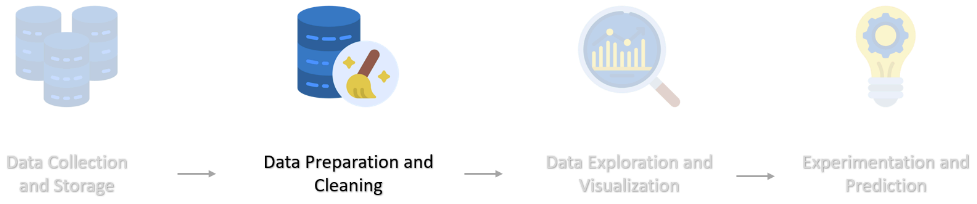
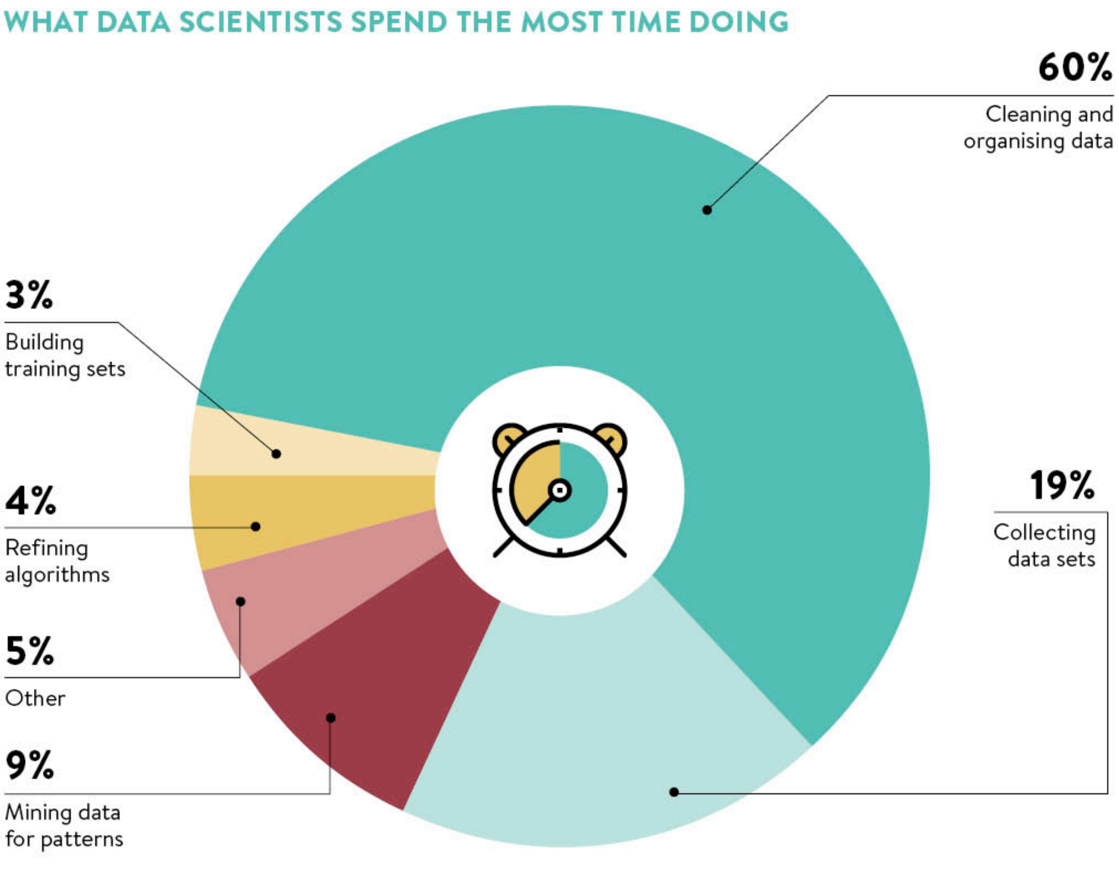
Data cleaning and preparation

Data Science Workflow



Data Collection and Storage → Data Preparation and Cleaning → Data Exploration and Visualization → Experimentation and Prediction



Source: CrowdFlower 2016

- **Pandas**: provide a high-level, flexible, and fast set of tools for cleaning the data

- **Cleaning tasks**:
  1. Validating data types
  2. Inconsistency
  3. Duplicates
  4. Missing data
  5. Extreme values / Outliers

The dataset we will work on:

In [118…

```
import pandas as pd
import numpy as np
df = pd.read_csv("data/online_retail_II 2_noisy.csv")
df
```

```
C:\Users\asabb\AppData\Local\Temp\ipykernel_19756\1257127318.py:3: DtypeWarning: Col
umns (3,4) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("data/online_retail_II 2_noisy.csv")
```

Out[118…

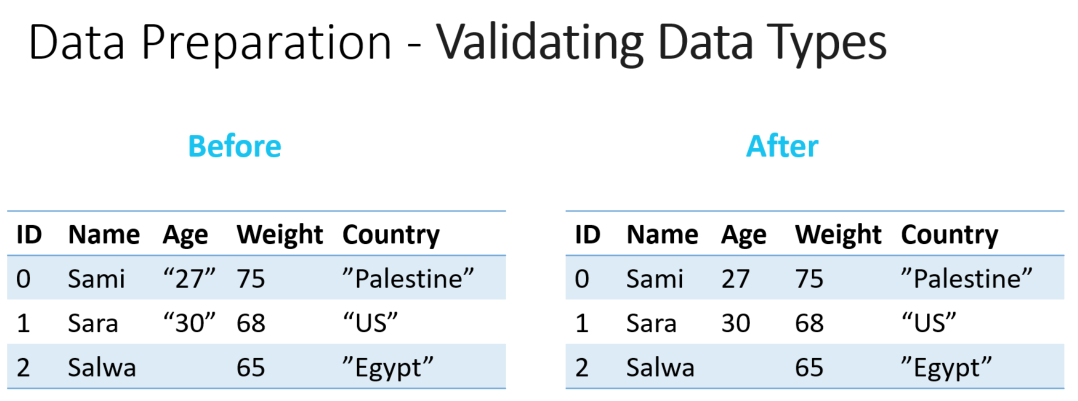
	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	Invoic
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	6.95	83.40	13.344	96.744	12/1
1	489434	79323P	PINK CHERRY LIGHTS	12	6.75	81.00	12.960	93.960	12/1
2	489434	79323W	WHITE CHERRY LIGHTS	12	6.75	81.00	12.960	93.960	12/1
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2.1	100.80	16.128	116.928	12/1
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	1.25	30.00	4.800	34.800	12/1
...	...	...	...	...	...	...	...	...	
805547	581587	22899	CHILDREN'S APRON DOLLY GIRL	6.0	2.1	12.60	2.016	14.616	12/9
805548	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4.0	4.15	16.60	2.656	19.256	12/9
805549	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4.0	4.15	16.60	2.656	19.256	12/9
805550	581587	22138	BAKING SET 9 PIECE RETROSPOT	3.0	4.95	14.85	2.376	17.226	12/9
805551	581587	POST	POSTAGE	1.0	18.0	18.00	2.880	20.880	12/9

