Part 1 –Extract, Staging

Download the two csv files Ships and CLIWOC15.csv. You just completed the "Extract" phase, it does get more complicated, but we keep it simple here.

We will begin by "staging" the data. The first step before "Transformation" is to get the data prepared or what's referred to as "staged" before we can load it, and that initial step is to load the data into "staging tables" in this case these are going to be staging data frames. The idea of staging tables is to keep the data as close to the source format as possible. We will begin by using Python, to load the two files into two data frames called Ship_df and Trip_df, respectively. Review the data and note that the two files share three columns: ShipName, ShipType, and Nationality. These will be used as "natural composite key" columns to join the two data frames (tables).

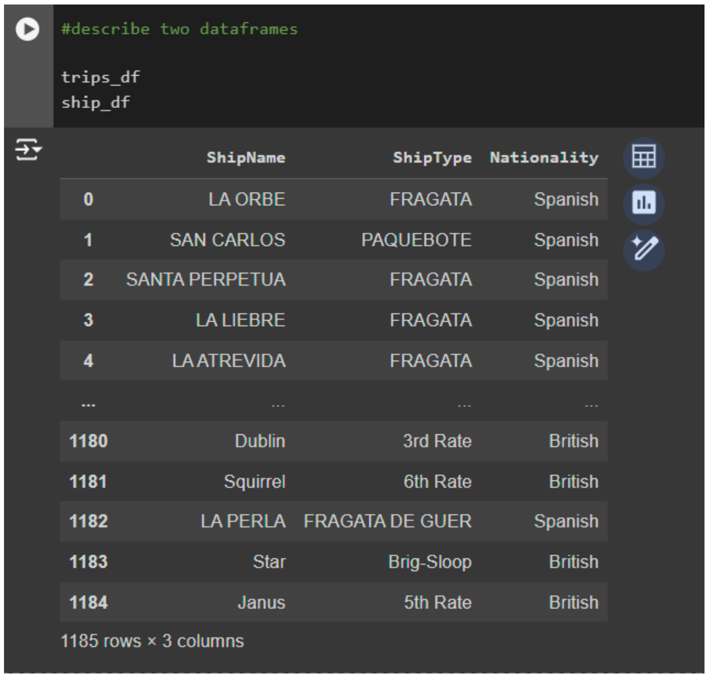
1. **Once you load the two data frames: How many rows and columns are in each data frame?**

	Rows	Columns
Ship_df	1185	3
Trip_df	280280	141

Show the record count of both Ship_df and Trip_df data frames as well as the count.
Python command:

```
display(trips_df)
display(ship_df)
```

Screenshots of the executed command:



```
#describe two dataframes

trips_df
ship_df
```

```
        ShipName          ShipType  Nationality
0       LA ORBE           FRAGATA   Spanish
1       SAN CARLOS        PAQUEBOTE Spanish
2       SANTA PERPETUA    FRAGATA   Spanish
3       LA LIEBRE         FRAGATA   Spanish
4       LA ATREVIDA       FRAGATA   Spanish
...     ...               ...       ...
1180    Dublin            3rd Rate  British
1181    Squirrel          6th Rate  British
1182    LA PERLA  FRAGATA DE GUER   Spanish
1183    Star              Brig-Sloop British
1184    Janus             5th Rate  British
1185 rows × 3 columns
```

Part 2 –Creating SCD1 Dimension and Key maintenance

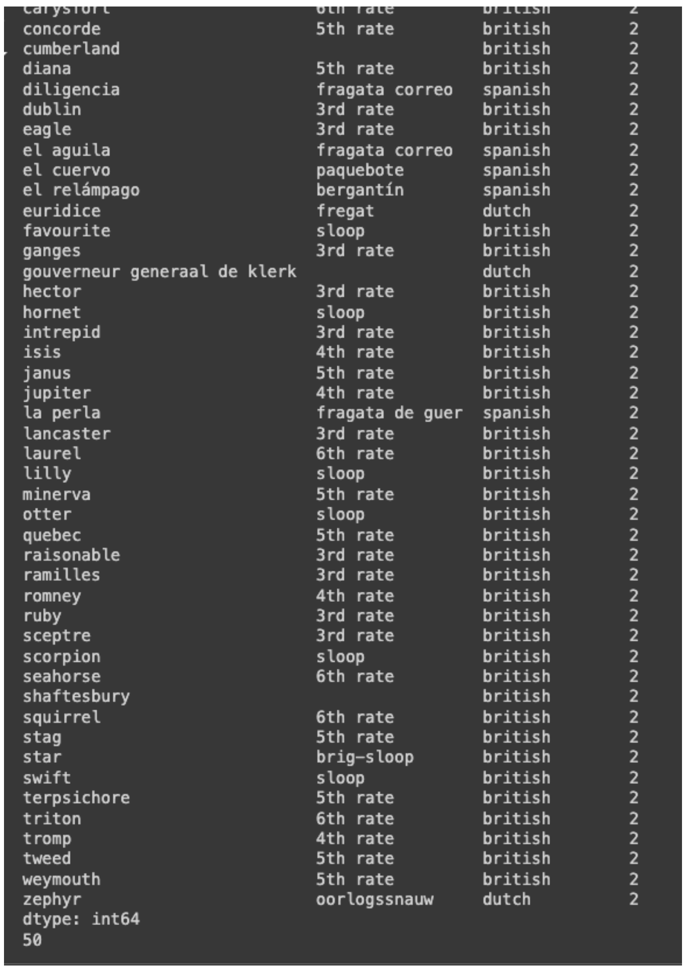
2. **Our end goal for this section of the assignment is to create a Ship dimension table ShipDim. In this case we are going to keep it simple and use SCD type1, meaning overwrite if there are any changes, or add if it's a new record. Recall that SCD type 1 needs to have a unique instance of each record, so let's check if there are any duplicates in the Ship_df**
   Show if there are any duplicate combinations of all three attributes in the Ship_df, this can just show the counts of combinations.

   Python command:

```
ship_df['ShipName'] = ship_df['ShipName'].str.lower()
ship_df['ShipType'] = ship_df['ShipType'].str.lower()
ship_df['Nationality'] = ship_df['Nationality'].str.lower()

ship_df = ship_df.fillna('')
grouped = ship_df.groupby(['ShipName', 'ShipType', 'Nationality']).size()
print(grouped[grouped>1])
```

   Screenshots of the executed command:

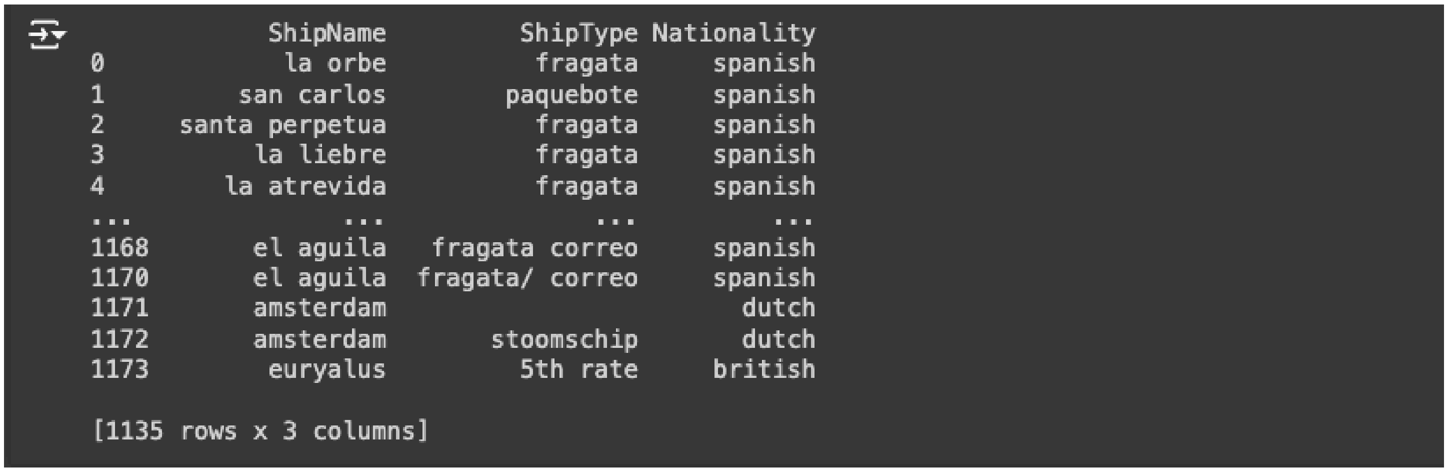


```
carysfort                     6th rate         british        2
concorde                      5th rate         british        2
cumberland                                     british        2
diana                         5th rate         british        2
diligencia                    fragata correo   spanish        2
dublin                        3rd rate         british        2
eagle                         3rd rate         british        2
el aguila                     fragata correo   spanish        2
el cuervo                     paquebote        spanish        2
el relámpago                  bergantín        spanish        2
euridice                      fregat           dutch          2
favourite                     sloop            british        2
ganges                        3rd rate         british        2
gouverneur generaal de klerk                   dutch          2
hector                        3rd rate         british        2
hornet                        sloop            british        2
intrepid                      3rd rate         british        2
isis                          4th rate         british        2
janus                         5th rate         british        2
jupiter                       4th rate         british        2
la perla                      fragata de guer  spanish        2
lancaster                     3rd rate         british        2
laurel                        6th rate         british        2
lilly                         sloop            british        2
minerva                       5th rate         british        2
otter                         sloop            british        2
quebec                        5th rate         british        2
raisonable                    3rd rate         british        2
ramilles                      3rd rate         british        2
romney                        4th rate         british        2
ruby                          3rd rate         british        2
sceptre                       3rd rate         british        2
scorpion                      sloop            british        2
seahorse                      6th rate         british        2
shaftesbury                                    british        2
squirrel                      6th rate         british        2
stag                          5th rate         british        2
star                          brig-sloop       british        2
swift                         sloop            british        2
terpsichore                   5th rate         british        2
triton                        6th rate         british        2
tromp                         4th rate         british        2
tweed                         5th rate         british        2
weymouth                      5th rate         british        2
zephyr                        oorlogssnauw     dutch          2
dtype: int64
50
```

