Data I/O and Preprocessing with SQL and Python

Module 2: APIs & Numerical Cleaning

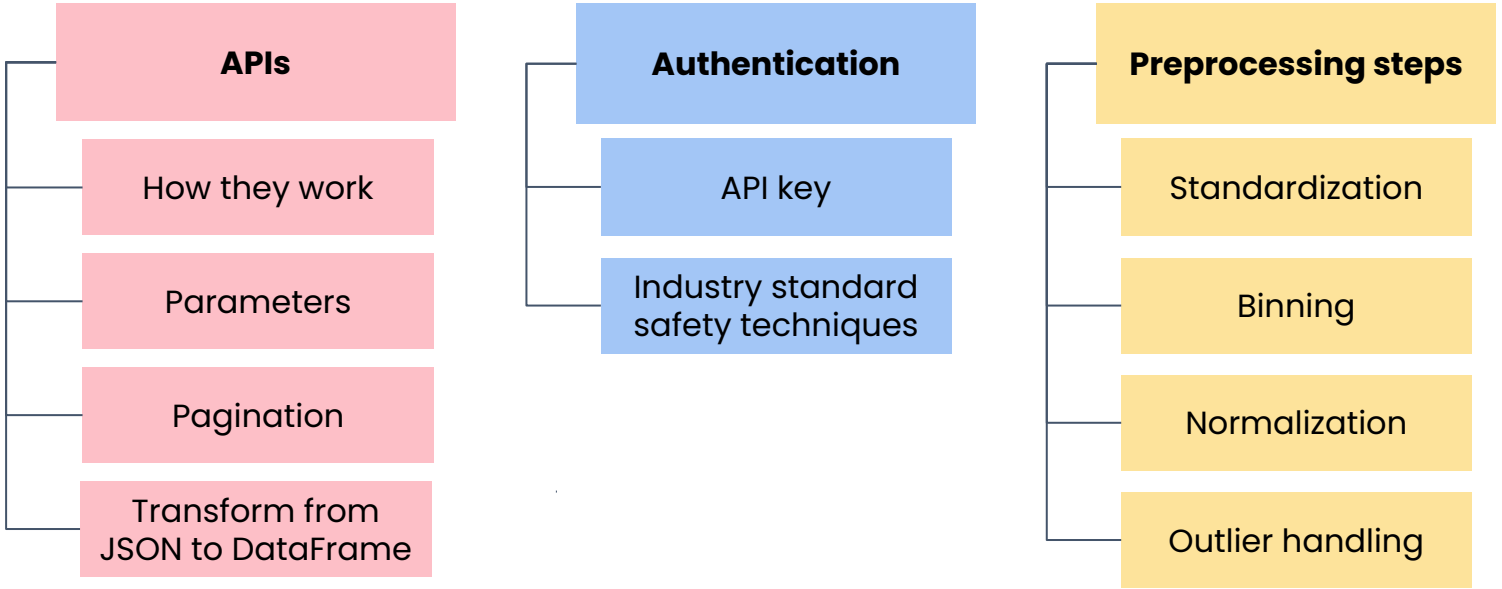
DeepLearning.AI

APIs & Numerical Cleaning

Module 2 introduction

DeepLearning.AI

Module 2 outline

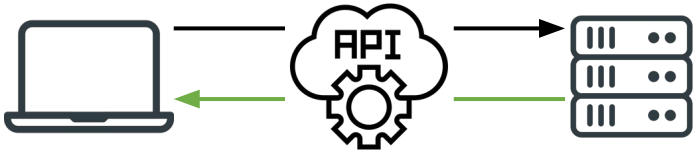


APIs & Numerical Cleaning

Introduction to APIs

DeepLearning.AI

Scraping vs. APIs



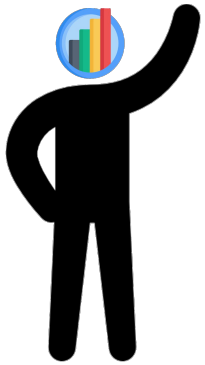
Scraping

- Extracting data from a webpage that has been formatted for human readers
- Data can be difficult to work with
  - ⛔ Websites aren't built to provide structured data
  - ⛔ Complexly formatted HTML
  - ⛔ Inconsistent layouts
  - ⛔ Difficult to write universal code
  - ⛔ Terms of service prohibit scraping