805552 rows × 12 columns

To do data cleaning properly, You should be familiar with:

- Business Domain
- Context how data is being collected
- Requirements
- Data dictionary

1. Validate data types



Data Preparation - Validating Data Types

Before

ID	Name	Age	Weight	Country
0	Sami	“27”	75	”Palestine”
1	Sara	“30”	68	“US”
2	Salwa		65	”Egypt”

After

ID	Name	Age	Weight	Country
0	Sami	27	75	”Palestine”
1	Sara	30	68	“US”
2	Salwa		65	”Egypt”

- Data types define the **kind of data** that can be stored in each column of your dataset
- **integers, floats, strings, and dates**
- Ensuring that each column has the correct data type is vital for **accurate analysis and functioning of algorithm**
- Determine type based on your **understanding of the dataset**
- **If discrepancies found -> convert**

In [119…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 805552 entries, 0 to 805551
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Invoice         805552 non-null  int64  
 1   StockCode       805526 non-null  object 
 2   Description     805526 non-null  object 
 3   Quantity        800989 non-null  object 
 4   Price           792162 non-null  object 
 5   Total           805552 non-null  float64
 6   Tax 16%         805552 non-null  float64
 7   Gross           805552 non-null  float64
 8   InvoiceDate     805552 non-null  object 
 9   Payment Method  805226 non-null  object 
 10  Customer ID     804307 non-null  float64
 11  Country         805552 non-null  object 
dtypes: float64(4), int64(1), object(7)
memory usage: 73.8+ MB
```

Convert *Quantity* to float/int

In [120…

```python
pattern = r'\d+(\.\d+)?$'


df2=df[(df['Quantity'].notna()) & (~df['Quantity'].astype(str).str.match(pattern))]
df2
```

Out[120…

	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	InvoiceDat
13	489436	21755	LOVE BUILDING BLOCK WORD	"18	5.45	98.10	15.6960	113.7960	12/1/200 9:0
15	489436	84879	ASSORTED COLOUR BIRD ORNAMENT	"16	1.69	27.04	4.3264	31.3664	12/1/200 9:0
34	489437	21364	PEACE SMALL WOOD LETTERS	"2	6.75	13.50	2.1600	15.6600	12/1/200 9:0
48	489437	22271	FELTCRAFT DOLL ROSIE	"6	2.95	17.70	2.8320	20.5320	12/1/200 9:0
447	489522	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	"1	3.75	3.75	0.6000	4.3500	12/1/200 11:4

In [121…

```
df['Quantity'] = df['Quantity'].astype(str).str.replace('"', '')

df['Quantity'] = df['Quantity'].astype(float)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 805552 entries, 0 to 805551
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Invoice         805552 non-null  int64  
 1   StockCode       805526 non-null  object 
 2   Description     805526 non-null  object 
 3   Quantity        800989 non-null  float64
 4   Price           792162 non-null  object 
 5   Total           805552 non-null  float64
 6   Tax 16%         805552 non-null  float64
 7   Gross           805552 non-null  float64
 8   InvoiceDate     805552 non-null  object 
 9   Payment Method  805226 non-null  object 
 10  Customer ID     804307 non-null  float64
 11  Country         805552 non-null  object 
dtypes: float64(5), int64(1), object(6)
memory usage: 73.8+ MB
```

Convert *Price* to float

In [122…

```
pattern = r'\d+(\.\d+)?$'

df[(df['Price'].notna()) & (~df['Price'].astype(str).str.match(pattern))]
```

Out[122…

	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	InvoiceDat
262288	522622	20966	SANDWICH BATH SPONGE	10.0	1,25	12.5	2.000	14.500	9/15/201 15:3
490597	547066	21080	SET/20 RED RETROSPOT PAPER NAPKINS	12.0	0.85$	10.2	1.632	11.832	3/20/201 13:4

In [123…

```
df['Price'] = df['Price'].astype(str).str.replace(',', '.',regex=False).str.replace
df['Price'] = df['Price'].astype(float)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 805552 entries, 0 to 805551
Data columns (total 12 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   Invoice         805552 non-null  int64  
 1   StockCode       805526 non-null  object 
 2   Description     805526 non-null  object 
 3   Quantity        800989 non-null  float64
 4   Price           792162 non-null  float64
 5   Total           805552 non-null  float64
 6   Tax 16%         805552 non-null  float64
 7   Gross           805552 non-null  float64
 8   InvoiceDate     805552 non-null  object 
 9   Payment Method  805226 non-null  object 
 10  Customer ID     804307 non-null  float64
 11  Country         805552 non-null  object 
dtypes: float64(6), int64(1), object(5)
memory usage: 73.8+ MB
```

convert *customer type* to string

In [124…

