Concatenating Data:

print pd.concat(Left, Right)

Output

id	Name	Subject_id
1	Priya	sub1
2	Riya	sub2
3	Amit	sub3
4	Neha	sub4
1	Ram	sub5
2	Raj	sub2
3	Shivansh	sub3
4	Kajal	sub6
5	Komal	sub8

Data Wrangling by Removing Duplication:

Syntax: DataFrame.duplicated(subset=None, keep='first')

For example, A university will organize the event, In order to participate Students have to fill in their details in the online form so that they will contact them. It may be possible that a student will fill out the form multiple times. It may cause Easily wrangles by removing dupliate values.

```
student_data={'Name': ['Priya', 'Riya', 'Amit'],
              'Roll_no': [23, 24, 25],
              'Email': ['xxxx@gmail.com', 'xxx@gmail.com',
                        'xxxxx@gmail.com']}
st = pd.DataFrame(student_data)
print(st)
```

Output

	Name	Roll_no	Email
0	Priya	23	xxxxx@gmail.com
1	Riya	24	xx@gmail.com
2	Amit	25	xxxxxx@gmail.com.

Removing Duplicate data from the Dataset using Data wrangling.

```
student = {'Name': ['F', 'C', 'K', 'F'],
           'Roll-no': [5, 6, 7, 5],
           'Email': ['x@gmail.com', 'xx@gmail.com',
                     'xxx@gmail.com', 'x@gmail.com']}
s1 = pd.DataFrame(student)   [n -> NOT]
non_duplicate = s1[~s1.duplicated('Roll-no')]
print(non_duplicate)
```

Output

	Name	Roll-no	Email
0	F	5	x@gmail.com
1	C	6	xx@gmail.com
2	K	7	xxx@gmail.com
3	F	5	x@gmail.com

Creating Two DataFrame for Concatenation

Working with indexes:

```
movies_indexed = movies.set_index("title")
movies_indexed.reset_index()
```

Joining and splitting columns

```
movies["release_year"].astype(str)\
  .str.cat(movies[['release_month','release_day']].
#split a column on a delimiter into several columns with
movies['directors'].str.split(',', expand=True)
astype(str), sep='-')
.str.split(expand=True)
# Combine several columns into a list column with.
values.tolist()

movies['release_list'] = movies[['release_year',
                                 'release_month','release_day']]\
# Split a list column into separate columns with .to_list()
movies[['release_year2','release_month2','release_day2']]
= movie['release_list'].tolist()
```

Melting and pivoting

```
#Move side-by-side columns to consecutive rows with .melt()
pop popcorn.melt(id_vars='brand', var_name='trial',
                 value_name='n_unpopped')
#Melt using row index as id_variable with .melt(
 ignore_index=False)
popcorn_indexed=popcorn.set_index('brand')
popcorn_indexed.melt(var_name='trial', value_name=
                     'n_unpopped', ignore_index=False)
#Where there is a column multi-index, specify id_
 vars with a list of tuples pig~~_feed.melt(id_vars=~~
 [('No','No')])
pig_feed.melt(id_vars=[('No','No')])

#Same as .melt(). plus cleanup of var_name with
 wide_to_long()
pd.wide_to_long(popcorn,stubnames='trial', i='brand',
                j='trial_no', sep='_')
#Move values in from rows to columns with .pivot()
#Reset the index to completely reverse a melting
 operation popcorn_long\
 .pivot(values='n_unpopped', index='brand', columns=
                'trial')\
 .reset_index()

#Move values in from rows to columns and aggregate
 with .pivot_table()
#df.pivot_table(values, index, columns, aggfunc) is
 equivalent to
#df.groupby([index, columns])[values].agg(aggfunc).
 reset_index().pivot(index, columns) popcorn_long\
 .pivot_table(values='n_unpopped', index='brand',
              columns='trial')\
 .reset_index()
```

Exploding and normalizing

Expand list columns with .explode()
Vectors inside the list are given their own row
The number of columns remains unchanged

```
music.explode('singles')
```

For dictionary columns, move items to their own columns with json_normalize()

```
pd.json_normalize(music_exploded['singles'])
```

Converting to and from JSON:

```
import json
```

Convert series containing nested elements to JSON string with json.dumps()

```
json_singles = json.dumps(music['singles'].to_list())
```

Add column from JSON string with json.loads()

```
music['singles2'] = json.loads(json_singles)
```

Stacking and unstacking:

Move (multi-) indexes from a column index to a row index with .stack()
level argument starts with 0 for the outer index

```
pig_feed_stacked = pig_feed.stack(level=0)
```

Move (multi-) indexes from a row index to a column index with .stack()

```
pig_feed_stacked.unstack(level=1)
```