Application Programming Interface

- ✅ Request data directly from website's server in a structured format
- ✅ Access by writing code
- ✅ Programming interface
- ✅ More reliable and efficient
- ❗ Only use web scraping if data isn't available from an API

Scenario



You
Data Analyst

🏆 **Goal:** Track and analyze product recalls to assess potential risks to the public

📊 **Data:** Food enforcement API by the United States Food and Drug Administration (FDA)

- Structured data on recalled food products

🎯 **Tasks:**

- ⬜ Preprocess data
- ⬜ Prepare report that summarizes recall patterns

APIs & Numerical Cleaning

JSON

DeepLearning.AI

What is JSON?

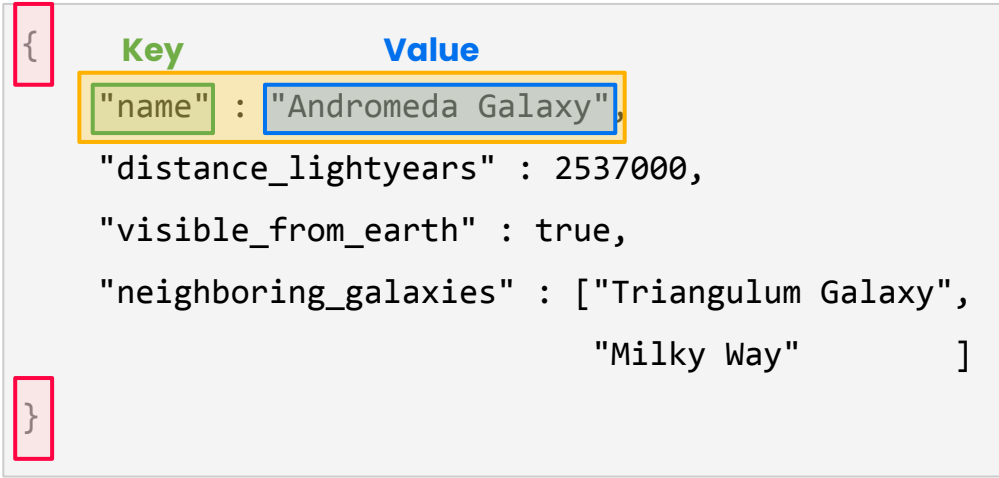
- You requested data from API managed by the FDA:

```
url = "https://api.fda.gov/food/enforcement.json"
```

- Stands for **J**ava**S**cript **O**bject **N**otation
- Format:
  - Used to transfer data between applications
  - Organizes data in a structured way
  - Originated from the web
  - Mimics the way JavaScript is written

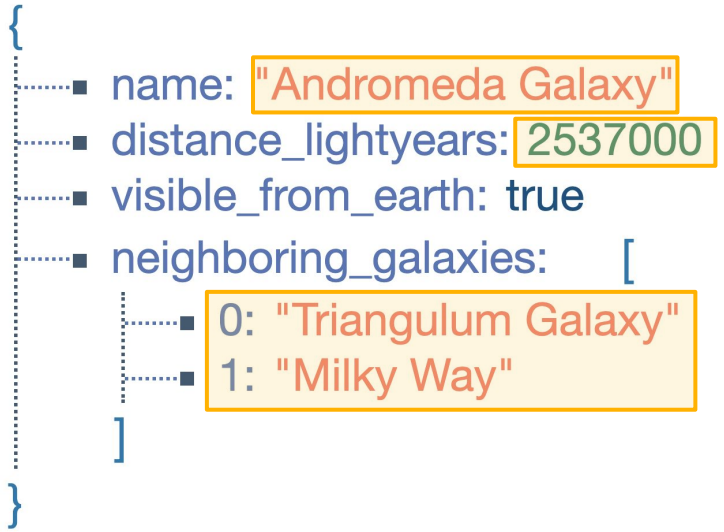
JSON example

- A collection structured as key-value pairs
  - Each key is used to access a value
- Previously, you worked with **lists**:
  - An ordered collection
  - Use **index** to access value (0, 1, 2,...)
- JSON use **keys**
  - Must be strings in quotes
  - Similar to a dictionary
- JSON loaded into Python notebook will be represented as a dictionary



