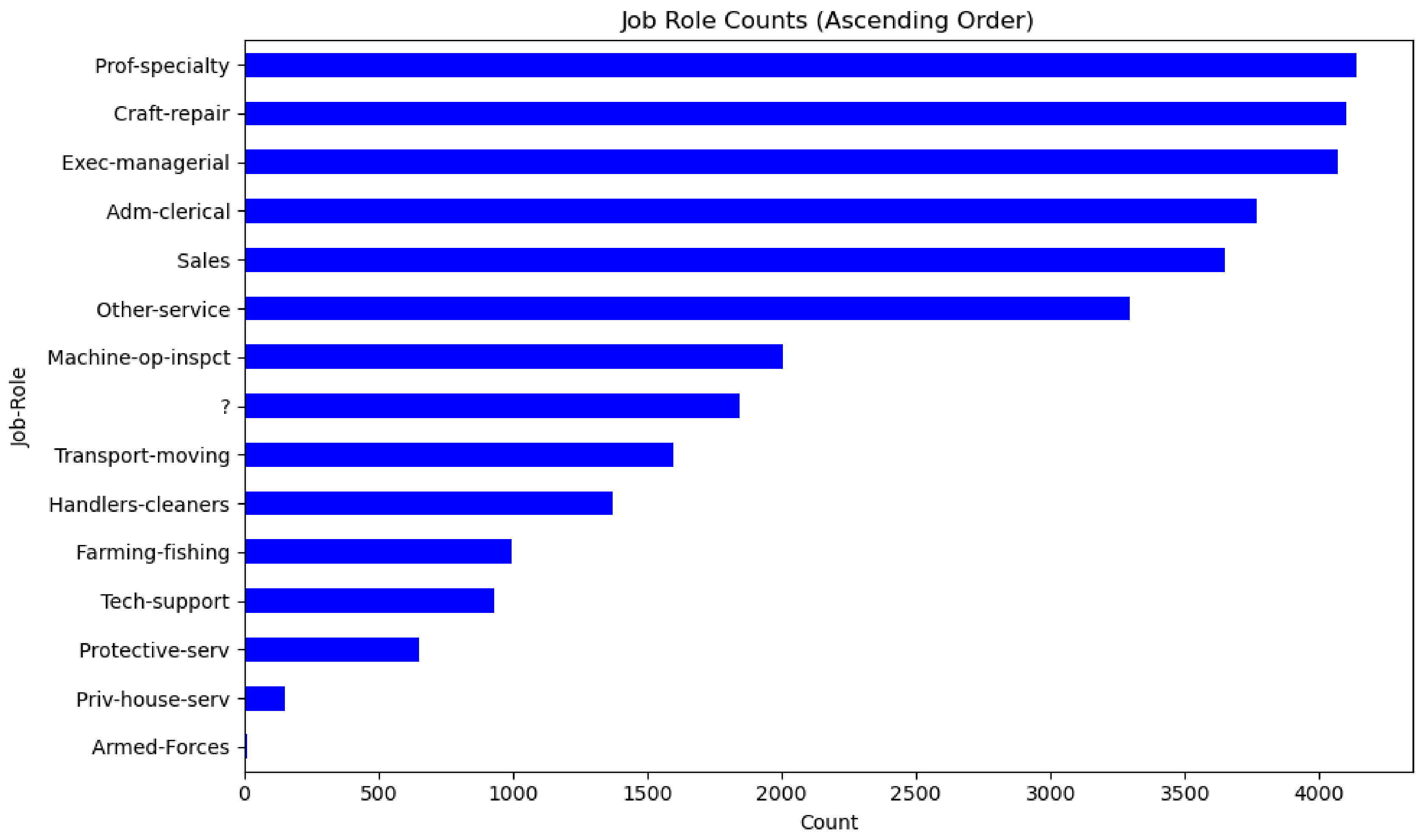
**Put your code below**

In [3]:
```python
df= df.rename(columns={
    ' State-gov': 'Employment Type',
    ' Bachelors':'Degree Status',
    ' Never-married': 'Marriage-Status',
    ' Adm-clerical':'Job-Role',
    ' Not-in-family': 'Family-Role',
    ' White': 'Ethnicity',
    ' Male': 'Gender',
    ' United-States': 'Country',
    ' <=50K': 'Earnings'})
```

