Sapio Analytics

GDP- Python Codes

Sapio DSS

Index

gdp1-

Works for- Warangal, and regions with formatting similar to Warangal

Arguments Required:

1. ‘df’ = variable; **type = DataFrame**; dataframe directly from Sapio Data Collection Master of the region.
2. ‘wardid’ = value; **type = int/pandas.core.frame.DataFrame**; should contain the last (maximum) wardid value present in the database, the function will start assigning from ‘wardid’ + 1
3. ‘cityid’ = value; **type = int/pandas.core.frame.DataFrame**; should contain the cityid of the wards
4. ‘rows’ = value; type = int; total number of rows containing values (necessary incase there are calculations below the data in google sheet.) (optional)

Argument Returned:

1. pandas.DataFrame

Dependencies:

1. In the google sheet of the region, Column name ‘**City**’ should be present at column ‘C’ or the column at index 2 (indexing starting from 0)
2. In the google sheet of region, Column name ‘**Ward No**.’ should be present at column ‘D’ or the column at index 3 (indexing starting from 0)
3. The GDP data should be present in the columns **BM:64, BN:65, BO:66, BP:67, BQ:68, BR:69, BS:70, BT:71, BU:72**

Code Explanation:

```
gdp_cols = [2,3,64,65,66,67,68,69,70,71,72]
df = df[df.columns[gdp_cols]]
```

Selects the City, Ward and GDP Data Columns

```
if 'rows' in kwargs:
    df = df.head(kwargs['rows'])
```

This code will only be executed if the optional argument ‘rows’ is passed

Selects first ‘N’ rows (optional)

```
col_map = {'Agriculture and Food':1,'Mining':2,'Manufacturing':3,
               'Electricity Gas Water':4,'Construction':5,'Trade Hotels
Restaurants':6,
               'Transport Storage Communication':7,'Financing Real Estate Business
Services':8,
               'Community Social Public Admin':9}

df = df.rename(columns=col_map)
df = df[['City','Ward No.',1,2,3,4,5,6,7,8,9]]
df = df[df.columns[2:11]]
```

Maps the GDP Data and changes to column names to their mappings (required for further numpy transformations)

After mapping, in ‘df’ only GDP data is subsetted.

```
ar_gdp = np.repeat(df.to_numpy().flatten(), 4)
```

to_numpy( )- converts dataframe to 2D array

flatten( )- converts the 2D array into 1D

np.repeat( )- repeats each number 4 times

```
ar_sec = np.resize(np.repeat(np.array(list(range(1,10))), 4),len(ar_gdp))
```

list(range( ))- creates a list from 1 to 9

np.array( )- converts the list to an array

np.repeat( )- repeats each number in array 4 times

np.resize( )- resizes the array to length of ar_gdp

```
ar_cnt_est = np.resize(100, len(ar_gdp))
```

np.resize( )- makes a new array (value=100) and resizes it to to the length of ar_gdp

```
ar_cityid = np.resize(cityid, len(ar_gdp))
```

np.resize( )- makes a new array (value=cityid) and resizes it to to the length of ar_gdp

```
ar_wardid =
np.repeat(range(wardid.iat[0,0]+1,wardid.iat[0,0]+int((len(ar_gdp))/36)+1),36)
```

range( )- creates a range starting from ‘wardid’+1 till the end (‘wardid’ + 1 + length of ar_gdp / 36); here 36 because each row in google sheet gets repeated 36 times.

np.repeat- repeats each number in array 36 times

```
quarter = pd.DataFrame({'quarter':['qtr1','qtr2','qtr3','qtr4']})
quarter = pd.concat([quarter]*int(len(ar_gdp)/4), ignore_index=True)
```

pd.DataFrame( )- creates a new DataFrame with column name = ‘quarter’ and contents ‘qtr1’, ‘qtr2’, ‘qtr3’, ‘qtr4’

pd.concat( )- contacts the previously created DataFrame ar_gdp / 4 times

```
gdp_form = pd.DataFrame({'cityid':ar_cityid,'wardid':ar_wardid,'sectorcode':ar_sec,
                                    'gdp':ar_gdp,'cnt_esta':ar_cnt_est})
gdp_form['quarter'] = quarter
gdp_form = gdp_form[['cityid','wardid','sectorcode','quarter','gdp','cnt_esta']]
```

here we combine the previously created numpy arrays and DataFrame into a DataFrame

gdp2-

Works for- Karimnagar MC, and regions with formatting similar to Karimnagar MC

Arguments Required:

