ND111 - Data Science II - Notebook

Contents

Course Info

Tags

- Author : AH Uyekita
- Dedication : 10 hours/week (suggested)
- Start : 14/12/2018
- End (Planned): 28/12/2018
- Title : Data Science II - Foundations Nanodegree Program
  - COD : ND111

Related Courses

- ND110 - Data Science I - Nanodegree Foundations

Objectives

I want to finish this course in two weeks. It includes the Optional videos and chapters.

Syllabus

- Chapter 01 - Welcome
  - Lesson 01 - Instructions
  - Lesson 02 - Tips
- Chapter 02 - SQL for Data Analysis
  - Lesson 01 - Basic SQL
  - Lesson 02 - SQL Joins
  - Lesson 03 - SQL Aggregations
  - Lesson 04 - (Optional) SQL Subqueries & Temporary Tables (Advanced)
  - Lesson 05 - (Optional) SQL Data Cleaning (Advanced)
  - Project 01 - Query a Digital Music Store Database
- Chapter 03 - Data Wrangling
  - Lesson 01 - Introduction to Data Wrangling
  - Lesson 02 - Gathering
  - Lesson 03 - Assessing Data
  - Lesson 04 - Cleaning Data
  - Project 02 - Wrangle and Analyze Data
- Chapter 04 - Advanced Statistics

  - Lesson 01 - Descriptive Statistics - Part 1
  - Lesson 02 - Descriptive Statistics - Part 2
  - Lesson 03 - Admissions Case Study
  - Lesson 04 - Probability
  - Lesson 05 - Binomial Distribution
  - Lesson 06 - Conditional Probability
  - Lesson 07 - Bayes Rule
  - Lesson 08 - Python Probability Practice
  - Lesson 09 - Normal Distribution Theory
  - Lesson 10 - Sampling Distributions and the Central Limit Theorem
  - Lesson 11 - Confidence Intervals
  - Lesson 12 - Hypothesis Testing
  - Lesson 13 - Case Study: A/B Tests
  - Lesson 14 - Regression
  - Lesson 15 - Multiple Linear Regression
  - Lesson 16 - Logistic Regression
  - Project 03 - Analyze A/B Test Results

- Chapter 05 - Intro to Machine Learning

  - Lesson 01 - Welcome to Machine Learning
  - Lesson 02 - Naive Bayes
  - Lesson 03 - SVM
  - Lesson 04 - Decision Trees
  - Lesson 05 - Choose Your Own Algorithm
  - Lesson 06 - Datasets and Questions
  - Lesson 07 - Regressions
  - Lesson 08 - Outliers
  - Lesson 09 - Clustering
  - Lesson 10 - Feature Scaling
  - Lesson 11 - Text Learning
  - Lesson 12 - Feature Selection
  - Lesson 13 - PCA
  - Lesson 14 - Validation
  - Lesson 15 - Evaluation Metrics
  - Lesson 16 - Tying It All Together
  - Project 04 - Identify Fraud from Enron Email

- Chapter 06 - (Optional) Data Visualization

  - Lesson 01 - Introduction to Data Visualization
  - Lesson 02 - Design
  - Lesson 03 - Data Visualization in Tableau
  - Lesson 04 - Making Dashboard & Stories in Tableau

Repository Structure

This is the structure of this repository, each course's chapters (or parts) will be stored in different folders.

```
ND111_data_science_foundation_02
|
+--  01-Chapter_01
|           |
|           +--  README.md                          # General information
|
+--  02-Chapter_02
```

```
|           |
|           +--  README.md                         # General information
|           +--  00-Project_01                     # Project 01
|           +--  01-Lesson_01                      # Files from Lesson 01
|           |        +--  README.md                # Notes from Lesson 01 from Chapter 02
|           +--  02-Lesson_02                      # Files from Lesson 02
|           |        +--  README.md                # Notes from Lesson 02 from Chapter 02
|           .
|
+--  03-Chapter_03
|           |
|           +--  README.md                         # General information
|           +--  00-Project_02                     # Project 02
|           +--  01-Lesson_01                      # Files from Lesson 01
|           |        +--  README.md                # Notes from Lesson 01 from Chapter 02
|           +--  02-Lesson_02                      # Files from Lesson 02
|           |        +--  README.md                # Notes from Lesson 02 from Chapter 02
|           .
```

Best practice

- Add all *deliverables* in the `GitKraken Glo`;
- Take notes using the `Markdown`.

Chapter 1

Welcome

This chapter is about the General aspects of the Udacity platform study.

Instructions

General information about the course.

- Projects Deadline
- Projects Review
- Mentoring

Tips

- Asking Help
- Keep in contact with the Slack Community
- Student Manual

Chapter 2

SQL for Data Analysis

2.1 SQL Basics

2.1.1 Entity Relationship Diagrams (ERD)

This is a way to see (visualize) the relationship between different spreadsheets, in other words, how is structure a database. In a database, there are several tables, and each table has your own attributes, based on the cardinality they could interact with each other.

2.1.1.1 Entities

This is a simple spreadsheet with information about anything you want, but keep in mind to: store new observations by rows and features/variables by column.

My example is a table called `Marks`, which has `mark id`, `student id`, `subject id`, `date` and `mark` as attributes. The other column is the variable's type.

2.1.1.2 Atributte

An attribute is a feature we want to keep track.

2.1.1.3 Relationship

Is a way to connect two tables.

Remember, this line has some properties, that is named as cardinality.

2.1.1.4 Cardinality

Cardinality represents a notation of how the information between tables will interact with each other.

Additional videos with good content.

Video 1 - Lucidchart Vídeo 2 - Lucidchart

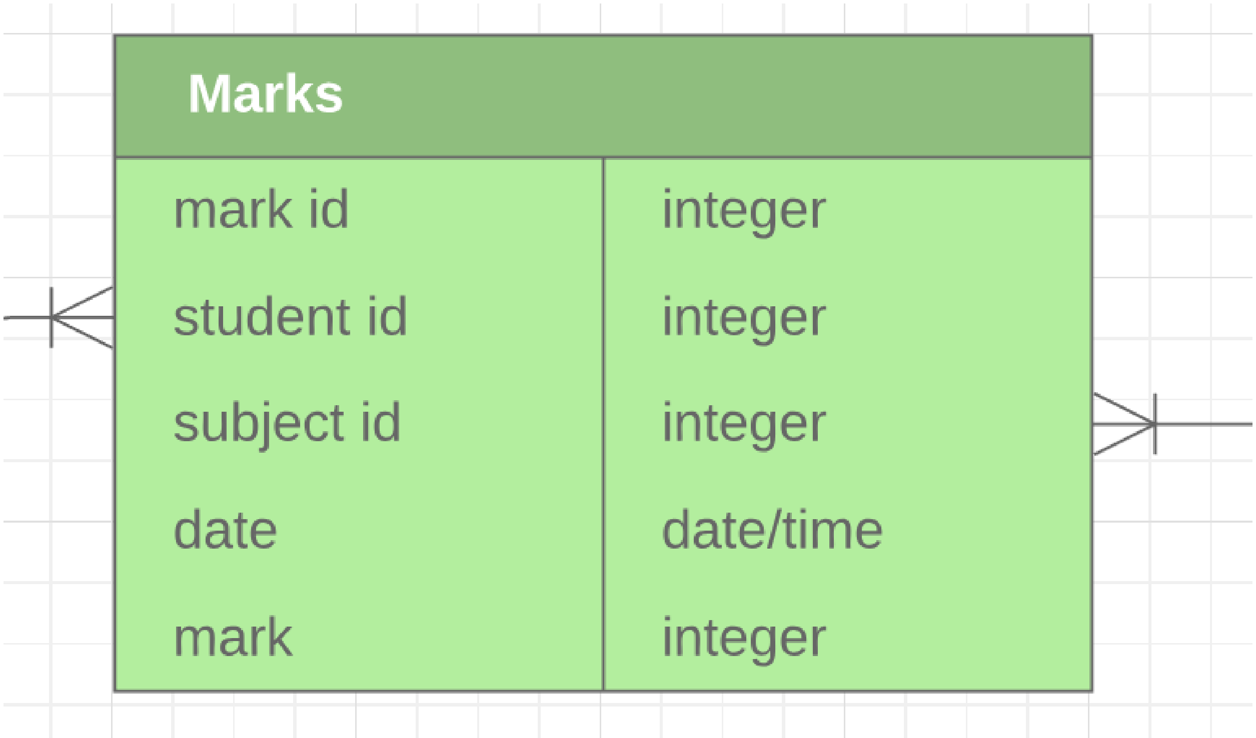

Figure 2.1: This is a entity.

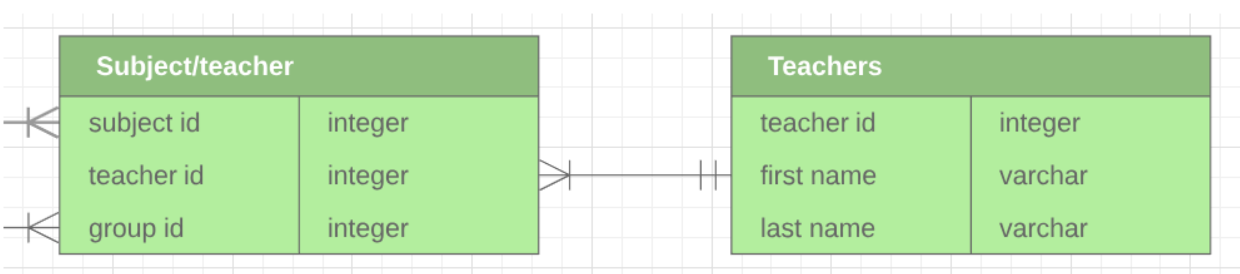

Figure 2.2: The line connecting two tables is a relationship.