**Q3. The Job Role Columns holds the job information for each individual in this census snapshot. Using this column, create a Bar Chart that shows the count of 'Unique' Jobs per Job Group in the "Job-Role" Column in ascending order, as per the provided image below**

**Put your code below**

In [4]:
```
job_counts = df['Job-Role'].value_counts().sort_values(ascending=True)
plt.figure(figsize=(10, 6))
job_counts.plot(kind='barh', color='blue')
plt.xlabel('Count')
plt.title('Job Role Counts (Ascending Order)')
plt.tight_layout()
plt.show()
```



**Q4. Please create two bar plots as per below that show:**

```
1) The number of individuals who have a High School Graduate Diploma AND earn <=50K in the United States
2) The number of individuals who have a High School Graduate Diploma AND earn >50K in the United States
```

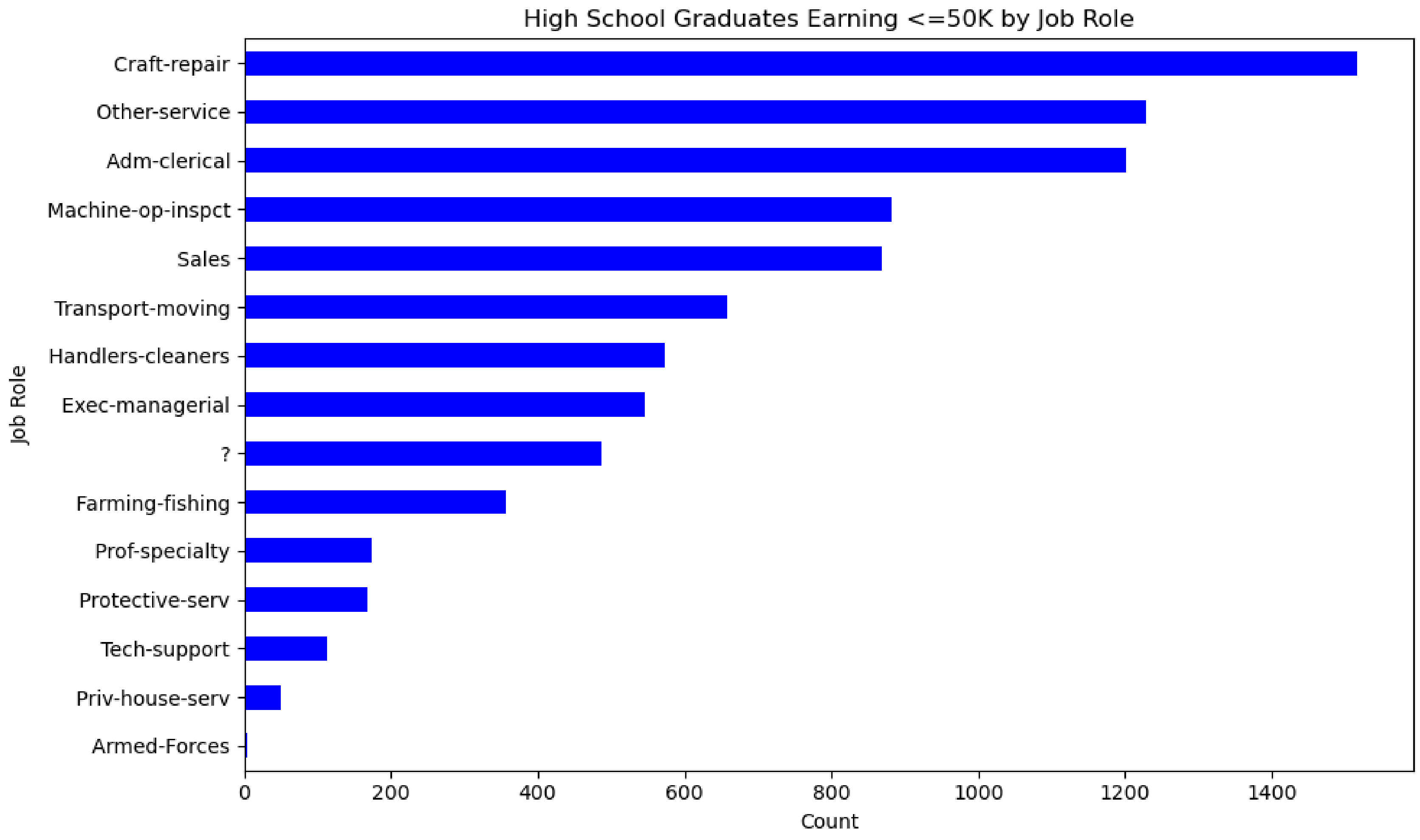
Please note you will be looking specifically at the *Job Role* column

