

Data type constraints

Course outline



Diagnose dirty
data



Side effects of
dirty data



Clean data

Why do we need to clean data?



Why do we need to clean data?



Garbage in Garbage out

Data type constraints

Datatype	Example
Text data	First name, last name, address ...
Integers	# Subscribers, # products sold ...
Decimals	Temperature, \$ exchange rates ...
Binary	Is married, new customer, yes/no, ...
Dates	Order dates, ship dates ...
Categories	Marriage status, gender ...

Python data type
<code>str</code>
<code>int</code>
<code>float</code>
<code>bool</code>
<code>datetime</code>
<code>category</code>

Strings to integers

```
# Import CSV file and output header  
sales = pd.read_csv('sales.csv')  
sales.head(2)
```

	SalesOrderID	Revenue	Quantity
0	43659	23153\$	12
1	43660	1457\$	2

```
# Get data types of columns  
sales.dtypes
```

```
SalesOrderID    int64  
Revenue         object  
Quantity        int64  
dtype: object
```

String to integers

```
# Get DataFrame information
```

```
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 31465 entries, 0 to 31464  
Data columns (total 3 columns):  
SalesOrderID      31465 non-null int64  
Revenue           31465 non-null object  
Quantity          31465 non-null int64  
dtypes: int64(2), object(1)  
memory usage: 737.5+ KB
```

String to integers

```
# Print sum of all Revenue column  
sales['Revenue'].sum()
```

```
'23153$1457$36865$32474$472$27510$16158$5694$6876$40487$807$6893$9153$6895$4216..'
```

```
# Remove $ from Revenue column  
sales['Revenue'] = sales['Revenue'].str.strip('$')  
sales['Revenue'] = sales['Revenue'].astype('int')
```

```
# Verify that Revenue is now an integer  
assert sales['Revenue'].dtype == 'int'
```


The assert statement

```
# This will pass  
assert 1+1 == 2
```

```
# This will not pass  
assert 1+1 == 3
```

```
AssertionError                                Traceback (most recent call last)  
      assert 1+1 == 3  
AssertionError:
```

Numeric or categorical?

```
... marriage_status ...  
...           3 ...  
...           1 ...  
...           2 ...
```

0 = Never married 1 = Married 2 = Separated 3 = Divorced

```
df['marriage_status'].describe()
```

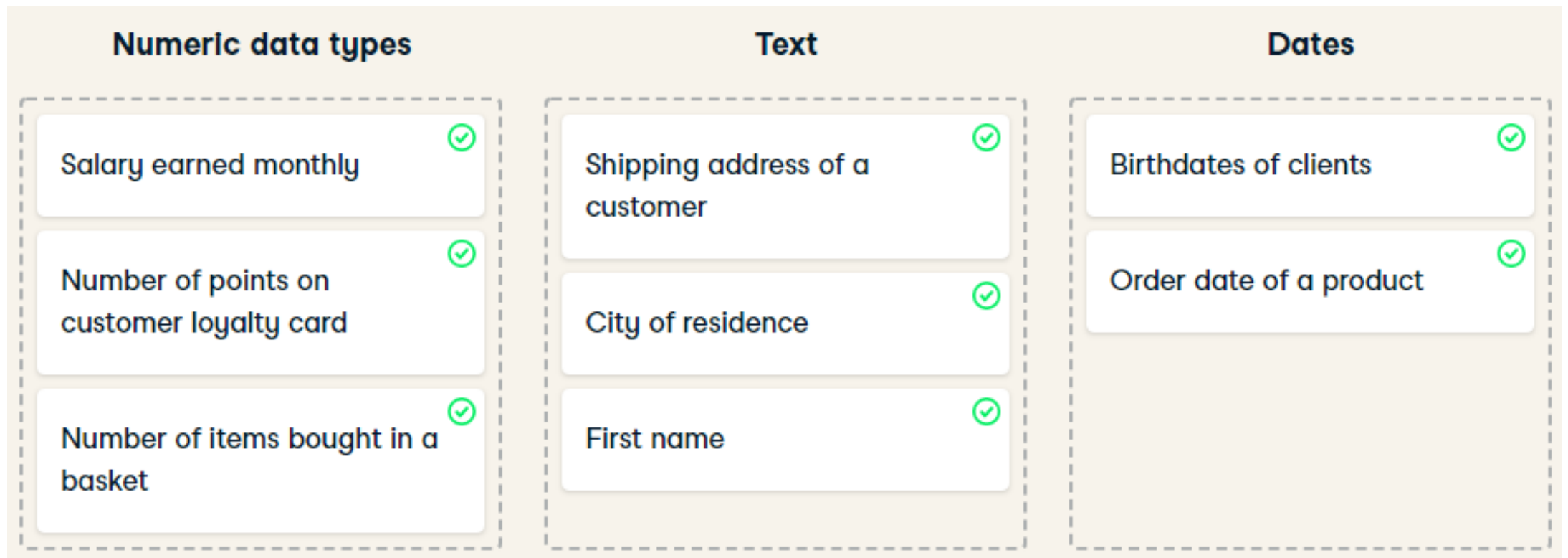
```
marriage_status  
...  
mean           1.4  
std            0.20  
min            0.00  
50%           1.8 ...
```