3. **You will notice that Ship_df has some duplicates that need to be removed before that data can be used to populate a ShipDim dimension table for it to be in SCD1 format. Use pandas to drop the duplicates and store the result into another data frame labelled ShipDistinct_df.**
   Show the above operation to create a new ShipDistinct_df data frame by removing the duplicate records.
   Python command:

```
shipdistinct_df = ship_df.drop_duplicates(subset=['ShipName', 'ShipType',
'Nationality'])
print(shipdistinct_df)
```

Screenshots of the executed command:



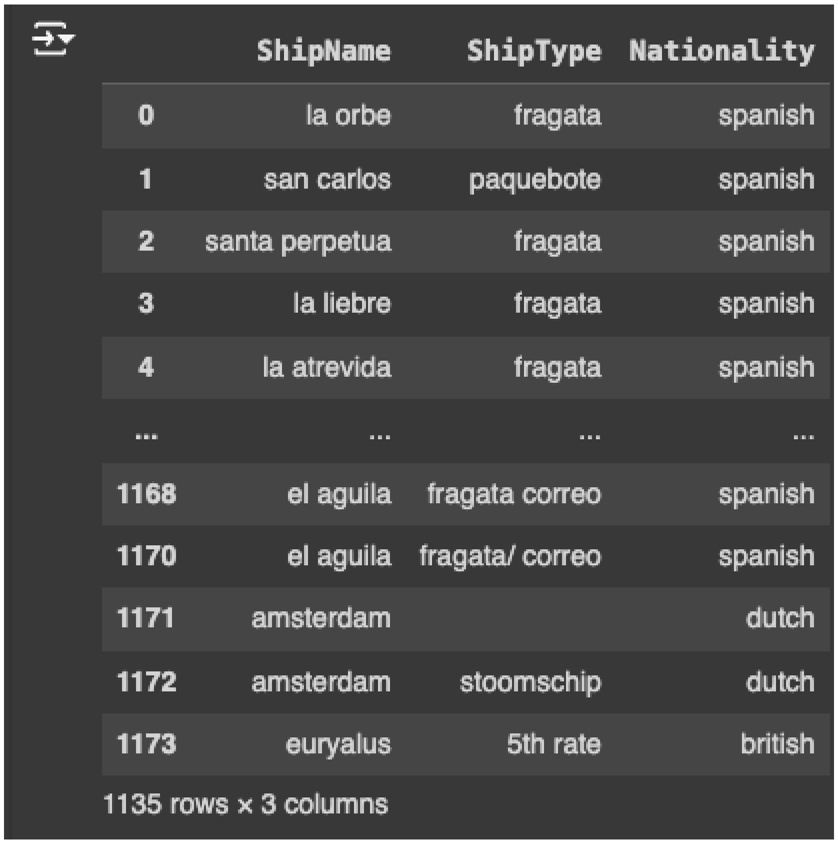
```
             ShipName         ShipType Nationality
0             la orbe          fragata     spanish
1          san carlos        paquebote     spanish
2      santa perpetua          fragata     spanish
3           la liebre          fragata     spanish
4         la atrevida          fragata     spanish
...               ...              ...         ...
1168        el aguila   fragata correo     spanish
1170        el aguila  fragata/ correo     spanish
1171        amsterdam                        dutch
1172        amsterdam       stoomschip       dutch
1173         euryalus         5th rate     british

[1135 rows x 3 columns]
```

4. **Show new ShipDistinct_df data frame record count which has no duplicates, use the commands you used in question 1 and 3 on this new data frame.**
   Python command:

```
display(shipdistinct_df)
```

   Screenshots of the executed command:



	ShipName	ShipType	Nationality
0	la orbe	fragata	spanish
1	san carlos	paquebote	spanish
2	santa perpetua	fragata	spanish
3	la liebre	fragata	spanish
4	la atrevida	fragata	spanish
...	...	...	...
1168	el aguila	fragata correo	spanish
1170	el aguila	fragata/ correo	spanish
1171	amsterdam		dutch
1172	amsterdam	stoomschip	dutch
1173	euryalus	5th rate	british

1135 rows × 3 columns

   How many rows are in ShipDistinct_df after duplicates have been dropped? _1135

Note: the new count should make sense in reviewing groupby count results between the two data frames.

5. **Now let's focus on the Trip_df. Trip has some additional ships (ShipName, ShipType, Nationality) that do not currently appear in ShipDistinct_df. These new dimension rows need to be pulled and added to the dimension table. Inspect the column names of the Trips_df and provide screenshot of the columns.**
   Hint: Look into columns.tolist() function, if not all columns display, modify settings to display all columns by using pd.set_option('display.max_columns', None)

   Python command:

```
pd.set_option('display.max_columns', None)
columns = trips_df.columns.tolist()
print(columns)
```

   Screenshots of the executed command:

```
['RecID', 'InstAbbr', 'InstName', 'InstPlace', 'InstLand', 'NumberEntry',
'NameArchiveSet', 'ArchivePart', 'Specification', 'LogbookIdent',
'LogbookLanguage', 'EnteredBy', 'DASnumber', 'ImageNumber', 'VoyageFrom',
'VoyageTo', 'ShipName', 'ShipType', 'Company', 'OtherShipInformation',
'Nationality', 'Name1', 'Rank1', 'Name2', 'Rank2', 'Name3', 'Rank3',
'ZeroMeridian', 'StartDay', 'TimeGen', 'ObsGen', 'ReferenceCourse',
'ReferenceWindDirection', 'DistUnits', 'DistToLandmarkUnits',
'DistTravelledUnits', 'LongitudeUnits', 'VoyageIni', 'UnitsOfMeasurement',
'Calendar', 'Year', 'Month', 'Day', 'DayOfTheWeek', 'PartDay', 'TimeOB',
'Watch', 'Glasses', 'UTC', 'CMG', 'ShipSpeed', 'Distance', 'drLatDeg',
'drLatMin', 'drLatSec', 'drLatHem', 'drLongDeg', 'drLongMin', 'drLongSec',
'drLongHem', 'LatDeg', 'LatMin', 'LatSec', 'LatHem', 'LongDeg', 'LongMin',
'LongSec', 'LongHem', 'Lat3', 'Lon3', 'LatInd', 'LonInd', 'PosCoastal',
'EncName', 'EncNat', 'EncRem', 'Anchored', 'AnchorPlace', 'LMname1',
'LMdirection1', 'LMdistance1', 'LMname2', 'LMdirection2', 'LMdistance2',
'LMname3', 'LMdirection3', 'LMdistance3', 'EstError', 'ApplError',
'WindDirection', 'AllWindDirections', 'WindForce', 'WindForceScale',
'AllWindForces', 'WindScale', 'Weather', 'ShapeClouds', 'DirClouds',
'Clearness', 'PrecipitationDescriptor', 'CloudFrac', 'Gusts', 'Rain', 'Fog',
'Snow', 'Thunder', 'Hail', 'SeaIce', 'Duplicate', 'Release', 'SSTReading',
'SSTReadingUnits', 'StateSea', 'CurrentDir', 'CurrentSpeed', 'TairReading',
'AirThermReadingUnits', 'ProbTair', 'BaroReading', 'AirPressureReadingUnits',
'BarometerType', 'BarTempReading', 'BarTempReadingUnits', 'HumReading',
'HumidityUnits', 'HumidityMethod', 'PumpWater', 'WaterAtThePumpUnits',
'LifeOnBoard', 'LifeOnBoardMemo', 'Cargo', 'CargoMemo', 'ShipAndRig',
'ShipAndRigMemo', 'Biology', 'BiologyMemo', 'WarsAndFights',
'WarsAndFightsMemo', 'Illustrations', 'TrivialCorrection', 'OtherRem']
```

