

MET CS 689 Designing and Implementing a Data Warehouse

Assignment 3A: ETL - Python



April 4, 2023

Vincent NgUYEn

Nvui@bu.edu

# Overview of the Assignment:

ETL takes most of the data warehouse developers’ and administrators’ time. This assignment will go through some of the more common ETL processes using Python. We will explore ETL via SQL in Assignment 3B.

# 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).

Pandas Reference/Hints/Notes

* [Pandas - DataFrame Reference (w3schools.com)](https://www.w3schools.com/python/pandas/pandas_ref_dataframe.asp) is a great reference to Pandas methods
* Review run LoadTitanic.py from assignment 1 on how to create a data frame from a CSV file. Note that you don’t have to have a separate Load.py, you can paste the load command directly in the notebook, however having separate load.py source files allow you to build modularity.
* display(df) method shows all the data in the data frame – this is very helpful in understanding the data
* df.head()method allows you to inspect the header and the first 10 rows – this is very helpful in understanding the data with performance in mind.
* df.count() method will show count of records, it’s good to know what was loaded or not.
* List(df.columns.values) method shows the attributes
* Loading Trips\_df might give you an error, understand why you are getting the error, and research how to solve it (you can use the “lazy option”, and not convert the data types- especially for the initial staging table)

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:**

1. **For Ship\_df**

**Ship\_df.count()** *# return record count and the count*

**Ship\_df.shape** *# return numbers of rows and columns*

**Ship\_df.shape[0]** *# number of rows*

**Ship\_df.shape[1]** # *number of columns*

1. **For Trip\_df**

**Trip\_df.count()** *# return record count and the count*

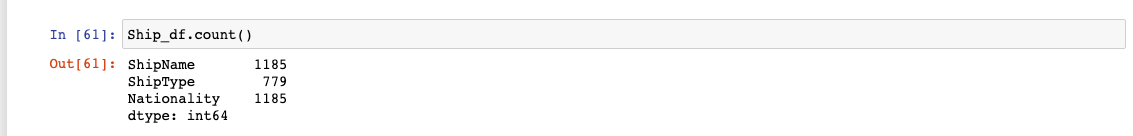
**Trip\_df.shape** *# return numbers of rows and columns*

**Trip\_df.shape[0]** *# number of rows*

**Trip\_df.shape[1]** *# number of columns*

**Screenshots of the executed command:**

1. **For Ship\_df**



Application

Description automatically generated with medium confidence

1. For Trip\_df

Table

Description automatically generated with low confidence

# Part 2 –Creating SCD1 Dimension and Key maintenance.

1. 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

Hints:

* Review the value\_counts() function – this might only help in seeing the count for a single attribute.
* Review the groupby() function – make sure to include all three attributes as we want to find a distinct combination of all three.
* You can apply multiple functions together on a data frame. For example, you can apply both groupby first and value\_counts after to get a count of distinct values.

Show if there are any duplicate combination of all three attributes in the Ship\_df, this can just show the counts of combinations.

**Python command:**

*# Count the combination of ShipName, ShipType, Nationality*

**Ship\_combine\_count = Ship\_df.groupby(['ShipName', 'ShipType', 'Nationality'], dropna = False).size().reset\_index(name='count')**

**print(Ship\_combine\_count)**

*# Get the duplicates combination of three attributes in the Ship\_df*

**Duplicate\_combination = Ship\_combine\_count[Ship\_combine\_count['count']>1]**

**print(Duplicate\_combination)**

**Screenshots of the executed command:**

**Table

Description automatically generated**

1. 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.

* Hint: Review the drop\_duplicates() function.

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()** *# drop duplicates*

**print(ShipDistinct\_df)**

**Screenshots of the executed command:**

Table

Description automatically generated with medium confidence

1. 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:**

**ShipDistinct\_df.count()** *# return record count and the count*

**ShipDistinct\_df.shape** *# return numbers of rows and columns*

**ShipDistinct\_df.shape[0]** *# number of rows*

**ShipDistinct\_df.shape[1]** *# number of columns*

**Screenshots of the executed command:**

Graphical user interface, text, application, email

Description automatically generated

How many rows are in ShipDistinct\_df after duplicates have been dropped? **1174**

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

1. 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 Trip\_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)**

**print(Trip\_df.columns.tolist())**

**Screenshots of the executed command:**

Graphical user interface, text, application, email

Description automatically generated

1. Our goal is to identify any Ships in Trip\_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 Trip\_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=Trip\_df.merge(ShipDistinct\_df, how = 'left', on = ['ShipName', 'ShipType', 'Nationality'], indicator=True)**