1. ‘df’ = variable; **type = DataFrame**; dataframe directly from Sapio Data Collection Master of the region.
2. ‘wardid’ = value; **type = int/pandas.core.frame.DataFrame**; should contain the last (maximum) wardid value present in the database, the function will start assigning from ‘wardid’ + 1
3. ‘cityid’ = value; **type = int/pandas.core.frame.DataFrame**; should contain the cityid of the wards
4. ‘rows’ = value; type = int; total number of rows containing values (necessary incase there are calculations below the data in google sheet.) (optional)

Argument Returned:

1. pandas.DataFrame

Dependencies:

1. In the google sheet of the region, Column name ‘**City**’ should be present at column ‘C’ or the column at index 2 (indexing starting from 0)
2. In the google sheet of region, Column name ‘**Ward**’ should be present at column ‘D’ or the column at index 3 (indexing starting from 0)
3. The GDP data should be present in the columns **BM:64, BN:65, BO:66, BP:67, BQ:68, BR:69, BS:70, BT:71, BU:72**

Code Explanation:

```
gdp_cols = [2,3,64,65,66,67,68,69,70,71,72]
df = df[df.columns[gdp_cols]]
```

Selects the City, Ward and GDP Data Columns

```
if 'rows' in kwargs:
    df = df.head(kwargs['rows'])
```

This code will only be executed if the optional argument ‘rows’ is passed

Selects first ‘N’ rows (optional)

```
col_map = {'Agriculture and Food':1,'Mining':2,'Manufacturing':3,
               'Electricity Gas Water':4,'Construction':5,'Trade Hotels
Restaurants':6,
               'Transport Storage Communication':7,'Financing Real Estate Business
Services':8,
               'Community Social Public Admin':9}

df = df.rename(columns=col_map)
df = df[['City','Ward',1,2,3,4,5,6,7,8,9]]
df = df[df.columns[2:11]]
```

Maps the GDP Data and changes to column names to their mappings (required for further numpy transformations)

After mapping, in ‘df’ only GDP data is subsetted.

```
ar_gdp = np.repeat(df.to_numpy().flatten(), 4)
```

to_numpy( )- converts dataframe to 2D array

flatten( )- converts the 2D array into 1D

np.repeat( )- repeats each number 4 times

```
ar_sec = np.resize(np.repeat(np.array(list(range(1,10))), 4),len(ar_gdp))
```

list(range( ))- creates a list from 1 to 9

np.array( )- converts the list to an array

np.repeat( )- repeats each number in array 4 times

np.resize( )- resizes the array to length of ar_gdp

```
ar_cnt_est = np.resize(100, len(ar_gdp))
```

np.resize( )- makes a new array (value=100) and resizes it to to the length of ar_gdp

```
ar_cityid = np.resize(cityid, len(ar_gdp))
```

np.resize( )- makes a new array (value=cityid) and resizes it to to the length of ar_gdp

```
ar_wardid =
np.repeat(range(wardid.iat[0,0]+1,wardid.iat[0,0]+int((len(ar_gdp))/36)+1),36)
```

range( )- creates a range starting from ‘wardid’+1 till the end (‘wardid’ + 1 + length of ar_gdp / 36); here 36 because each row in google sheet gets repeated 36 times.

np.repeat- repeats each number in array 36 times

```
quarter = pd.DataFrame({'quarter':['qtr1','qtr2','qtr3','qtr4']})
quarter = pd.concat([quarter]*int(len(ar_gdp)/4), ignore_index=True)
```

pd.DataFrame( )- creates a new DataFrame with column name = ‘quarter’ and contents ‘qtr1’, ‘qtr2’, ‘qtr3’, ‘qtr4’

pd.concat( )- contacts the previously created DataFrame ar_gdp / 4 times

```
gdp_form = pd.DataFrame({'cityid':ar_cityid,'wardid':ar_wardid,'sectorcode':ar_sec,
                                    'gdp':ar_gdp,'cnt_esta':ar_cnt_est})
gdp_form['quarter'] = quarter
gdp_form = gdp_form[['cityid','wardid','sectorcode','quarter','gdp','cnt_esta']]
```

here we combine the previously created numpy arrays and DataFrame into a DataFrame

gdp3-

Works for- Rajasthan, and regions with formatting similar to Rajasthan

Arguments Required:

1. ‘df’ = variable; **type = DataFrame**; dataframe directly from Sapio Data Collection Master of the region.
2. ‘max_cityid’ = value; **type = int/pandas.core.frame.DataFrame**; should contain the last (maximum) cityid value present in the database, the function will start assigning from ‘max_cityid’ + 1
3. ‘rows’ = value; type = int; total number of rows containing values (necessary incase there are calculations below the data in google sheet.) (optional)

Argument Returned:

1. pandas.DataFrame

Dependencies:

1. In the google sheet of the region, Column name ‘**City**’ should be present at column ‘C’ or the column at index 2 (indexing starting from 0)
2. Input ‘df’ should contain a column ‘**cityid**’ having cityid corresponding to column ‘**City**’
3. Input ‘df’ should contain a column ‘**stateid**’ having stateid corresponding to the region
4. The GDP data should be present in the columns **BM:64, BN:65, BO:66, BP:67, BQ:68, BR:69, BS:70, BT:71, BU:72**
5. The names and spellings of cities in google sheet and database should match.

Function Behaviour:

1. If such cities are present whose cityid is null, a new city id (‘max_cityid’ will be assigned to it)
2. If there are **inconsistencies** in the city names and spellings, the function will consider it to be a new city and a **new city id** will be **assigned** to it.

Code Explanation:

```
if 'rows' in kwargs:
    df = df.head(kwargs['rows'])
```

This code will only be executed if the optional argument ‘rows’ is passed

Selects first ‘N’ rows (optional)

```
stateid = df['stateid']
cityid = pd.DataFrame(df['cityid'])
```

Extracting ‘cityid’ and ‘stateid’ from ‘df’

```
if 'max_cityid' in kwargs:

    cityid_max = kwargs['max_cityid']
    stateid = stateid.fillna(value=stateid.dropna()[0])
    cityid.loc[cityid.cityid.isnull(), 'cityid'] =
np.repeat(range(cityid_max.iat[0,0]+1,cityid_max.iat[0,0]+int(cityid.isnull().sum())
+1),1).tolist()
    cityid = cityid.iloc[:,0]

else:
    print('Argument required:  \'max_cityid\' type pandas.DataFrame')
    return 0
```

stateid.dropna( )- gets a list in which there are no NA values

stateid.dropna( )[0]- selects the first value from the list (we do this to get the ‘stateid’

stateid.fillna( )- replaces any NA values in ‘stateid’ with the above value

cityid.isnull( ).sum( )- counts the number of NA values in cityid

range( )- creates a range of cityids starting from ‘max_cityid’ + 1 consisting of [*cityid.isnull( ).sum( )*] cityids

np.repeat(,1)- converts the range into an array

.to_list( )- converts the array to a list

cityid.loc[cityid.cityid.isnull( ), ‘cityid’]- those values in ‘cityid’ where cityid is not present

```
gdp_cols = [2,3,64,65,66,67,68,69,70,71,72]
df = df[df.columns[gdp_cols]]
```

Selects the City and GDP Data Columns

```
col_map = {'Agriculture and Food':1,'Mining':2,'Manufacturing':3,
               'Electricity Gas Water':4,'Construction':5,'Trade Hotels
Restaurants':6,
               'Transport Storage Communication':7,'Financing Real Estate Business
Services':8,
               'Community Social Public Admin':9}

df = df.rename(columns=col_map)
df = df[[1,2,3,4,5,6,7,8,9]]
```

Maps the GDP Data and changes to column names to their mappings (required for further numpy transformations)