Numeric or categorical?

```
# Convert to categorical  
df["marriage_status"] = df["marriage_status"].astype('category')  
df.describe()
```

	marriage_status
count	241
unique	4
top	1
freq	120

Example of Data type



Data range constraints

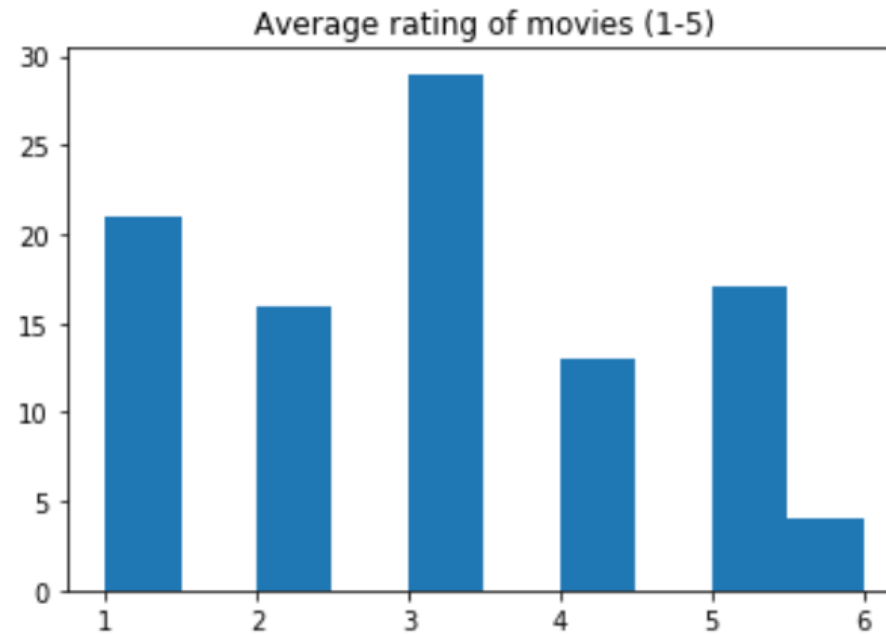
Motivation

```
movies.head()
```

```
   movie_name  avg_rating
0  The Godfather         5
1   Frozen 2         3
2    Shrek         4
...
```

Motivation

```
import matplotlib.pyplot as plt
plt.hist(movies['avg_rating'])
plt.title('Average rating of movies (1-5)')
```



Motivation

Can future sign-ups exist?

```
# Import date time
import datetime as dt
today_date = dt.date.today()
user_signups[user_signups['subscription_date'] > dt.date.today()]
```

	subscription_date	user_name	...	Country
0	01/05/2021	Marah	...	Nauru
1	09/08/2020	Joshua	...	Austria
2	04/01/2020	Heidi	...	Guinea
3	11/10/2020	Rina	...	Turkmenistan
4	11/07/2020	Christine	...	Marshall Islands
5	07/07/2020	Ayanna	...	Gabon

How to deal with out of range data?

- Dropping data
- Setting custom minimums and maximums
- Treat as missing and impute
- Setting custom value depending on business assumptions

Movie example

```
import pandas as pd
# Output Movies with rating > 5
movies[movies['avg_rating'] > 5]
```

	movie_name	avg_rating
23	A Beautiful Mind	6
65	La Vita e Bella	6
77	Amelie	6

1 `# Drop values using filtering`
`movies = movies[movies['avg_rating'] <= 5]` → a boolean series
2 `# Drop values using .drop()`
`movies.drop(movies[movies['avg_rating'] > 5].index, inplace = True)` → a selected DataFrame
`# Assert results`
`assert movies['avg_rating'].max() <= 5`

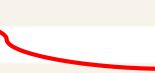
Movie example

```
# Convert avg_rating > 5 to 5
movies.loc[movies['avg_rating'] > 5, 'avg_rating'] = 5
```



True shows the row index

```
# Assert statement
assert movies['avg_rating'].max() <= 5
```