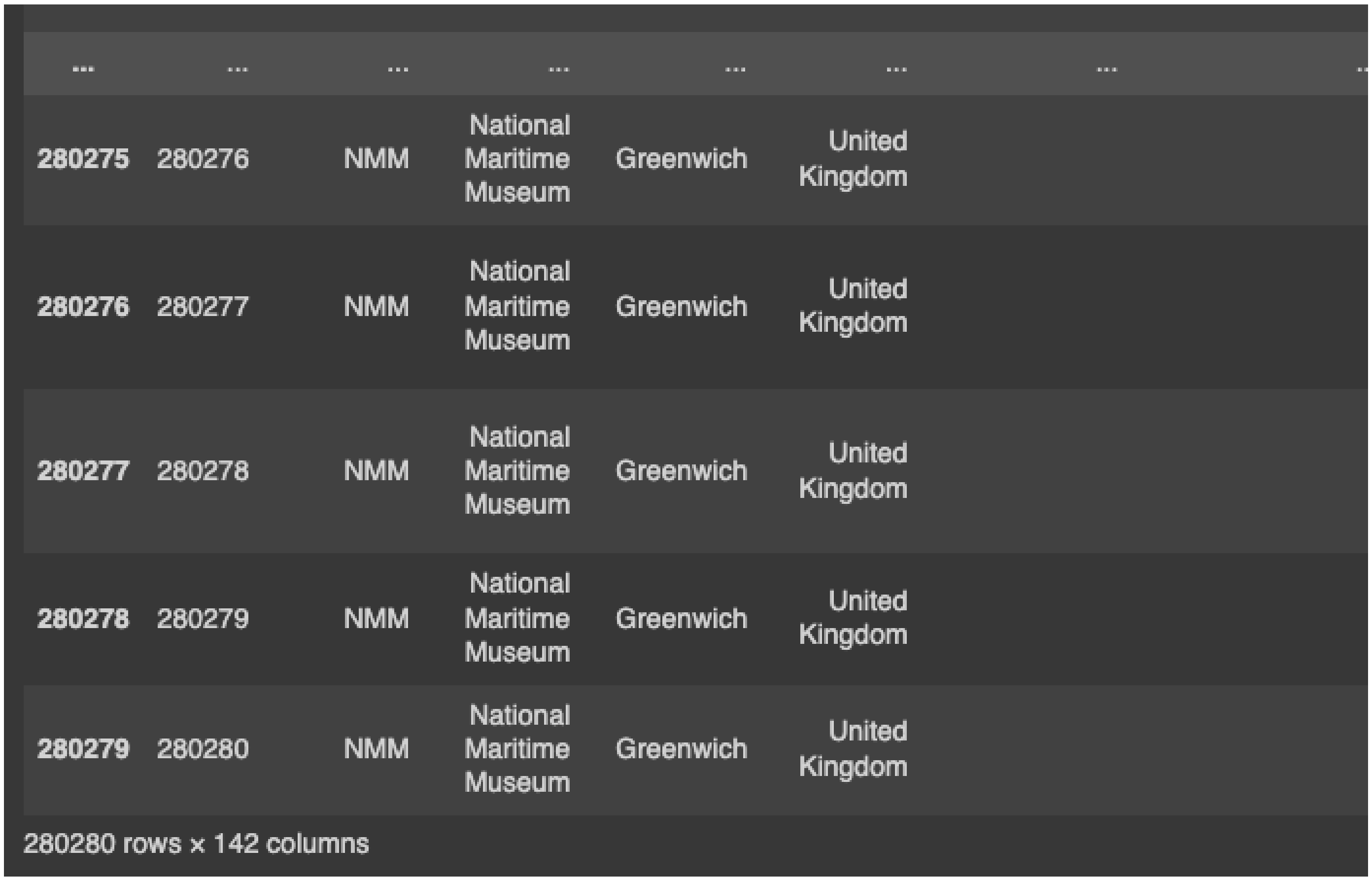
6. **Our goal is to identify any Ships in Trips_df that is not in ShipDistinct. We can use a LEFT JOIN on these dataframes to determine which values of (ShipName, ShipType, Nationality) are in Trip_df but not in ShipDistinct_df.**

   First create a joined data frame called ShipTrips_df by using the merge() function on the Trips_df and the ShipDistinct_df; use a left join on (ShipName, ShipType, Nationality) and set the indicator to True.

   Show the merge command to create the ShipTrips_df data frame.
   Python command:

```
shiptrips_df =pd.merge(trips_df,
                   shipdistinct_df,
                   how = 'left',
                   on=['ShipName', 'ShipType', 'Nationality'],
                   indicator=True)
display(shiptrips_df)
```

   Screenshots of the executed command:



...	...	...	...	...	...	...	..
280275	280276	NMM	National Maritime Museum	Greenwich	United Kingdom		
280276	280277	NMM	National Maritime Museum	Greenwich	United Kingdom		
280277	280278	NMM	National Maritime Museum	Greenwich	United Kingdom		
280278	280279	NMM	National Maritime Museum	Greenwich	United Kingdom		
280279	280280	NMM	National Maritime Museum	Greenwich	United Kingdom		

280280 rows × 142 columns

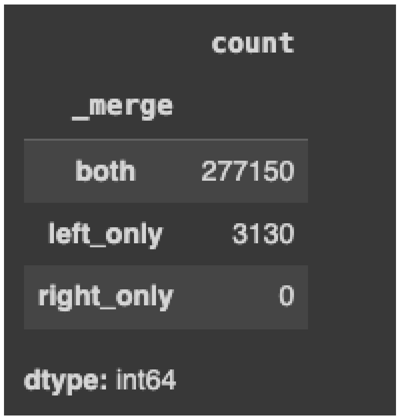
7. **Inspect the resulting ShipsTrips_df data frame (use the head() function), specifically scroll all the way to the right and note the _merge column that has been added. Let's determine the unique combinations of _merge column by using the value_counts() function.**

Show the value_counts() of the _merge column.

Python command:

```
shiptrips_df['_merge'].value_counts()
```

Screenshots of the executed command:



```
             count
    _merge
      both  277150
 left_only    3130
right_only       0

dtype: int64
```

For each of the resulting values (you should see three) of the results above**, very briefly explain what it means – short, bulleted list.**

- both – the ship details exist in both tables
- left_only – the ship details exist in trip_df but not ship_df
- right_only – the ship details exist in ship_df but not trip_df