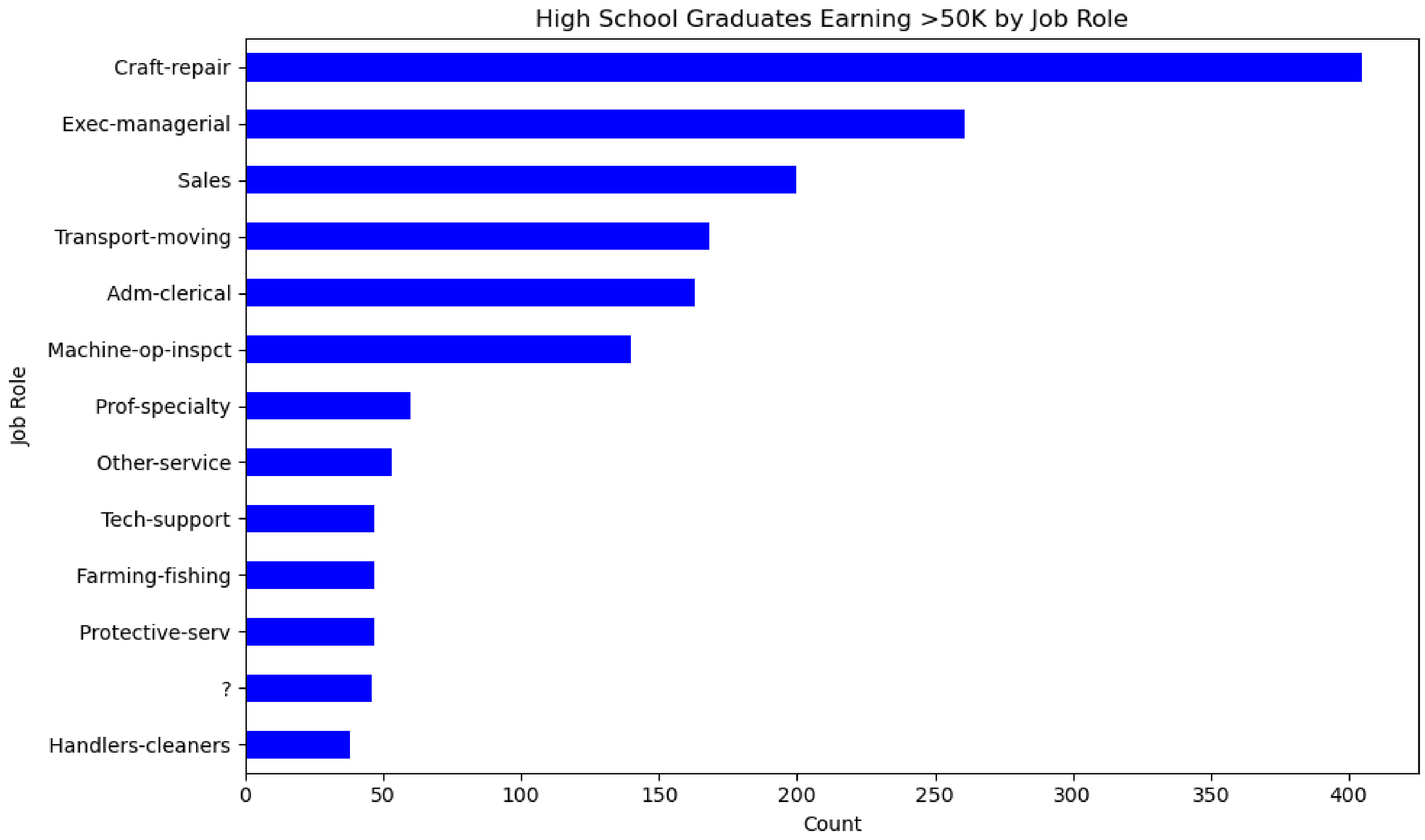
**Put Your Code Below**

```
In [5]: print(df['Degree Status'].head(10))
        print(df['Earnings'].unique())
```

```
0        Bachelors
1          HS-grad
2             11th
3        Bachelors
4          Masters
5              9th
6          HS-grad
7          Masters
8        Bachelors
9     Some-college
Name: Degree Status, dtype: object
[' <=50K' ' >50K']
```