**ShipTrips\_df**

**Screenshots of the executed command:**

**Table

Description automatically generated with low confidence**

Background pattern

Description automatically generated

1. Inspect the resulting ShipTrips\_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.head()** *# use head() to inspect \_merge*

**ShipTrips\_df['\_merge'].value\_counts()** *# count unique combinations of \_merge columns*

**Screenshots of the executed command:**

A picture containing table

Description automatically generated

Table

Description automatically generated

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

* **both 277150** This result means that there are 277150 rows in ShipTrips\_df that came from both Trip\_df (left dataframe) and ShipDistinct\_df ( right dataframe)
* **left\_only 3130** This result means that there are 3130 rows in ShipTrips\_df that came from Trip\_df (left dataframe) and did not have matching values in the ShipDistinct\_df (right dataframe)
* **right\_only 0** This result means there are no rows that only come from ShipDistinct\_df (right dataframe)

1. 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 ShipTrips\_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:**

*# filter out (filter function) ShipTrips\_df with 4 columns: ShipName, ShipType, Nationality, \_merge*

*# and then use query function to filter the rows which has \_merge == left\_only*

**ShipTrips2\_df = ShipTrips\_df.filter(['ShipName','ShipType','Nationality','\_merge']).query("\_merge == 'left\_only'")**

**ShipTrips2\_df** *# Show results*

**Screenshots of the executed command:**

Graphical user interface, text, application

Description automatically generated

1. Your rows count should match the count in question 7. How many new records were found? 3130
2. 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:**

*# Count the combination of ShipName, ShipType, Nationality*

**Ship\_new\_count=ShipTrips2\_df.groupby(['ShipName','ShipType','Nationality'],dropna=False).size().reset\_index(name='count')**

**print(Ship\_new\_count)**

*# Get the duplicates combination of three attributes in the ShipTrips2\_df (Result of ShipTrips\_df in question 8)*

**Duplicate\_combination = Ship\_new\_count[Ship\_new\_count['count']>1]**

**print(Duplicate\_combination)**

*# drop duplicates and drop the \_merge columns to get ShipsTrips\_Distinct\_df*

**ShipsTrips\_Distinct\_df = ShipTrips2\_df.drop\_duplicates().drop(['\_merge'], axis = 1)**

**print(ShipsTrips\_Distinct\_df)**

**Screenshots of the executed command:**

Graphical user interface, table

Description automatically generated

1. Show the resulting data frame – list all the data.

**Python command:**

*# List all the data*

**print(ShipsTrips\_Distinct\_df)**

*# number of records*

**ShipsTrips\_Distinct\_df.shape[0]**

**Screenshots of the executed command:**

**Table

Description automatically generated**

How many new records are in the ShipsTrips\_Distinct\_df? **11**

1. Combine the two distinct data frames (ShipDistinct\_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:**

*# combine the two distinct data frames (ShipDistinct\_df and ShipsTrips\_Distinct\_df)*

**data = [ShipDistinct\_df,ShipsTrips\_Distinct\_df]**

**DimShip = pd.concat(data, ignore\_index = True)**

**print(DimShip)**

*# number of rows*

**DimShip.shape[0]**

**Screenshots of the executed command:**

Graphical user interface, text

Description automatically generated with medium confidence

How many records are in DimShip now? **1185**

1. 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:**

*# add the column "Id" start the index 1 to DimShip*

**DimShip['Id'] = DimShip.reset\_index().index + 1**

**print(DimShip)**

**display(DimShip)** *# Display DimShip*

**Screenshots of the executed command:**

Table

Description automatically generated

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.

1. List the three measures columns you will use:

* **Measure 1: Distance**
* **Measure 2: ShipSpeed**
* **Measure 3: drLatDeg**

1. Create a new dataframe called FactTrip that includes the following attributes:
   1. ShipName, ShipType, Nationality – these are our natural key to connect the fact table to the ShipDim
   2. RecID, Year, Month, Day (we will work with dates in a later question)
   3. The Id from the DimShip dataframe – you will need to join to the DimShip dataFrame to get this attribute. Make sure to call this attribute DimShipId
   4. 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:**

*# Create FactTrip\_filter by filter ShipName, ShipType, Nationality, RecID, Year, Month, Day, ShipSpeed, Distance, drLatDeg from the Trip\_df*