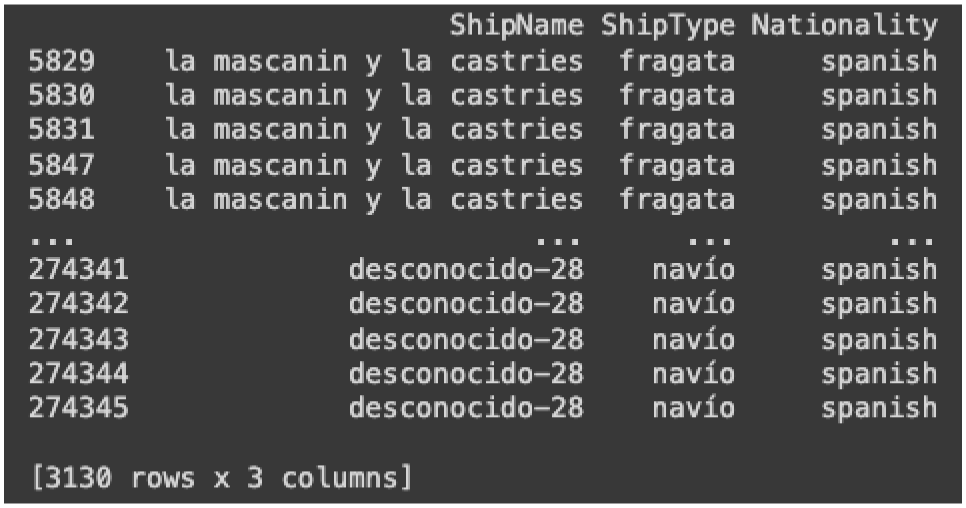
8. **Now we can filter out the new records we will need to bring into our Ships from the joined data frame, decide the filter condition based on the results from question 7.**
Hint: Look into query and filter functions to use on the ShipsTrips_df. The filter function will show the attributes that we want to look at, while the query function will help us filter the results

Provide the function call for the ShipsTrips_df showing only the columns that we need (ShipName, ShipType, Nationality), as well as the _merge column. The function call should filter (query) _merge column as outlined in the directions above.

Python command:

```
filtered_ships = shiptrips_df.query('_merge ==
"left_only"').filter(['ShipName', 'ShipType', 'Nationality'])
print(filtered_ships)
```

Screenshots of the executed command:



```
                         ShipName ShipType Nationality
5829    la mascanin y la castries  fragata     spanish
5830    la mascanin y la castries  fragata     spanish
5831    la mascanin y la castries  fragata     spanish
5847    la mascanin y la castries  fragata     spanish
5848    la mascanin y la castries  fragata     spanish
...                           ...      ...         ...
274341             desconocido-28    navío     spanish
274342             desconocido-28    navío     spanish
274343             desconocido-28    navío     spanish
274344             desconocido-28    navío     spanish
274345             desconocido-28    navío     spanish

[3130 rows x 3 columns]
```

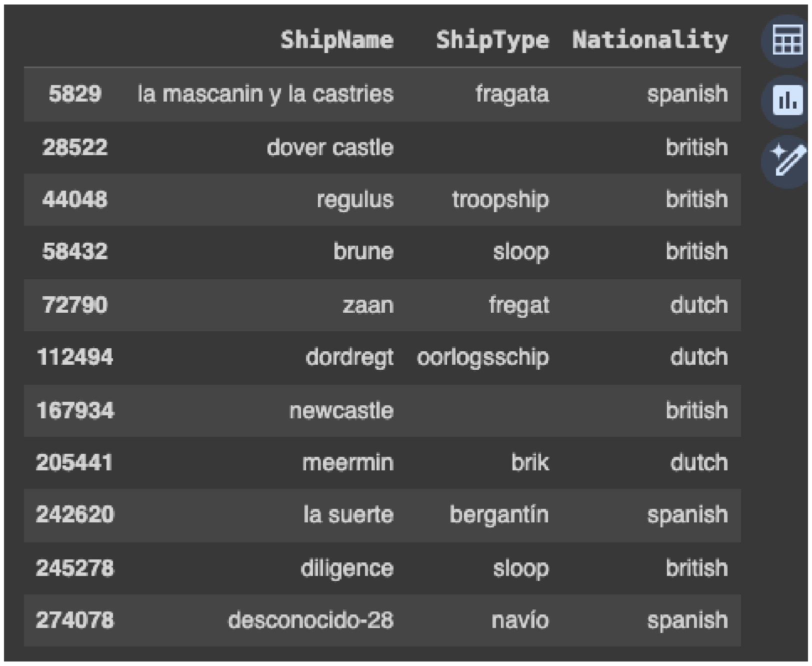
9. **Your rows count should match the count in question 7. How many new records were found?**
   3130

10. **Now let's find the distinct instances of the ShipName, ShipType, and Nationality. Perform the same operations as outlined in steps 2 through 4 to create a ShipsTrips_Distinct_df (this new data frame should not have any duplicates)**

Python command:

```
#filter out the distinct values and update the table
filtered_ships_distinct = filtered_ships.drop_duplicates(subset=['ShipName',
'ShipType', 'Nationality'])
display(filtered_ships_distinct)
```

Screenshots of the executed command:

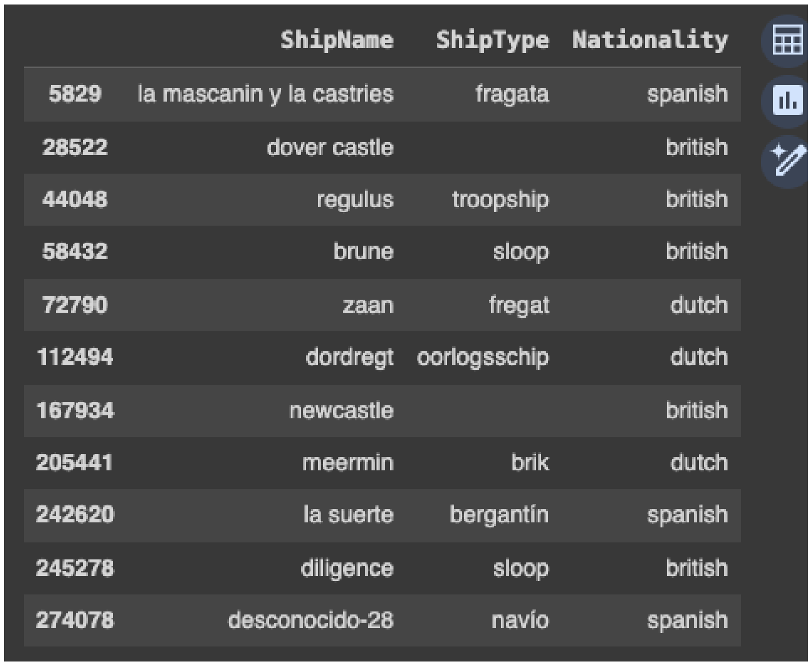


	ShipName	ShipType	Nationality
5829	la mascanin y la castries	fragata	spanish
28522	dover castle		british
44048	regulus	troopship	british
58432	brune	sloop	british
72790	zaan	fregat	dutch
112494	dordregt	oorlogsschip	dutch
167934	newcastle		british
205441	meermin	brik	dutch
242620	la suerte	bergantín	spanish
245278	diligence	sloop	british
274078	desconocido-28	navío	spanish

11. **Show the resulting data frame – list all the data.**

    Python command: `display(filtered_ships_distinct)`

    Screenshots of the executed command:



	ShipName	ShipType	Nationality
5829	la mascanin y la castries	fragata	spanish
28522	dover castle		british
44048	regulus	troopship	british
58432	brune	sloop	british
72790	zaan	fregat	dutch
112494	dordregt	oorlogsschip	dutch
167934	newcastle		british
205441	meermin	brik	dutch
242620	la suerte	bergantín	spanish
245278	diligence	sloop	british
274078	desconocido-28	navío	spanish

How many new records are in the ShipsTrips_Distinct_df? 11

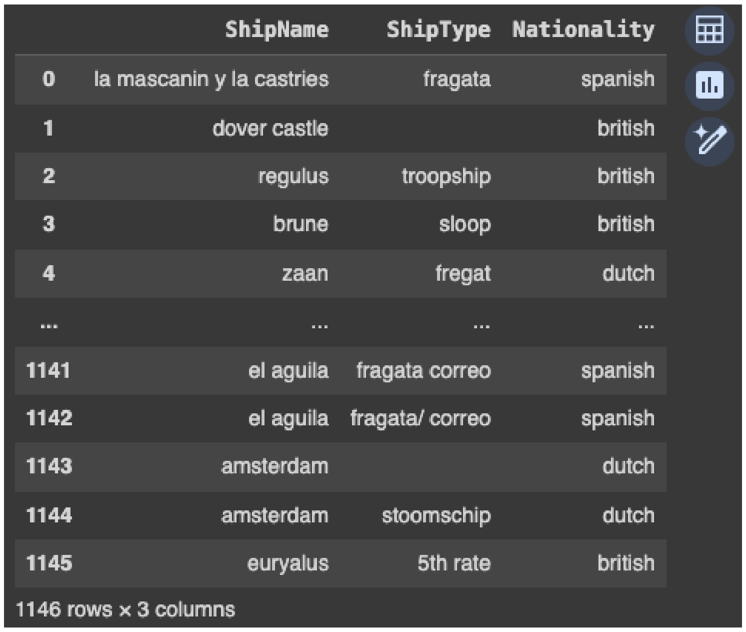
12. **Combine the two distinct data frames (ShipsDistinct_df and ShipsTrips_Distinct_df) into DimShip data frame. Hint: Use the pandas append() or the pd.concat() function, look to ignore the existing index as we will create a new surrogate primary key in the next step.**

    Show the command combining the data frames into a single DimShip.

