Writing Data Frames to Files

Pandas also provides simple APIs to write the data back to files.

- Let us write the revenue per order along with order_id to a file.

Here are the steps which you need to follow before writing Data Frame to a file.

- Make sure you have the Data Frame that is supposed to be written to file.
- You need to ensure that you have write permissions on the folder under which files are supposed to be written.
- Make sure to use appropriate key word arguments to write the Data Frame into file as per the requirements.

```
%run 06_csv_to_pandas_data_frame.ipynb
```

```
order_items.to_csv?
```

```
Signature:
order_items.to_csv(
    path_or_buf:Union[str, pathlib.Path, IO[~AnyStr], NoneType]=None,
    sep:str=',',
    na_rep:str='',
    float_format:Union[str, NoneType]=None,
    columns:Union[Sequence[collections.abc.Hashable], NoneType]=None,
    header:Union[bool, List[str]]=True,
    index:bool=True,
    index_label:Union[bool, str, Sequence[collections.abc.Hashable], NoneType]=None,
    mode:str='w',
    encoding:Union[str, NoneType]=None,
    compression:Union[str, Mapping[str, str], NoneType]='infer',
    quoting:Union[int, NoneType]=None,
    quotechar:str='"',
    line_terminator:Union[str, NoneType]=None,
    chunksize:Union[int, NoneType]=None,
    date_format:Union[str, NoneType]=None,
    doublequote:bool=True,
    escapechar:Union[str, NoneType]=None,
    decimal:Union[str, NoneType]='.',
    errors:str='strict',
) -> Union[str, NoneType]
Docstring:
Write object to a comma-separated values (csv) file.

.. versionchanged:: 0.24.0
    The order of arguments for Series was changed.

Parameters
----------
path_or_buf : str or file handle, default None
    File path or object, if None is provided the result is returned as
    a string.  If a file object is passed it should be opened with
    `newline=''`, disabling universal newlines.

    .. versionchanged:: 0.24.0

       Was previously named "path" for Series.

sep : str, default ','
    String of length 1. Field delimiter for the output file.
na_rep : str, default ''
    Missing data representation.
float_format : str, default None
    Format string for floating point numbers.
columns : sequence, optional
    Columns to write.
header : bool or list of str, default True
    Write out the column names. If a list of strings is given it is
    assumed to be aliases for the column names.

    .. versionchanged:: 0.24.0

       Previously defaulted to False for Series.

index : bool, default True
    Write row names (index).
index_label : str or sequence, or False, default None
    Column label for index column(s) if desired. If None is given, and
    `header` and `index` are True, then the index names are used. A
    sequence should be given if the object uses MultiIndex. If
    False do not print fields for index names. Use index_label=False
    for easier importing in R.
mode : str
    Python write mode, default 'w'.
encoding : str, optional
    A string representing the encoding to use in the output file,
    defaults to 'utf-8'.
compression : str or dict, default 'infer'
    If str, represents compression mode. If dict, value at 'method' is
    the compression mode. Compression mode may be any of the following
    possible values: {'infer', 'gzip', 'bz2', 'zip', 'xz', None}. If
    compression mode is 'infer' and `path_or_buf` is path-like, then
    detect compression mode from the following extensions: '.gz',
    '.bz2', '.zip' or '.xz'. (otherwise no compression). If dict given
    and mode is one of {'zip', 'gzip', 'bz2'}, or inferred as
    one of the above, other entries passed as
    additional compression options.

    .. versionchanged:: 1.0.0

       May now be a dict with key 'method' as compression mode
       and other entries as additional compression options if
       compression mode is 'zip'.

    .. versionchanged:: 1.1.0
```

```
        Passing compression options as keys in dict is
        supported for compression modes 'gzip' and 'bz2'
        as well as 'zip'.

quoting : optional constant from csv module
    Defaults to csv.QUOTE_MINIMAL. If you have set a `float_format`
    then floats are converted to strings and thus csv.QUOTE_NONNUMERIC
    will treat them as non-numeric.
quotechar : str, default '\"'
    String of length 1. Character used to quote fields.
line_terminator : str, optional
    The newline character or character sequence to use in the output
    file. Defaults to `os.linesep`, which depends on the OS in which
    this method is called ('\n' for linux, '\r\n' for Windows, i.e.).

    .. versionchanged:: 0.24.0
chunksize : int or None
    Rows to write at a time.
date_format : str, default None
    Format string for datetime objects.
doublequote : bool, default True
    Control quoting of `quotechar` inside a field.
escapechar : str, default None
    String of length 1. Character used to escape `sep` and `quotechar`
    when appropriate.
decimal : str, default '.'
    Character recognized as decimal separator. E.g. use ',' for
    European data.
errors : str, default 'strict'
    Specifies how encoding and decoding errors are to be handled.
    See the errors argument for :func:`open` for a full list
    of options.

    .. versionadded:: 1.1.0

Returns
-------
None or str
    If path_or_buf is None, returns the resulting csv format as a
    string. Otherwise returns None.

See Also
--------
read_csv : Load a CSV file into a DataFrame.
to_excel : Write DataFrame to an Excel file.

Examples
--------
>>> df = pd.DataFrame({'name': ['Raphael', 'Donatello'],
...                    'mask': ['red', 'purple'],
...                    'weapon': ['sai', 'bo staff']})
>>> df.to_csv(index=False)
'name,mask,weapon\nRaphael,red,sai\nDonatello,purple,bo staff\n'

Create 'out.zip' containing 'out.csv'

>>> compression_opts = dict(method='zip',
...                         archive_name='out.csv')  # doctest: +SKIP
>>> df.to_csv('out.zip', index=False,
...           compression=compression_opts)  # doctest: +SKIP
File:      /opt/anaconda3/envs/beakerx/lib/python3.6/site-packages/pandas/core/generic.py
Type:      method
```

```
import getpass
username = getpass.getuser()
```

```
username
```

```
'itversity'
```

```
base_dir = f"/home/{username}/data/retail_db"
base_dir
```

```
'/home/itversity/data/retail_db'
```

```
output_dir = f'{base_dir}/revenue_per_order'
output_dir
```

```
'/home/itversity/data/retail_db/revenue_per_order'
```

```
%%sh

rm -rf /home/`whoami`/data/retail_db/revenue_per_order
```

```
%%sh

ls -ltr /home/`whoami`/data/retail_db
```

```
total 0
```

```
import subprocess

subprocess.call(['rm', '-rf', output_dir])
```

```
0
```

```
import subprocess

subprocess.call(['mkdir', '-p', output_dir])
```

```
0
```

```
import subprocess
#ls -ltr /Users/itversity/Research/data/retail_db/revenue_per_order
subprocess.check_output(['ls', '-ltr', output_dir])
```

```
b'total 0\n'
```

```
%%sh
ls -ltr /data/retail_db
```

```
total 20156
drwxr-xr-x 2 root root     4096 Nov 22 16:08 categories
-rw-r--r-- 1 root root      806 Nov 22 16:08 README.md
drwxr-xr-x 2 root root     4096 Nov 22 16:08 customers
-rw-r--r-- 1 root root     1748 Nov 22 16:08 create_db_tables_pg.sql
-rw-r--r-- 1 root root 10303297 Nov 22 16:08 create_db.sql
drwxr-xr-x 2 root root     4096 Nov 22 16:08 departments
drwxr-xr-x 2 root root     4096 Nov 22 16:08 order_items
-rw-r--r-- 1 root root 10297372 Nov 22 16:08 load_db_tables_pg.sql
drwxr-xr-x 2 root root     4096 Nov 22 16:08 orders
drwxr-xr-x 2 root root     4096 Nov 22 16:08 products
```

```
orders
```