```
## Customer
df['Customer ID'] = df['Customer ID'].fillna(-1)
df['Customer ID'] = df['Customer ID'].astype(int).astype(str)
df['Customer ID'] = df['Customer ID'].replace("-1", np.nan)
df['Customer ID'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 805552 entries, 0 to 805551
Series name: Customer ID
Non-Null Count   Dtype 
--------------   ----- 
804307 non-null  object
dtypes: object(1)
memory usage: 6.1+ MB
```

Convert *Invoice_Date* to date time

In [125…

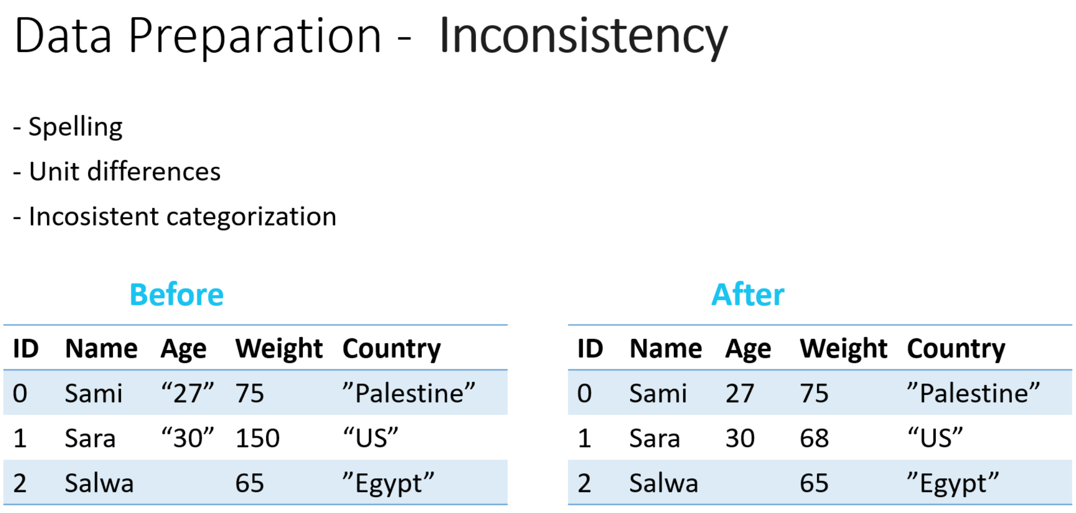
```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'], format='%m/%d/%Y %H:%M')
df
```

Out[125…

	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	Invoic
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12.0	6.95	83.40	13.344	96.744	2009-07
1	489434	79323P	PINK CHERRY LIGHTS	12.0	6.75	81.00	12.960	93.960	2009-07
2	489434	79323W	WHITE CHERRY LIGHTS	12.0	6.75	81.00	12.960	93.960	2009-07
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48.0	2.10	100.80	16.128	116.928	2009-07
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24.0	1.25	30.00	4.800	34.800	2009-07
...	...	...	...	...	...	...	...	...	
805547	581587	22899	CHILDREN'S APRON DOLLY GIRL	6.0	2.10	12.60	2.016	14.616	2011-12
805548	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4.0	4.15	16.60	2.656	19.256	2011-12
805549	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4.0	4.15	16.60	2.656	19.256	2011-12
805550	581587	22138	BAKING SET 9 PIECE RETROSPOT	3.0	4.95	14.85	2.376	17.226	2011-12
805551	581587	POST	POSTAGE	1.0	18.00	18.00	2.880	20.880	2011-12

805552 rows × 12 columns

2. Data inconsistency



Data Preparation - Inconsistency

- Spelling
- Unit differences
- Incosistent categorization

Before

ID	Name	Age	Weight	Country
0	Sami	“27”	75	”Palestine”
1	Sara	“30”	150	“US”
2	Salwa		65	”Egypt”

After

ID	Name	Age	Weight	Country
0	Sami	27	75	”Palestine”
1	Sara	30	68	“US”
2	Salwa		65	”Egypt”

Data inconsistencies occur when similar data is recorded in **different formats** or **representations**, leading to **unreliable analysis**

The popular approach in finding inconsistency is using **value_counts()** function

Find inconsistencies in *Payment Method*

In [126…

```
df['Payment Method'].value_counts()
```

Out[126…

```
Payment Method
Credit Card    804002
Cash             1193
credit card        23
CC                  7
cash                1
Name: count, dtype: int64
```

In [127…

```
df['Payment Method'] = df['Payment Method'].replace({"CC": "Credit Card", "credit c
df['Payment Method'].value_counts()
```

Out[127…

```
Payment Method
Credit Card    804032
Cash             1194
Name: count, dtype: int64
```

Find inconsistencies in *Country*

In [128…

```
df['Country'].value_counts()
```

Out[128…

```
Country
United Kingdom          725242
Germany                  16694
EIRE                     15743
France                   13813
Netherlands               5088
Spain                     3719
Belgium                   3068
Switzerland               3011
Portugal                  2446
Australia                 1810
Channel Islands           1569
Italy                     1468
Norway                    1436
Sweden                    1319
Cyprus                    1155
Finland                   1032
Austria                    922
Denmark                    798
Greece                     657
Unspecified                521
Poland                     512
Japan                      485
United Arab Emirates       383
USA                        373
Singapore                  339
Israel                     322
Malta                      282
Iceland                    253
Canada                     228
Lithuania                  189
RSA                        122
Brazil                      94
Thailand                    76
European Community          60
Bahrain                     59
West Indies                 54
Korea                       53
Lebanon                     45
United States               36
Nigeria                     30
Czech Republic              25
UK                          10
Saudi Arabia                 9
AUS                          2
Name: count, dtype: int64
```

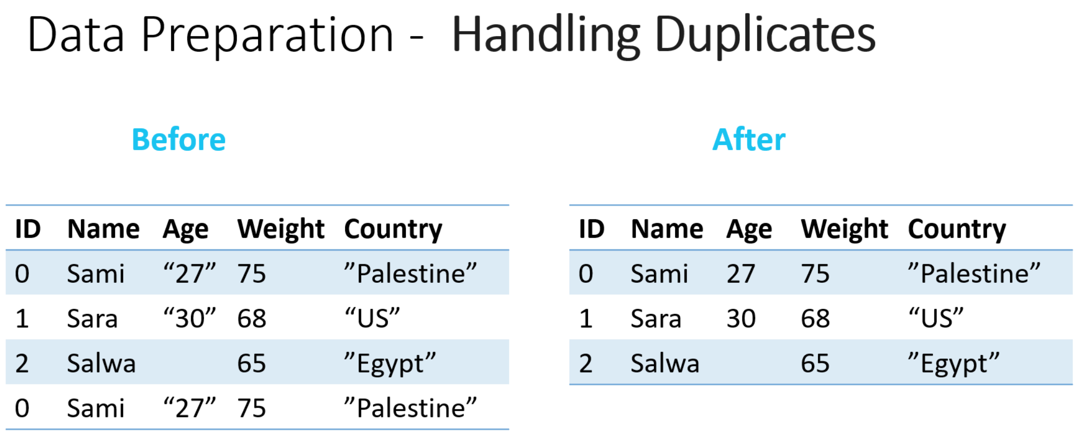
In [129…