    Python command:

```
dimship = pd.concat([filtered_ships_distinct, shipdistinct_df],
ignore_index=True)
display(dimship)
```

Screenshots of the executed command:



	ShipName	ShipType	Nationality
0	la mascanin y la castries	fragata	spanish
1	dover castle		british
2	regulus	troopship	british
3	brune	sloop	british
4	zaan	fregat	dutch
...	...	...	...
1141	el aguila	fragata correo	spanish
1142	el aguila	fragata/ correo	spanish
1143	amsterdam		dutch
1144	amsterdam	stoomschip	dutch
1145	euryalus	5th rate	british

1146 rows × 3 columns

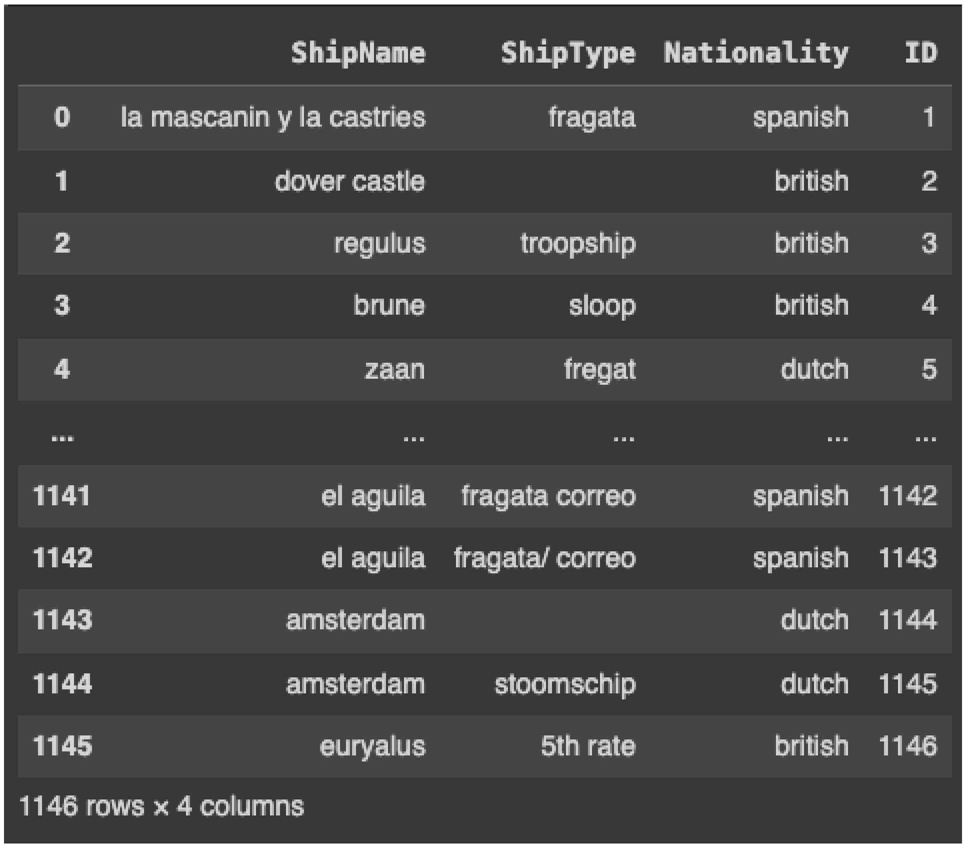
How many records are in DimShip now?
1146 rows

13. Now we need to create a surrogate primary key for the DimShip data frame. Use the reset_index() function to add a column to the DimShip data frame and call the column "Id", start the index at 1. Hint: investigate how to add a new column to the existing data frame. Show the command creating the surrogate key for DimShip and a separate command showing the new index column. Use display() method.

Python command:

```
dimship = dimship.reset_index(drop=True)
dimship['ID'] = dimship.index + 1
display(dimship)
```

Screenshots of the executed command:



	ShipName	ShipType	Nationality	ID
0	la mascanin y la castries	fragata	spanish	1
1	dover castle		british	2
2	regulus	troopship	british	3
3	brune	sloop	british	4
4	zaan	fregat	dutch	5
...	...	...	...	...
1141	el aguila	fragata correo	spanish	1142
1142	el aguila	fragata/ correo	spanish	1143
1143	amsterdam		dutch	1144
1144	amsterdam	stoomschip	dutch	1145
1145	euryalus	5th rate	british	1146

1146 rows × 4 columns

Congratulations, you have now created a clean data frame called DimShip which includes distinct record combinations as well as a primary key.

Part 3 –Creating Fact data frame

Now that we have a DimShip dimension, lets focus on creating a FactTrip dataframe. There are many columns in the Trip_df – if you recall fact tables contain measures. Some of the columns are good candidates for additional dimensions, we are going to keep it simple for now and focus on creating a Fact table. Select three or four numeric columns from the original Trip_df dataframe which you will use.

14. List the three measures columns you will use:

- Measure 1: PosCoastal
- Measure 2:LatInd
- Measure 3:LonInd

15. Create a new dataframe called FactTrip that includes the following attributes:

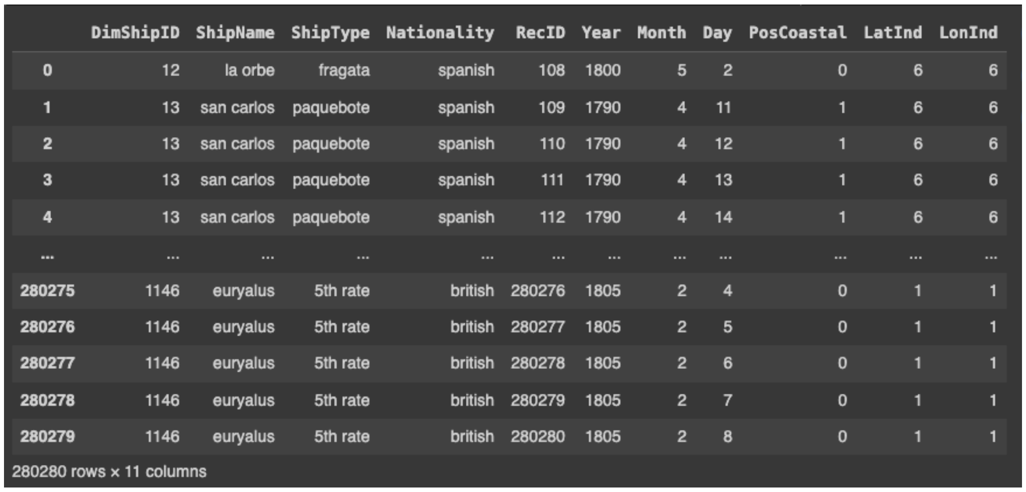
a. **ShipName, ShipType, Nationality – these are our natural key to connect the fact table to the ShipDim**
b. **RecID, Year, Month, Day (we will work with dates in a later question)**
c. **The DimShipId foreign key from the DimShip dataframe – you will need to join to the DimShip dataFrame to get this attribute. Make sure to call this attribute DimShipId**
d. **Three numeric values which you selected in question 15.**

Provide the command(s) creating the FactTrip data frame and the results of displaying some sample data from the FactTrip. Provide a third screenshot verifying counts between the original Trip_df and FactTrip.