After mapping, in df only GDP data is subsetted.

```
ar_gdp = np.repeat(df.to_numpy().flatten(), 4)
```

to_numpy( )- converts dataframe to 2D array

flatten( )- converts the 2D array into 1D

np.repeat( )- repeats each number 4 times

```
ar_sec = np.resize(np.repeat(np.array(list(range(1,10))), 4),len(ar_gdp))
```

list(range( ))- creates a list from 1 to 9

np.array( )- converts the list to an array

np.repeat( )- repeats each number in array 4 times

np.resize( )- resizes the array to length of ar_gdp

```
ar_cnt_est = np.resize(100, len(ar_gdp))
```

np.resize( )- makes a new array (value=100) and resizes it to to the length of ar_gdp

```
ar_stateid = np.resize(stateid, len(ar_gdp))
```

np.resize( )- makes a new array (value=stateid) and resizes it to to the length of ar_gdp

```
ar_cityid = np.repeat(cityid,36)
```

np.repeat- repeats each number in array 36 times

```
quarter = pd.DataFrame({'quarter':['qtr1','qtr2','qtr3','qtr4']})
quarter = pd.concat([quarter]*int(len(ar_gdp)/4), ignore_index=True)

ar_quarter = quarter.to_numpy().flatten()
```

pd.DataFrame( )- creates a new DataFrame with column name = ‘quarter’ and contents ‘qtr1’, ‘qtr2’, ‘qtr3’, ‘qtr4’

pd.concat( )- contacts the previously created DataFrame ar_gdp / 4 times

.to_numpy( )- converts the DataFrame into a numpy

```
gdp_form =
pd.DataFrame({'stateid':ar_stateid,'cityid':ar_cityid,'sectorcode':ar_sec,

'quarter':ar_quarter,'gdp':ar_gdp,'cnt_esta':ar_cnt_est})
```

here we combine the previously created numpy arrays and DataFrame into a DataFrame

gdp4-

Works for- Lucknow, and regions with formatting similar to Lucknow

Arguments Required:

5. ‘df’ = variable; **type = DataFrame**; dataframe directly from Sapio Data Collection Master of the region.
6. ‘wardid’ = value; **type = int/pandas.core.frame.DataFrame**; should contain the last (maximum) wardid value present in the database, the function will start assigning from ‘wardid’ + 1
7. ‘cityid’ = value; **type = int/pandas.core.frame.DataFrame**; should contain the cityid of the wards
8. ‘rows’ = value; type = int; total number of rows containing values (necessary incase there are calculations below the data in google sheet.) (optional)

Argument Returned:

2. pandas.DataFrame

Dependencies:

4. In the google sheet of region, Column name ‘**City**’ should be present at column ‘C’ or the column at index 2 (indexing starting from 0)
5. In the google sheet of the region, Column name ‘**Ward Number**.’ should be present at column ‘D’ or the column at index 3 (indexing starting from 0)
6. The GDP data should be present in the columns **BM:63, BN:64, BO:65, BP:66, BQ:67, BR:68, BS:69, BT:70, BU:71**

Code Explanation:

```
gdp_cols = [2,63,64,65,66,67,68,69,70,71]
df = df[df.columns[gdp_cols]]
```

Selects the City, Ward and GDP Data Columns

```
if 'rows' in kwargs:
    df = df.head(kwargs['rows'])
```

This code will only be executed if the optional argument ‘rows’ is passed

Selects first ‘N’ rows (optional)

```
col_map = {'Agriculture and Food':1,'Mining':2,'Manufacturing':3,
               'Electricity Gas Water':4,'Construction':5,'Trade Hotels
Restaurants':6,
               'Transport Storage Communication':7,'Financing Real Estate Business
Services':8,
               'Community Social Public Admin':9}

df = df.rename(columns=col_map)
df = df[['Ward Number',1,2,3,4,5,6,7,8,9]]
df = df[df.columns[1:10]]
```

Maps the GDP Data and changes to column names to their mappings (required for further numpy transformations)

After mapping, in ‘df’ only GDP data is subsetted.

```
ar_gdp = np.repeat(df.to_numpy().flatten(), 4)
```

to_numpy( )- converts dataframe to 2D array

flatten( )- converts the 2D array into 1D

np.repeat( )- repeats each number 4 times

```
ar_sec = np.resize(np.repeat(np.array(list(range(1,10))), 4),len(ar_gdp))
```

list(range( ))- creates a list from 1 to 9

np.array( )- converts the list to an array

np.repeat( )- repeats each number in array 4 times

np.resize( )- resizes the array to length of ar_gdp

```
ar_cnt_est = np.resize(100, len(ar_gdp))
```

np.resize( )- makes a new array (value=100) and resizes it to to the length of ar_gdp

```
ar_cityid = np.resize(cityid, len(ar_gdp))
```

np.resize( )- makes a new array (value=cityid) and resizes it to to the length of ar_gdp

```
ar_wardid =
np.repeat(range(wardid.iat[0,0]+1,wardid.iat[0,0]+int((len(ar_gdp))/36)+1),36)
```

range( )- creates a range starting from ‘wardid’+1 till the end (‘wardid’ + 1 + length of ar_gdp / 36); here 36 because each row in google sheet gets repeated 36 times.