Column index

Remember, no output means it passed

Date range example

```
import datetime as dt
import pandas as pd
# Output data types
user_signups.dtypes
```

```
subscription_date    object
user_name             object
Country              object
dtype: object
```

```
# Convert to DateTime
user_signups['subscription_date'] = pd.to_datetime(user_signups['subscription_date'])
```

```
# Assert that conversion happened
assert user_signups['subscription_date'].dtype == 'datetime64[ns]'
```

Date range example

```
today_date = dt.date.today()
```

Drop the data

New DataFrame = DataFrame with out of range data

```
# Drop values using filtering
user_signups = user_signups[user_signups['subscription_date'] < today_date]
# Drop values using .drop()
user_signups.drop(user_signups[user_signups['subscription_date'] > today_date].index, inplace = True)
```



Hardcode dates with upper limit

True



```
# Drop values using filtering
user_signups.loc[user_signups['subscription_date'] > today_date, 'subscription_date'] = today_date
# Assert is true
assert user_signups.subscription_date.max().date() <= today_date
```

Uniqueness constraints

What are duplicate values?

- ✓ All columns have the same values

first_name	last_name	address	height	weight
Justin	Saddlemyer	Boulevard du Jardin Botanique 3, Bruxelles	193 cm	87 kg
Justin	Saddlemyer	Boulevard du Jardin Botanique 3, Bruxelles	193 cm	87 kg

What are duplicate values?

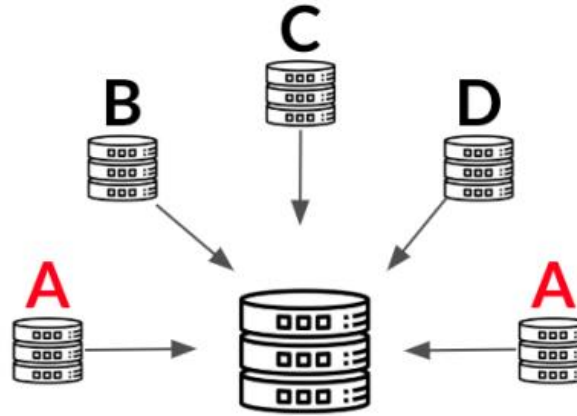
- ✓ Most columns have the same values

first_name	last_name	address	height	weight
Justin	Saddlemyer	Boulevard du Jardin Botanique 3, Bruxelles	193 cm	87 kg
Justin	Saddlemyer	Boulevard du Jardin Botanique 3, Bruxelles	194 cm	87 kg

Why do they happen?



Data Entry &
Human Error



Join or merge
Errors



Bugs and design
errors

How to find duplicate values?

```
# Print the header  
height_weight.head()
```

	first_name	last_name	address	height	weight
0	Lane	Reese	534-1559 Nam St.	181	64
1	Ivor	Pierce	102-3364 Non Road	168	66
2	Roary	Gibson	P.O. Box 344, 7785 Nisi Ave	191	99
3	Shannon	Little	691-2550 Consectetuer Street	185	65
4	Abdul	Fry	4565 Risus St.	169	65

How to find duplicate values?

```
# Get duplicates across all columns
duplicates = height_weight.duplicated()
print(duplicates)
```

```
1      False
...     ....
22     True
23     False
...     ...
```

How to find duplicate values?

```
# Get duplicate rows
duplicates = height_weight.duplicated()
height_weight[duplicates]
```

	first_name	last_name	address	height	weight
100	Mary	Colon	4674 Ut Rd.	179	75
101	Ivor	Pierce	102-3364 Non Road	168	88
102	Cole	Palmer	8366 At, Street	178	91
103	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	196	83

How to find duplicate rows?

The `.duplicated()` method

`subset` : List of column names to check for duplication.

`keep` : Whether to keep **first** (`'first'`), **last** (`'last'`) or **all** (`False`) duplicate values.

```
# Column names to check for duplication
```

```
column_names = ['first_name', 'last_name', 'address']
```

```
★ duplicates = height_weight.duplicated(subset = column_names, keep = False)
```

How to find duplicate rows?

```
# Output duplicate values  
height_weight[duplicates]
```

	first_name	last_name	address	height	weight
1	Ivor	Pierce	102-3364 Non Road	168	66
22	Cole	Palmer	8366 At, Street	178	91
28	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	195	83
37	Mary	Colon	4674 Ut Rd.	179	75
100	Mary	Colon	4674 Ut Rd.	179	75
101	Ivor	Pierce	102-3364 Non Road	168	88
102	Cole	Palmer	8366 At, Street	178	91
103	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	196	83