JSON structure

- JSON has a **nested** structure
- When working with JSON:
  - Look at entire result to understand:
    - What data is available
    - Parts you'll need to access
  - Navigate different levels to get data you need



JSON and dictionaries

- Dictionaries:
  - Written with curly braces
  - Represent key-value pairs
- Few small differences:
  - Empty value
    - Represented by “null”
    - In Python → “None”
  - `true` and `false`
    - Capitalized in Python
    - Lower case in JavaScript

Important things to remember

- JSON is a structured format for data when using APIs
- Represented similarly by dictionaries
- Conversion will be done by Pandas

Recap: JSON

- A nested structure of key-value pairs for organizing data
- Formatted as a dictionary in Python
  - Defined by curly braces
  - Nested structures of key-value pairs

```
{
    "name" : "Andromeda Galaxy",
    "distance_lightyears" : 2537000,
    "visible_from_earth" : true,
    "neighboring_galaxies" : [ "Triangulum Galaxy",
                               "Milky Way"         ]
}
```

- Use chained indexing to unpack layers:

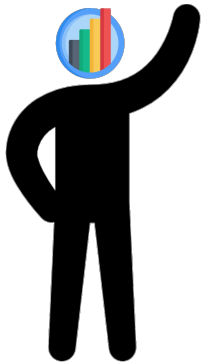
```
data["results"][0]["product_quantity"]
```

APIs & Numerical Cleaning

API requests and responses

DeepLearning.AI

Scenario



You
Data Analyst

📊 **Data:** FDA food enforcement API

🏆 **Goal:** Track and analyze food product recalls to assess potential risks to the public

Recap: API requests and responses

```
import requests
import pandas as pd

# Define the URL
url = "https://api.fda.gov/food/enforcement.json"

# Send a GET request
response = requests.get(url)

# Check the status code
print(response.status_code)

# Extract JSON data
data = response.json()
```

