Student (s) Number as per student card:

10595913	Pavan Seerapu

Course Title: MSc in Data Analytics

Lecturer Name: Clive Gargan

Module/Subject Title: Programming for Data Analysis

Assignment Title: Python ETL Solution Design & Implementation

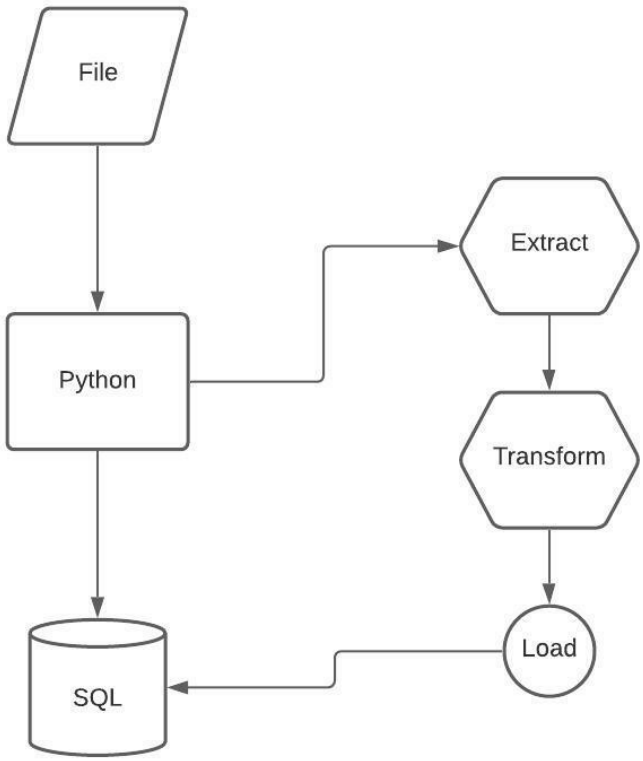
Contents

Scope of the document

ETL is an acronym for Extract, Transform, and Load and its scope is to extract data from a source (file) and perform some operations on the data so that it is free of missing and NULL values, this is called transforming the data, this extraction and transformation happens in one platform but loading of data is on a different platform either inserting all the data we have at the end of the above two phases to a SQL Server and by following this ETL process you can work on cross platforms.

Technical Design to include

ETL Architecture



Pandas operations detailed for each requirement

.read_csv() :

We can read comma separated values and load them into pandas with the help of this function.

.shape :

Provides the dimension of a DataFrame, from which we can calculate the total number of rows and columns.

.count() :

Counts the number of rows in each column, omitting null values.

.isnull().sum() :

By applying, isnull() determines whether or not null values exist. sum() also returns the total number of null values for each column.

.isnull().sum().sum() :

isnull(), as previously explained, by using the sum() function, you can get the total count of each column. Using the sum() method again, you can get the total null values of all the columns.

.groupby().size() :

If we pass a column inside .groupby() it combines similar names into a group and .size() gives the total count of each group inside a column.

.loc[].groupby().size() :

loc[] is used to access rows by column name; we may also put conditions within to access rows that match certain conditions, such as. loc[(df['Description'] == 'Fire CAR') | loc['Description'] == 'Fire ALARM')] and .sortby (). size is used to calculate the total number of each group supplied as a parameter.

.replace() :

This is used to replace any character or string with another character or string.

.dropna() :

This deletes rows with Null/NaN values, and we can also add a criteria such as subset = ['column name'], which just checks these specific columns and removes any matches.

.drop_duplicates() :

This eliminates several instances of a certain row and includes a condition to keep the first occurrence.

.to_datetime() :

Converts the time into a correct format and returns it as a Series.

.min() :

Determines the minimum value of a column.

.to_pydatetime() :

Because we already converted using .to_datetime(), this function can only be applied to Series datatypes and returns an array. We performed all conversions in this manner because numerical operations are easier to apply to array datatypes.

pypyodbc.drivers() :

This method checks your local system for available drivers.

pypyodbc.connect() :

Setting up a connection between a SQL Server and Python. Furthermore, we pass parameters to this procedure, one of which is the server, which we discovered by using the pypyodbc.drivers() function, and the other is the database, which is to be physically created on SQL Management Studio.

.cursor() :

The cursor function helps in referring to the newly generated database.

Table creation and Table Insertion:

The table create command with data description can be placed in a single variable inside quotation marks (""), such as 'CREATE TABLE tablename (variable1 datatype1, variable2 datatype2)'. Similarly, inserting goes as follows:'INSERT INTO tablename VALUES (?,?)

.values().tolist():

Converts the DataFrame values to a list since SQL insertion queries may be executed when the values are in a list or tuples.