```python
df['Country'] = df['Country'].replace({"UK": "United Kingdom", "United States": "US

df['Country'].value_counts()
```

```
Out[129…
Country
United Kingdom          725252
Germany                  16694
EIRE                     15743
France                   13813
Netherlands               5088
Spain                     3719
Belgium                   3068
Switzerland               3011
Portugal                  2446
Australia                 1812
Channel Islands           1569
Italy                     1468
Norway                    1436
Sweden                    1319
Cyprus                    1155
Finland                   1032
Austria                    922
Denmark                    798
Greece                     657
Unspecified                521
Poland                     512
Japan                      485
USA                        409
United Arab Emirates       383
Singapore                  339
Israel                     322
Malta                      282
Iceland                    253
Canada                     228
Lithuania                  189
RSA                        122
Brazil                      94
Thailand                    76
European Community          60
Bahrain                     59
West Indies                 54
Korea                       53
Lebanon                     45
Nigeria                     30
Czech Republic              25
Saudi Arabia                 9
Name: count, dtype: int64
```

3. Duplicates



Data Preparation - Handling Duplicates

Before

ID	Name	Age	Weight	Country
0	Sami	“27”	75	”Palestine”
1	Sara	“30”	68	“US”
2	Salwa		65	”Egypt”
0	Sami	“27”	75	”Palestine”

After

ID	Name	Age	Weight	Country
0	Sami	27	75	”Palestine”
1	Sara	30	68	“US”
2	Salwa		65	”Egypt”

- Duplicates can lead to **incorrect analysis**
- Duplicates arise:
    - Repeated data entry
    - Mergeing datasets
    - Data Collection errors

Check for diplicates

In [130…

```
df[df.duplicated()]
```

Out[130…

	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	InvoiceD
342	489517	21912	VINTAGE SNAKES & LADDERS	1.0	3.75	3.75	0.6000	4.3500	2009-12 11:34
354	489517	22130	PARTY CONE CHRISTMAS DECORATION	6.0	0.85	5.10	0.8160	5.9160	2009-12 11:34
355	489517	22319	HAIRCLIPS FORTIES FABRIC ASSORTED	12.0	0.65	7.80	1.2480	9.0480	2009-12 11:34
356	489517	21913	VINTAGE SEASIDE JIGSAW PUZZLES	1.0	3.75	3.75	0.6000	4.3500	2009-12 11:34
357	489517	21821	GLITTER STAR GARLAND WITH BELLS	1.0	3.75	3.75	0.6000	4.3500	2009-12 11:34
...	...	...	...	...	...	...	...	...	
805320	581538	22068	BLACK PIRATE TREASURE CHEST	1.0	0.39	0.39	0.0624	0.4524	2011-12 11:34
805334	581538	23318	BOX OF 6 MINI VINTAGE CRACKERS	1.0	2.49	2.49	0.3984	2.8884	2011-12 11:34
805337	581538	22992	REVOLVER WOODEN RULER	1.0	1.95	1.95	0.3120	2.2620	2011-12 11:34
805344	581538	22694	WICKER STAR	1.0	2.10	2.10	0.3360	2.4360	2011-12 11:34
805346	581538	23343	JUMBO BAG VINTAGE CHRISTMAS	1.0	2.08	2.08	0.3328	2.4128	2011-12 11:34

25108 rows × 12 columns

Drop duplicates

In [131…

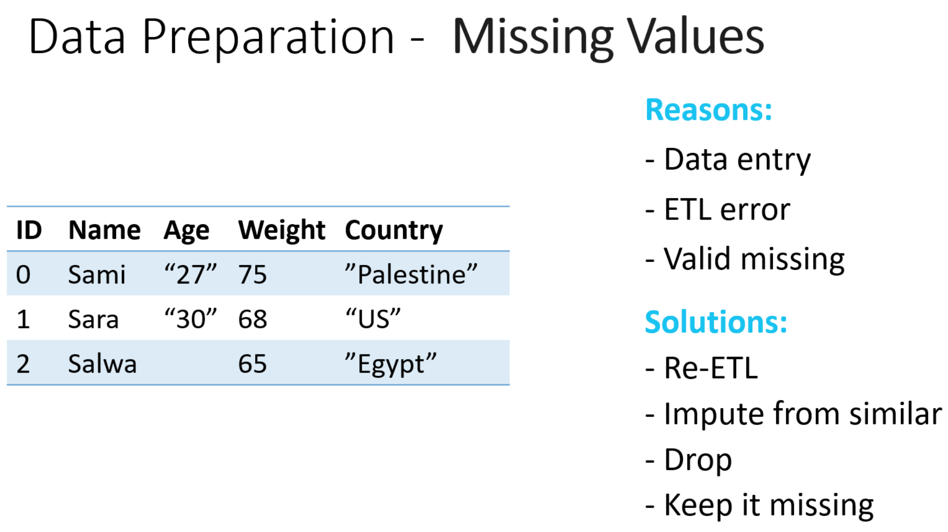
```
df = df.drop_duplicates()
df
```

Out[131…

	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	Invoic
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12.0	6.95	83.40	13.344	96.744	2009-07
1	489434	79323P	PINK CHERRY LIGHTS	12.0	6.75	81.00	12.960	93.960	2009-07
2	489434	79323W	WHITE CHERRY LIGHTS	12.0	6.75	81.00	12.960	93.960	2009-07
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48.0	2.10	100.80	16.128	116.928	2009-07
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24.0	1.25	30.00	4.800	34.800	2009-07
...	...	...	...	...	...	...	...	...	
805547	581587	22899	CHILDREN'S APRON DOLLY GIRL	6.0	2.10	12.60	2.016	14.616	2011-12
805548	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4.0	4.15	16.60	2.656	19.256	2011-12
805549	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4.0	4.15	16.60	2.656	19.256	2011-12
805550	581587	22138	BAKING SET 9 PIECE RETROSPOT	3.0	4.95	14.85	2.376	17.226	2011-12
805551	581587	POST	POSTAGE	1.0	18.00	18.00	2.880	20.880	2011-12

780444 rows × 12 columns

Missing Values



Data Preparation - Missing Values

ID	Name	Age	Weight	Country
0	Sami	“27”	75	”Palestine”
1	Sara	“30”	68	“US”
2	Salwa		65	”Egypt”

Reasons:

- Data entry
- ETL error
- Valid missing

Solutions:

- Re-ETL
- Impute from similar
- Drop
- Keep it missing

Check missing values

```
In [132…  df.isna().sum()
```

```
Out[132…  Invoice               0
          StockCode            26
          Description          26
          Quantity           4562
          Price             13386
          Total                 0
          Tax 16%               0
          Gross                 0
          InvoiceDate           0
          Payment Method      326
          Customer ID        1245
          Country               0
          dtype: int64
```

Process for handling missing values:

1. **Understand missing data**: revist the data source and understand why the data is missing

2. **Calculate from the data itself**

3. **Impute**
    - From similar data points
    - constant value (mean, most frequent, 0,..)
    - bfill, ffill (time series data)
    - ML models (knn)

4. **Drop**

Example

In [133…

