Lecture #4: Intro to SQL + PostgreSQL

Date: Monday, October 1, 2018
Lecturer: Alex Nakagawa

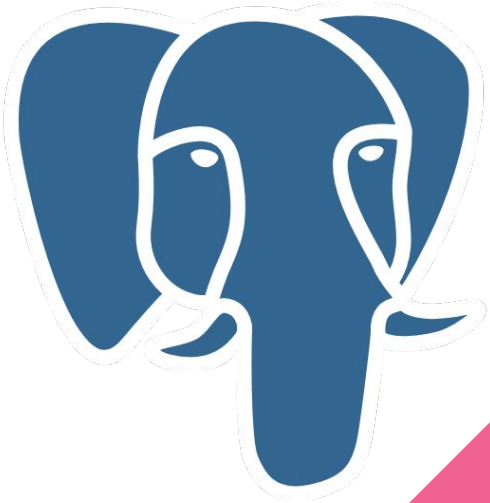
Special Slide Credits to: Paul Bitutsky, Alexander Ivanoff & Heather Chen

Announcements

- Instant Runoff Voting Project is out! Due October 14, 11:59:59 PM
- All knowledge required will be explained in next two lectures
- Piazza is your friend! 💪
- Utilize the Datacamp account

Lesson Plan

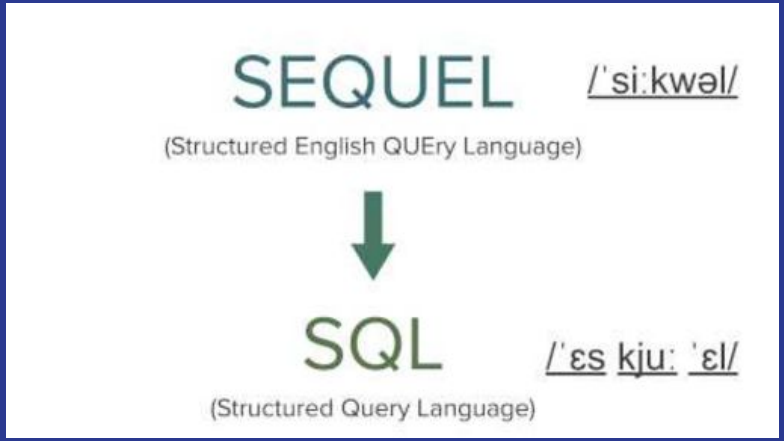
- Intro + PostgreSQL
- SQL Syntax / Basic Commands: CREATE, INSERT, SELECT, ALTER TABLE, UPDATE, DELETE, DROP (JOINS)
- Project Introduction