**FactTrip\_filter = Trip\_df.filter(['ShipName', 'ShipType', 'Nationality', 'RecID', 'Year', 'Month', 'Day', 'Distance', 'ShipSpeed', 'drLatDeg'])**

*# merge FactTrip\_filter with DimShip by natural key to get the Id*

**FactTrip\_merge = FactTrip\_filter.merge(DimShip, how = 'left', on = ['ShipName','ShipType','Nationality'])**

*# rename Id column to be DimShipId*

**FactTrip = FactTrip\_merge.rename(columns = {'Id': 'DimShipId'})**

*# Display FactTrip*

**FactTrip**

**Screenshots of the executed command:**

Table

Description automatically generated

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

**Your answer goes here:**

*# Review count for FactTrip*

FactTrip.groupby(['ShipName', 'ShipType', 'Nationality', 'DimShipId'], dropna = False).size()

*# Review count for Trip\_df*

Trip\_df.groupby(['ShipName', 'ShipType', 'Nationality'],dropna =False).size()

* The counts from two data frames above give me condifidence in the Factrip data frame because the count of FactTrip matches the original data frame count (Trip\_df). The results are illustrated in the screenshot below:

Table

Description automatically generated

* Another way to get confidence in FactTrip is when we merge Fact\_filter with DimShip, we keep indicator = True. And then we use value\_count of FactTrip on \_merge colum. Will see both is 280280 which is there are 280280 rows have data from both data frames. And number of rows of Trip\_df is 280280

Table

Description automatically generated

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

FactTrip['FactShipId'] = FactTrip.reset\_index().index + 1 *# Create surrogate key*

FactTrip *# Display result*

Graphical user interface, application, table

Description automatically generated

1. 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['FactShipId'] = FactTrip.reset\_index().index +** 1 *# Create surrogate key*

**FactTrip** *# Display result*

Graphical user interface, application, table

Description automatically generated

**FactTrip['FactShipId'] = FactTrip.reset\_index().index + 1** *# Create surrogate key*

**display(FactTrip)** *# Display result use display() method*

**Screenshots of the executed commands:**

Table

Description automatically generated

# 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.

1. 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 Date\_str column*

**FactTrip['Date\_str'] = FactTrip['Year'].astype(str) + FactTrip['Month'].astype(str) + FactTrip['Day'].astype(str)**

*# Create Date column from Date\_str column*

**FactTrip['Date'] = pd.to\_datetime(FactTrip['Date\_str'], errors = 'coerce', format = '%Y%m%d')**

*# Display FactTrip*

**FactTrip**

**Screenshots of the executed command:**

Table

Description automatically generated

1. Take a screen shot showing the first 10-20 rows of the updated FactTrip data frame.

**Python command:**

**FactTrip.head(15)**

**Screenshots of the executed command:**

****

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

Create a new database in your system and create a table called Dim\_Ship and load it with the DimShip data frame. The Dim\_Ship table should have all the attributes from the DimShip including the primary key. You can choose to use SQL or Python to create the Dim\_Ship table. Using Python, populate the new Dim\_Ship table.

1. Take a screen shot of your command to create the Dim\_Ship table in your database

**Python/or SQL command:**

*# Connect to database*

**import psycopg2 as pg**

**conn = pg.connect(user = "postgres",**

**password = "BU669",**

**host = "127.0.0.1",**

**port = "5432",**

**database = "Assignment\_3A")**

**cur = conn.cursor()**

*# Create Dim\_Ship table in database*

**cur.execute("CREATE TABLE IF NOT EXISTS Dim\_Ship (Id int Primary Key, ShipName varchar, ShipType varchar, Nationality varchar);")**

**conn.commit()**

**Screenshots of the executed command:**

**Graphical user interface, text, application, email

Description automatically generated**

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

**Python command:**

**for index, row in DimShip.iterrows():**

**ShipName = row.ShipName if not pd.isna(row.ShipName) else None**

**ShipType = row.ShipType if not pd.isna(row.ShipType) else None**

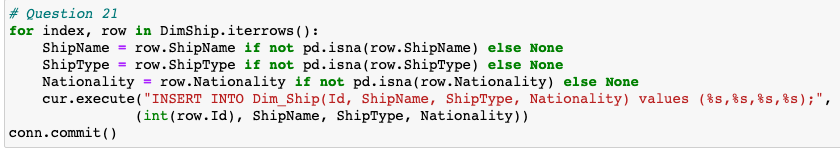
**Nationality = row.Nationality if not pd.isna(row.Nationality) else None**