Python commands:

```
newdimship = dimship[['ID','ShipName', 'ShipType', 'Nationality']]
facttrip = trips_df.merge(newdimship, on=['ShipName', 'ShipType',
'Nationality'], how='left')
facttrip.rename(columns={'ID': 'DimShipID'}, inplace=True)
facttrip=facttrip[['DimShipID','ShipName','ShipType','Nationality','RecID','Yea
r','Month','Day','PosCoastal','LatInd','LonInd']]
display(facttrip)
```

Screenshots of the executed command:



	DimShipID	ShipName	ShipType	Nationality	RecID	Year	Month	Day	PosCoastal	LatInd	LonInd
0	12	la orbe	fragata	spanish	108	1800	5	2	0	6	6
1	13	san carlos	paquebote	spanish	109	1790	4	11	1	6	6
2	13	san carlos	paquebote	spanish	110	1790	4	12	1	6	6
3	13	san carlos	paquebote	spanish	111	1790	4	13	1	6	6
4	13	san carlos	paquebote	spanish	112	1790	4	14	1	6	6
...	...	...	...	...	...	...	...	...	...	...	...
280275	1146	euryalus	5th rate	british	280276	1805	2	4	0	1	1
280276	1146	euryalus	5th rate	british	280277	1805	2	5	0	1	1
280277	1146	euryalus	5th rate	british	280278	1805	2	6	0	1	1
280278	1146	euryalus	5th rate	british	280279	1805	2	7	0	1	1
280279	1146	euryalus	5th rate	british	280280	1805	2	8	0	1	1

280280 rows × 11 columns

```
FactTrips count :  280280
Trips count :  280280
```

Which counts specifically from the two data frames gives you confidence in the FactTrip data frame? (Short answer- single sentence)

Since both facttrips and trips_df have the same row count means that all the records from trips_df have been accounted for. Facttrip will have repeating data but that is because there are unique values for all the other numerical columns that we have added to the fact.

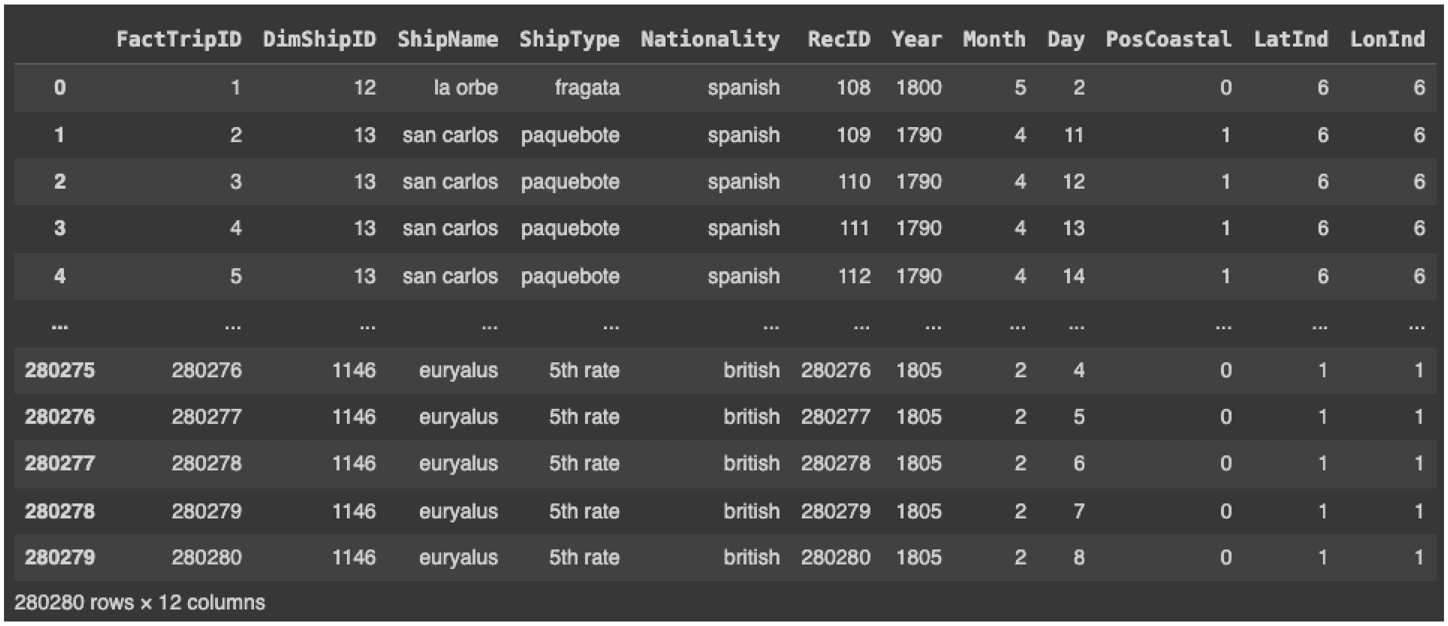
16. **Add a surrogate primary key to the FactTrip data frame like we did in step 13 for the DimShip data frame.**

Provide the command and results of creating the surrogate key for FactTrip and a second command and results showing the new index column including several rows of the FactTrip data frame. Use display() method.

Python commands:

```
facttrip =facttrip.reset_index(drop=True)
facttrip['FactTripID'] = facttrip.index + 1
facttrip = facttrip[['FactTripID'] + [col for col in facttrip.columns if col !=
'FactTripID']]
display(facttrip)
```

Screenshots of the executed commands:



	FactTripID	DimShipID	ShipName	ShipType	Nationality	RecID	Year	Month	Day	PosCoastal	LatInd	LonInd
0	1	12	la orbe	fragata	spanish	108	1800	5	2	0	6	6
1	2	13	san carlos	paquebote	spanish	109	1790	4	11	1	6	6
2	3	13	san carlos	paquebote	spanish	110	1790	4	12	1	6	6
3	4	13	san carlos	paquebote	spanish	111	1790	4	13	1	6	6
4	5	13	san carlos	paquebote	spanish	112	1790	4	14	1	6	6
...	...	...	...	...	...	...	...	...	...	...	...	...
280275	280276	1146	euryalus	5th rate	british	280276	1805	2	4	0	1	1
280276	280277	1146	euryalus	5th rate	british	280277	1805	2	5	0	1	1
280277	280278	1146	euryalus	5th rate	british	280278	1805	2	6	0	1	1
280278	280279	1146	euryalus	5th rate	british	280279	1805	2	7	0	1	1
280279	280280	1146	euryalus	5th rate	british	280280	1805	2	8	0	1	1

280280 rows × 12 columns

Part 4 – Transformation: Dates

There are a few columns in the FactTrip data frame that, together, indicate a specific date: Year, Month, Day. We will use the next step to transform those columns into a date column.

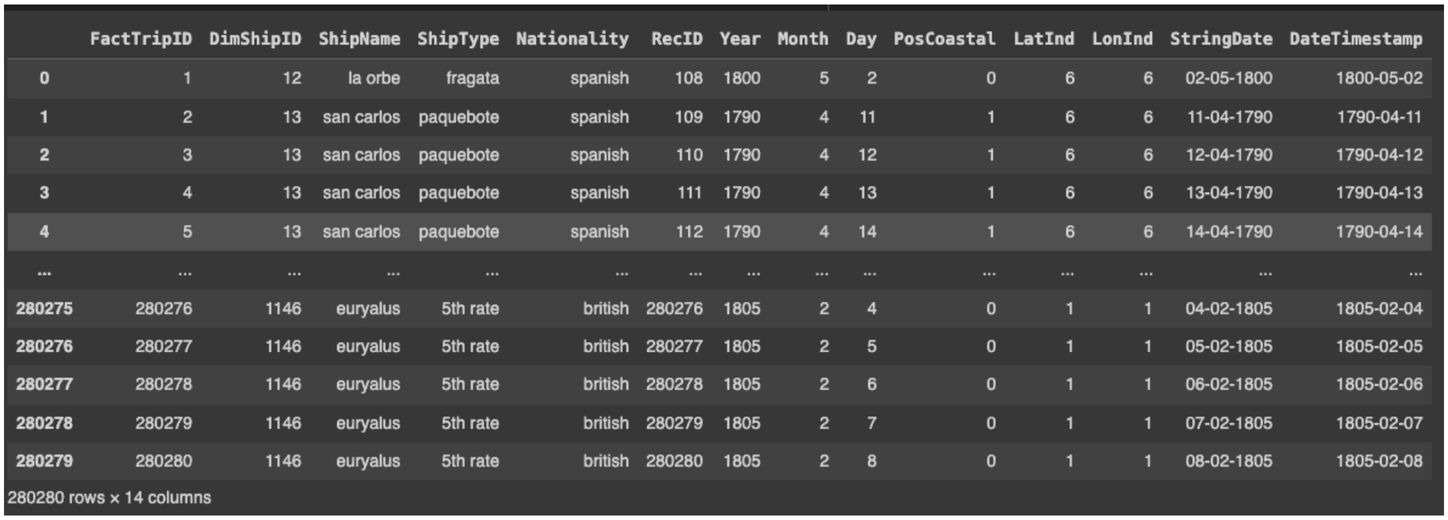
17. **Add two new columns to the data frame and populate one with a string for the date and the second with the date calculated from the string. Hint, look into date formatting and string conversion as well as the to_datetime() function and pay attention to how to handle errors (errors='coerce') and format.**