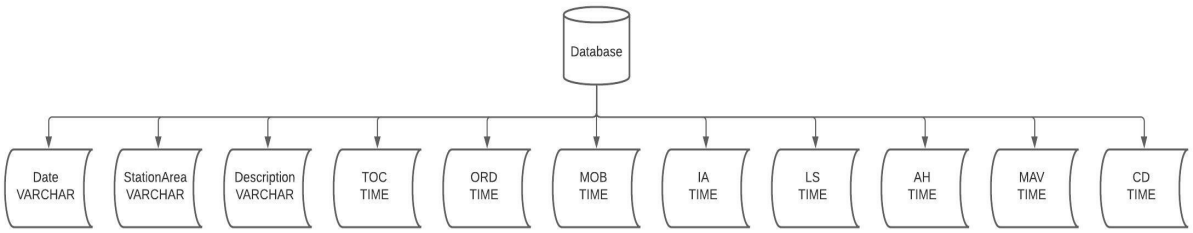
cursor.execute() :

This is used to perform a query and to run insertion queries, which accept two arguments, one for the query and the other for the values to be inserted.

.commit() :

Using the .commit() function, we can save any queries we've run up to this point.

Data Model



- Date (varchar(50), null)
- StationArea (varchar(50), null)
- Description (varchar(50), null)
- TOC (time(7), null)
- ORD (time(7), null)
- MOB (time(7), null)
- IA (time(7), null)
- LS (time(7), null)
- AH (time(7), null)
- MAV (time(7), null)
- CD (time(7), null)

Testing

Testing must be carried out in order to compare the result to the expected output.

To see if data is obtained from the .csv file.

```
Display first 10 rows of the dataframe :
         Date   Station Area   Description  ...        AH       MAV       CD
0  01/01/2013    Rathfarnham     S/S OTHER  ...  02:28:54  02:48:54  03:08:54
1  01/01/2013       Tallaght      Fire CAR  ...  02:25:43  02:45:43  03:05:43
2  01/01/2013   North Strand       S/S RTA  ...  04:11:20  04:31:20  04:51:20
3  01/01/2013       Tallaght      Fire CAR  ...  04:50:48  05:10:48  05:30:48
4  01/01/2013        Finglas  Fire DOM PER  ...  04:50:47  05:10:47  05:30:47
5  01/01/2013       Tallaght     S/S OTHER  ...  05:39:48  05:59:48  06:19:48
6  01/01/2013  Dun Laoghaire    Fire ALARM  ...  05:55:13  06:15:13  06:35:13
7  01/01/2013        Tara St    Fire ALARM  ...  00:30:00  00:50:00  01:10:00
8  01/01/2013     Balbriggan    Fire ALARM  ...  06:40:10  07:00:10  07:20:10
9  01/01/2013     Kilbarrack    Fire ALARM  ...  06:43:47  07:03:47  07:23:47

[10 rows x 11 columns]
```

Shape of the DataFrame or Dimension of the DataFrame.

```
Rows : 38556 columns : 11
```

Counting non null values in each column.

```
Total count of non null rows by each column :
 Date            38556
Station Area     38556
Description      38556
TOC              38556
ORD              38556
MOB              37311
IA               30962
LS               38556
AH                 153
MAV              38530
CD               38530
dtype: int64
```

Counting null values in each column.

```
Total count of null values by each column :
 Date                0
Station Area         0
Description          0
TOC                  0
ORD                  0
MOB               1245
IA                7594
LS                   0
AH               38403
MAV                 26
CD                  26
dtype: int64
```

Summation of NaN values of all columns.

```
Total count of null values of all the columns :  47294
```

Count of each group in Station Area column.

```
Count of call outs by Station Area :
Station Area
Balbriggan         688
Blanchardstown    2089
Dolphins Barn     4018
Donnybrook        1787
Dun Laoghaire     3271
Dunshaughlin         2
Finglas           3030
Kilbarrack        3548
MH14                 1
North Strand      1645
Phibsborough      3160
Rathfarnham       2129
Skerries           449
Swords            2614
Tallaght          6525
Tara St           3600
dtype: int64
```

Count of each group by Date and Station Area.

```
Total number of call outs by Date and Station Area :
Date        Station Area
01/01/2013  Balbriggan       3
            Blanchardstown   1
            Dolphins Barn    4
            Donnybrook       1
            Dun Laoghaire    3
                            ..
31/12/2015  Donnybrook       1
            Dun Laoghaire    1
            Finglas          3
            North Strand     1
            Tallaght         5
Length: 12483, dtype: int64
Station Area  Date
Balbriggan    01/01/2013    3
              01/01/2014    1
              01/04/2013    1
              01/05/2013    1
              01/11/2015    1
                           ..
Tara St       31/07/2015    2
              31/08/2013    1
              31/10/2014    2
              31/10/2015    2
              31/12/2013    1
Length: 6792, dtype: int64
```