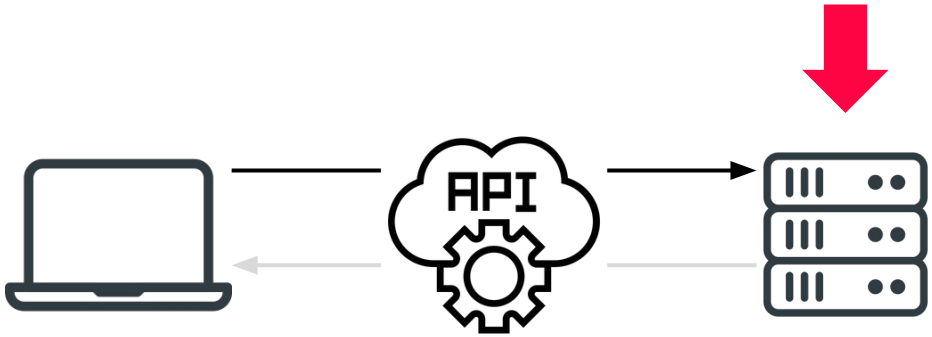
APIs & Numerical Cleaning

Query parameters

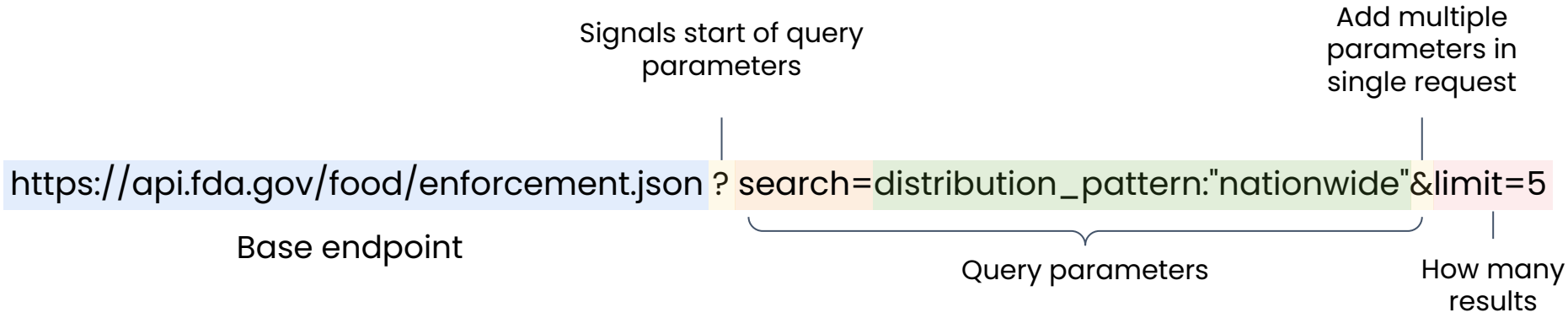
DeepLearning.AI

Queries and requests

- When you send a request:
  - Goes to specific URL called **API endpoint**
    - Destination of request
    - Where data lives
    - Determined by what the company providing API makes available
- Refine request with **query parameters**
  - “Query” is a synonym for “request”
  - Both mean “asking for information”
  - Best place to know what options are available is the API documentation




```
# Define the URL
url = "https://api.fda.gov/food/enforcement.json"
```



- **`skip`** - offset or ignore a certain number of records
- Instead of manually adding parameters:

```
params = {...}
requests.get(url, params=params)
```

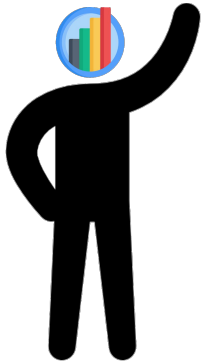
Automatically handles encoding and formatting

APIs & Numerical Cleaning

From JSON to a dataframe

DeepLearning.AI

Scenario



You
Data Analyst

📊 **Data:** FDA's food enforcement API

🏆 **Goal:** Track and analyze food product recalls to assess potential risks to the public

Recap: JSON to a dataframe

- Create a dataframe from a list of dictionaries:

```
pd.DataFrame(data["results"])
```

- Resulting dataframe:
  - Same length as list
  - Column names were keys of each dictionary

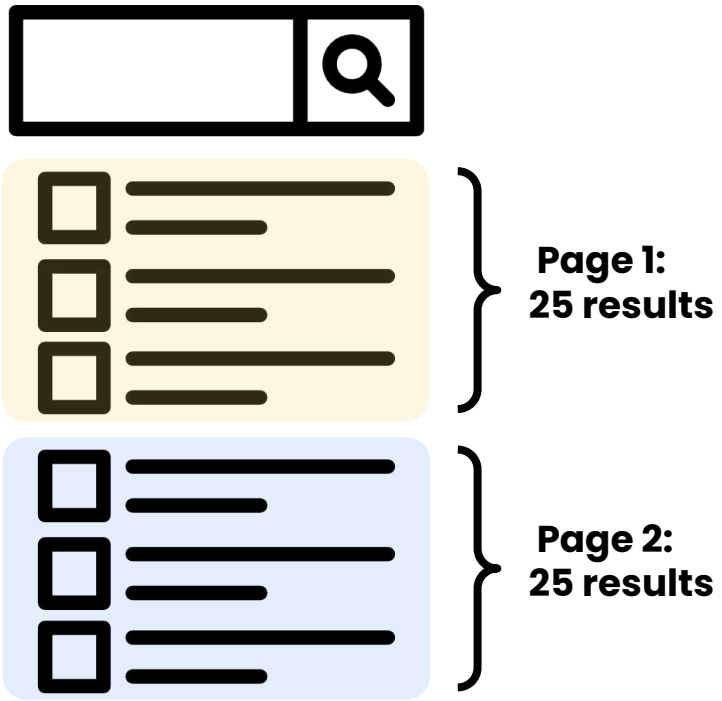
APIs & Numerical Cleaning

Pagination

DeepLearning.AI

Pagination

- Requested **1,000** records from FDA API
  - Each item was a dictionary representing a unique product recall
  - Limited to 1000 results in a response
- **Challenge**: There are **26,000** records
- **Solution**: Use pagination
  - Get different sets of records at once
  - Comes from the word “page”
  - Implement when making requests to get more than max from a single call