    Python command:

```
#create new column with date as a string
facttrip['StringDate'] = facttrip['Day'].astype(str).str.zfill(2) + '-' +
facttrip['Month'].astype(str).str.zfill(2) + '-' + facttrip['Year'].astype(str)

#create new column to convert DateString to a datetime object
facttrip['DateTimestamp'] = pd.to_datetime(facttrip['StringDate'], format='%d-%m-%Y',
errors='coerce')
display(facttrip)
```

Screenshots of the executed command:

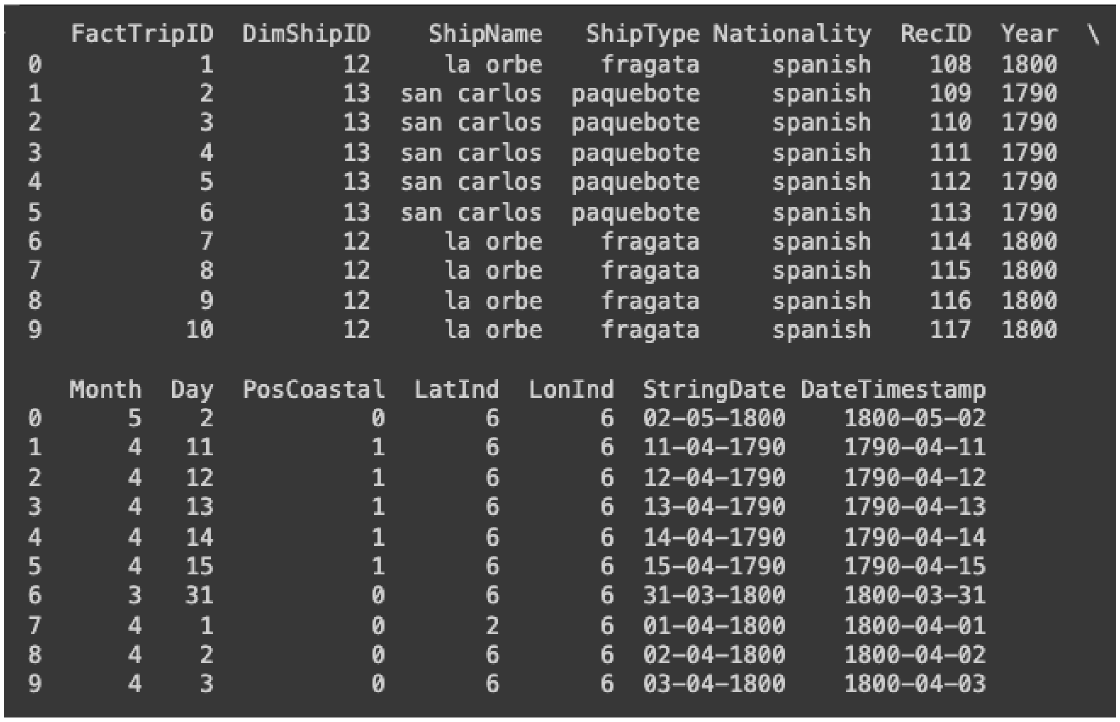


	FactTripID	DimShipID	ShipName	ShipType	Nationality	RecID	Year	Month	Day	PosCoastal	LatInd	LonInd	StringDate	DateTimestamp
0	1	12	la orbe	fragata	spanish	108	1800	5	2	0	6	6	02-05-1800	1800-05-02
1	2	13	san carlos	paquebote	spanish	109	1790	4	11	1	6	6	11-04-1790	1790-04-11
2	3	13	san carlos	paquebote	spanish	110	1790	4	12	1	6	6	12-04-1790	1790-04-12
3	4	13	san carlos	paquebote	spanish	111	1790	4	13	1	6	6	13-04-1790	1790-04-13
4	5	13	san carlos	paquebote	spanish	112	1790	4	14	1	6	6	14-04-1790	1790-04-14
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
280275	280276	1146	euryalus	5th rate	british	280276	1805	2	4	0	1	1	04-02-1805	1805-02-04
280276	280277	1146	euryalus	5th rate	british	280277	1805	2	5	0	1	1	05-02-1805	1805-02-05
280277	280278	1146	euryalus	5th rate	british	280278	1805	2	6	0	1	1	06-02-1805	1805-02-06
280278	280279	1146	euryalus	5th rate	british	280279	1805	2	7	0	1	1	07-02-1805	1805-02-07
280279	280280	1146	euryalus	5th rate	british	280280	1805	2	8	0	1	1	08-02-1805	1805-02-08

280280 rows × 14 columns

18. Take a screen shot showing the first 10-20 rows of the updated FactTrip data frame.
    Python command: `print(facttrip[0:10])`

Screenshots of the executed command:



```
   FactTripID  DimShipID    ShipName   ShipType Nationality  RecID  Year  \
0           1         12     la orbe    fragata     spanish    108  1800
1           2         13  san carlos  paquebote     spanish    109  1790
2           3         13  san carlos  paquebote     spanish    110  1790
3           4         13  san carlos  paquebote     spanish    111  1790
4           5         13  san carlos  paquebote     spanish    112  1790
5           6         13  san carlos  paquebote     spanish    113  1790
6           7         12     la orbe    fragata     spanish    114  1800
7           8         12     la orbe    fragata     spanish    115  1800
8           9         12     la orbe    fragata     spanish    116  1800
9          10         12     la orbe    fragata     spanish    117  1800

   Month  Day  PosCoastal  LatInd  LonInd  StringDate DateTimestamp
0      5    2           0       6       6  02-05-1800    1800-05-02
1      4   11           1       6       6  11-04-1790    1790-04-11
2      4   12           1       6       6  12-04-1790    1790-04-12
3      4   13           1       6       6  13-04-1790    1790-04-13
4      4   14           1       6       6  14-04-1790    1790-04-14
5      4   15           1       6       6  15-04-1790    1790-04-15
6      3   31           0       6       6  31-03-1800    1800-03-31
7      4    1           0       2       6  01-04-1800    1800-04-01
8      4    2           0       6       6  02-04-1800    1800-04-02
9      4    3           0       6       6  03-04-1800    1800-04-03
```

Transforming date is a single example of transformation, usually this is one of the more complicated steps. In your project, you will want to focus transforming strings and aggregating data to create measures as part of transformation instead of just extracting measures from the source file.

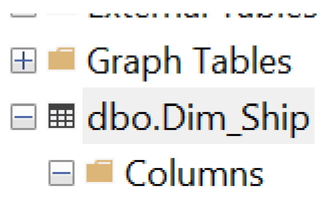
Part 5 – Load the data frames into the database tables

19. Take a screen shot of your command to create the Dim_Ship table in your database

Python/or SQL command:

```
#creating table dimship
create_table_query = '''
CREATE TABLE Dim_Ship (
    Id INT PRIMARY KEY,
    ShipName NVARCHAR(255),
    ShipType NVARCHAR(255),
    Nationality NVARCHAR(255)
    );
    '''
cursor.execute(create_table_query)
conn.commit()
```

Screenshots of the executed command:



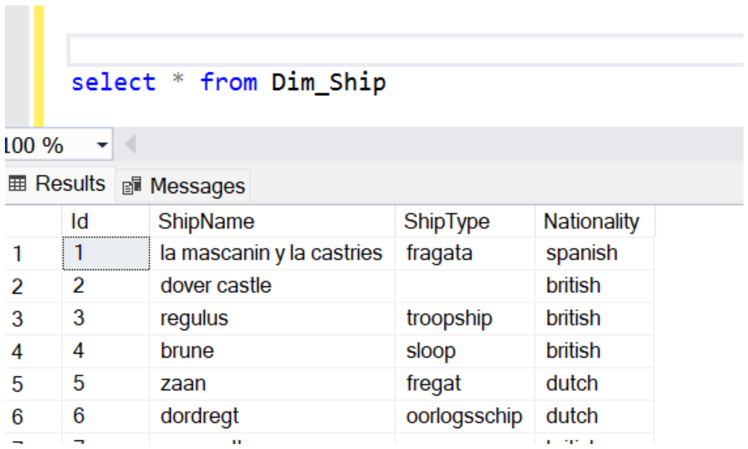
20. Show your command of loading the DimShip into the database using Python.