Quick Survey

https://yellkey.com/single

Introduction

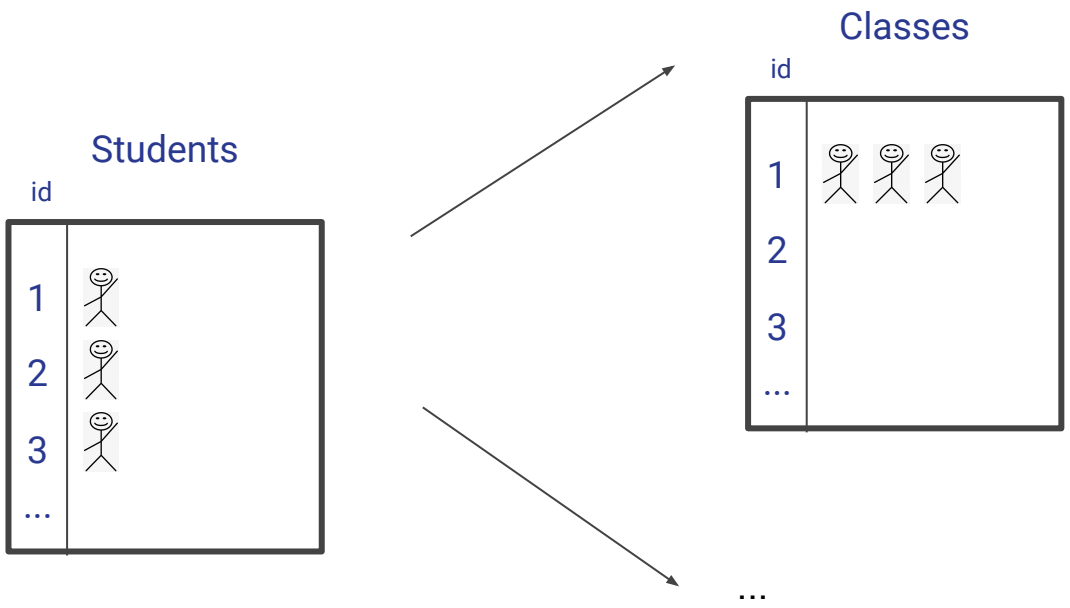


Database Management System (DBMS)

➔ The software (+ server) that helps store and manipulate data for individuals, companies, etc.

AND MORE

The Relational Database Model



How to recognize it:
Information is spread across multiple relational schema

Pros:

- More efficient
- Schema represent real-world relationships

Cons

- Requires complex db system to process

vs. ... Tabular Database Model

How to recognize it:
All information for a single record is contained in a row for that record

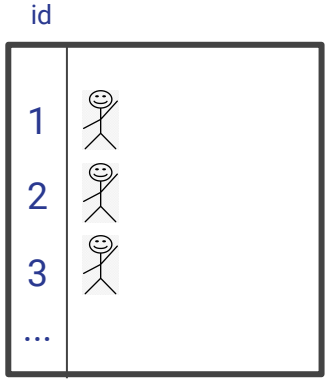
Pros:

- Easy to read

Cons:

- Sparse values
- Takes up more space/is slower

Students + Teacher + Class + ...



KEY

Primary Key

- The primary key consists of one or more columns whose data contained within is used to *uniquely identify each row* in the table.
- i.e. mailing address

Foreign Key

- The foreign key is used to identify a row from a different table in the database.

The Structured Query Language (SQL)

```
SELECT      ptype, AVG(cost)
FROM        products
WHERE       cost > 50
GROUP BY    ptype
ORDER BY    cost
```

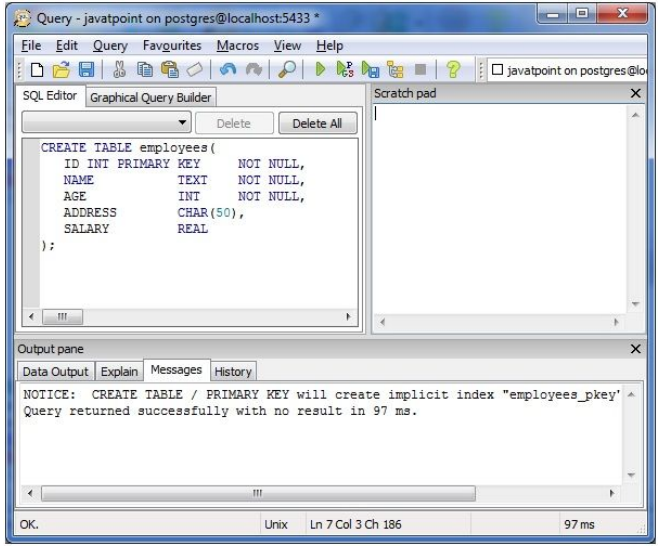
What you want in your table

What table is your data from?

A “filter” for data in your table

SQL Technologies

Starting from scratch?



From existing data sources?

Answer: Depends on the DBMS

Schemas

- Need to clearly identify the tables we're working with and the structure of the data
- SQL requires defined **schemas,** (a structure) for databases.

