# Data Handling

## Virtual environment Setup

```
pip install virtualenv
```

```
python -m virtualenv env
```

## Active virtual environment

```
cd env/Scripts
activate
cd ..
cd ..
cd BI
```

## Install Jupyter Notebook

```
pip install jupyter
```

## Install Pandas

```
Pip install pandas
```

## Install openpyxl Engine

```
pip install openpyxl
```

# Data Loading

## Open Jupyter Notebook

```
Jupyter-notebook
```

## Download Dataset:

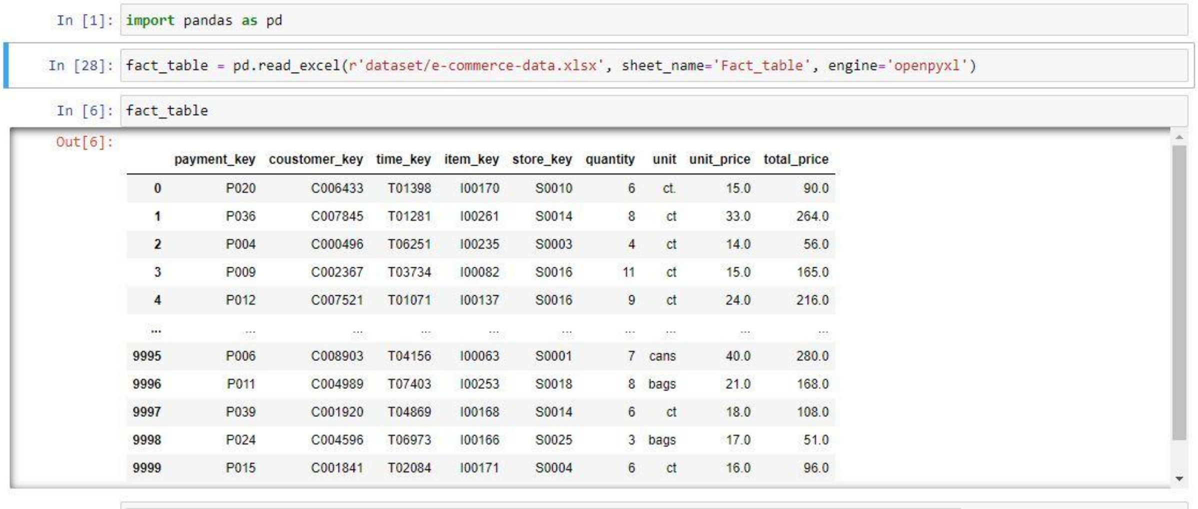
https://docs.google.com/spreadsheets/d/1sFPtZwMPO6aeK_sDmXuXwTK-lv48kPja/edit?usp=sharing&ouid=111861362152580698478&rtpof=true&sd=true

## Download Code:

https://drive.google.com/file/d/16A0VcHpNd4IRsdfVoQnaw1XJM54rZWce/view?usp=sharing

## Example:1

Load the dataset using openxyl python library



```
In [1]: import pandas as pd
```

```
In [28]: fact_table = pd.read_excel(r'dataset/e-commerce-data.xlsx', sheet_name='Fact_table', engine='openpyxl')
```

```
In [6]: fact_table
```

Out[6]:

| | payment_key | coustomer_key | time_key | item_key | store_key | quantity | unit | unit_price | total_price |
|---|---|---|---|---|---|---|---|---|---|
| 0 | P020 | C006433 | T01398 | I00170 | S0010 | 6 | ct. | 15.0 | 90.0 |
| 1 | P036 | C007845 | T01281 | I00261 | S0014 | 8 | ct | 33.0 | 264.0 |
| 2 | P004 | C000496 | T06251 | I00235 | S0003 | 4 | ct | 14.0 | 56.0 |
| 3 | P009 | C002367 | T03734 | I00082 | S0016 | 11 | ct | 15.0 | 165.0 |
| 4 | P012 | C007521 | T01071 | I00137 | S0016 | 9 | ct | 24.0 | 216.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9995 | P006 | C008903 | T04156 | I00063 | S0001 | 7 | cans | 40.0 | 280.0 |
| 9996 | P011 | C004989 | T07403 | I00253 | S0018 | 8 | bags | 21.0 | 168.0 |
| 9997 | P039 | C001920 | T04869 | I00168 | S0014 | 6 | ct | 18.0 | 108.0 |
| 9998 | P024 | C004596 | T06973 | I00166 | S0025 | 3 | bags | 17.0 | 51.0 |
| 9999 | P015 | C001841 | T02084 | I00171 | S0004 | 6 | ct | 16.0 | 96.0 |