np.repeat- repeats each number in array 36 times

```
quarter = pd.DataFrame({'quarter':['qtr1','qtr2','qtr3','qtr4']})
quarter = pd.concat([quarter]*int(len(ar_gdp)/4), ignore_index=True)
```

pd.DataFrame( )- creates a new DataFrame with column name = ‘quarter’ and contents ‘qtr1’, ‘qtr2’, ‘qtr3’, ‘qtr4’

pd.concat( )- contacts the previously created DataFrame ar_gdp / 4 times

```
gdp_form = pd.DataFrame({'cityid':ar_cityid,'wardid':ar_wardid,'sectorcode':ar_sec,
                                    'gdp':ar_gdp,'cnt_esta':ar_cnt_est})
gdp_form['quarter'] = quarter
gdp_form = gdp_form[['cityid','wardid','sectorcode','quarter','gdp','cnt_esta']]
```

here we combine the previously created numpy arrays and DataFrame into a DataFrame

gdp5-

Works for- Uttar Pradesh, and regions with formatting similar to Uttar Pradesh

Arguments Required:

4. ‘df’ = variable; **type = DataFrame**; dataframe directly from Sapio Data Collection Master of the region.
5. ‘max_cityid’ = value; **type = int/pandas.core.frame.DataFrame**; should contain the last (maximum) cityid value present in the database, the function will start assigning from ‘max_cityid’ + 1
6. ‘rows’ = value; type = int; total number of rows containing values (necessary incase there are calculations below the data in google sheet.) (optional)

Argument Returned:

2. pandas.DataFrame

Dependencies:

6. In the google sheet of the region, Column name ‘**District**’ should be present at column ‘D’ or the column at index 3 (indexing starting from 0)
7. Input ‘df’ should contain a column ‘**cityid**’ having cityid corresponding to column ‘**City**’
8. Input ‘df’ should contain a column ‘**stateid**’ having stateid corresponding to the region
9. The GDP data should be present in the columns **BN:65, BO:66, BP:67, BQ:68, BR:69, BS:70, BT:71, BU:72, BV:73**
10. The names and spellings of cities in google sheet and database should match.

Function Behaviour:

3. If such cities are present whose cityid is null, a new city id (‘max_cityid’ will be assigned to it)
4. If there are **inconsistencies** in the city names and spellings, the function will consider it to be a new city and a **new city id** will be **assigned** to it.

Code Explanation:

```
if 'rows' in kwargs:
    df = df.head(kwargs['rows'])
```

This code will only be executed if the optional argument ‘rows’ is passed

Selects first ‘N’ rows (optional)

```
stateid = df['stateid']
cityid = pd.DataFrame(df['cityid'])
```

Extracting ‘cityid’ and ‘stateid’ from ‘df’

```
if 'max_cityid' in kwargs:

    cityid_max = kwargs['max_cityid']
    stateid = stateid.fillna(value=stateid.dropna()[0])
    cityid.loc[cityid.cityid.isnull(), 'cityid'] =
np.repeat(range(cityid_max.iat[0,0]+1,cityid_max.iat[0,0]+int(cityid.isnull().sum())
+1),1).tolist()
    cityid = cityid.iloc[:,0]

else:
    print('Argument required:  \'max_cityid\' type pandas.DataFrame')
    return 0
```

stateid.dropna( )- gets a list in which there are no NA values

stateid.dropna( )[0]- selects the first value from the list (we do this to get the ‘stateid’

stateid.fillna( )- replaces any NA values in ‘stateid’ with the above value

cityid.isnull( ).sum( )- counts the number of NA values in cityid

range( )- creates a range of cityids starting from ‘max_cityid’ + 1 consisting of [*cityid.isnull( ).sum( )*] cityids

np.repeat(,1)- converts the range into an array

.to_list( )- converts the array to a list

cityid.loc[cityid.cityid.isnull( ), ‘cityid’]- those values in ‘cityid’ where cityid is not present

```
gdp_cols = [3,65,66,67,68,69,70,71,72,73]
df = df[df.columns[gdp_cols]]
```