```
import numpy as np
import pandas as pd

# Create dataframe
import numpy as np
data = pd.DataFrame(np.random.standard_normal((7, 3)))
data.iloc[2:3, 1] = np.nan
data.iloc[4:5, 1] = np.nan
data.iloc[4:5, 2] = np.nan
data.iloc[0,1]= np.nan
data.iloc[-1,2]= np.nan
data
```

Out[133…

	0	1	2
0	0.071148	NaN	-0.690006
1	-0.740496	-2.121285	0.870374
2	-0.563426	NaN	-0.598694
3	-0.381680	1.630362	0.908357
4	1.015674	NaN	NaN
5	1.664753	0.029204	-0.509784
6	-0.752856	0.869271	NaN

In [134…

```
# Fill with constant value
data.fillna(0)
```

Out[134...

	0	1	2
0	0.071148	0.000000	-0.690006
1	-0.740496	-2.121285	0.870374
2	-0.563426	0.000000	-0.598694
3	-0.381680	1.630362	0.908357
4	1.015674	0.000000	0.000000
5	1.664753	0.029204	-0.509784
6	-0.752856	0.869271	0.000000

In [135...

```
# Fill with mean/median/max/min...
data.fillna(data.mean())
```

Out[135...

	0	1	2
0	0.071148	0.101888	-0.690006
1	-0.740496	-2.121285	0.870374
2	-0.563426	0.101888	-0.598694
3	-0.381680	1.630362	0.908357
4	1.015674	0.101888	-0.003951
5	1.664753	0.029204	-0.509784
6	-0.752856	0.869271	-0.003951

In [136...

```
# ‘forward fill’: propagate last valid observation forward
data.ffill()
```

Out[136...

	0	1	2
0	0.071148	NaN	-0.690006
1	-0.740496	-2.121285	0.870374
2	-0.563426	-2.121285	-0.598694
3	-0.381680	1.630362	0.908357
4	1.015674	1.630362	0.908357
5	1.664753	0.029204	-0.509784
6	-0.752856	0.869271	-0.509784

In [137...

```
# backward fill
data.bfill()
```

Out[137…

	0	1	2
0	0.071148	-2.121285	-0.690006
1	-0.740496	-2.121285	0.870374
2	-0.563426	1.630362	-0.598694
3	-0.381680	1.630362	0.908357
4	1.015674	0.029204	-0.509784
5	1.664753	0.029204	-0.509784
6	-0.752856	0.869271	NaN

In [138…

```
# Drop all rows that have any missing value
data.dropna()
```

Out[138…

	0	1	2
1	-0.740496	-2.121285	0.870374
3	-0.381680	1.630362	0.908357
5	1.664753	0.029204	-0.509784

Back to Retail dataset

In [139…

```
df.isna().sum()
```

Out[139…

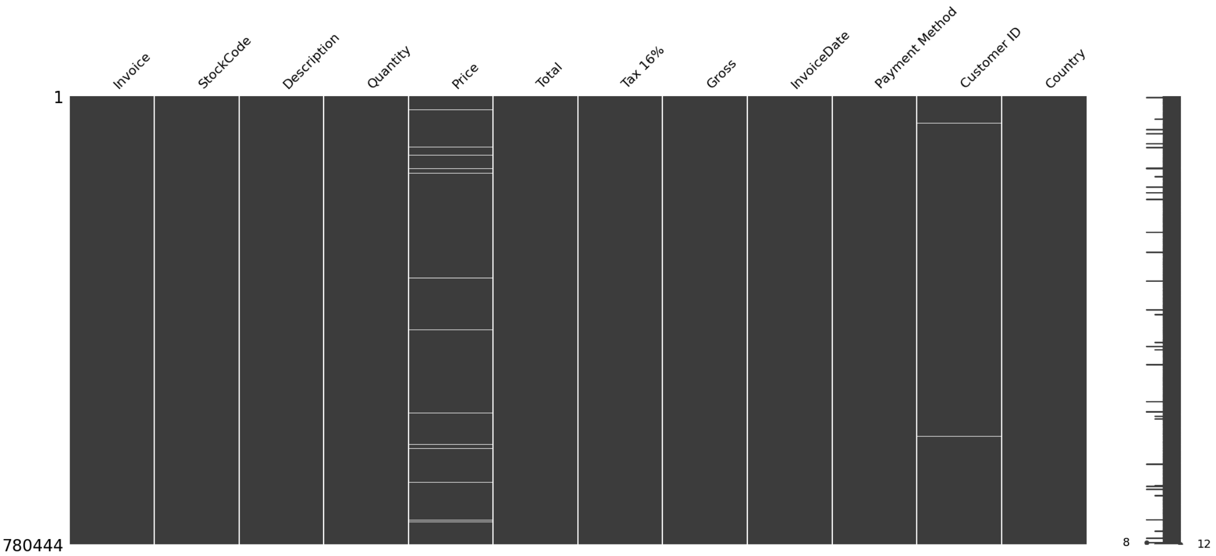
```
Invoice              0
StockCode           26
Description         26
Quantity          4562
Price            13386
Total                0
Tax 16%              0
Gross                0
InvoiceDate          0
Payment Method     326
Customer ID       1245
Country              0
dtype: int64
```

In [140…

```
#!pip install missingno
import missingno as msno
msno.matrix(df.sort_values(by="Payment Method"))
#print(df[df['Quantity'].isna()]['Payment Method'].value_counts(normalize=True))
#df[df['Quantity'].notna()]['Payment Method'].value_counts(normalize=True)
```

Out[140…