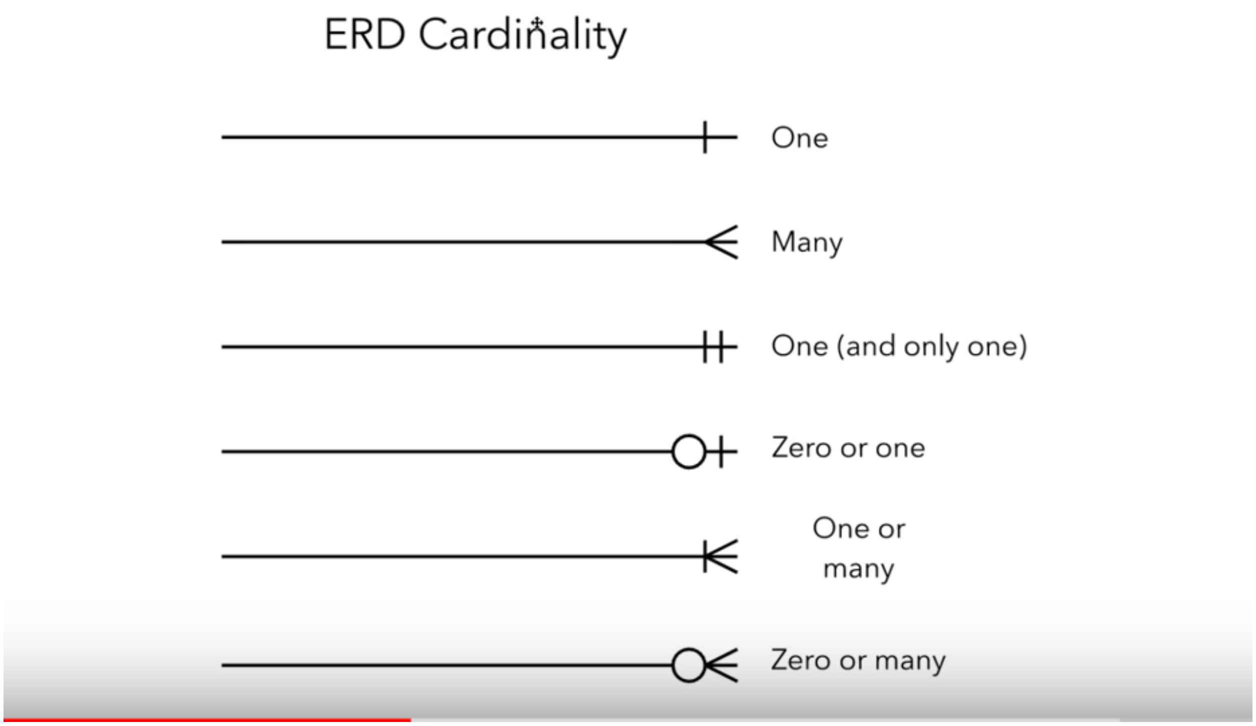

Figure 2.3: In a nutshell of Cardinality - Extracted from the Lucidchart Video.

2.1.2 SQL Introduction

SQL is a Language used to manage this interactions between tables, allowing us to access the stored database. The meaning of SQL is:

> Structured Query Language

It is very popular in Data Analysis because:

- Easy to understand
- Easy to learn
- Used to access very large datasets directly where is stored
- Easy to audit and replicate
- It is possible to run multiple queries at onde
- Almost do not have a limit of rows/observations
- Ensure the data Integrity, it is not possible to register a half child if you have defined this field as an integer
- SQL is very fast
- Database provide the data sharing, everybody could access the data simultaneously, which is good due to a standardization of database

SQL provides also functions such as:

- Summation
- Count
- Max and min

- Mean, etc.

Have in mind, probably we are going to manipulate data, and rarely updating or change values.

SQL is not case sensitive, so the best practices is to write the clauses/staments in upper case.

Best practices

```
SELECT first_column
  FROM my_table
```

Bad one

```
SelecT first_column
from my_table
```

Bear in mind, the indentation is not a requirements but helps a lot to understand your code.

2.1.2.1 SQL vs. NoSQL

Extracted from the class notes.

> You may have heard of NoSQL, which stands for not only SQL. Databases using NoSQL allow for you to write code that interacts with the data a bit differently than what we will do in this course. These NoSQL environments tend to be particularly popular for web based data, but less popular for data that lives in spreadsheets the way we have been analyzing data up to this point. One of the most popular NoSQL languages is called MongoDB. Udacity has a full course on MongoDB that you can take for free here, but these will not be a focus of this program. NoSQL is not a focus of analyzing data in this Nanodegree program, but you might see it referenced outside this course!

2.1.3 Clauses

Tell the database what to do.

2.1.3.1 `DROP TABLE`

Remove a table from the database.

2.1.3.2 `CREATE TABLE`

Create a new table.

2.1.3.3 `SELECT`

Is also know as query, is used to create a new table with the selected variables. You can use `*` if you want to select all columns.

```sql
SELECT first_column, second_column, last_column
  FROM first_table;
```

2.1.3.4 LIMIT

This is the same of `.head()` but this could only load a few lines to analyses the table.

```sql
SELECT first_column
  FROM my_table
LIMIT 1000            /* Will load the firs 1000 lines*/
```

2.1.3.5 ORDER BY

It is possible to order by in ascendant and descendent way.

ascendant

```sql
SELECT first_column, second_column, last_column
  FROM my_table
ORDER BY last_column /*ascendanting*/
LIMIT 1000
```

descendent

```sql
SELECT first_column, second_column, last_column
  FROM my_table
ORDER BY last_column DESC, second_column /*descending for last_column*/
LIMIT 1000
```

This last query will returns:

- Last_column ordered by the highest to lowest;
- The second_column will be the lowest to highest.

2.1.3.6 WHERE

Apply a filter to find a specific customer or anything else.

```sql
SELECT first_column, second_column, last_column
  FROM my_table
WHERE first_column = 100
ORDER BY second_column
LIMIT 100
```

All staments possible to use. * `>` (greater than) * `<` (less than) * `>=` (greater than or equal to) * `<=` (less than or equal to) * `=` (equal to) * `!=` (not equal to)

If the argument of the WHERE clause is not a number, you must use single quotes.

```sql
SELECT first_column, second_column, last_column
  FROM my_table
WHERE first_column = 'Hello World!'
ORDER BY second_column
LIMIT 100
```

2.1.4 Derived Columns

Is a new column created from the query. It is similar to the `mutate` function from R.

This is the operator to create a derived column:

- `*` (Multiplication)
- `+` (Addition)
- `-` (Subtraction)
- `/` (Division)

```sql
SELECT id, (standard_amt_usd/total_amt_usd)*100
FROM orders
LIMIT 10;
```

Will display without a specific name (?column?).

2.1.4.1 AS

If you use the `AS` the derived column will be name as you define (in other words "alias").

```sql
SELECT id, (standard_amt_usd/total_amt_usd)*100 AS std_percent, total_amt_usd
FROM orders
LIMIT 10;
```

Best pratices: No capital letters, descriptive names, etc.

2.1.5 Introduction to "Logical Operators"

In the next concepts, you will be learning about Logical Operators. Logical Operators include:

2.1.5.1 LIKE

Using with WHERE clause could search some patterns.

```sql
SELECT first_column, second_column, last_column
  FROM my_table
WHERE last_column LIKE '%ello%'
```

The `%` is called wild-card.

2.1.5.2 IN

It is the same in Python or R. `IN` will be used to filter the dataset based on a list.

```
SELECT first_column, second_column, last_column
  FROM my_table
WHERE last_column IN (100, 200)
```

This example will filter the rows of last_column with values of 100 or 200.

2.1.5.3 NOT

`NOT` return the reverse/opposite.

```
SELECT first_column, second_column, last_column
  FROM my_table
WHERE last_column NOT IN (100, 200)
```

This example will remove all observations equals to 100 or 200.

Possible uses:

- NOT IN
- NOT LIKE

2.1.5.4 AND

Logical statment usually to make some filtration.

```
SELECT *
FROM orders
WHERE standard_qty > 1000 AND poster_qty = 0 AND gloss_qty = 0;
```

2.1.5.5 BETWEEN

Sometimes AND statment could be replaced by BETWEEN, this is much clearly to understand. BUT the BETWEEN is inclusive, which means the endpoints will be included in the filter.

```
SELECT name
FROM accounts
WHERE name NOT LIKE 'C%' AND name LIKE '%s';
```

2.1.5.6 OR

Well, this is a logical operator.

```
SELECT id
FROM orders
WHERE gloss_qty > 4000 OR poster_qty > 4000;
```

2.2 SQL Joins

2.2.1 Joins

When a table is splited the performance to update or just to make a query is better than a big one. The reason is the quantity of data to read. This is one of the reason to split dataset in several tables, even more, sometimes in convinient to split because the type of data stored.

The reason of JOIN is to "bind" two datasets into one. Here we need to use the period `.` (table.colums) to reference which column/variable we want to select.

```
SELECT accounts.name, orders.occurred_at
  FROM orders
JOIN accounts
ON orders.account_id = accounts.id;
```

The result of this query is two columns (`name` and `occured_at`), and to linked by the `account_id` and `id`.

2.2.1.1 Primary Key (PK)

Is a columns with unique values used to map a variable.

2.2.1.2 Foreign Key (FK)

Is a Primary Key from the other table. We use the PK and FK to link the tables.

Based on the new information about `PK` and `FK`. Let's insert a picture to visualize the database.

I want to Join these tables. My query:

```
SELECT orders.*
FROM orders
JOIN accounts
ON orders.account_id = accounts.id;
```

What I need to realize:

- `PK` and `FK` **always** will be allocated in `ON`.
- `FROM` and `JOIN` each one with one table.

2.2.1.3 Binding three tables

It is possible to "chaining" three tables.

```
SELECT *
FROM web_events
JOIN accounts
ON web_events.account_id = accounts.id
JOIN orders
ON accounts.id = orders.account_id
```

In this case, I will import all columns, but I may want few columns.