	order_id	order_date	order_customer_id	order_status
0	1	2013-07-25 00:00:00.0	11599	CLOSED
1	2	2013-07-25 00:00:00.0	256	PENDING_PAYMENT
2	3	2013-07-25 00:00:00.0	12111	COMPLETE
3	4	2013-07-25 00:00:00.0	8827	CLOSED
4	5	2013-07-25 00:00:00.0	11318	COMPLETE
...	...	...	...	...
68878	68879	2014-07-09 00:00:00.0	778	COMPLETE
68879	68880	2014-07-13 00:00:00.0	1117	COMPLETE
68880	68881	2014-07-19 00:00:00.0	2518	PENDING_PAYMENT
68881	68882	2014-07-22 00:00:00.0	10000	ON_HOLD
68882	68883	2014-07-23 00:00:00.0	5533	COMPLETE

68883 rows × 4 columns

```
order_items
```

	order_item_id	order_item_order_id	order_item_product_id	order_item_quantity	order
0	1	1	957	1	
1	2	2	1073	1	
2	3	2	502	5	
3	4	2	403	1	
4	5	4	897	2	
...	...	...	...	...	
172193	172194	68881	403	1	
172194	172195	68882	365	1	
172195	172196	68882	502	1	
172196	172197	68883	208	1	
172197	172198	68883	502	3	