```
<Axes: >
```



Handle Price

In [141…

```
df[df['Price'].isna()]
```

Out[141…

	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	Invoice
11	489435	22353	LUNCHBOX WITH CUTLERY FAIRY CAKES	NaN	NaN	30.60	4.8960	35.4960	2009-1 07:4
47	489437	20971	PINK BLUE FELT CRAFT TRINKET BOX	NaN	NaN	15.00	2.4000	17.4000	2009-1 09:0
52	489437	22111	SCOTTIE DOG HOT WATER BOTTLE	NaN	NaN	14.85	2.3760	17.2260	2009-1 09:0
60	489438	21411	GINGHAM HEART DOORSTOP RED	32.0	NaN	80.00	12.8000	92.8000	2009-1 09:2
77	489439	16161P	WRAP ENGLISH ROSE	25.0	NaN	10.50	1.6800	12.1800	2009-1 09:2
...	...	...	...	...	...	...	...	...	
805154	581496	22112	CHOCOLATE HOT WATER BOTTLE	NaN	NaN	29.70	4.7520	34.4520	2011-1 10:2
805165	581496	22190	LOCAL CAFE MUG	24.0	NaN	9.36	1.4976	10.8576	2011-1 10:2
805192	581501	21564	PINK HEART SHAPE LOVE BUCKET	24.0	NaN	18.96	3.0336	21.9936	2011-1 10:4
805349	581567	22464	HANGING METAL HEART LANTERN	24.0	NaN	18.96	3.0336	21.9936	2011-1 11:5
805517	581585	84879	ASSORTED COLOUR BIRD ORNAMENT	16.0	NaN	27.04	4.3264	31.3664	2011-1 12:3

13386 rows × 12 columns

In [142…

```
print(df['Price'].isna().sum())
df['Price'] = df['Price'].fillna(df['Total'] / df['Quantity'])
```

```
df['Price'].isna().sum()
```

```
13386
```

```
C:\Users\asabb\AppData\Local\Temp\ipykernel_19756\4184607728.py:2: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df['Price'] = df['Price'].fillna(df['Total'] / df['Quantity'])
```

Out[142…

```
np.int64(4562)
```

In [143…

```
### Get help from StockCode, assuming the product usually sold with the same price
df = df[df['StockCode'].notna()]

# Step 2: Keep only numeric StockCodes
df = df[df['StockCode'].astype(str).str.isnumeric()]

# Step 3: Convert StockCode to integer (optional)
df["StockCode"] = df["StockCode"].astype(int)

# Step 4: Fill missing prices using the average price per StockCode
df['Price'] = df.groupby('StockCode')['Price'].transform(lambda x: x.fillna(x.mean(
```

In [144…

```
df[df['Price'].isna()]
```

Out[144…

Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	InvoiceDate	Payme Meth

Handle Quantity

In [145…

```
df[df['Quantity'].isna()]
```

Out[145…

	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	Inv
11	489435	22353	LUNCHBOX WITH CUTLERY FAIRY CAKES	NaN	2.533019	30.60	4.8960	35.4960	2(
47	489437	20971	PINK BLUE FELT CRAFT TRINKET BOX	NaN	1.253953	15.00	2.4000	17.4000	2(
52	489437	22111	SCOTTIE DOG HOT WATER BOTTLE	NaN	4.894426	14.85	2.3760	17.2260	2(
441	489522	21479	WHITE SKULL HOT WATER BOTTLE	NaN	3.877635	7.50	1.2000	8.7000	2(
598	489529	20657	TROPICAL LUGGAGE TAG	NaN	1.241286	1.25	0.2000	1.4500	2(
...	...	...	...	...	...	...	...	...	
804775	581467	84976	RECTANGULAR SHAPED MIRROR	NaN	1.229000	2.37	0.3792	2.7492	2(
804802	581469	21158	MOODY GIRL DOOR HANGER	NaN	0.956357	0.39	0.0624	0.4524	2(
804908	581473	20718	RED RETROSPOT SHOPPER BAG	NaN	1.249688	1.25	0.2000	1.4500	2(
805046	581479	22087	PAPER BUNTING WHITE LACE	NaN	2.892199	29.50	4.7200	34.2200	2(
805154	581496	22112	CHOCOLATE HOT WATER BOTTLE	NaN	4.879060	29.70	4.7520	34.4520	2(

4002 rows × 12 columns

In [146…

```
print(df['Quantity'].isna().sum())
df['Quantity'] = df['Quantity'].fillna(df['Total'] / df['Price'])
df['Quantity'].isna().sum()
```

```
4002
```

Out[146…

```
np.int64(0)
```

Handle payment method

```
In [147…  df[df['Payment Method'].isna()]
          # df.isna().sum()
```

Out[147…

	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	Invoice
1171	489560	90093	CLEAR CRYSTAL STAR PHONE CHARM	24.0	0.85	20.40	3.2640	23.6640	2009-1 12:5
2915	489791	20621	ECONOMY PASSPORT COVER	1.0	2.10	2.10	0.3360	2.4360	2009-1 12:0
12036	490685	48197	DOOR MAT BIRD ON THE WIRE	1.0	6.75	6.75	1.0800	7.8300	2009-1 13:4
14512	491009	22353	LUNCHBOX WITH CUTLERY FAIRY CAKES	6.0	2.55	15.30	2.4480	17.7480	2009-1 18:0
19280	491645	20764	ABSTRACT CIRCLES SKETCHBOOK	1.0	3.75	3.75	0.6000	4.3500	2009-1 16:5
...	...	...	...	...	...	...	...	...	
792635	580399	23355	HOT WATER BOTTLE KEEP CALM	2.0	4.95	9.90	1.5840	11.4840	2011-1 11:4
801801	581166	22722	SET OF 6 SPICE TINS PANTRY DESIGN	1.0	3.95	3.95	0.6320	4.5820	2011-1 14:4
802115	581181	22144	CHRISTMAS CRAFT LITTLE FRIENDS	6.0	2.10	12.60	2.0160	14.6160	2011-1 15:5
802550	581230	22617	BAKING SET SPACEBOY DESIGN	3.0	4.95	14.85	2.3760	17.2260	2011-1 10:2
804537	581443	22758	LARGE PURPLE BABUSHKA NOTEBOOK	12.0	0.39	4.68	0.7488	5.4288	2011-1 16:5

288 rows × 12 columns

In [148…

```
#This fills missing payment methods with the first non-null value from the same Inv
print(df['Payment Method'].isna().sum())

def find_payment_method(group):
    rows = group.dropna()
    v = rows.iloc[0] if rows.shape[0] > 0 else np.nan
    return group.fillna(v).astype(object)

df['Payment Method'] = df['Payment Method'].groupby(df['Invoice']).transform(find_p

df['Payment Method'].isna().sum()
```

```
288
```

```
C:\Users\asabb\AppData\Local\Temp\ipykernel_19756\1119359148.py:7: FutureWarning: Do
wncasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will chan
ge in a future version. Call result.infer_objects(copy=False) instead. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  return group.fillna(v).astype(object)
```

Out[148…