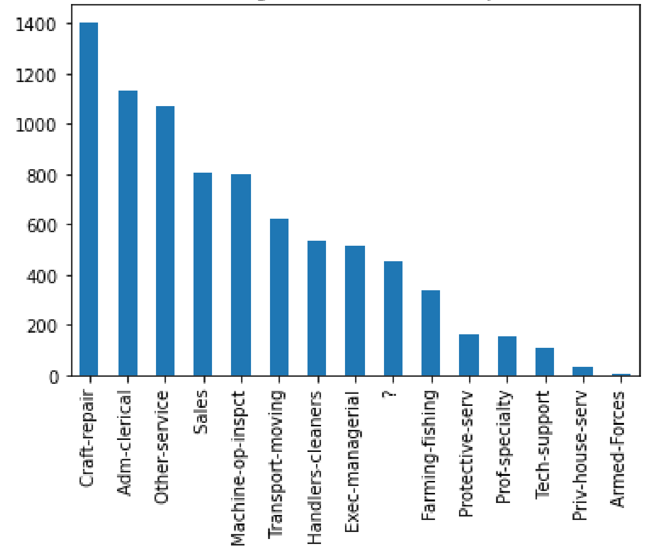
In [6]:
```python
df['Degree Status'] = df['Degree Status'].str.strip()
df['Earnings'] = df['Earnings'].str.strip()
hs_grads = df[df['Degree Status'] == 'HS-grad']

hs_le_50k = hs_grads[hs_grads['Earnings'] == '<=50K']
count_le_50k = hs_le_50k['Job-Role'].value_counts().sort_values(ascending=True)

hs_gt_50k = hs_grads[hs_grads['Earnings'] == '>50K']
count_gt_50k = hs_gt_50k['Job-Role'].value_counts().sort_values(ascending=True)

plt.figure(figsize=(10, 6))
count_le_50k.plot(kind='barh', color='blue')
plt.title('High School Graduates Earning <=50K by Job Role')
plt.xlabel('Count')
plt.ylabel('Job Role')
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 6))
count_gt_50k.plot(kind='barh', color='blue')
plt.title('High School Graduates Earning >50K by Job Role')
plt.xlabel('Count')
plt.ylabel('Job Role')
plt.tight_layout()
plt.show()
```