172198 rows × 6 columns

```
order_items. \
    groupby('order_item_order_id')['order_item_subtotal']. \
    agg(['sum', 'min', 'max', 'count'])
```

	sum	min	max	count
order_item_order_id				
1	299.98	299.98	299.98	1
2	579.98	129.99	250.00	3
4	699.85	49.98	299.95	4
5	1129.86	99.96	299.98	5
7	579.92	79.95	299.98	3
...	...	...	...	...
68879	1259.97	129.99	999.99	3
68880	999.77	149.94	250.00	5
68881	129.99	129.99	129.99	1
68882	109.99	50.00	59.99	2
68883	2149.99	150.00	1999.99	2

57431 rows × 4 columns

```
order_items
```

	order_item_id	order_item_order_id	order_item_product_id	order_item_quantity	order
0	1	1	957	1	
1	2	2	1073	1	
2	3	2	502	5	
3	4	2	403	1	
4	5	4	897	2	
...	...	...	...	...	
172193	172194	68881	403	1	
172194	172195	68882	365	1	
172195	172196	68882	502	1	
172196	172197	68883	208	1	
172197	172198	68883	502	3	

172198 rows × 6 columns

```
order_items. \
    groupby('order_item_order_id')['order_item_subtotal']. \
    agg(['sum', 'min', 'max', 'count']). \
    reset_index()
```

	order_item_order_id	sum	min	max	count
0	1	299.98	299.98	299.98	1
1	2	579.98	129.99	250.00	3
2	4	699.85	49.98	299.95	4
3	5	1129.86	99.96	299.98	5
4	7	579.92	79.95	299.98	3
...	...	...	...	...	...
57426	68879	1259.97	129.99	999.99	3
57427	68880	999.77	149.94	250.00	5
57428	68881	129.99	129.99	129.99	1
57429	68882	109.99	50.00	59.99	2
57430	68883	2149.99	150.00	1999.99	2

57431 rows × 5 columns

```
order_items. \
    groupby('order_item_order_id')['order_item_subtotal']. \
    agg(['sum', 'min', 'max', 'count']). \
    rename(columns={'count': 'item_count', 'sum': 'revenue'}). \
    to_json(f'{output_dir}/revenue_per_order.json', orient='table')
```

```
%%sh
ls -ltr /home/`whoami`/data/retail_db/revenue_per_order
```

```
total 4884
-rw-rw-r-- 1 itversity itversity 4999377 Dec 14 10:54 revenue_per_order.json
```

```
%%sh
head -10 /home/`whoami`/data/retail_db/revenue_per_order/revenue_per_order.json
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

```
order_items. \
    groupby('order_item_order_id')['order_item_subtotal']. \
    agg(['sum', 'min', 'max', 'count']). \
    rename(columns={'count': 'item_count', 'sum': 'revenue'})
```

order_item_order_id	revenue	min	max	item_count
1	299.98	299.98	299.98	1
2	579.98	129.99	250.00	3
4	699.85	49.98	299.95	4
5	1129.86	99.96	299.98	5
7	579.92	79.95	299.98	3
...	...	...	...	...
68879	1259.97	129.99	999.99	3
68880	999.77	149.94	250.00	5
68881	129.99	129.99	129.99	1
68882	109.99	50.00	59.99	2
68883	2149.99	150.00	1999.99	2

57431 rows × 4 columns

```
order_items. \
    groupby('order_item_order_id')['order_item_subtotal']. \
    agg(['sum', 'min', 'max', 'count']). \
    rename(columns={'count': 'item_count', 'sum': 'revenue'}). \
    round(2). \
    to_csv(output_dir + '/revenue_per_order.csv')
```

```
%%sh
ls -ltr /home/`whoami`/data/retail_db/revenue_per_order
```

```
total 6460
-rw-rw-r-- 1 itversity itversity 4999377 Dec 14 10:54 revenue_per_order.json
-rw-rw-r-- 1 itversity itversity 1610716 Dec 14 10:55 revenue_per_order.csv
```

```
%%sh
head /home/`whoami`/data/retail_db/revenue_per_order/revenue_per_order.csv
```

```
order_item_order_id,revenue,min,max,item_count
1,299.98,299.98,299.98,1
2,579.98,129.99,250.0,3
4,699.85,49.98,299.95,4
5,1129.86,99.96,299.98,5
7,579.92,79.95,299.98,3
8,729.84,50.0,299.95,4
9,599.96,199.98,199.99,3
10,651.92,21.99,199.99,5
11,919.79,49.98,399.96,5
```