**cur.execute("INSERT INTO Dim\_Ship(Id, ShipName, ShipType, Nationality) values (%s,%s,%s,%s);",**

**(int(row.Id), ShipName, ShipType, Nationality))**

**conn.commit()**

**Screenshots of the executed command:**



1. 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:Table

Description automatically generated**

Next, we will create a table called Fact\_Trip and load it with the FactTrip data frame data. Fact\_Trip table will contain the following attributes from the FactTrip data frame The primary key, for foreign key to the Dim\_Ship, the Date you converted, and the three measures. You will want to skip loading ShipName, ShipType and Nationality fields as you now have a foreign key to it and these are no longer needed. You can choose to use SQL or Python to create the table. Use Python to populate the FactTrip table.

1. Take a screen shot of your command to create the Fact\_Trip table in your database

**Python/SQL command:**

*# Create Fact\_Trip in database*

**cur.execute("""CREATE TABLE IF NOT EXISTS Fact\_Trip (FactShipID Int Primary Key,**

**DimShipId Int References Dim\_Ship (Id),**

**Date Date, Distance Float, ShipSpeed Float, drLatDeg Int);""")**

**conn.commit()**

**Screenshots of the executed command:**

Text

Description automatically generated

1. Take a screen shot of your command loading the FactTrip into the database using Python

**Python command:**

**for index, row in FactTrip.iterrows():**

**FactShipId = row.FactShipId**

**DimShipId = row.DimShipId**

**Date = row.Date if not pd.isna(row.Date) else None**

**Distance = row.Distance if not pd.isna(row.Distance) else None**

**ShipSpeed = row.ShipSpeed if not pd.isna(row.ShipSpeed) else None**

**drLatdeg = row.drLatDeg if not pd.isna(row.drLatDeg) else None**

**cur.execute("""INSERT INTO Fact\_Trip (FactShipID, DimShipId, Date, Distance, ShipSpeed, drLatDeg)**

**values (%s,%s,%s,%s,%s,%s)""",**

**(FactShipId, DimShipId, Date, Distance, ShipSpeed, drLatdeg))**

**conn.commit()**

**Screenshots of the executed command:**

**Text

Description automatically generated**

1. Take a screen shot selecting the loaded data from the Fact\_Trip table in your Database using SQL

**Python command:**

**select \* from fact\_trip;**

**Screenshots of the executed command:**

Table

Description automatically generated

Table

Description automatically generated

# 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:**

*# Filter to get more infomation about Ship*

**Ship\_add\_info\_f = Trip\_df.filter(['ShipName','ShipType','Nationality','Company','OtherShipInformation'])**

**Ship\_add\_info\_f**

*# Count the combination of ShipName, ShipType, Nationality*

**Ship\_new\_inf=Ship\_add\_info\_f.groupby(['ShipName','ShipType','Nationality','Company',**

**'OtherShipInformation'],dropna=False).size().reset\_index(name='count')**

**Ship\_new\_inf**

*# Drop duplicates to get distinct values*

**Ship\_inf\_add = Ship\_add\_info\_f.drop\_duplicates()**

**Ship\_inf\_add**

*# Create key*

**Ship\_inf\_add['new\_id'] = Ship\_inf\_add.reset\_index().index + 1**

*# Distinct Ship with added information*

**Ship\_inf\_add**

*## Doing Fact table*

*##Filter columns*

**FactTrip2\_filter = Trip\_df.filter(['ShipName','ShipType','Nationality','Company','OtherShipInformation',**

**'RecID', 'Year', 'Month', 'Day', 'Distance', 'ShipSpeed', 'drLatDeg'])**

*# Merge to Ship\_inf\_add*

**FactTrip2\_merge = FactTrip2\_filter.merge(Ship\_inf\_add, on = ['ShipName','ShipType','Nationality',**

**'Company','OtherShipInformation'])**

*# Create key*

**FactTrip2\_merge['FactTripId'] = FactTrip2\_merge.reset\_index().index + 1**

**# Change ship\_inf\_add key name**

**FactTrip\_new = FactTrip2\_merge.rename(columns = {'new\_id' : 'newDimShipid'})**

**FactTrip\_new**

*# Transform Date:*

**#Create Date\_str\_n column**

**FactTrip\_new['Date\_str2'] = FactTrip\_new['Year'].astype(str) + FactTrip\_new['Month'].astype(str) + FactTrip\_new['Day'].astype(str)**