Pagination

- To get a large number of results from API:
  - Use pagination to make many requests
  - Build up dataframe from each one
- How it work for 25,000 records:
  - Request in sets of 1,000
  - Uses `skip` and `limit` query parameters
    - `"limit"` - always will be 1000
    - `"skip"` - how many initial results to ignore

1. `params = { "limit": 1000, "skip": 0 }`
2. `params = { "limit": 1000, "skip": 1000 }`

... `params = { "limit": 1000, "skip": 25000 }`

Recap: Pagination

- Used pagination to get thousands of results from an API
- Wrote a loop with consistent `limit` and increasing `skip` parameter
- Each batch was transformed into a DataFrame
- Used **`pd.concat()`** to combine the individual DataFrames into one
- Setting `ignore_index = True` renumbers indices in combined dataframe starting from 0

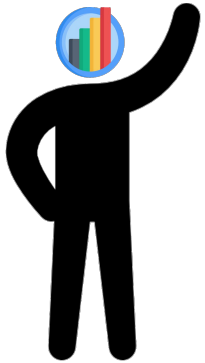
```
for i in range(25):
    params = {
        "limit": 1000,
        "skip": i * 1000,
    }

    data = requests.get(url, params=params).json()
    dfs.append(pd.DataFrame(data["results"]))

df = pd.concat(dfs, ignore_index = True )
```

APIs & Numerical Cleaning

Analyzing the
combined DataFrame

DeepLearning.AI

Scenario



You
Data Analyst

📊 **Data:** FDA's food enforcement API

🏆 **Goal:** Track and analyze food product recalls to assess potential risks to the public

APIs & Numerical Cleaning

API keys

DeepLearning.AI

What is API authentication?

- **Authentication** - process of verifying who is making the request to the API

🌐 **APIs without authentication:**

- Send request without identifying information and receive response
- **Why?**
  - They provide data that doesn't require significant resources to serve
  - Government agencies provide public APIs to give access to important data
  - Intended to maximize accessibility

🔐 **Most APIs require authentication:**

- Provide proof of your identity before you can access the data
- **Why?**
  - Requires computing power, and therefore costs money
  - Offer access to private data
  - Safeguard against malicious requests

What is an API key?

- Unique identifier assigned to you by provider
- Acts as a digital signature
- Often look like this:

  `"9a2b6c4d-e8f0-4g12-h345-6ijk7lm8no9p"`

  - Long string of letters and numbers
  - Sometimes include special characters

API key authentication

1. You include API key as part of request
2. API server:
   - Receives request
   - Checks against authorized keys
   - If valid, send back response with the data requested

Common API key errors

⛔ Missing, incorrect, or exceeded limits

⛔ Server will:

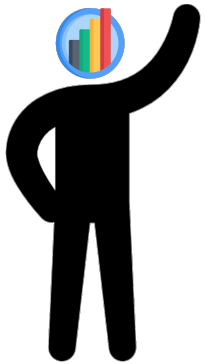
- Respond rejecting request
- Include error message (e.g. “Invalid API Key”) depending on the specific issue

APIs & Numerical Cleaning

Using an API key

DeepLearning.AI

Scenario



You
Data Analyst

🏆 **Goal:** Develop reporting system that gives access to information on safety of food manufacturing plants across the U.S.