High School Graduates Earning <=50K by Job Role
Job Role
Craft-repair
Other-service
Adm-clerical
Machine-op-inspct
Sales
Transport-moving
Handlers-cleaners
Exec-managerial
?
Farming-fishing
Prof-specialty
Protective-serv
Tech-support
Priv-house-serv
Armed-Forces
0
200
400
600
800
1000
1200
1400
Count

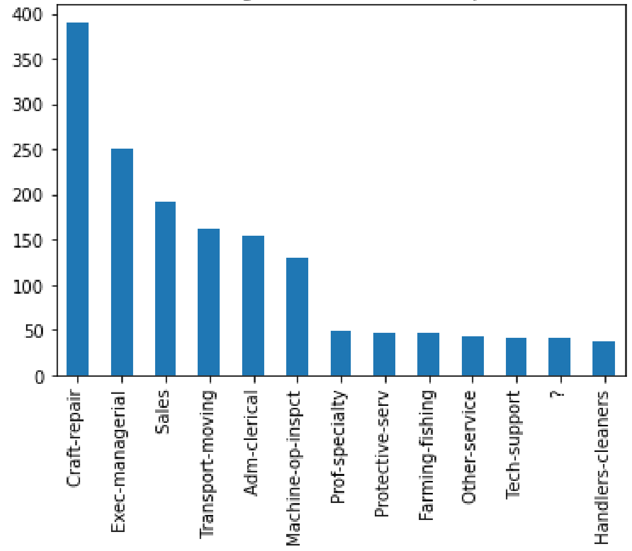

High School Graduates Earning >50K by Job Role
Job Role
Craft-repair
Exec-managerial
Sales
Transport-moving
Adm-clerical
Machine-op-inspct
Prof-specialty
Other-service
Tech-support
Farming-fishing
Protective-serv
?
Handlers-cleaners
0
50
100
150
200
250
300
350
400
Count

```
In [42]:
```

Individuals who earn < 50K, have a High School Graduate Diploma and are in the United States



Individuals who earn > 50K, have a High School Graduate Diploma and are in the United States



```
In [ ]:
```

## Challenge Question

**Q5. Which Job Role has the highest *proportion* of individuals who earn >50K?**

**Put your code below**

```
In [7]: df['Job-Role'] = df['Job-Role'].str.strip()
        df['Earnings'] = df['Earnings'].str.strip()
```

```
job_counts = df.groupby('Job-Role')['Earnings'].value_counts(normalize=False).unstack(fill_value=0)

if '<=50K' not in job_counts.columns:
    job_counts['<=50K'] = 0
if '>50K' not in job_counts.columns:
    job_counts['>50K'] = 0

job_counts['Total'] = job_counts['<=50K'] + job_counts['>50K']
job_counts['>50K Proportion'] = job_counts['>50K'] / job_counts['Total']

top_job_role = job_counts['>50K Proportion'].idxmax()
top_proportion = job_counts['>50K Proportion'].max()

print(f"The job role with the highest proportion of individuals earning >50K is: **{top_job_role}**")
print(f"Proportion: {top_proportion:.2%}")
```

```
The job role with the highest proportion of individuals earning >50K is: **Exec-managerial**
Proportion: 48.40%
```

In [7]:

```
!jupyter nbconvert --to html MockInterviewQuestions.ipynb
```

```
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_directory= --ClearOutputPreprocessor.ena
bled=True]
--coalesce-streams
    Coalesce consecutive stdout and stderr outputs into one stream (within each cell).
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_directory= --CoalesceStreamsPreprocesso
r.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExporter.exclude_output_prompt=True]
```

```
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporter.exclude_input=True --TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                    results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                    results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    Overwrite base name use for output files.
                Supports pattern replacements '{notebook_name}'.
    Default: '{notebook_name}'
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                  to output to the directory of each notebook. To recover
                                  previous default behaviour (outputting to the current
                                  working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides', 'we
bpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb
            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```

```
[NbConvertApp] WARNING | pattern 'MockInterviewQuestions.ipynb' matched no files
```