**Practice problem 1.1**
Load the tables (item_dim, customer_dim, time_dim, store_dim) from the e-commerce dataset.

# Data Preprocessing

## What is Data Preprocessing?

It is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources, it is collected in raw format, which is not feasible for analysis. Therefore, certain steps are executed to convert the data into a small clean data set. This technique is performed before the execution of the Iterative Analysis. The set of steps is known as Data Preprocessing. It includes -Data Transformation and Data Integration

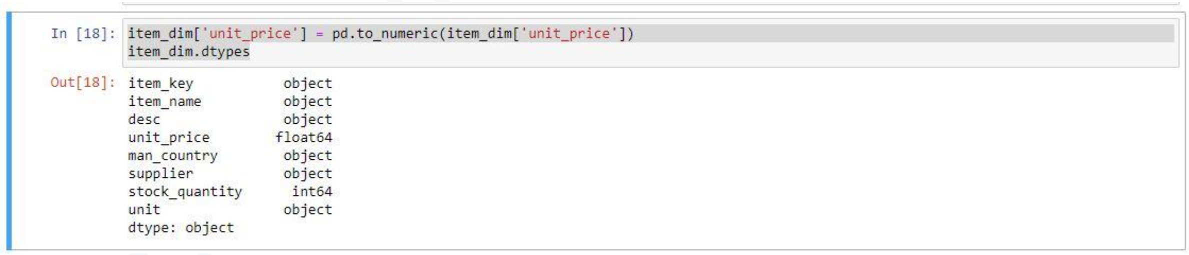
## Data Integration in Data Warehousing

Data integration is one of the significant aspects of Data Warehousing. At the highest level, if we talk about Data Warehousing, it is nothing but the innovation, manipulation, and mapping practices to match the correct set of requested data with the data to be forwarded as a response to the end-user. ETL(Extract, Transform and Load) is a significant data integration component in data warehousing.

## Example-2:

```
item_dim['unit_price'] = pd.to_numeric(item_dim['unit_price'])
Item_dim.dtypes
```

**Output:**



```
In [18]: item_dim['unit_price'] = pd.to_numeric(item_dim['unit_price'])
         item_dim.dtypes

Out[18]: item_key             object
         item_name            object
         desc                 object
         unit_price          float64
         man_country          object
         supplier             object
         stock_quantity        int64
         unit                 object
         dtype: object
```

**Practice problem 2.1**

1. Change the date from time_dim to pandas date-time series format, and unit_price, total_price from fact_table to pandas numeric format.
2. Check the data types of the fact_table.

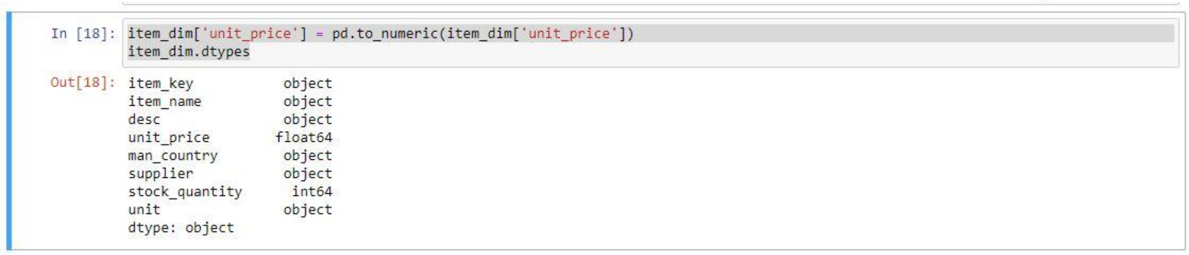
## Data Reduction

Data reduction is the process of reducing the amount of capacity required to store data. Data reduction can increase storage efficiency and reduce costs. Storage vendors will often describe storage capacity in terms of raw capacity and effective capacity, which refers to data after the reduction.