After removing null values, replacing spaces and null values with None these are the leftover rows.

```
            Date    Station Area   Description  ...        AH       MAV       CD
0      01/01/2013     Rathfarnham     S/S OTHER  ...  02:28:54  02:48:54  03:08:54
1      01/01/2013        Tallaght      Fire CAR  ...  02:25:43  02:45:43  03:05:43
2      01/01/2013   North Strand       S/S RTA  ...  04:11:20  04:31:20  04:51:20
3      01/01/2013        Tallaght      Fire CAR  ...  04:50:48  05:10:48  05:30:48
4      01/01/2013         Finglas  Fire DOM PER  ...  04:50:47  05:10:47  05:30:47
...           ...             ...           ...  ...       ...       ...       ...
38490  16/10/2015        Tallaght    Fire SMALL  ...  00:38:32  00:20:00  00:40:00
38491  13/08/2015        Tallaght    Fire GRASS  ...  00:38:32  00:20:00  00:40:00
38496  23/06/2015         Finglas    Fire SMALL  ...  00:38:32  00:20:00  00:40:00
38497  23/12/2015   Dun Laoghaire     S/S TREEDN  ...  00:38:32  00:20:00  00:40:00
38500  30/09/2015  Blanchardstown    Fire SMALL  ...  00:38:32  00:20:00  00:40:00

[152 rows x 11 columns]
```

```
The minimum time difference between TOC and ORD:  0:02:32
```

Checking drivers in the local system, we are going to use SQL Server in the program.

```
SQL Server
MySQL ODBC 8.0 ANSI Driver
MySQL ODBC 8.0 Unicode Driver
SQL Server Native Client 11.0
SQL Server Native Client RDA 11.0
ODBC Driver 17 for SQL Server
Microsoft Access Driver (*.mdb, *.accdb)
Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
Microsoft Access Text Driver (*.txt, *.csv)
Microsoft Access Text Driver (*.txt, *.csv)
```

Loading the data is successful as we can see from below image of SQL Management Studio.

	Date	StationArea	Description	TOC	ORD	MOB	IA	LS	AH	MAV	CD
1	01/01/2013	Rathfarnham	S/S OTHER	01:27:19.0000000	01:28:58.0000000	01:30:47.0000000	01:58:54.0000000	02:08:54.0000000	02:28:54.0000000	02:48:54.0000000	03:08:54.0000000
2	01/01/2013	Tallaght	Fire CAR	01:49:57.0000000	01:50:40.0000000	01:51:47.0000000	01:55:43.0000000	02:05:43.0000000	02:25:43.0000000	02:45:43.0000000	03:05:43.0000000
3	01/01/2013	North Strand	S/S RTA	03:35:24.0000000	03:36:14.0000000	03:38:28.0000000	03:41:20.0000000	03:51:20.0000000	04:11:20.0000000	04:31:20.0000000	04:51:20.0000000
4	01/01/2013	Tallaght	Fire CAR	04:12:51.0000000	04:13:56.0000000	04:14:32.0000000	04:20:48.0000000	04:30:48.0000000	04:50:48.0000000	05:10:48.0000000	05:30:48.0000000
5	01/01/2013	Finglas	Fire DOM PER	04:14:24.0000000	04:15:36.0000000	04:17:37.0000000	04:20:47.0000000	04:30:47.0000000	04:50:47.0000000	05:10:47.0000000	05:30:47.0000000
6	01/01/2013	Tallaght	S/S OTHER	04:54:30.0000000	04:59:08.0000000	05:02:07.0000000	05:09:48.0000000	05:19:48.0000000	05:39:48.0000000	05:59:48.0000000	06:19:48.0000000
7	01/01/2013	Dun Laoghaire	Fire ALARM	05:12:03.0000000	05:15:00.0000000	05:17:23.0000000	05:25:13.0000000	05:35:13.0000000	05:55:13.0000000	06:15:13.0000000	06:35:13.0000000
8	01/01/2013	Tara St	Fire ALARM	05:45:45.0000000	05:47:55.0000000	05:51:19.0000000	05:25:13.0000000	00:10:00.0000000	00:30:00.0000000	00:50:00.0000000	01:10:00.0000000
9	01/01/2013	Balbriggan	Fire ALARM	05:59:23.0000000	06:00:13.0000000	06:03:34.0000000	06:10:10.0000000	06:20:10.0000000	06:40:10.0000000	07:00:10.0000000	07:20:10.0000000
10	01/01/2013	Kilbarrack	Fire ALARM	06:07:42.0000000	06:08:51.0000000	06:10:41.0000000	06:13:47.0000000	06:23:47.0000000	06:43:47.0000000	07:03:47.0000000	07:23:47.0000000

To check the whether the data loaded into SQL Server or not, I have retrieved and showed in the below image and you can take a look at the code also.

```
        Date  Station Area  ...                MAV                CD
0 01/01/2013   Rathfarnham  ... 02:48:54.0000000 03:08:54.0000000
1 01/01/2013      Tallaght  ... 02:45:43.0000000 03:05:43.0000000
2 01/01/2013  North Strand  ... 04:31:20.0000000 04:51:20.0000000
3 01/01/2013      Tallaght  ... 05:10:48.0000000 05:30:48.0000000
4 01/01/2013       Finglas  ... 05:10:47.0000000 05:30:47.0000000
```

Reflection on Learning

Throughout the task, I did not struggle to apply principles learned in class about the Python language, numpy, and pandas, but I spent more time studying SQL, which I could have done in less time if this had been thought of or if I had known this ability previously. Overall, the ETL method allows us to collaborate across platforms.

References

- https://devdocs.io/
- https://pandas.pydata.org/docs/pandas.pdf
- https://numpy.org/doc/stable/
- DBS Moodle