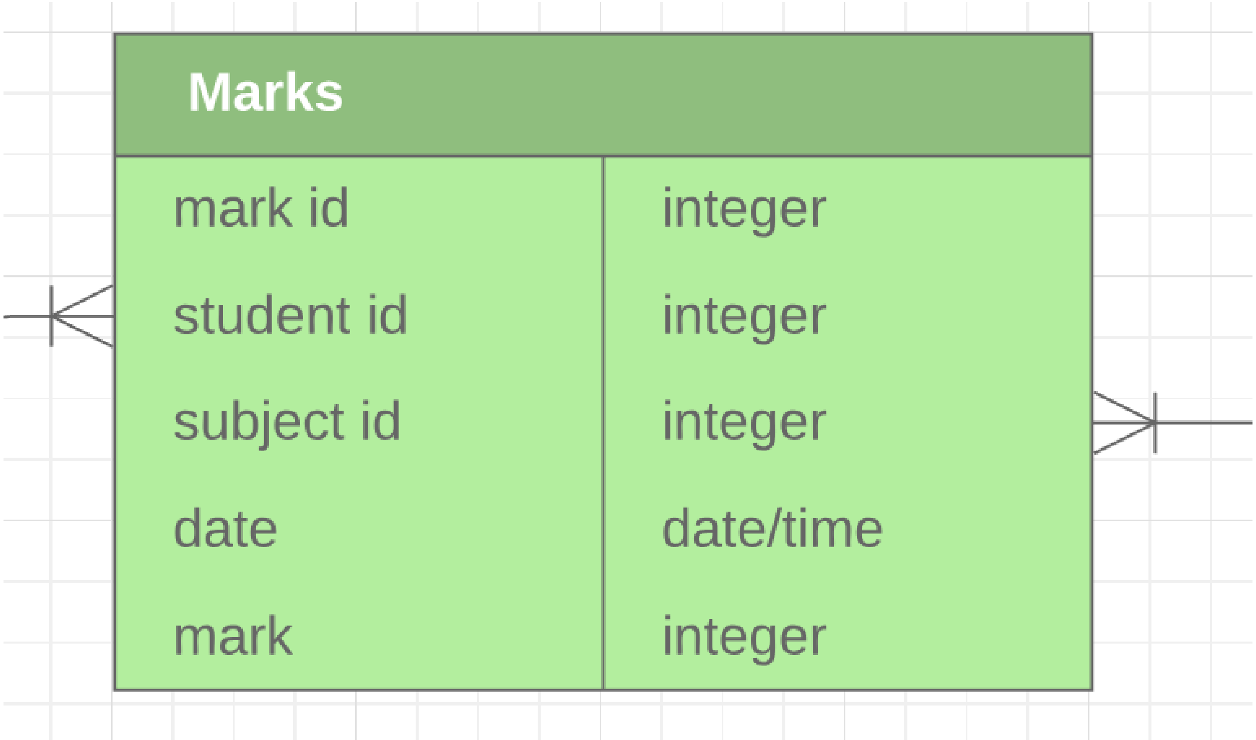

Figure 2.4: Example of Join

```
SELECT web_events.channel, accounts.name, orders.total
FROM web_events
JOIN accounts
ON web_events.account_id = accounts.id
JOIN orders
ON accounts.id = orders.account_id
```

2.2.1.4 Alias

Alias is a form to "short" the name of columns, the first method is using `AS`, but it could be simplified by only a space.

- Example 1

```
Select t1.column1 aliasname, t2.column2 aliasname2
FROM tablename AS t1
JOIN tablename2 AS t2
```

or

```
Select t1.column1 aliasname, t2.column2 aliasname2
FROM tablename t1
JOIN tablename2 t2
```

- Example 2

```sql
SELECT col1 + col2 AS total, col3
```

or

```sql
SELECT col1 + col2 total, col3
```

or

2.2.1.5 `INNER JOIN`

Returns rows which appears in both tables.

```sql
SELECT table_1.id, table_1.name, table_2.total
  FROM table_2
    JOIN table_1
      ON table_2.account_id = table_1.id
```

These last examples are all `INNER JOINS`, and will return a new dataframe (intersection between two dataframes).

2.2.1.6 `OUTER JOIN`

There are two kinds of OUTER JOINs

- Left outer JOIN, and;
- Right outer JOIN.

This two new JOINs has a property to pull rows that only exist in one table, it means some rows might have NULL values. The standard for this course will be to use only the left outer join.

2.3 SQL Aggregations

2.3.1 Aggregations Functions

This is functions return a single row with the aggregated value.

- sum;
- min;
- max;
- mean, etc.

2.3.1.1 NULL

NULL is no a value, it is different from ZERO or a space, for this reason you can not use equal (`=`) to find it, for do so you must use `IS`. The NULL is ignored in all aggregatins functions, and it is defined as a property of the data.

For the *Parch and Posey* dataset, NULL is equal to zero.

```
WHERE something IS NULL
WHERE something IS NOT NULL
```

2.3.1.1.1 NULLs - Expert Tip

There are two common ways in which you are likely to encounter NULLs:

- NULLs frequently occur when performing a LEFT or RIGHT JOIN. You saw in the last lesson - when some rows in the left table of a left join are not matched with rows in the right table, those rows will contain some NULL values in the result set.
- NULLs can also occur from simply missing data in our database.

2.3.2 Functions

2.3.2.1 COUNT()

Count the number of rows. If the entire line has only NULLs, this line will be noted counted.

Simple Example:

```
SELECT COUNT(*)
FROM accounts;
```

Example with filter

```
SELECT COUNT(*) AS order_count
FROM some_table
WHERE any_column > 100 AND any_column < 200;
```

Example with column selection

```
SELECT COUNT(account.id)
FROM accounts;
```

2.3.2.2 SUM()

Perform the summation among rows. You must define which columns will be applied the sum function.

```
SELECT SUM(poster_qty)
FROM demo.orders;
```

2.3.2.3 MAX() and MIN()

Return a rows with the minimun or maximum of a given column.

```
SELECT MAX(poster_qty) AS max_poster_qty,
       MIN(standard_qty) AS min_standard_qty
FROM demo.orders;
```

2.3.2.4 GROUP BY

Divide the non-grouped column into groups, which means the aggregated function will be calculated by group.

- The GROUP BY always goes between WHERE and ORDER BY.

Example 1:

```sql
SELECT a.name, o.occurred_at
FROM accounts a
JOIN orders o
ON a.id = o.account_id
ORDER BY o.occurred_at
LIMIT 1;
```

Same example but indexing by number:

```sql
SELECT a.name, o.occurred_at
FROM accounts a
JOIN orders o
ON a.id = o.account_id
ORDER BY 2
LIMIT 1;
```

OBS.: The index used in ORDER BY clause is to refence o.occurred_at.

2.3.2.5 DISTINCT

DISTINCT is always used in SELECT statements, and it provides the unique rows for all columns written in the SELECT statement. Therefore, you only use DISTINCT once in any particular SELECT statement.

```sql
SELECT DISTINCT column1, column2, column3
FROM table1;
```

2.3.2.6 HAVING

> HAVING is the “clean” way to filter a query that has been aggregated, but this is also commonly done using a subquery. Essentially, any time you want to perform a WHERE on an element of your query that was created by an aggregate, you need to use HAVING instead.

Note extracted from the class notes.

```sql
SELECT s.id, s.name, COUNT(*) num_accounts
FROM accounts a
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.id, s.name
HAVING COUNT(*) > 5
ORDER BY num_accounts;
```

2.3.3 DATE

To GROUP BY a date is quite complicated because each time is (obviously) different, for this, reason is necessary to "round" the time/date to group them.

2.3.3.1 DATE_TRUNC

Common trunctions are:

- day;
- month, and;
- year.

Sintaxe:

DATE_TRUNC('[interval]', time_column)

Where:

- microsecond
- millisecond
- second
- minute
- hour
- day
- week
- month
- quarter
- year
- century
- decade
- millenium

For further explanaition about date

```
SELECT demo.accounts.name,
       DATE_TRUNC('month', demo.orders.occurred_at) AS year_month,
       SUM(demo.orders.gloss_amt_usd) AS sum_gloss_usd
FROM demo.orders
JOIN demo.accounts
ON demo.orders.account_id = demo.accounts.id
WHERE demo.accounts.name = 'Walmart'
GROUP BY year_month, demo.accounts.name
ORDER BY sum_gloss_usd DESC
LIMIT 1;
```

2.3.3.2 DATE PART

Extract part of the date

2.3.4 CASE

Create a new column, derivate column, with a kind classification (assign a value into this new column according to the statment).

```
SELECT account_id,
       occurred_at,
       total,
       CASE WHEN total > 500 THEN 'Over 500'
            WHEN total > 300 AND total <= 500 THEN '301 - 500'
            WHEN total > 100 AND total <= 300 THEN '101 - 300'
            ELSE '100 or under' END AS total_group
FROM demo.orders
LIMIT 10;
```

Creates the total_group column.

2.3.4.1 With AGGREGATION

Combining the CASE clause with aggregations function could be a power tool, because the WHERE clause only evaluate one statement, using WHEN CASE it is possible to evaluate several staments.

```
SELECT demo.orders.account_id,
       demo.orders.total_amt_usd,
       CASE WHEN demo.orders.total_amt_usd >= 3000 THEN 'Large'
            ELSE 'Small' END AS level
FROM demo.orders
LIMIT 10;
```

2.4 SQL Subqueries & Temporary Tables (Advanced)

2.4.1 Subqueries

This is a way to nest queries, it means: The result of one query will be used as FROM to the next query.

```
SELECT *
FROM(SELECT something
     FROM   interesting) AS table_1
```

In the example above, I have one query nested to another. Bear in mind, I must give a alias to the nested query.

If the result of the subquery is a single value, you are allowed to insert this subquery wherever you want.

2.4.2 WITH

Also known as *Common Table Expression* (CTE), is a kinf of subquery but could be more helpful if someone is going to read the code. Due to the possibility to write the code in fragments an assign name, this is very handy.

Example

```
WITH my_with_example AS (SELECT ... MY CODE)

SELECT something
FROM my_with_example
```

As you can see it provide a better way to code because the code became more readable.

2.5 Data Cleaning (Advanced)

2.5.1 Data Cleaning

2.5.1.1 LEFT and RIGHT

It is the same of Excel functions.

```
SELECT LEFT(2, something) AS lefty_part_of_simething
FROM interesting
```

The example above will create a new column with the first two, from the left to right, character of something.

```
SELECT RIGHT(2, something) AS lefty_part_of_simething
FROM interesting
```

Almost the same, but start from the right to the left.

2.5.1.2 LEN

Returns the string length.

```
SELECT LEN(something)
FROM interesting
```

2.5.1.3 POSITION and STRPOS

POSITION will find a pattern in the string and will return the position (from the left to the right).

```
SELECT POSITION(',', something) /*Looking for a coma*/
FROM interesting
```

The STRPOS has the same use and same results.

```
SELECT STRPOS(something, ',') /*Looking for a coma*/
FROM interesting
```

Both functions are case sensitive.