## Example-3:

visualize the first 10 data of the fact table.

```
fact_table.head(10)
```



```
In [18]: item_dim['unit_price'] = pd.to_numeric(item_dim['unit_price'])
         item_dim.dtypes

Out[18]: item_key          object
         item_name         object
         desc              object
         unit_price       float64
         man_country       object
         supplier          object
         stock_quantity     int64
         unit              object
         dtype: object
```

## Practice Problem 3.1

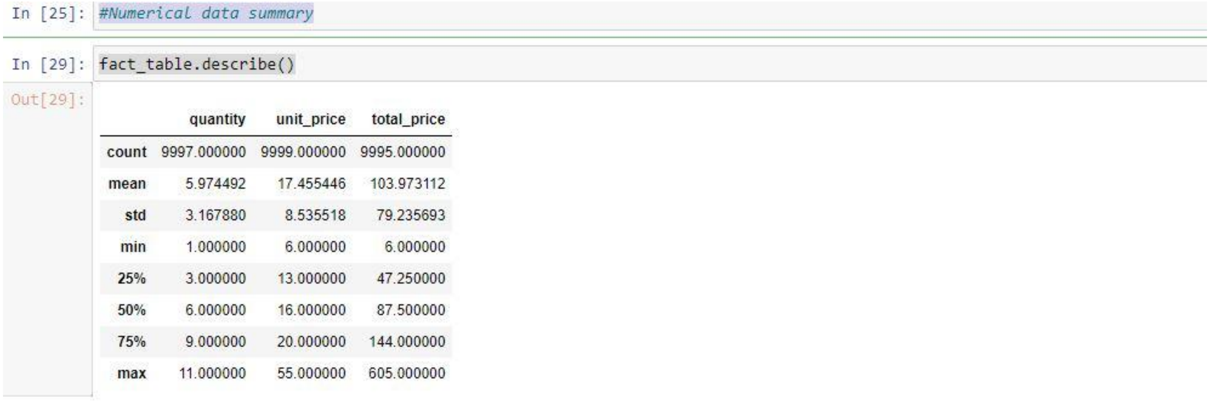
Visualize the last 10 data of the fact table.

## Data Cleaning

- Filling the missing values manually - This is one of the best-chosen methods of Data Preparation process. But there is one limitation that when there are large data set, and missing values are significant then, this approach is not efficient as it becomes a time-consuming task.
- Filling using computed values - The missing values can also be occupied by computing mean, mode or median of the observed given values. Another method could be the predictive values in Preprocessing of Data is that are computed by using any Machine Learning or Deep Learning tools and algorithms. But one drawback of this approach is that it can generate bias within the data as the calculated values are not accurate concerning the observed values.

### Numerical data summary

```
fact_table.describe()
```



```
In [25]: #Numerical data summary

In [29]: fact_table.describe()
Out[29]:
```

|  | quantity | unit_price | total_price |
|---|---|---|---|
| count | 9997.000000 | 9999.000000 | 9995.000000 |
| mean | 5.974492 | 17.455446 | 103.973112 |
| std | 3.167880 | 8.535518 | 79.235693 |
| min | 1.000000 | 6.000000 | 6.000000 |
| 25% | 3.000000 | 13.000000 | 47.250000 |
| 50% | 6.000000 | 16.000000 | 87.500000 |
| 75% | 9.000000 | 20.000000 | 144.000000 |
| max | 11.000000 | 55.000000 | 605.000000 |

**Missing data can have a severe impact on building predictive models because the missing values might contain some vital information that could help in making better predictions. So, it becomes imperative to carry out missing data imputation.**

## Example 4:

### Check missing values

```
fact_table.apply(lambda x: sum(x.isnull()))
```

### Unique value type count

```
fact_table['unit'].value_counts()
```

### Fill in the missing value using the top value count of unit

```
fact_table.unit = fact_table.unit.fillna('ct')
```

## Practice problem 4.1

Fill in the unit_price and total_price missing values of fact_table

## What Is Data Wrangling?

Data Wrangling is a technique that is executed at the time of making an interactive model. In other words, it is used to convert the raw data into a format that is convenient for the consumption of data. This technique is also known as Data Munging. This method also follows certain steps such as extracting the data from different data sources, sorting data using certain algorithms are performed, decomposing the data into a different structured format, and finally storing the data in another database.