**# Create Date column from Date\_str column**

**FactTrip\_new['Date2'] = pd.to\_datetime(FactTrip\_new['Date\_str2'], errors = 'coerce', format = '%Y%m%d')**

**FactTrip\_new**

*## Load Data into DBMS*

*# Create Dim\_Ship\_new table in database*

**cur.execute("""CREATE TABLE IF NOT EXISTS Dim\_Ship\_new**

**(Id\_new int Primary Key, ShipName varchar, ShipType varchar, Nationality varchar,**

**Company varchar, OtherShipInformation varchar);""")**

**conn.commit()**

*#Load data into DBMS*

**for index, row in Ship\_inf\_add.iterrows():**

**cur.execute("""INSERT INTO Dim\_Ship\_new(Id\_new, ShipName, ShipType, Nationality, Company,**

**OtherShipInformation) values (%s,%s,%s,%s,%s,%s);""",**

**(int(row.new\_id), str(row.ShipName), str(row.ShipType),**

**str(row.Nationality), str(row.Company), str(row.OtherShipInformation)))**

**conn.commit()**

*# Create Fact\_Trip\_New in database*

**cur.execute("""CREATE TABLE IF NOT EXISTS FactTrip\_new (FactTripid Int Primary Key,**

**DimShipId\_new Int References Dim\_Ship\_new (Id\_new),**

**Date Date, Distance Float, ShipSpeed Float, drLatDeg Int);""")**

**conn.commit()**

*#Load data into DBMS*

**for index, row in FactTrip\_new.iterrows():**

**FactTripId = row.FactTripId**

**DimShipId\_new = row.newDimShipid**

**Date = row.Date2 if not pd.isna(row.Date2) else None**

**Distance = row.Distance if not pd.isna(row.Distance) else None**

**ShipSpeed = row.ShipSpeed if not pd.isna(row.ShipSpeed) else None**

**drLatdeg = row.drLatDeg if not pd.isna(row.drLatDeg) else None**

**cur.execute("""INSERT INTO FactTrip\_new (FactTripid, DimShipId\_new, Date, Distance, ShipSpeed, drLatDeg)**

**values (%s,%s,%s,%s,%s,%s)""",**

**(FactTripId, DimShipId\_new, Date, Distance, ShipSpeed, drLatdeg))**

**conn.commit()**

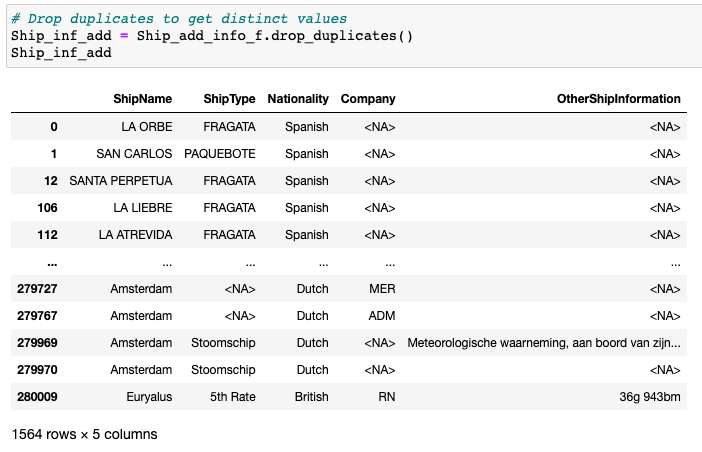
**Screenshots of the executed commands:**

**Table

Description automatically generated with medium confidence**

**Graphical user interface, text, application, email

Description automatically generated**

****

**A picture containing diagram

Description automatically generated**

**Table

Description automatically generated**

**Table

Description automatically generated**

**Table

Description automatically generated**

**Table

Description automatically generatedGraphical user interface, text, application

Description automatically generated**

Graphical user interface, application, table

Description automatically generated

**Table

Description automatically generated**

Use the **Ask the Teaching Team Discussion Forum** if you have any questions regarding the how to approach this assignment.

Save your assignment as ***lastnameFirstname\_assign3\_A.docx*** and submit it in the *Assignments* section of the course.

For help uploading files please refer to the *Technical Support* page in the syllabus.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 supporting explanations. | Good use and integration of appropriate SQL/Python constructs and supporting explanations | Mediocre use and integration of appropriate SQL/Python constructs and supporting explanations | Substandard use and integration of appropriate SQL/Python constructs and supporting explanations | Virtually all SQL/Python constructs and supporting explanations are unsuitable or improperly integrated |  |
|  |  |  |  |  | Assignment Grade: |  |