2.5.1.4 LOWER and UPPER

Converts string into all lower or all upper cases.

```
SELECT LOWER(something)
FROM interesting
```

2.5.1.5 CONCAT

Bind/Combine/Concatenate strings (in different) columns into a new column.

Example 1

```
SELECT CONCAT(first_name, ' ',last_name) AS complete_name /* The ' ' is the space between strings*/
FROM interesting
```

You can use ||.

Example 2

```
SELECT first_name || ' ' || last_name AS complete_name /* The ' ' is the space between strings*/
FROM interesting
```

2.5.1.6 CAST

CAST allow to convert one type to another.

Example 1

```
SELECT CAST(year || month || day AS date) AS formatted_date
FROM interesting
```

The same of Example 1, but with a different notation to CAST clause.

Example 2:

```
SELECT (year || month || day AS date)::date AS formatted_date
FROM interesting
```

CAST is useful to converter strings into numbers or dates.

2.5.1.7 COALESCE

Converts NULL fields into Zero.

2.6 Project 01 - Chinook

Questions

All exercises of this chapter I have stored in the Mode Analytics platform.

Optional Questions

Project Submitted

I have written all the project in Mode Analytics because is a better place to coding.

- I can perform SQL queries;
- I can create graphics;
- An opportunity to get knowledge in a new tool.

Project 01 in Mode Analytic

2.6.1 Project Submission

To submit your project, please do the following:

- Review your project against the project Rubric. Reviewers will use this to evaluate your work.
- Create your slides with whatever presentation software you'd like (e.g. Google Slides, PowerPoint, Keynote, etc.).

In order to review your presentation, you will need to save your slides as a PDF. You can do this from within Google Slides by selecting File > Download as > PDF Document.

Chapter 3

Data Wrangling

3.1 Introduction to Data Wrangling

There are roughly three steps in the Data Wrangling.

- Gathering;
- Assessing, and;
- Cleaning.

This is a iterate process between these three steps.

> Data wrangling is about gathering the right pieces of data, assessing your data's quality and structure, then modifying your data to make it clean. But the assessments you make and convert to cleaning operations won't make your analysis, viz, or model better, though. The goal is to just make them possible, i.e., functional.
>
> EDA is about exploring your data to later augment it to maximize the potential of our analyses, visualizations, and models. When exploring, simple visualizations are often used to summarize your data's main characteristics. From there you can do things like remove outliers and create new and more descriptive features from existing data, also known as feature engineering. Or detect and remove outliers so your model's fit is better.
>
> ETL: You also may have heard of the extract-transform-load process also known as ETL. ETL differs from data wrangling in three main ways: * The users are different * The data is different * The use cases are different This article (Data Wrangling Versus ETL: What's the Difference?) by Wei Zhang explains these three differences well.

All text extracted from the class notes.

3.1.1 Gathering

Gathering is the first step of a Data Wrangling, is also known as Collecting or Acquiring. The Armenian Online Job Post has 19,000 jobs postings from 2004 to 2015.

> Best Practice: Downloading Files Programmatically

This is the reasons:

- Scalability: This automation will save time, and prevents erros;
- Reproducibility: Key point to any research. Anyone could reproduce your work and check it.

3.1.2 Assessing

The assessing in divided into two mains aspects:

- Quality of the dataset
- Tidiness of the dataset

3.1.2.1 Quality

Low quality dataset is related to a dirty dataset, which means the content quality of data.

Commom issues:

- Missing values
- Non standard units (km, meters, inches, etc. all mixed)
- Innacurate data, invalid data, inconsistent data, etc.

  One dataset may be high enough quality for one application but not for another.

3.1.2.2 Tidiness

Untidy data or *messy* data, is about the structure of the dataset.

- Each obsevation by rows, and;
- Each variable/features by column.

This is the Hadley Wickham definition of tidy data.

3.1.3 Assessing the data

There are two ways to assess the data.

- Visual, and;
- Programmatic.

3.1.3.1 Visual Assessment

Using regular tools, such as Graphics, Excel, tables, etc. It means, there is a human assessing the data.

3.1.3.2 Programmatic Assessment

Using automation to dataset evaluation is scalable, and allows you to handle a very huge quantity of data.

Examples of "Programmatic Assessment": Analysing the data using `.info()`, `.head()`, `.describe()`, plotting graphics (`.plot()`), etc..

Bear in mind, in this step we do not use "verbs" to describe any erros/problem, because the "verbs" will be actions to the next step.

3.1.4 Cleaning

Improving the quality of a dataset or cleaning the dataset do not means: Changing the data (because it could be **data fraud**).

The meaning of Cleaning is correcting the data or removing the data.

- Innacurate, wrong or irrelevant data.
- Replacing or filling (NULL or NA values) data.
- Combining/Merging datasets.

Improving the tidiness is transform the dataset to follow:

- each observation = row
- each variable = column

There are two ways to cleaning the data: manually and programmatic.

3.1.4.1 Manually

To be avoided.

3.1.4.2 Programmatic

There are three steps:

1. Define
2. Code
3. Test

> **Defining** means defining a data cleaning plan in writing, where we turn our assessments into defined cleaning tasks. This plan will also serve as an instruction list so others (or us in the future) can look at our work and reproduce it.
>
> **Coding** means translating these definitions to code and executing that code.
>
> **Testing** means testing our dataset, often using code, to make sure our cleaning operations worked.

Text from the class notes.

3.2 Data Gathering

This is the first step of any Data Wrangling, sometimes this process is a bit complicated because you need to find these data (probably from different sources and then merge).

3.2.1 Flat File

This is the way to store data into a single text file, usually, this file has another extension (.csv, tsv, etc.), each one of this extension has your own characteristic.

- Each variable/features is separated by a comma and each row is an observation;
- Each variable/features is separated by a tab and each row is an observation.

There are some **advantages** for using the flat files.

- Anyone could read, even a human;
- Is lightweight;
- You do not need to install a specific software;
- Simple to understand (each variable is delimited by a coma/tab);
- Any software could open it;
- Very good to small dataset.

But has disadvantages also:

- Do not have standard;
- Do not have data integrity checks;
  - Duplicated rows;
  - You can record any value in any field;
- Not great to large datasets.

3.2.1.0.1 Importing the tsv file

I have used the `read_csv` to load the data, but I have set the sep argument as `\t`, which means tabular. Sometimes, the flat files use `;` or `,`, so it is necessary to define what is the delimiter.

Example:

```
import pandas as pd
df = pd.read_cvs('bestofrt.tsv', sep= '\t')
```

3.2.2 Web Scraping

This is terminology is used to say the data extracted from a website (usually using code to do it). Due to this code depends on the HTML file, if any change of the website happens, all the code used to web scrapping could stopping to working properly, which requires an adjustments. For this reason, web scraping is not a definitive solution.

3.2.3 HTTP Request

This is useful to access archives from the internet, combining with the OS package, it is possible to download and store locally the file.

3.2.4 Encoding and Character Set

This explanation is based on this Stack Overflow thread.

Encoding: Is a proccess to convert a something into bytes.

- Audio is encoded into MP3, WAV, etc.
- Images is encoded into PNG, JPG, TIFF, etc.
- Text files is enconded into ASCII, UTF-8, etc.

The Character Set is as the name, is a set of charaters which I can use to write a phrase, each character has a code which represents the letter/character. There are several character set such as ASCII and UTF-8.

3.2.5 Application Programming Interfaces - API

The API let you access the data from the internet in a resonable easy manner.

There are several API available in the internet for many social media:

- Facebook;
- Instagram;
- Twitter, etc.;

This lesson will use the Mediawiki, which is a Open Source API to Wikipedia.

Most of the file from the API are formated as JSON or XML.

3.2.6 JSON and XML

JSON stands for Javascript Object Notation and XML for Extensible Markup Language.

Sometimes the regular tabular way to structure the data is not a good solution, and for this reason, there are other forms to store data as JSON and XML.

They use a kind of "dictionary" to store data, which allows storing more than one information per variable.

There are some similarities in JSON and Python:

- JSON Array = Python list
- JSON Object = Python dictonary

3.2.7 Methods in this Lesson

3.2.7.1 `.find()`

This method is used to find tags and containers.

Example:

```
soup.find('title')
```

This code above will find the tag title, and return the content.

3.2.7.2 `.find_all()`

It is almost the same of `.find()`, but will find in all document the given pattern.

Example:

```
something.find_all('div')
```

This code will return all `div` in the document. It could be used with `limit = 1`, which will return the first `div`.

3.2.7.3 `.contents`

The `.contents` get the elements from the `find` and `find_all`.You are capable to select, which element you want (indexing).

```
something.find_all('div')[1].contents[2]
```

In this fragment of code, I am selecting only the third element of `something.find_all('div')[1]`.

3.2.7.4 `os.listdir()`

This function list all files inside a given folder/directory.

```
os.listdir(my_path)
```

3.2.7.5 `.glob()`

This method is a part of the glob package.

If you are familiar with Linux CLI, you have alread used the globbing to find a file in a folder.

```
import glob
glob.glob('any_folder/*.txt')
```

The result of the `.glob()` will be a list with all files which matches the `.txt`.

3.2.7.6 `.read()`

Convert the file into a in memory variable.

```
my_new_variable = file.read() # my_new_variable is a variable which contains the file.
```

3.2.7.7 `.readline()`

Read line by line every instance which is used this method.

```
file.readline() # Read the first line of the document file
file.readline() # Read the second line of the document file
file.read()     # Read the rest of the content.
```

3.2.7.8 `.DataFrame()`

This method from the pandas package converts a simple dictonary to a Pandas DataFrame.

```
pd.DataFrame(my_dataframe, columns = ['column_1', 'column_2', 'column_3'])
```

3.2.7.9 `.page()`

The `.page()` method from the wptools package converts a Wikipedia page into a object.

```
any_website = wptools.page('E.T._the_Extra-Terrestrial')
```

3.2.7.10 `.get()`

The `.get()` method from the wptools package extract all info from the wptools object.

```
any_website = wptools.page('E.T._the_Extra-Terrestrial').get()
```

3.3 Assessing Data

This is the second step of the Data Wrangling, and the aims of this lesson is to explain some details. There are two kind of *unclean* data:

- Quality issues: Dirty data;
  - Missing, duplicated, or incorrect data;
- Lack of tidiness: Also known as messy data.
  - Strucutural issues

There are two ways to assess:

- Visual: Plotting a simple graphic or visualizing the table (rows and columns);
- Programmatic: Using code to summarize the data frame using .info(), .describe(), average, summation, max, min, etc..