Selects the City and GDP Data Columns

```
col_map = {'Agriculture and Food':1,'Mining':2,'Manufacturing':3,
               'Electricity Gas Water':4,'Construction':5,'Trade Hotels
Restaurants':6,
               'Transport Storage Communication':7,'Financing Real Estate Business
Services':8,
               'Community Social Public Admin':9}

df = df.rename(columns=col_map)
df = df[[1,2,3,4,5,6,7,8,9]]
```

Maps the GDP Data and changes to column names to their mappings (required for further numpy transformations)

After mapping, in df only GDP data is subsetted.

```
ar_gdp = np.repeat(df.to_numpy().flatten(), 4)
```

to_numpy( )- converts dataframe to 2D array

flatten( )- converts the 2D array into 1D

np.repeat( )- repeats each number 4 times

```
ar_sec = np.resize(np.repeat(np.array(list(range(1,10))), 4),len(ar_gdp))
```

list(range( ))- creates a list from 1 to 9

np.array( )- converts the list to an array

np.repeat( )- repeats each number in array 4 times

np.resize( )- resizes the array to length of ar_gdp

```
ar_cnt_est = np.resize(100, len(ar_gdp))
```

np.resize( )- makes a new array (value=100) and resizes it to to the length of ar_gdp

```
ar_stateid = np.resize(stateid, len(ar_gdp))
```

np.resize( )- makes a new array (value=stateid) and resizes it to to the length of ar_gdp

```
ar_cityid = np.repeat(np.array(cityid),36)
```

np.repeat- repeats each number in array 36 times

```
quarter = pd.DataFrame({'quarter':['qtr1','qtr2','qtr3','qtr4']})
quarter = pd.concat([quarter]*int(len(ar_gdp)/4), ignore_index=True)

ar_quarter = quarter.to_numpy().flatten()
```

pd.DataFrame( )- creates a new DataFrame with column name = ‘quarter’ and contents ‘qtr1’, ‘qtr2’, ‘qtr3’, ‘qtr4’

pd.concat( )- contacts the previously created DataFrame ar_gdp / 4 times

.to_numpy( )- converts the DataFrame into a numpy

```
gdp_form =
pd.DataFrame({'stateid':ar_stateid,'cityid':ar_cityid,'sectorcode':ar_sec,
                          'quarter':ar_quarter,'gdp':ar_gdp,'cnt_esta':ar_cnt_est})
```

here we combine the previously created numpy arrays and DataFrame into a DataFrame

gdp6-

Works for- Telangana Wards, and regions with formatting similar to Telangana Wards (Adilabad, Bhadradri, etc.)

Arguments Required:

9. ‘df’ = variable; **type = DataFrame**; dataframe directly from Sapio Data Collection Master of the region.
10. ‘wardid’ = value; **type = int/pandas.core.frame.DataFrame**; should contain the last (maximum) wardid value present in the database, the function will start assigning from ‘wardid’ + 1
11. ‘cityid’ = value; **type = int/pandas.core.frame.DataFrame**; should contain the cityid of the wards
12. ‘rows’ = value; type = int; total number of rows containing values (necessary incase there are calculations below the data in google sheet.) (optional)

Argument Returned:

3. pandas.DataFrame

Dependencies:

7. In the google sheet of region, Column name ‘**City**’ should be present at column ‘C’ or the column at index 2 (indexing starting from 0)
8. In the google sheet of the region, Column name ‘**Mandal**.’ should be present at column ‘D’ or the column at index 3 (indexing starting from 0)
9. The GDP data should be present in the columns **BM:63, BN:64, BO:65, BP:66, BQ:67, BR:68, BS:69, BT:70, BU:71**

Code Explanation:

```
gdp_cols = [2,3,64,65,66,67,68,69,70,71,72]
df = df[df.columns[gdp_cols]]
```

Selects the City, Mandal and GDP Data Columns

```
if 'rows' in kwargs:
    df = df.head(kwargs['rows'])
```