🎯 **Task:** Analyze publicly available data from government inspection reports

📊 **Data:** Food Safety Inspection Reports API

- Provides up-to-date inspection reporting data for food processing facilities
- Requires authentication via an API key

Recap: Using an API key

- To use an `api_key`, you can:

```
params = {"api_key": api_key}

url = "https://2eraiuh.dlai.link/api/facilities"
response = requests.get(url, params=params)
```

- Check API documentation to make sure you have:
  - ⬜ Base endpoint
  - ⬜ Parameter name

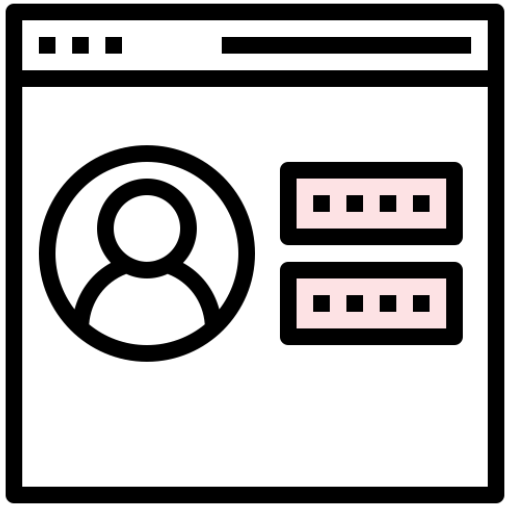
APIs & Numerical Cleaning

Environment variables

DeepLearning.AI

Keeping your API keys safe

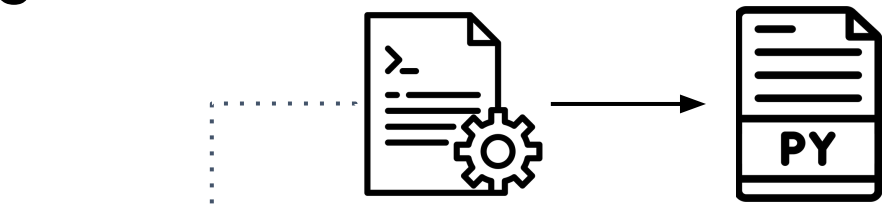
- Why hide your API key:
  - **`api_key`** is like a password
  - It's very common to share code
- While API key is technically a string:
  - You'll never see experienced programmers store it this way
- API keys provide access to services
- If someone gets your API key, they could:
  - Send requests as you
  - Rack up huge charges
  - Access your API usage history



⚠️ You don't want to share your API key

Environment variables

- Values stored outside of Python file
- Accessed by multiple programs or notebooks running on computer
- "Jupyter notebook **environment**"
  - Workspace inside computer where code runs
- Environment variables are:
  - Part of that workspace
  - Managed at higher level, outside any single file
- Main reasons for using is security
  - Prevent sensitive information from being accidentally shared



.env

```
INSPECTION_SAFETY_API_KEY =
"sk_live_x9Y2NkR7vL4mP8qJ3tW5hB1c_6nVwZdKfM"

FDA_STANDARDS_API_KEY =
"xk_2f90a8d31ee745b6a8995c9e8d22b012"
```

How environment variables work

os module - allows Python to interact with operating system

dotenv module - helps load environment variables from .env file

load_dotenv() - reads .env file and makes its contents available

Retrieves value from environment and stores it in **api_key**

```python
import os
from dotenv import load_dotenv


# Load the .env file
load_dotenv(".env")

 # Retrieve the API key
api_key = os.getenv("FOOD_SAFETY_API_KEY")
```

APIs & Numerical Cleaning

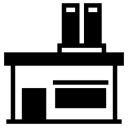
Scaling

DeepLearning.AI

Recap

🎯 **Goal:** Develop reporting system that provides consumers with information on food manufacturers

- At first glance, you might:
  - Publish violation count directly
  - Compare different facilities' violation counts
- To fairly compare, scale your data
  - **Scaling** - adjusts to a consistent measurement to make comparison fair across different contexts
  - **Example:** Calculating violations by facility size gives more accurate risk measurement