3.3.0.1 Dirty Data

Is related to the content issues, as known as low quality data.

- Innacurated data: Typos, corrupted, and duplicated data;

3.3.0.2 Messy Data

Messy data is related to structural issues, as known as untidy data.

- Each observation is a row;
- Each variable/features is a column;
- Based on the Hadley Wickham principles of tidy data.

3.3.1 Assess Proccess

In both cases (visual or programmatic), we could be divided into two main steps:

- Detect;
- Document.

3.3.2 Data Quality Dimensions

> Data quality dimensions help guide your thought process while assessing and also cleaning. The four main data quality dimensions are:

- Completeness: Missing values;
- Validity: Invalid value (like negative height or weight, zip code with only 4 digits, etc.);
- Accuracy: Wrong data which is valid (like the typo in the height);
- Consistency: Data without a standard notation (New York and NY, Colorado and CO, same information but differents notations).

The severity of this problem is decreasing order: Completeness, Validity, Accuracy, and Consistency.

3.4 Cleaning Data

Always opt to clean the data using the Programmatic way because manually it is more error prone.

This is the steps of Data Cleaning:

- Define: Defining a Data Cleaning Plan (usually writting down);
- Code: Converts the Data Cleaning Plan into code;
- Test: Evaluates the outuput of the code.

3.4.1 Tidiness

It is the standard preconized by Hadley Wickham.

Usually, the tidiness issues is the first to be solved.

3.4.2 Quality

After fixing tidiness issues, the quality issues could be fixed.

3.4.3 Methods

3.4.3.1 `.melt()`

Convert a wide format to a long format. It is the same of gather and spread functions from tidyr R package.

Good Video - Explaning the melt

3.5 Project 02 - Wrangle and Analyze Data

Project Submitted

Please, find below the URL to redirect to the project Jupyter Notebook.

Project 02 - WeRateDogs™ - Wrangle and Analyze Data

3.5.1 Project Submission

In this project, you'll gather, assess, and clean data then act on it through analysis, visualization and/or modeling.

Before you submit:

1. Ensure you meet specifications for all items in the Project Rubric. Your project "meets specifications" only if it meets specifications for all of the criteria.

2. Ensure you have not included your API keys, secrets, and tokens in your project files.

3. If you completed your project in the Project Workspace, ensure the following files are present in your workspace, then click "Submit Project" in the bottom righthand corner of the Project Workspace page:

- `wrangle_act.ipynb`: code for gathering, assessing, cleaning, analyzing, and visualizing data
- `wrangle_report.pdf` or wrangle_report.html: documentation for data wrangling steps: gather, assess, and clean
- `act_report.pdf` or `act_report.html`: documentation of analysis and insights into final data
- `twitter_archive_enhanced.csv`: file as given
- `image_predictions.tsv`: file downloaded programmatically
- `tweet_json.txt`: file constructed via API
- `twitter_archive_master.csv`: combined and cleaned data
- any additional files (e.g. files for additional pieces of gathered data or a database file for your stored clean data)

4. If you completed your project outside of the Udacity Classroom, package the above listed files into a zip archive or push them from a GitHub repo, then click the "Submit Project" button on this page.

As stated in point 4 above, you can submit your files as a zip archive or you can link to a GitHub repository containing your project files. If you go with GitHub, note that your submission will be a snapshot of the linked repository at time of submission. It is recommended that you keep each project in a separate repository to avoid any potential confusion: if a reviewer gets multiple folders representing multiple projects, there might be confusion regarding what project is to be evaluated.

It can take us up to a week to grade the project, but in most cases it is much faster. You will get an email once your submission has been reviewed. If you are having any problems submitting your project or wish to check on the status of your submission, please email us at review-support@udacity.com. In the meantime, you should feel free to proceed with your learning journey by continuing on to the next module in the program.

Chapter 4

Advanced Statistics

4.1 Descriptive Statistics

This lesson is a kind of review.

4.1.1 What is data?

Data could be whatever thing: Text, Spreadsheets, video, images, database, etc.

4.1.2 Data types

- Quantitative Data: Allow us to perform mathematical operations with data (1, 2, 3, 4, etc.);
  - Continuous: Could be any real number (Age);
  - Discrete: Only integer number (Number of persons);
- Categorical Data: Used to label a group or a set of items (blue, yellow, red, etc.);
  - Ordinal: There are a way to put the categories in a scale (Very good, so-so, very poor);
  - Nominal: It is impossible to put the categories in order (blue, yellow, orange, etc.);

4.1.3 Measures of Center

4.1.3.1 Categorical

The categorical data is analyzed doing a simple summary to count the total of each category has.

4.1.3.2 Quantitative

Four main aspects when analysing **quantitative** data:

- Measures of `Center`
  - Mean
  - Median: The median splits our data so that 50% of our values are lower and 50% are higher.
    * Even number of elements: single values.

    * Odd number of elements: average between two "center" values.
  - Mode: The mode is the most frequently observed value in our dataset.
    * No mode: If all observations in our dataset are observed with the same frequency, there is no mode.
    * Many modes: If two (or more) numbers share the maximum value, then there is more than one mode.

- Measures of `Spread`
- The `Shape` of the Data
- `Outliers`

4.2 Quantitative Data

4.2.1 Measures of Spread

> How far are points from one another.

Common values of spread:

- Range;
- Interquartile range (IQR);
- Standard Deviation, and;
- Variance.

4.2.1.1 Histogram

Figure 1 shows an example of histogram.

This is a way to visualize the quantitative data.

4.2.2 Five Number Summary

These are the number:

- Maximum;
- Third quartile or Q3 (75%);
- Second quartile (it is the same of mean) or Q2 (or 50%);
- First quartile or Q1 (25%), and;
- Minimum.

First step to do is order the values, as you can see in Figure 2 (odd set of values).

As you can see, Q1 and Q3 are the median of the data on either sides of Q2.

The range is defined as:

$$\text{Range} = maximum - minimum \tag{1}$$

The Interquartile is define as:

$$\text{Interquartile} = Q3 - Q1 \tag{2}$$

For a even set of values I need to calculate the "average" of two values.

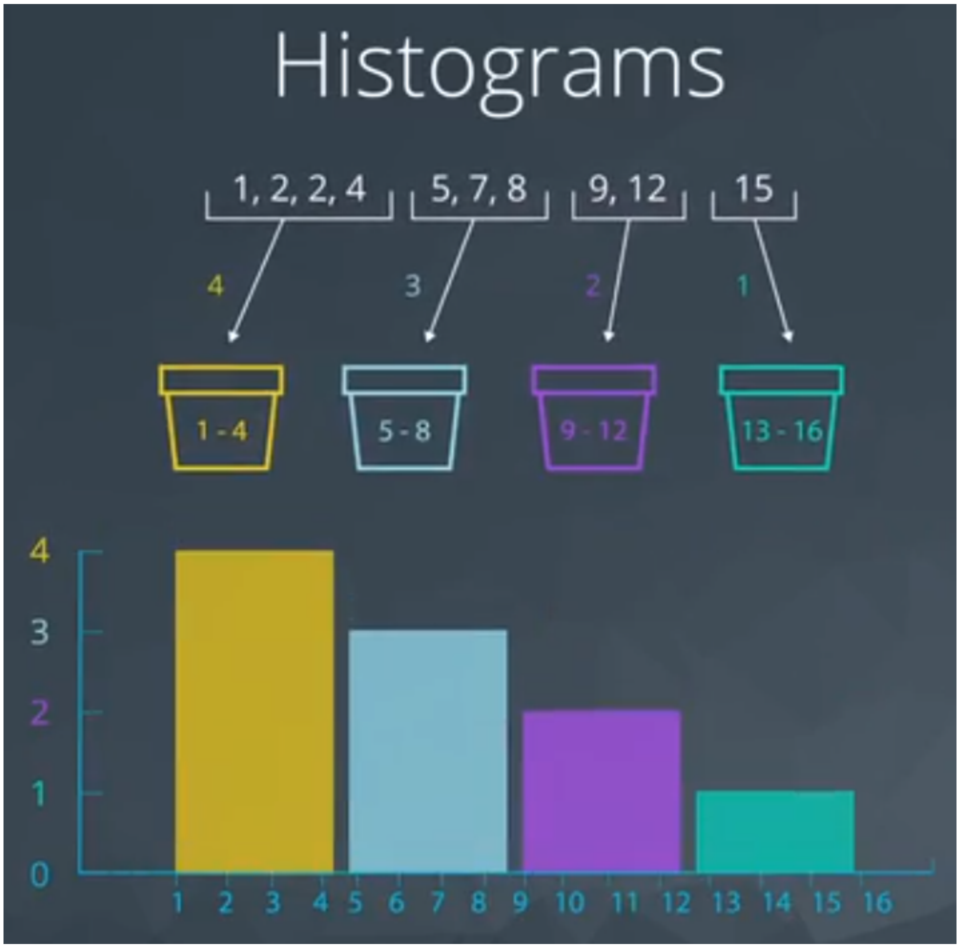

Figure 4.1:

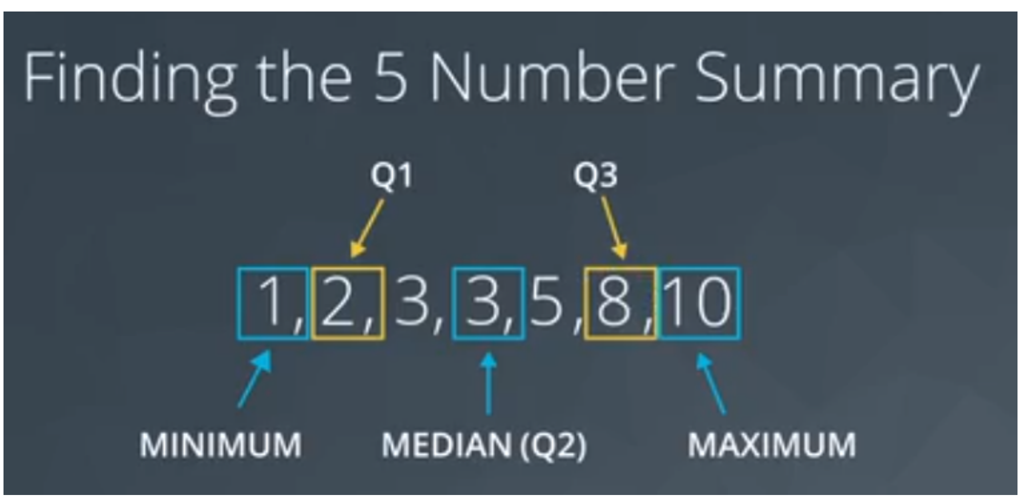

Figure 4.2:

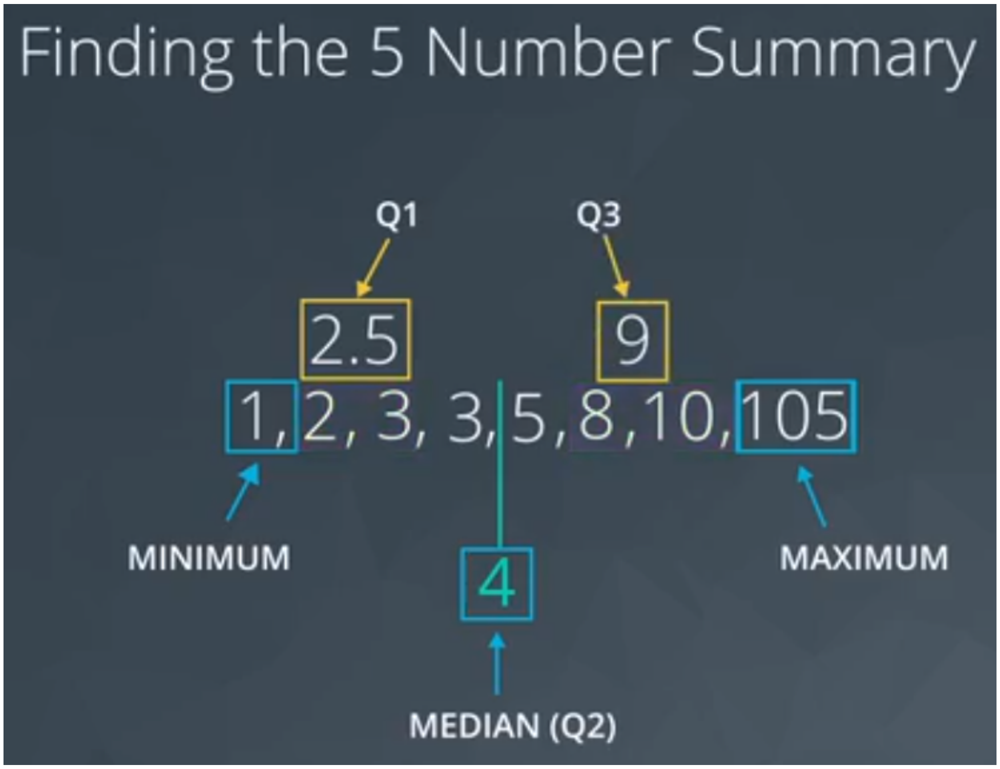

Figure 4.3:

4.2.2.1 Boxplot

The boxplot graphic is a way to visualize the spread of the data.

It could be useful for quickly comparing the spread of two data sets.

Based on the Figure 4, the graphic on the right shows that in the weekends the number of dogs varies much more than on weekdays (looking to the range).

4.2.3 Standard Deviation

Meaning: On average, how much each point varies from the mean of the points.

First, I need to define the "distance" between mean and each observation. "Distance" could be interpreted as the difference of these two values. The issue observed in this difference are positive and negative values. For this reason, the square is used to turn everything positive (because later I can square root).

- Standard Deviation is frequently used to compare spread of different groups.
- Having higher standard deviation is associated with having higher risk.

$$\text{Standard Deviation} = \frac{1}{n}\sum_{i=1}^{n}(\bar{x} - x_i)^2 \tag{3}$$

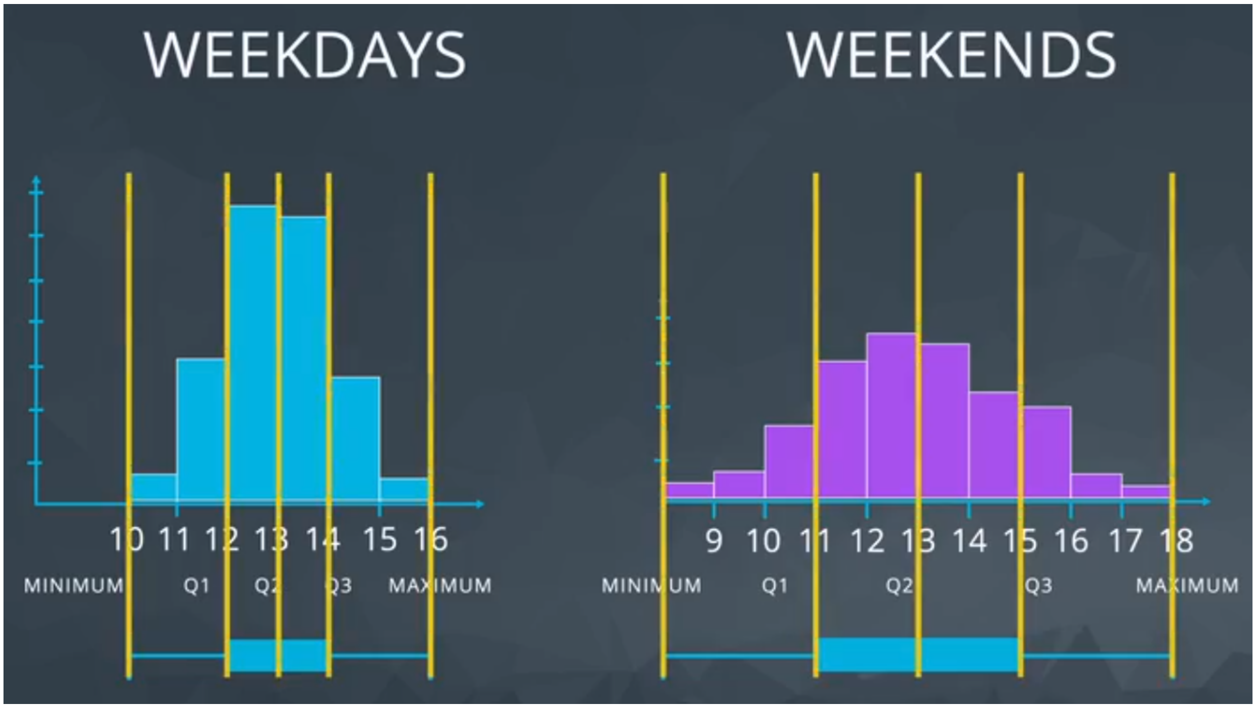

Figure 4.4:

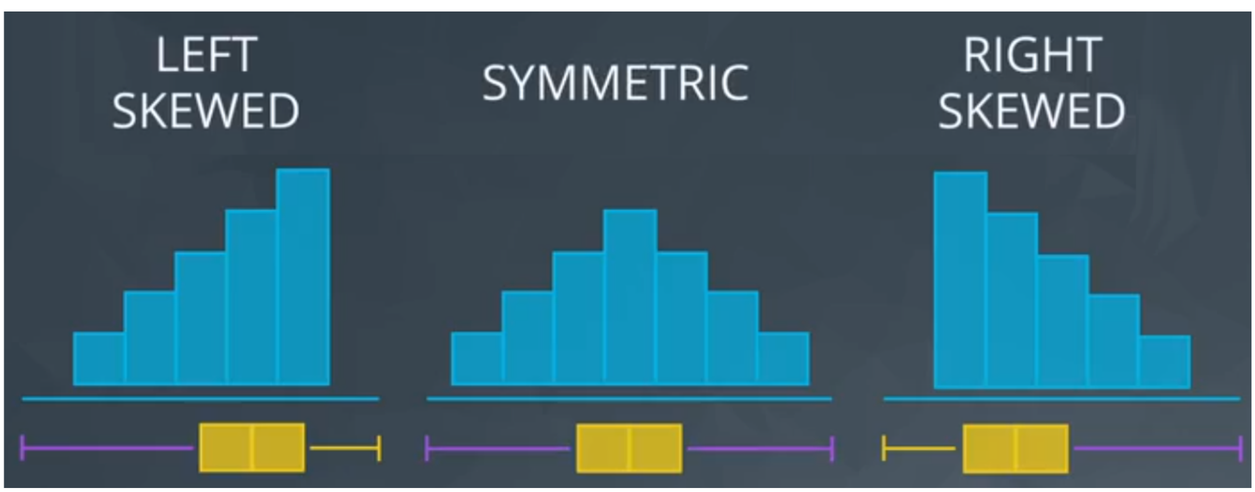

Figure 4.5:

4.2.4 Shape

The shape is related to the histogram form, Figure 5 shows an example.

- Left Skewed
  - is pulled to the "begining"
  - median stays close to the mode
  - GPA, Age of death, Asset price changes
- Simmetric (example: Normal distribution or bell curve)
  - mean = median = mode;
  - Examples: heights, weights, scores, precipitation, etc.
- Right Skewed
  - mean is pulled to the tail
  - median stays close to the mode
  - Amount of drug left in your bloodstream over time, distribution of wealth, human athletic abilities.

Side note: If you aren't sure if your data are normally distributed, there are plots called normal quantile plots and statistical methods like the Kolmogorov-Smirnov test that are aimed to help you understand whether or not your data are normally distributed. Implementing this test is beyond the scope of this class, but can be used as a fun fact.

4.2.5 Outliers

Data points thah fall very far from the rest of the values in our dataset.

The "very far" is quite generic and could be interpreted in many forms. One way to visualize it is plotting a histogram, as you can see in Figure 6.