```
np.int64(1)
```

Handle customer ID

In [149…

```
df[df['Customer ID'].isna()]
```

Out[149…

	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	Gross	In
1951	489640	84006	MAGIC TREE - PAPER FLOWERS	72.0	0.85	61.20	9.7920	70.9920	2
2187	489667	22246	GARLAND, MAGIC GARDEN 1.8M	12.0	1.95	23.40	3.7440	27.1440	2
2372	489683	48116	DOOR MAT MULTICOLOUR STRIPE	2.0	6.75	13.50	2.1600	15.6600	2
2483	489702	22086	PAPER CHAIN KIT 50'S CHRISTMAS	280.0	2.55	714.00	114.2400	828.2400	2
2691	489780	22083	PAPER CHAIN KIT RETRO SPOT	12.0	2.95	35.40	5.6640	41.0640	2
...	...	...	...	...	...	...	...	...	
802494	581220	20992	JAZZ HEARTS PURSE NOTEBOOK	24.0	0.39	9.36	1.4976	10.8576	2
802538	581225	23341	PINK DINER WALL CLOCK	2.0	8.50	17.00	2.7200	19.7200	2
803284	581376	22767	TRIPLE PHOTO FRAME CORNICE	1.0	9.95	9.95	1.5920	11.5420	2
803707	581405	22940	FELTCRAFT CHRISTMAS FAIRY	1.0	4.25	4.25	0.6800	4.9300	2
805385	581571	21169	YOU'RE CONFUSING ME METAL SIGN	1.0	1.69	1.69	0.2704	1.9604	2

1109 rows × 12 columns

In [150…

```
print(df['Customer ID'].isna().sum())
df['Customer ID'] = df.groupby('Invoice').apply(lambda x: x['Customer ID'].fillna(x
df['Customer ID'].isna().sum()
df.isna().sum()
```

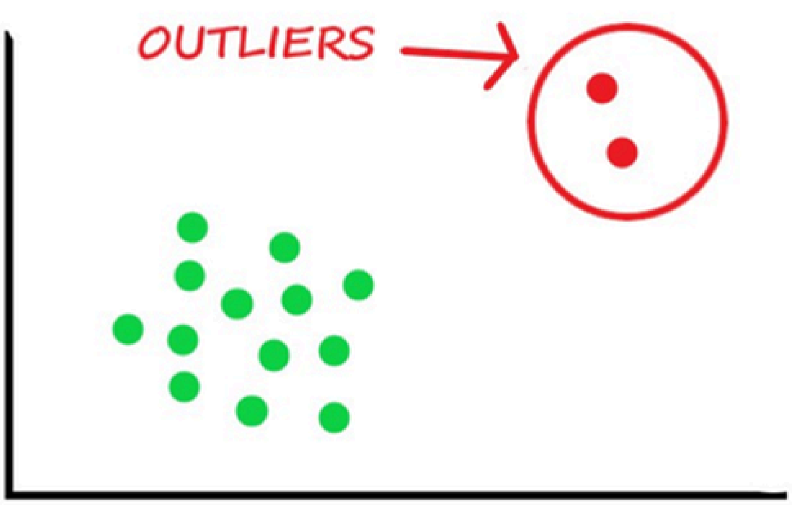
```
1109
```

```
C:\Users\asabb\AppData\Local\Temp\ipykernel_19756\3373735653.py:2: FutureWarning: Do
wncasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will chan
ge in a future version. Call result.infer_objects(copy=False) instead. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df['Customer ID'] = df.groupby('Invoice').apply(lambda x: x['Customer ID'].fillna
(x['Customer ID'].values[0]).astype(object)).reset_index('Invoice', drop=True)
C:\Users\asabb\AppData\Local\Temp\ipykernel_19756\3373735653.py:2: FutureWarning: Do
wncasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will chan
ge in a future version. Call result.infer_objects(copy=False) instead. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df['Customer ID'] = df.groupby('Invoice').apply(lambda x: x['Customer ID'].fillna
(x['Customer ID'].values[0]).astype(object)).reset_index('Invoice', drop=True)
C:\Users\asabb\AppData\Local\Temp\ipykernel_19756\3373735653.py:2: FutureWarning: Do
wncasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will chan
ge in a future version. Call result.infer_objects(copy=False) instead. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df['Customer ID'] = df.groupby('Invoice').apply(lambda x: x['Customer ID'].fillna
(x['Customer ID'].values[0]).astype(object)).reset_index('Invoice', drop=True)
C:\Users\asabb\AppData\Local\Temp\ipykernel_19756\3373735653.py:2: FutureWarning: Do
wncasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will chan
ge in a future version. Call result.infer_objects(copy=False) instead. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df['Customer ID'] = df.groupby('Invoice').apply(lambda x: x['Customer ID'].fillna
(x['Customer ID'].values[0]).astype(object)).reset_index('Invoice', drop=True)
C:\Users\asabb\AppData\Local\Temp\ipykernel_19756\3373735653.py:2: FutureWarning: Do
wncasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will chan
ge in a future version. Call result.infer_objects(copy=False) instead. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df['Customer ID'] = df.groupby('Invoice').apply(lambda x: x['Customer ID'].fillna
(x['Customer ID'].values[0]).astype(object)).reset_index('Invoice', drop=True)
C:\Users\asabb\AppData\Local\Temp\ipykernel_19756\3373735653.py:2: FutureWarning: Do
wncasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and will chan
ge in a future version. Call result.infer_objects(copy=False) instead. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df['Customer ID'] = df.groupby('Invoice').apply(lambda x: x['Customer ID'].fillna
(x['Customer ID'].values[0]).astype(object)).reset_index('Invoice', drop=True)
C:\Users\asabb\AppData\Local\Temp\ipykernel_19756\3373735653.py:2: DeprecationWarnin
g: DataFrameGroupBy.apply operated on the grouping columns. This behavior is depreca
ted, and in a future version of pandas the grouping columns will be excluded from th
e operation. Either pass `include_groups=False` to exclude the groupings or explicit
ly select the grouping columns after groupby to silence this warning.
  df['Customer ID'] = df.groupby('Invoice').apply(lambda x: x['Customer ID'].fillna
(x['Customer ID'].values[0]).astype(object)).reset_index('Invoice', drop=True)
```

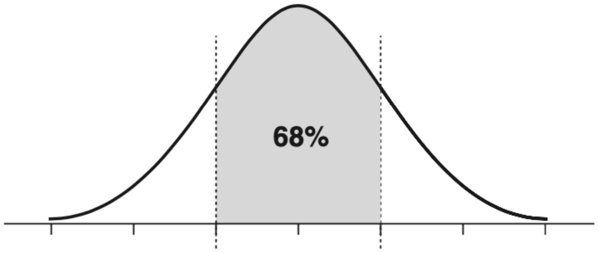
Out[150…