Python command:

```
#inserting data
for index,row in dimship.iterrows():
    cursor.execute("""
        INSERT INTO Dim_Ship(Id,ShipName,ShipType,Nationality)
        VALUES(?,?,?,?)
        """,

row['ID'],row['ShipName'],row['ShipType'],row['Nationality']
                  )
    conn.commit()
```

Screenshots of the executed command:



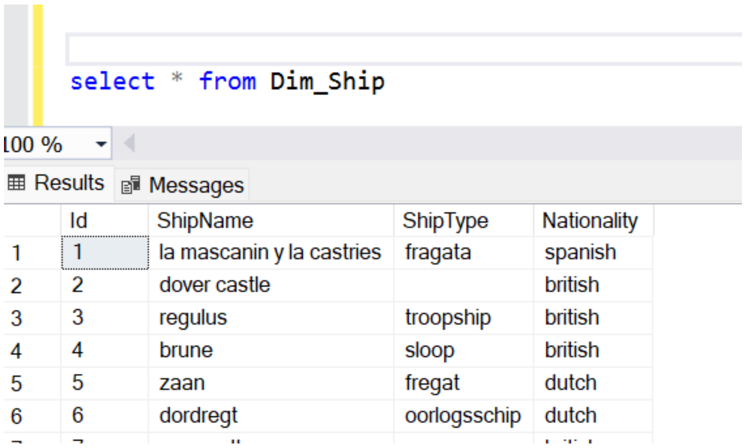
```
select * from Dim_Ship
```

	Id	ShipName	ShipType	Nationality
1	1	la mascanin y la castries	fragata	spanish
2	2	dover castle		british
3	3	regulus	troopship	british
4	4	brune	sloop	british
5	5	zaan	fregat	dutch
6	6	dordregt	oorlogsschip	dutch

21. Display the loaded data from the Dim_Ship table in your Database using SQL

SQL command: `SELECT * FROM Dim_Ship`

Screenshots of the executed command:



```
select * from Dim_Ship
```

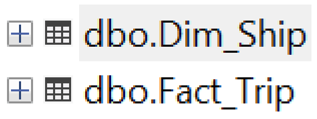
	Id	ShipName	ShipType	Nationality
1	1	la mascanin y la castries	fragata	spanish
2	2	dover castle		british
3	3	regulus	troopship	british
4	4	brune	sloop	british
5	5	zaan	fregat	dutch
6	6	dordregt	oorlogsschip	dutch

22. Take a screen shot of your command to create the Fact_Trip table in your database

Python/SQL command:

```
create_table_query = '''
CREATE TABLE Fact_Trip (
    FactTripID INT PRIMARY KEY,
    DimShipId INT,
    DateTimestamp DATE,
    PosCoastal INT,
    LatInd INT,
    LonInd INT,
    FOREIGN KEY (DimShipId) REFERENCES Dim_Ship(Id)
);
'''
cursor.execute(create_table_query)
conn.commit()
```

Screenshots of the executed command:



23. Take a screenshot of your command loading the FactTrip into the database using Python

Python command:

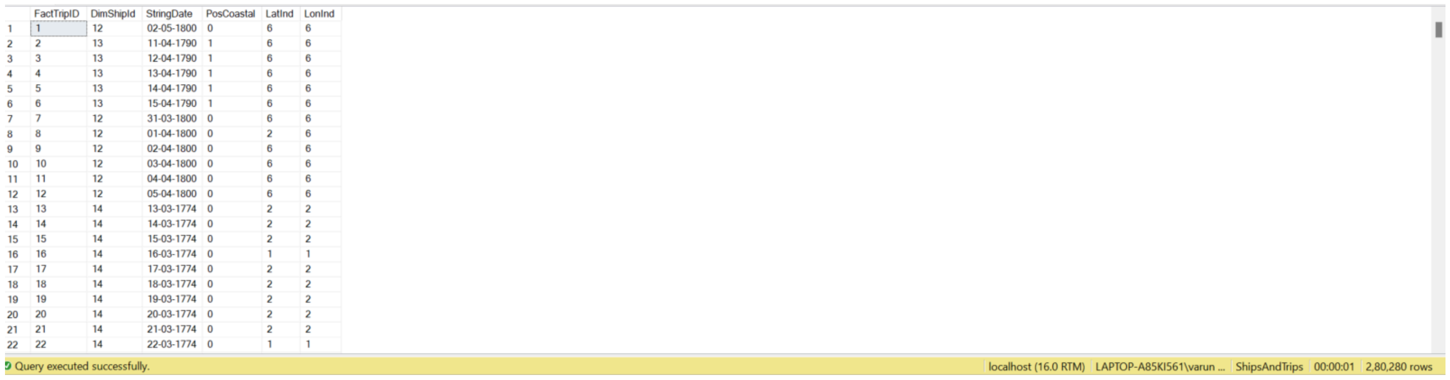
```
#inserting data into Fact_Trip
for index,row in facttrip.iterrows():
    cursor.execute("""
    INSERT INTO Fact_Trip(FactTripID,DimShipID,StringDate,PosCoastal,LatInd,LonInd)
    VALUES(?,?,?,?,?,?)
    """,
                   row['FactTripID'],
                   row['DimShipID'],
                   row['StringDate'],
                   row['PosCoastal'],
                   row['LatInd'],
                   row['LonInd']
                  )
    conn.commit()
```

24. Take a screenshot selecting the loaded data from the Fact_Trip table in your Database using SQL

SQL command:

```
select * from Fact_Trip
```

Screenshots of the executed command:



	FactTripID	DimShipId	StringDate	PosCoastal	LatInd	LonInd
1	1	12	02-05-1800	0	6	6
2	2	13	11-04-1790	1	6	6
3	3	13	12-04-1790	1	6	6
4	4	13	13-04-1790	1	6	6
5	5	13	14-04-1790	1	6	6
6	6	13	15-04-1790	1	6	6
7	7	12	31-03-1800	0	6	6
8	8	12	01-04-1800	0	2	6
9	9	12	02-04-1800	0	6	6
10	10	12	03-04-1800	0	6	6
11	11	12	04-04-1800	0	6	6
12	12	12	05-04-1800	0	6	6
13	13	14	13-03-1774	0	2	2
14	14	14	14-03-1774	0	2	2
15	15	14	15-03-1774	0	2	2
16	16	14	16-03-1774	0	1	1
17	17	14	17-03-1774	0	2	2
18	18	14	18-03-1774	0	2	2
19	19	14	19-03-1774	0	2	2
20	20	14	20-03-1774	0	2	2
21	21	14	21-03-1774	0	2	2
22	22	14	22-03-1774	0	1	1

```
Query executed successfully.    localhost (16.0 RTM) | LAPTOP-A85KI561\varun ... | ShipsAndTrips | 00:00:01 | 2,80,280 rows
```

Extra Credit - Extending the dimension

(Up-to 5 extra credit points)

Some of the fields in the CLIWOC.csv could be a new dimension, or part of the Dim_Ship table. Outline the new dimension you want to create, extract, transform and load both the dimension and the fact appropriately. A suggestion is to focus on some complexity within transformation. This will give you practice and prepare you for the term project.

Show the commends and appropriate screenshots demonstrating your work and that the data has been loaded into the database.

Python commands:

Screenshots of the executed commands:

Criterion	A	B	C	D	F	Letter Grade
Correctness and Completeness of Results (70%)	All steps' results are entirely complete and correct	About ¾ of the steps' results are correct and complete	About half of the steps' results are correct and complete	About ¼ of the steps' results are correct and complete	Virtually none of the step's results are correct and complete	
Constitution of SQL/Python and Explanations (30%)	Excellent use and integration of appropriate SQL/Python constructs and	Good use and integration of appropriate SQL/Python constructs and	Mediocre use and integration of appropriate SQL/Python constructs and	Substandard use and integration of appropriate SQL/Python constructs and	Virtually all SQL/Python constructs and supporting explanations are unsuitable	

	supporting explanation s	supporting explanation s	supporting explanations	supporting explanations	or improperly integrated	
					Assignment Grade:	