Size: 1,000 ft²
Violations: 1

Size: 30,000 ft²
Violations: 2

```
{  'critical_violations': 0,
   'days_since_last_inspection': 267,
   'employee_count': 182,
   'facility_id': 'FAC001',
   'facility_name': 'Tropical Harvest Oasis Operations',
   'inspection_date': '2024-05-16',
   'non_critical_violations': 2,
   'previous_score': 75,
   'production_volume': 11383,
   'risk_category': 'Medium',
   'shifts_per_day': 2,
   'square_footage': 182969,
   'state': 'AZ',
   'total_score': 76,
   'training_hours_monthly': 482   }
```

Recap: Scaling

Calculated violations per 100,000 square feet to make comparisons across facilities:

- Scaled violation count

```
df["scaled_violations"] = df["total_violations"] / df["square_footage"]

df["scaled_violations"] = df["scaled_violations"] * 100000
```

- Use **`.round()`** method on Series

```
df["scaled_violations"] = round(df["scaled_violations"], 2 )
```

Number of decimal places

APIs & Numerical Cleaning

Binning

DeepLearning.AI

Binning

- Involves categorizing numerical data into distinct groups or "bins"
- Helps simplify data analysis by reducing noise

Equal-size (“quantile”) bins

- Bins contain equal number of observations
- Uniform sample sizes across categories
- Quick approach to group numerical data

Custom bins

- Industry standards define specific breakpoints
- Use percentiles to define bins
- Useful when:
  - ✅ Data is unevenly distributed
  - ✅ Certain ranges are more meaningful
    - **Example**: Segmenting top 5% of customers separately

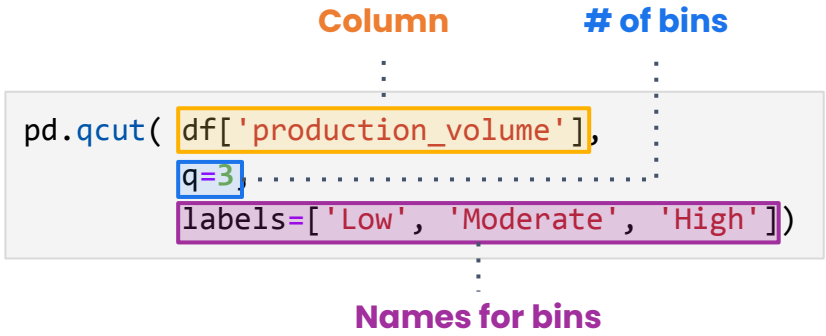
Recap: Binning

`qcut()`:

- Automatically determines bin cutoff
- Each bin contains an equal number of observations

`cut()`:

- Creates custom bins
- Requires you to explicitly specify the bin boundaries




```
boundaries = [0, 75, 90, 100]
pd.cut( df['employee_count'],
        bins=boundaries,
        labels=['Low', 'Medium', 'High'])
```

APIs & Numerical Cleaning

Normalization

DeepLearning.AI

Normalization

- **Problem**: Compare health and safety performance for food manufacturing plants
- **Challenge**: A lot of factors that contribute:
  - Size
  - Violations
  - Employee training hours
  - More!
- **Task**: Create composite score from multiple variables that combines safety aspects

Factor	Typical range
Employee count	100s
Square footage	100,000s

Normalization

- Transforming data so that different variables are scaled consistently
- Adjusts values to common range, often between 0 and 1, by:

$$X_{\text{normalized}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

- Z-scoring is another method
- Ensures no single variable dominates because of its scale
- Allows you to compare different variables on same scale

Setting the score

🏆 **Goal**: Combining multiple inspection factors into a single score

📊 **Data**: Food Safety API

⛔ Using compliance score isn't enough

✅ Build safety composite score between 0 and 100, where higher is better

ℹ️ Score will be based on two inputs:

1. Days since last inspection
   - More recent inspections are better
2. Violations per 100,000 sq ft
   - Fewer violations are better

✅ Normalize each so it's on scale of 0 to 1

🪨 Weight each component:

- "Days since last inspection" → 30%
- "Violations per 100,000 sq ft" → 70%

Recap: Normalization

- Normalized factors to a common 0-1 scale:

```
df["days_normalized"] = 1 - normalize(df["days_since_last_inspection"])
df["violations_normalized"] = 1 - normalize(df["violations_per_1000_sqft"])
```