This code will only be executed if the optional argument ‘rows’ is passed

Selects first ‘N’ rows (optional)

```
col_map = {'Agriculture and Food':1,'Mining':2,'Manufacturing':3,
               'Electricity Gas Water':4,'Construction':5,'Trade Hotels
Restaurants':6,
               'Transport Storage Communication':7,'Financing Real Estate Business
Services':8,
               'Community Social Public Admin':9}

df = df.rename(columns=col_map)
df = df[['City','Mandal',1,2,3,4,5,6,7,8,9]]
df = df[df.columns[2:11]]
```

Maps the GDP Data and changes to column names to their mappings (required for further numpy transformations)

After mapping, in ‘df’ only GDP data is subsetted.

```
ar_gdp = np.repeat(df.to_numpy().flatten(), 4)
```

to_numpy( )- converts dataframe to 2D array

flatten( )- converts the 2D array into 1D

np.repeat( )- repeats each number 4 times

```
ar_sec = np.resize(np.repeat(np.array(list(range(1,10))), 4),len(ar_gdp))
```

list(range( ))- creates a list from 1 to 9

np.array( )- converts the list to an array

np.repeat( )- repeats each number in array 4 times

np.resize( )- resizes the array to length of ar_gdp

```
ar_cnt_est = np.resize(100, len(ar_gdp))
```

np.resize( )- makes a new array (value=100) and resizes it to to the length of ar_gdp

```
ar_cityid = np.resize(cityid, len(ar_gdp))
```

np.resize( )- makes a new array (value=cityid) and resizes it to to the length of ar_gdp

```
ar_wardid =
np.repeat(range(wardid.iat[0,0]+1,wardid.iat[0,0]+int((len(ar_gdp))/36)+1),36)
```

range( )- creates a range starting from ‘wardid’+1 till the end (‘wardid’ + 1 + length of ar_gdp / 36); here 36 because each row in google sheet gets repeated 36 times.

np.repeat- repeats each number in array 36 times

```
quarter = pd.DataFrame({'quarter':['qtr1','qtr2','qtr3','qtr4']})
quarter = pd.concat([quarter]*int(len(ar_gdp)/4), ignore_index=True)
```

pd.DataFrame( )- creates a new DataFrame with column name = ‘quarter’ and contents ‘qtr1’, ‘qtr2’, ‘qtr3’, ‘qtr4’

pd.concat( )- contacts the previously created DataFrame ar_gdp / 4 times

```
gdp_form = pd.DataFrame({'cityid':ar_cityid,'wardid':ar_wardid,'sectorcode':ar_sec,
                                    'gdp':ar_gdp,'cnt_esta':ar_cnt_est})
gdp_form['quarter'] = quarter
gdp_form = gdp_form[['cityid','wardid','sectorcode','quarter','gdp','cnt_esta']]
```

here we combine the previously created numpy arrays and DataFrame into a DataFrame

How to use Functions-

Type 1 (gdp1, gdp2, gdp4, gdp6)-

Insert Here

```
df = get_data(master_sheet_name = 'Data Collection Master',master_sheet_no =
Sheet_Number, region_name = 'Ward_Name', region_sheet_no = 0, cred = path)

cityid = pd.read_sql(('SELECT
covid19_uat.func_get_cityid_from_citynm(\'Ward_Name\');'),db_connect.database_connec
tion)

wardid_max = pd.read_sql(('SELECT MAX(wardid) FROM covid19_uat.tran_eco_gdp_ward;'),
db_connect.database_connection)

no_rows =

df = function_name(df = df, cityid = cityid, wardid = wardid_max, rows = no_rows)
```

Type 2 (gdp3, gdp5)-

Insert Here

```
df = get_data(master_sheet_name = 'Data Collection Master',master_sheet_no =
Sheet_Number, region_name = 'Ward_Name', region_sheet_no = 0, cred = path)

cityid_max = pd.read_sql(('SELECT MAX(cityid) FROM covid19_uat.mas_city;'),
                     db_connect.database_connection)

mas_city = pd.read_sql(('SELECT * FROM covid19_uat.mas_city;'),
                     db_connect.database_connection)
mas_city = mas_city[mas_city['stateid']=='State_ID']
mas_city = mas_city.rename(columns={'citynm':'City'})
df = df.rename(columns={'Google_Sheet_City_Column_Name':'City'})
df = pd.merge(df,mas_city, on='City', how='left')
no_rows =
df = gdp5(df = df, rows = no_rows, max_cityid = cityid_max)
```