1. Note they exist and the impact on summary Statistics
2. If typo, remove or fix it.
3. Understand why they exist, and the impact on questions we are trying to answer
4. Reporting the 5 number summary is better than mean and standard deviation when outliers are present
5. Be careful in reporting know how to ask the right questions

4.2.6 Descriptive vs Inferential

Descriptive Statistics: Describing Collected Data Inferential Statistics: Drawing conclusions about a population based on data collected from sample of individuals from that population.

4.3 Admissions Case Study

4.3.1 Simpson's Paradox

In this example lesson, you learned about Simpson's Paradox, and you had the opportunity to apply it to a small example with Sebastian, as well as work through similar example in Python.

Figure 4.6:

In the lessons ahead, you will be learning a lot by following along with Sebastian, but it is really important to put these ideas to practice using data and computing, because that is how you will apply these skills in a day to day environment as a Data Analyst or Data Scientist.

It is so easy to get caught up in looking at full aggregates of your data. Hopefully, the examples here serve as a reminder to look at your data multiple ways.

4.3.2 Case Study

4.4 Probability

4.4.1 Introduction to Probability

Do not confound Statistics and Probability.

- Probability: Make preditcions about the future events based on models, and;
  - Here I want to predict data!
- Statistics: Analyze data from past events to infer what those models or causes could be.
  - Here I use data to preditc!

Figure 1 shows the relation between these two subjects.

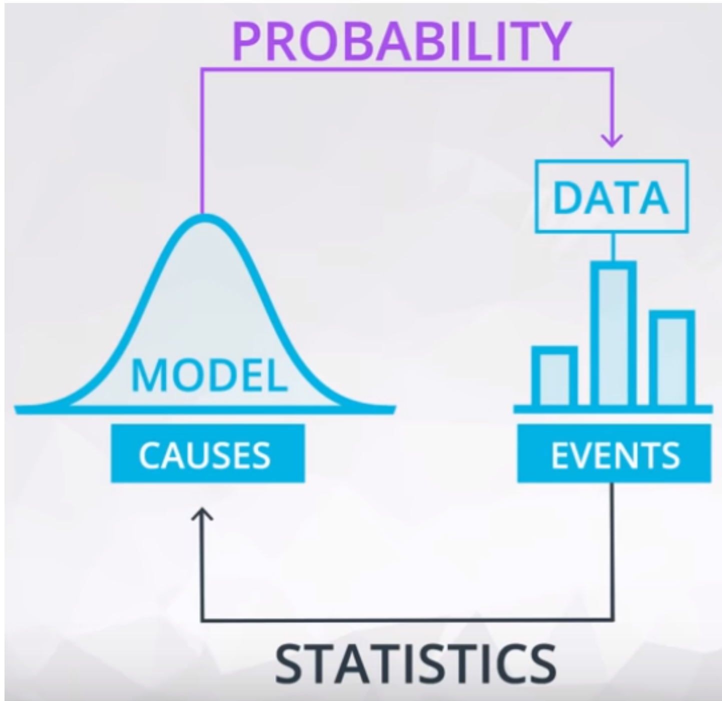

Figure 4.7:

4.4.1.1 Fair Coin

The probability notation is based on the 0 to 1 scale, where 0 means zero percentage and 1 means 100 percentage. The example below is a 50%.

$$P(HEADS) = 0.5$$

To be a fair coin the tail probability it is the same of heads.

$$P(TAILS) = 0.5$$

4.4.1.2 Loaded Coin

Its occurs when the $P(HEADS)$ is different of $P(TAILS)$. Bear in mind, in the equation 1.

$$P(HEADS) + P(TAILS) = 1 \tag{1}$$

Example 1: {HEADS, HEADS} = $P(H, H)$ for a fair coin.

$P(H) = P(T) = 0.5$

To ilustrate this solution, let's draw the Truth Table (Table 1)

Table 1 - Truth Table for a Fair Coin

Flip 1	Flip 2	Probability
H	H	$ 0.5 * 0.5 = 0.25 $

Flip 1	Flip 2	Probability
H	T	$ 0.5 * 0.5 = 0.25 $
T	H	$ 0.5 * 0.5 = 0.25 $
T	T	$ 0.5 * 0.5 = 0.25 $
		$ SUM = 1.0 $

The probability of P(H, H) is 0.25.

Example 2: {HEADS, HEADS} = $P(H, H)$ for a loaded coin.

$ P(H) = 0.6 $ $ P(T) = 0.4 $

To ilustrate this solution, let's draw the Truth Table (Table 2)

Table 2 - Truth Table for a Loaded Coin

Flip 1	Flip 2	Probability
H	H	$ 0.6 * 0.6 = 0.36 $
H	T	$ 0.6 * 0.4 = 0.24 $
T	H	$ 0.4 * 0.6 = 0.24 $
T	T	$ 0.4 * 0.4 = 0.16 $
		$ SUM = 1.0 $

The probability of P(H, H) is 0.36.

Example 3: Three coins flipped. What is the probability of only one heads in three coins flipped. Adopting a fair coin ($ P(H) = 0.5 $).

$P_1(OnlyoneH)$

Table 3 - Truth Table for a Loaded Coin

Flip 1	Flip 2	Flip 3	Probability	Has only one heads?	P_1
H	H	H	$ 0.5 * 0.5 * 0.5 = 0.125 $	No	0
H	H	T	$ 0.5 * 0.5 * 0.5 = 0.125 $	No	0
H	T	H	$ 0.5 * 0.5 * 0.5 = 0.125 $	No	0
H	T	T	$ 0.5 * 0.5 * 0.5 = 0.125 $	Yes	0.125
T	H	H	$ 0.5 * 0.5 * 0.5 = 0.125 $	No	0
T	H	T	$ 0.5 * 0.5 * 0.5 = 0.125 $	Yes	0.125
T	T	H	$ 0.5 * 0.5 * 0.5 = 0.125 $	Yes	0.125
T	T	T	$ 0.5 * 0.5 * 0.5 = 0.125 $	No	0
			$ SUM = 1.0 $	$ SUM = 3 cases $	$ SUM = 0.375 $

The $ P_1 $ is 0.375.

Example 4: Three coins flipped. What is the probability of only one heads in three coins flipped. Adopting a loaded coin ($ P(H) = 0.6 $).

$P_2(OnlyoneH)$

Table 3 - Truth Table for a Loaded Coin

Flip 1	Flip 2	Flip 3	Probability	Has only one heads?	P_2
H	H	H	$ 0.6 * 0.6 * 0.6 = 0.216 $	No	0

Flip 1	Flip 2	Flip 3	Probability	Has only one heads?	P_2
H	H	T	$ 0.6 * 0.6 * 0.4 = 0.144 $	No	0
H	T	H	$ 0.6 * 0.4 * 0.6 = 0.144 $	No	0
H	T	T	$ 0.6 * 0.4 * 0.4 = 0.096 $	Yes	0.096
T	H	H	$ 0.4 * 0.6 * 0.6 = 0.144 $	No	0
T	H	T	$ 0.4 * 0.6 * 0.4 = 0.096 $	Yes	0.096
T	T	H	$ 0.4 * 0.4 * 0.6 = 0.096 $	Yes	0.096
T	T	T	$ 0.4 * 0.4 * 0.4 = 0.064 $	No	0
			$ SUM = 1.0 $	$ SUM = 3 cases $	$ SUM = 0.288 $

The $ P_2 $ is 0.288.

4.4.2 Bernoulli Distribution

Founded on this introduction, let's generalize this concept using the Bernoulli Distribution.

> In probability theory and statistics, the Bernoulli distribution, named after Swiss mathematician Jacob Bernoulli, is the discrete probability distribution of a random variable which takes the value 1 with probability p and the value 0 with probability $q = 1 - p$, that is, the probability distribution of any single experiment that asks a yes–no question; the question results in a boolean-valued outcome, a single bit of information whose value is success/yes/true/one with probability p and failure/no/false/zero with probability q. It can be used to represent a (possibly biased) coin toss where 1 and 0 would represent "heads" and "tails" (or vice versa), respectively, and p would be the probability of the coin landing on heads or tails, respectively. In particular, unfair coins would have $p \neq 1/2$.
>
> The Bernoulli distribution is a special case of the binomial distribution where a single trial is conducted (so n would be 1 for such a binomial distribution). It is also a special case of the two-point distribution, for which the possible outcomes need not be 0 and 1. – Wikipedia

Rede more in wolfram.

4.4.2.1 Summary

> Here you learned some fundamental rules of probability. Using notation, we could say that the outcome of a coin flip could either be T or H for the event that the coin flips tails or heads, respectively.
>
> Then the following rules are true:

- Probability of a Event >

$$\mathbf{P}(\mathbf{H}) = 0.5$$

- Probability of opposite event >

$$\mathbf{1} - \mathbf{P}(\mathbf{H}) = \mathbf{P}(\text{not H}) = 0.5$$

> where not H is the event of anything other than heads. Since, there are only two possible outcomes, we have that $\mathbf{P}(\text{not H}) = \mathbf{P}(\mathbf{T}) = 0.5$. In later concepts, you will see this with the following notation: $\neg\mathbf{H}$.

- Probability of composite event

$$P * P * P * \cdots * P$$

It is only true because the events are independent of one another, which means the outcome of one does not affect the outcome of another.

- Across multiple coin flips, we have the probability of seeing n heads as $\mathbf{P(H)^n}$. This is because these events are independent.

We can get two generic rules from this:

1. The probability of any event must be between 0 and 1, inclusive.

2. The probability of the compliment event is 1 minus the probability of an event. That is the probability of all other possible events is 1 minus the probability an event itself. Therefore, the sum of all possible events is equal to 1.

3. If our events are independent, then the probability of the string of possible events is the product of those events. That is the probability of one event AND the next AND the next event, is the product of those events.

4.4.2.2 Looking Ahead

You will be working with the Binomial Distribution, which creates a function for working with coin flip events like the first events in this lesson. These events are independent, and the above rules will hold. from Text: Recap + Next Steps

4.5 Conditional Probability

Here the first event will affect the second one. Figure 1 shows an example of it.

Figure 1 - Example of conditional probability.

The first event is to determine the bird type, and the second event the probability to run on the morning. Have in mind, these two birds has different probability to run on the morning.

- The early bird has 0.02;
- The night owl has 0.00.