- Weighted the factors to reflect their importance in food safety

```
df["composite_score"] = round( (df["days_normalized"] * 0.30 +
                                df["violations_normalized"] * 0.50 * 100)
```

- Created a single composite score that can be used to compare the facilities

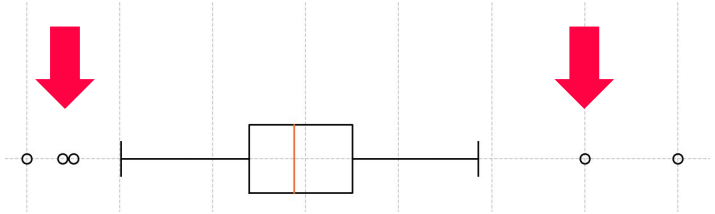
APIs & Numerical Cleaning

Identifying outliers

DeepLearning.AI

What are outliers?

- Values that lie far outside the range of most of data
- Sometimes they indicate:
  - Genuine variability
  - Errors or anomalies
- Before deciding what to do outliers:
  - Figure out why they exists
  - Have range of possible values in mind for each feature



Example: Casting column to datetime

- `'02121207'` → Year 212
- Other data was between 2008-2025
- Clearly an error

Detecting outliers

1️⃣ Interquartile range (IQR) method

- Define outliers as values:
  - 📉 Below $\mathbf{Q1 - (1.5 \times IQR)}$
  - 📈 Above $\mathbf{Q3 + 1.5 * IQR}$

- **IQR** is robust to skewed data
- When data may follow a normal distribution, **z-scoring** may be more appropriate

2️⃣ Z-Score method:

- Calculate Z-Score for each data point:

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

- Data points considered outliers:
  - 📈 $\mathbf{Z > 3}$
  - 📉 $\mathbf{Z < -3}$
    - **3-sigma rule**: 99.7% are expected 3σ of mean in normal distribution

      → Outliers are the most extreme 0.3%

APIs & Numerical Cleaning

Handling outliers

DeepLearning.AI

How to deal with outliers

✂️ **Remove them**

- If clearly an error, dropping makes sense
- Be careful not to remove outliers without justification

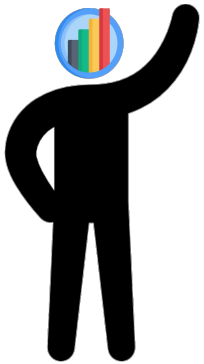
🔍 **Keep but analyze them separately**

- Outliers tell an interesting story
- Skew analyses of the other data points

🔄 **Transform them**

- Replace extreme values with a predefined percentile
- Use a “log transformation”

Scenario



You
Data Analyst

🎯 **Task**: Look at training hours different facilities provide

- ⬜ Colleague suggested dropping most of outliers
- ⬜ Calculate mean training hours per employee

APIs & Numerical Cleaning

Data quality

DeepLearning.AI

What is data quality?

⛔ **Completeness**:
Are all required values present?

⛔ **Accuracy**:
Does the data correctly reflect the real world?

⛔ **Consistency**:
Does the same data appear the same way across all records?

⛔ **Timeliness**:
Is the data up-to-date?

✅ Consider it good quality data

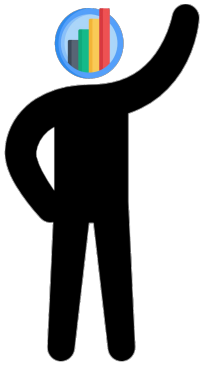
⛔ **Incomplete data**: May make analysis impossible

⛔ **Inaccurate data**: Doesn't lead to reliable conclusions

⛔ **Data that's inconsistent**: Requires a lot of time preprocessing

⛔ **Data that isn't timely**: Could make conclusions less generalizable

Scenario



You
Data Analyst

📊 **Data:** Ask a manager salary dataset

🎯 **Task**: Look at data quality dimensions to estimate true average salary for people in different industries