Pandas Framework of Python is used for Data Wrangling. The process like data fillter, remove, etc.

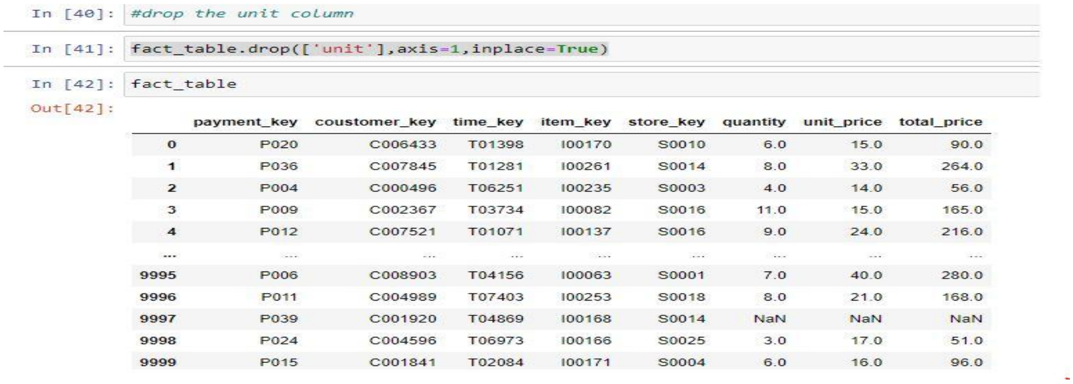
## Data wrangling in python deals with the below functionalities:

1. **Data exploration:**
   In this process, the data is studied, analyzed, and understood by visualizing representations of data.
2. **Dealing with missing values:**
   Most of the datasets having a vast amount of data contain missing values of NaN, they are needed to be taken care of by replacing them with mean, mode, the most frequent value of the column, or simply by dropping the row having a NaN value.
3. **Filtering data:** Some times datasets are comprised of unwanted rows or columns which are required to be removed or filtered

## Example 5:

Drop unit column from the fact_table

```
fact_table.drop(['unit'],axis=1,inplace=True)
```



```
In [40]: #drop the unit column

In [41]: fact_table.drop(['unit'],axis=1,inplace=True)

In [42]: fact_table
Out[42]:
```

| | payment_key | coustomer_key | time_key | item_key | store_key | quantity | unit_price | total_price |
|---|---|---|---|---|---|---|---|---|
| 0 | P020 | C006433 | T01398 | I00170 | S0010 | 6.0 | 15.0 | 90.0 |
| 1 | P036 | C007845 | T01281 | I00261 | S0014 | 8.0 | 33.0 | 264.0 |
| 2 | P004 | C000496 | T06251 | I00235 | S0003 | 4.0 | 14.0 | 56.0 |
| 3 | P009 | C002367 | T03734 | I00082 | S0016 | 11.0 | 15.0 | 165.0 |
| 4 | P012 | C007521 | T01071 | I00137 | S0016 | 9.0 | 24.0 | 216.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9995 | P006 | C008903 | T04156 | I00063 | S0001 | 7.0 | 40.0 | 280.0 |
| 9996 | P011 | C004989 | T07403 | I00253 | S0018 | 8.0 | 21.0 | 168.0 |
| 9997 | P039 | C001920 | T04869 | I00168 | S0014 | NaN | NaN | NaN |
| 9998 | P024 | C004596 | T06973 | I00166 | S0025 | 3.0 | 17.0 | 51.0 |
| 9999 | P015 | C001841 | T02084 | I00171 | S0004 | 6.0 | 16.0 | 96.0 |

**Practice Problem 5.1**

Drop the unit_price column from the fact_table

## Save the new file to csv,

## Example -6:

Export the data into csv

```
fact_table.to_csv("dataset/csv/fact_table.csv",index=False)
```

****Must create a folder into the dataset folder. The folder name should be named “csv”.**