How to find duplicate rows?

```
# Output duplicate values
```

```
height_weight[duplicates].sort_values(by = 'first_name')
```

	first_name	last_name	address	height	weight
22	Cole	Palmer	8366 At, Street	178	91
102	Cole	Palmer	8366 At, Street	178	91
28	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	195	83
103	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	196	83
1	Ivor	Pierce	102-3364 Non Road	168	66
101	Ivor	Pierce	102-3364 Non Road	168	88
37	Mary	Colon	4674 Ut Rd.	179	75
100	Mary	Colon	4674 Ut Rd.	179	75

How to find duplicate rows?

```
# Output duplicate values
```

```
height_weight[duplicates].sort_values(by = 'first_name')
```

	first_name	last_name	address	height	weight
22	Cole	Palmer	8366 At, Street	178	91
102	Cole	Palmer	8366 At, Street	178	91
28	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	195	83
103	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	196	83
1	Ivor	Pierce	102-3364 Non Road	168	66
101	Ivor	Pierce	102-3364 Non Road	168	88
37	Mary	Colon	4674 Ut Rd.	179	75
100	Mary	Colon	4674 Ut Rd.	179	75

How to find duplicate rows?

```
# Output duplicate values
```

```
height_weight[duplicates].sort_values(by = 'first_name')
```

	first_name	last_name	address	height	weight
22	Cole	Palmer	8366 At, Street	178	91
102	Cole	Palmer	8366 At, Street	178	91
28	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	195	83
103	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	196	83
1	Ivor	Pierce	102-3364 Non Road	168	66
101	Ivor	Pierce	102-3364 Non Road	168	88
37	Mary	Colon	4674 Ut Rd.	179	75
100	Mary	Colon	4674 Ut Rd.	179	75

How to treat duplicate values?

```
# Output duplicate values
```

```
height_weight[duplicates].sort_values(by = 'first_name')
```

	first_name	last_name	address	height	weight
22	Cole	Palmer	8366 At, Street	178	91
102	Cole	Palmer	8366 At, Street	178	91
28	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	195	83
103	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	196	83
1	Ivor	Pierce	102-3364 Non Road	168	66
101	Ivor	Pierce	102-3364 Non Road	168	88
37	Mary	Colon	4674 Ut Rd.	179	75
100	Mary	Colon	4674 Ut Rd.	179	75

How to treat duplicate values?

The `.drop_duplicates()` method

`subset` : List of column names to check for duplication.

`keep` : Whether to keep **first** ('first'), **last** ('last') or **all** (False) duplicate values.

`inplace` : Drop duplicated rows directly inside DataFrame without creating new object (True).

```
# Drop duplicates
height_weight.drop_duplicates(inplace = True)
```

How to treat duplicate values?

```
# Output duplicate values
column_names = ['first_name', 'last_name', 'address']
duplicates = height_weight.duplicated(subset = column_names, keep = False)
height_weight[duplicates].sort_values(by = 'first_name')
```

	first_name	last_name	address	height	weight
28	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	195	83
103	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	196	83
1	Ivor	Pierce	102-3364 Non Road	168	66
101	Ivor	Pierce	102-3364 Non Road	168	88

How to treat duplicate values?

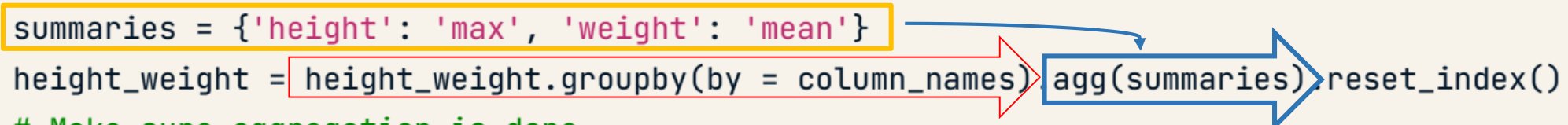
```
# Output duplicate values
column_names = ['first_name', 'last_name', 'address']
duplicates = height_weight.duplicated(subset = column_names, keep = False)
height_weight[duplicates].sort_values(by = 'first_name')
```

	first_name	last_name	address	height	weight
28	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	195	83
103	Desirae	Shannon	P.O. Box 643, 5251 Consectetuer, Rd.	196	83
1	Ivor	Pierce	102-3364 Non Road	168	66
101	Ivor	Pierce	102-3364 Non Road	168	88

How to treat duplicate values?

The `.groupby()` and `.agg()` methods

```
# Group by column names and produce statistical summaries
column_names = ['first_name', 'last_name', 'address']
summaries = {'height': 'max', 'weight': 'mean'}
height_weight = height_weight.groupby(by = column_names).agg(summaries).reset_index()
# Make sure aggregation is done
duplicates = height_weight.duplicated(subset = column_names, keep = False)
height_weight[duplicates].sort_values(by = 'first_name')
```



first_name	last_name	address	height	weight
------------	-----------	---------	--------	--------

Let's practice!