Data Cleaning:
Missing Data and Optimising

Missing Data

Data Cleaning

3 Main Categories

Before we dive into handling missing data, let's explore the three basic categories of missing data.

- **MCAR (Missing Completely at Random):**

  Missing values are unrelated to any variable in the dataset.

  Example: Data entry error.

- **MAR (Missing at Random):**

  Missing values are related to other observed variables, not the missing variable itself.

  Example: Income missing for younger people.

- **MNAR (Missing Not at Random):**

  Missing values depend on the value of the missing variable.

  Example: Non-payment amounts not reported by customers.

Data Cleaning

Delete, Recode, or Impute? That's the question!

Deleting
Missing values or columns with missing values can be deleted in certain situations. For example, you have a large sample of data you plan to use for hypothesis testing. If you have three thousand rows, deleting a few is usually fine.
Recoding
When you have missing data such as payments with a 0 value becuase a customer has not yet paid. You could create a new boolean column names not_paid to show why there is a zero (MNAR).
Imputing
Sometimes missing values can be imputed as none for categorial data or 0 for numerical data if it makes sense to do so. We'll explore this further in the upcoming videos.

Action Steps
Research and talk to stakeholders, then consider one of the following:
1.Remove missing data
2.Recode missing data
3.Impute missing data

dropna

```
df.dropna(
            axis: 'Axis' = 0,
            how: 'Any'/'All'
            subset: 'IndexLabel | None' = None,
            inplace: 'bool' = False)
```

- **Use axis**: Select rows or columns for the operation.
- **Use how**: Decide whether to delete any row/column with a NaN or only those where all values are NaN.
- **Use subset**: Apply the operation to specific columns.
- **Use inplace**: Make changes permanent in the dataset.

fillna

```
df.fillna(
    value,
    inplace: 'bool_t' = False )
```

fillna replaces missing values (NaN) in a dataset with a specified value or method.
For example:

- Replace NaN with a fixed value (e.g., df.fillna(0) fills all NaN with 0).

isnull

```
df.isnull()
```

	club	last_name	first_name	position	base_salary	guaranteed_compensation
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
610	False	False	False	False	False	False
611	False	False	False	False	False	False
612	False	False	False	False	False	False
613	True	True	True	True	False	True
614	False	False	False	False	False	False

Mask of bool values for each element in DataFrame that indicates whether an element is an NA value.

Optimising

astype

```
df['column name'].astype(category)
```

Cast a pandas object to a specified dtype ``dtype``.

The **category** data type in Pandas is used for data with a fixed number of unique values, such as labels or categories.
It is memory-efficient and speeds up operations compared to the object type, making it ideal for columns with repetitive text values like "Yes/No" or "City names."