4.5.1 Medical Example

Supose a patient with a disease, the probability of this patient has cancer is 0.9 and to be free cancer is 0.1.

$$P(cancer) = 0.1 P(\neg\ cancer) = 0.9 \tag{1}$$

To be honest, we do not know if this patient has cancer, so it is necessary to apply a test. This test is not perfect, it means, there are a probability to indicates a false positive and a false negative.

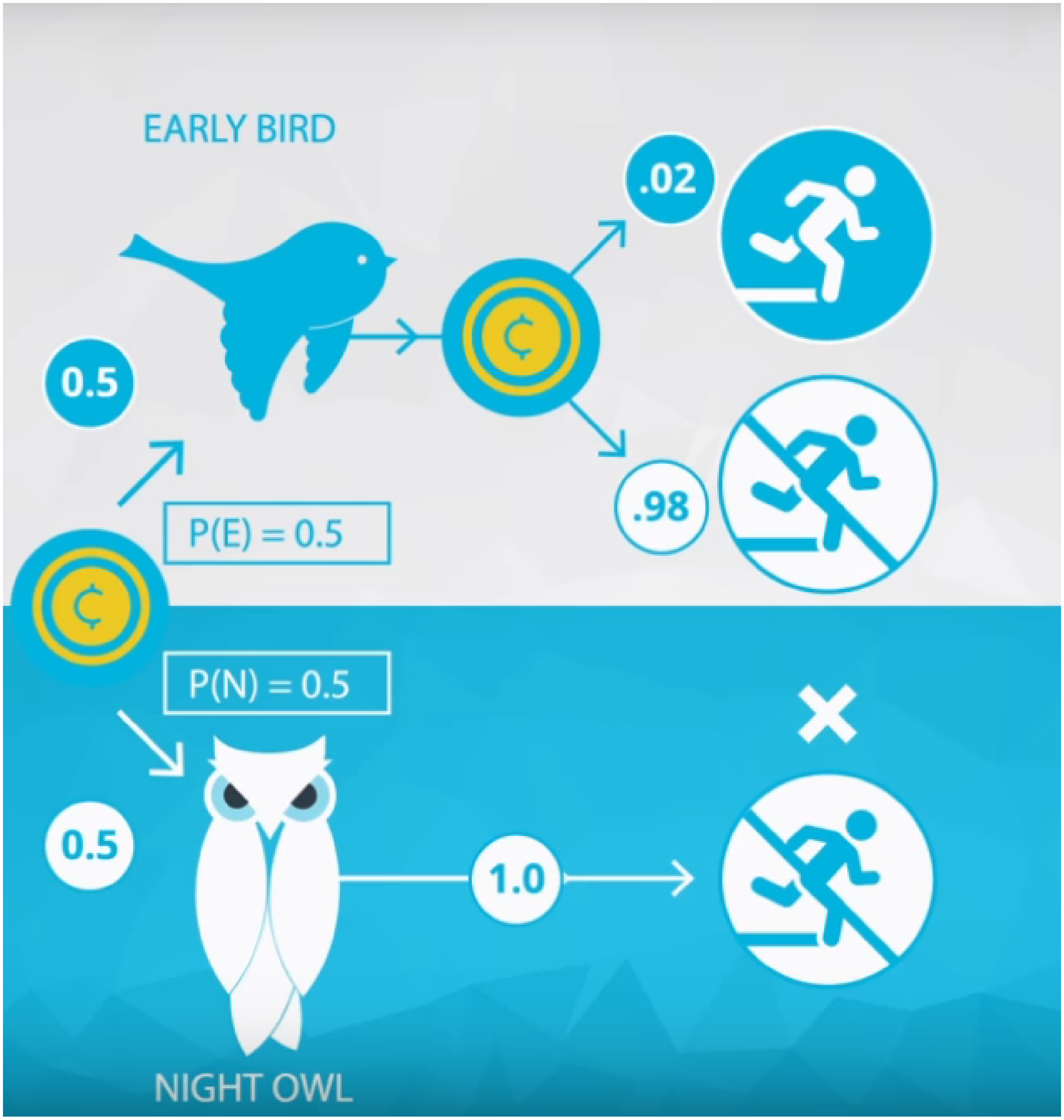

Figure 4.8: Figure 1

For this reason, I introduce the conditional probability.

$$P(Positive|cancer) = 0.9 \tag{2}$$

What is the meaning of this notation?

Given the patient has cancer, the probability of this test indicates positive is 0.9. Thus, given the patient has cancer and the test indicates negative is 0.1, as shown in equation (3).

$$P(Negative|cancer) = 0.1 \tag{3}$$

Analogous to the case of the patient do not has cancer.

$$P(Positive|\neg\ cancer) = 0.2 P(Negative|\neg\ cancer) = 0.8 \tag{2}$$

Table 1 shows a representation in a tabular way.

Table 1 - Truth Table for Medical Example

Disease	Test	$P_{disease}$	P_{test}	P	Q1: Test Positive?	Q1: Answer
No	Negative	$P(\neg\ cancer) = 0.9$	$P(Negative\Vert\neg\ cancer) = 0.8$	0.72	No	0
No	Positive	$P(\neg\ cancer) = 0.9$	$P(Positive\Vert\neg\ cancer) = 0.2$	0.18	Yes	0.18
Yes	Negative	$P(cancer) = 0.1$	$P(Negative\Vert cancer) = 0.1$	0.01	No	0
Yes	Positive	$P(cancer) = 0.1$	$P(Positive\Vert cancer) = 0.9$	0.09	Yes	0.09
				$SUM =$ 1		$SUM =$ 0.27

What is the probability the test is positive?

Q1: 0.27

4.5.1.1 Coin flip example

Two coins, one fair and other loaded.

- Coin 1: $P_1(HEADS) = P_1(TAILS) = 0.5$;
- Coin 2: $P_2(HEADS) = 0.9$ and $P_2(TAILS) = 0.1$.

Figure 2 - Coin Example of conditional probability.

What is the probability of this sequence HEADS and TAILS?

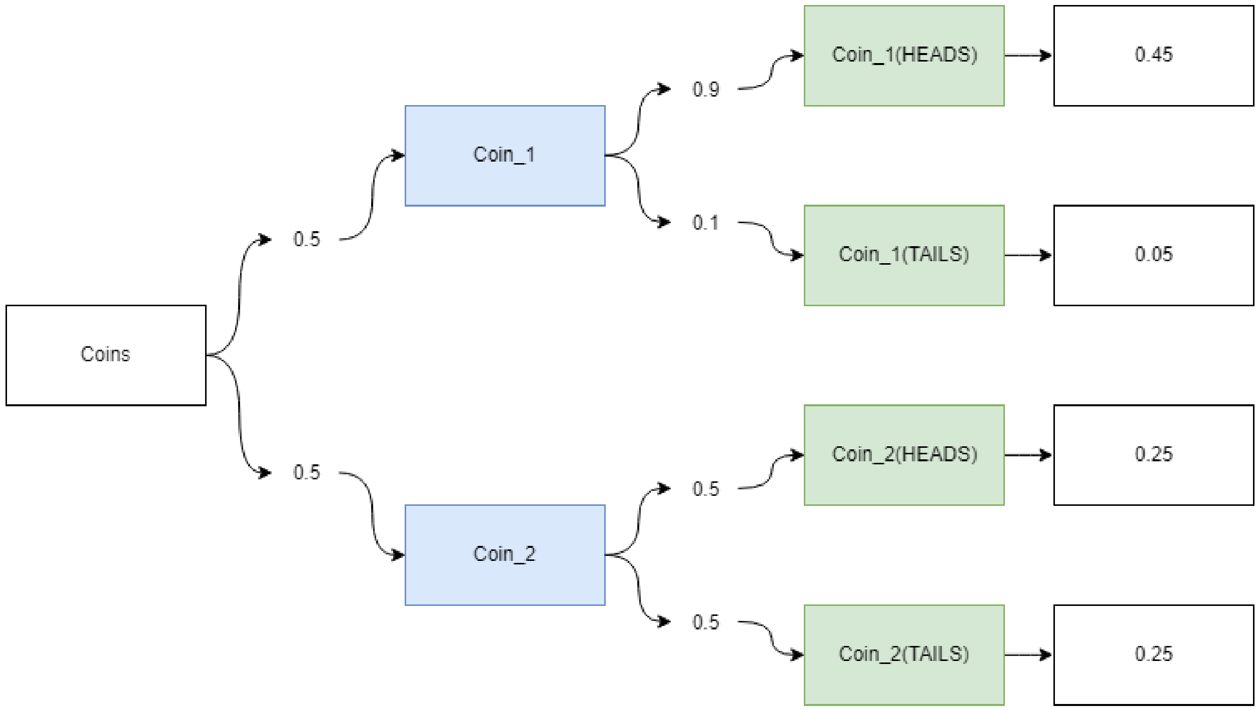

Figure 4.9: Figure 2

Coin	Flip 1	Flip 2	P_{coin}	P_{Flip1}	P_{Flip2}	P	Q2: HEADS then TAILS?	Q2: answer
1	H	H	0.5	0.9	0.9	0.405	No	0
1	H	T	0.5	0.9	0.1	0.045	Yes	0.045
1	T	H	0.5	0.1	0.9	0.045	No	0
1	T	T	0.5	0.1	0.1	0.005	No	0
2	H	H	0.5	0.5	0.5	0.125	No	0
2	H	T	0.5	0.5	0.5	0.125	Yes	0.125
2	T	H	0.5	0.5	0.5	0.125	No	0
2	T	T	0.5	0.5	0.5	0.125	No	0
						$SUM =$ 1		$SUM =$ 0.170

4.5.1.2 Summary

In this lesson you learned about conditional probability. Often events are not independent like with coin flips and dice rolling. Instead, the outcome of one event depends on an earlier event.

For example, the probability of obtaining a positive test result is dependent on whether or not you have a particular condition. If you have a condition, it is more likely that a test result is positive. We can formulate conditional probabilities for any two events in the following way:

- $P(A|B) = \frac{P(A \cap B)}{P(B)}$
- $P(AB)$

In this case, we could have this as:

$$P(positive|disease) = \frac{P(\text{positive } \cap \text{ disease})}{P(disease)}$$

where represents “given” and ∩ represents “and”. — Class notes - Text: Summary

Chapter 5

Intro to Machine Learning

We have finished a nice book.

Chapter 6

Data Visualization (Optional)

We have finished a nice book.