```
Invoice           0
StockCode         0
Description       0
Quantity          0
Price             0
Total             0
Tax 16%           0
Gross             0
InvoiceDate       0
Payment Method    1
Customer ID      55
Country           0
dtype: int64
```

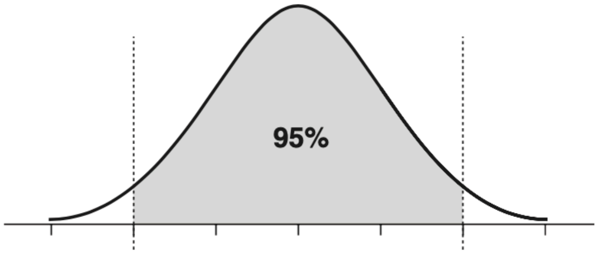
Outliers



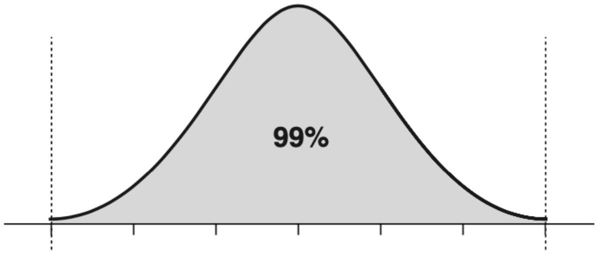
The Impirical Rule



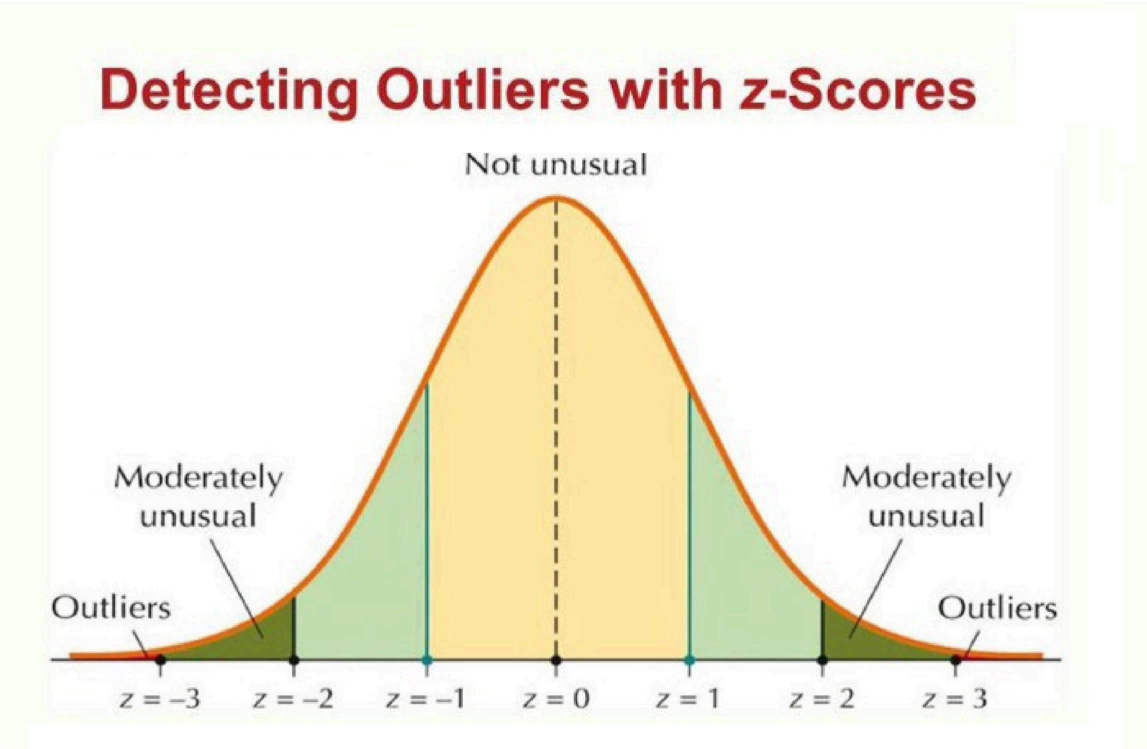
Approximately 68% of the area under a normal curve is between one standard deviation above and below the mean.



Approximately 95% of the area under a normal curve is between two standard deviations above and below the mean.



More than 99% of the area under a normal curve is between three standard deviations above and below the mean.



Recap

$$Z = \frac{X - \mu}{\sigma}$$

Find extreme/outlier prices

In [151…

```
df['price_zscore'] = (df['Price'] - df['Price'].mean()) / df['Price'].std()
df['outlier'] = df['price_zscore'].apply(lambda x: abs(x) >= 3)
df[df['outlier'] == True].sort_values(by="price_zscore", ascending=False)
```

Out[151…

	Invoice	StockCode	Description	Quantity	Price	Total	Tax 16%	
563073	556446	22502	PICNIC BASKET WICKER 60 PIECES	1.000000	649.500000	649.50	103.9200	75
563063	556444	22502	PICNIC BASKET WICKER 60 PIECES	60.000000	649.500000	38970.00	6235.2000	4520
215781	516913	22656	VINTAGE BLUE KITCHEN CABINET	1.000000	295.000000	295.00	47.2000	34
274680	523946	22655	VINTAGE RED KITCHEN CABINET	1.000000	295.000000	295.00	47.2000	34
210051	516164	22656	VINTAGE BLUE KITCHEN CABINET	1.000000	295.000000	295.00	47.2000	34
...	...	...	...	...	...	...	...	
782842	579196	21922	UNION STRIPE WITH FRINGE HAMMOCK	1.000000	16.630000	16.63	2.6608	1
385679	535576	22847	BREAD BIN DINER STYLE IVORY	2.056394	16.485169	33.90	5.4240	3
569496	557222	22847	BREAD BIN DINER STYLE IVORY	2.056394	16.485169	33.90	5.4240	3
783671	579283	22846	BREAD BIN DINER STYLE RED	2.072802	16.354677	33.90	5.4240	3
553272	555164	22848	BREAD BIN DINER STYLE PINK	11.070295	16.205530	179.40	28.7040	20

3846 rows × 14 columns