Number of petals	Name	Color
8	lotus	pink
34	sunflower	yellow
5	rose	red

Schemas

This table is named **flowers** and has three columns: **Number of petals, Name,** and **Color.**

Every entry that goes into this table must be in the format “Number of petals, Name, Color” and must follow the type specified by that column. Number of petals is an **integer**, and Name and Color are both **strings**.

Number of petals	Name	Color
8	lotus	pink
34	sunflower	yellow
5	rose	red

Syntax

CREATE TABLE

```sql
CREATE TABLE Cal_Sports (
    ID        INT,
    Sport     VARCHAR (255),
    Gender    VARCHAR (255),
    Season    VARCHAR (255)
);
```

ID	Sport	Gender	Season

INSERT

```sql
INSERT INTO Cal_Sports
(ID, Sport, Gender, Season)
VALUES
(1, 'Baseball', 'Mens', 'Spring');
```

ID	Sport	Gender	Season
1	Baseball	Mens	Spring

Data can be inserted in any column order

If you want, you can skip a column, and then that cell will contain a predetermined default value or NULL (depending on database)

INSERT (cont.)

```
INSERT INTO Cal_Sports
(ID, Sport, Gender, Season)VALUES
(2, 'Basketball', 'Mens', 'Winter');
INSERT INTO Cal_Sports
(ID, Sport, Gender, Season) VALUES
(3, 'Basketball', 'Basketball',
'Winter');
INSERT INTO Cal_Sports
(ID, Sport, Gender, Season) VALUES
(4, 'Beach Volleyball', 'Womens',
'Spring');
```

ID	Sport	Gender	Season
1	Baseball	Mens	Spring
2	Basketball	Mens	Winter
3	Basketball	Womens	Winter
4	Beach Volleyball	Womens	Spring

SELECT

```
SELECT *
FROM Cal_Sports
```

ID	Sport	Gender	Season
1	Baseball	Mens	Spring
2	Basketball	Mens	Winter
3	Basketball	Womens	Winter
4	Beach Volleyball	Womens	Spring

Tip: Use SELECT * to select all columns

SELECT

```
SELECT Sport, Gender, Season
FROM Cal_Sports
WHERE Season = 'Winter'
ORDER BY Sport DESC
```

	Sport	Gender	Season
	Basketball	Mens	Winter
3	Basketball	Womens	Winter

ALTER TABLE

```
ALTER TABLE Cal_Sports
ADD Ticket_Price int
```

ID	Sport	Gender	Season	Ticket_Price
1	Baseball	Mens	Spring	[null]
2	Basketball	Mens	Winter	[null]
3	Basketball	Womens	Winter	[null]
4	Beach Volleyball	Womens	Spring	[null]

UPDATE

```
UPDATE Cal_Sports
SET Ticket_Price = 0
WHERE id = 4 OR id = 1;

UPDATE Cal_Sports
SET Ticket_Price = 25
WHERE Sport IN ('Basketball');
```

ID	Sport	Gender	Season	Ticket_Price
1	Baseball	Mens	Spring	0
2	Basketball	Mens	Winter	25
3	Basketball	Womens	Winter	25
4	Beach Volleyball	Womens	Spring	0

DELETE

```sql
DELETE FROM Cal_Sports
WHERE id = 1;
```

ID	Sport	Gender	Season	Ticket_Price
2	Basketball	Mens	Winter	25
3	Basketball	Womens	Winter	25
4	Beach Volleyball	Womens	Spring	0

DROP

```sql
DROP TABLE Cal_Sports;
```

ID				Ticket_Price
2	Basketb		ter	25
3	Basket		er	25
4	Vo			0

Categorize the Commands

Which commands can alter the rows/columns of the **original** table?

SELECT

CREATE TABLE

DELETE

INSERT

UPDATE

ALTER TABLE

Categorize the Commands

Which commands can alter the rows/columns of the **<u>original</u>** table?

~~SELECT~~

CREATE TABLE

DELETE

INSERT

UPDATE

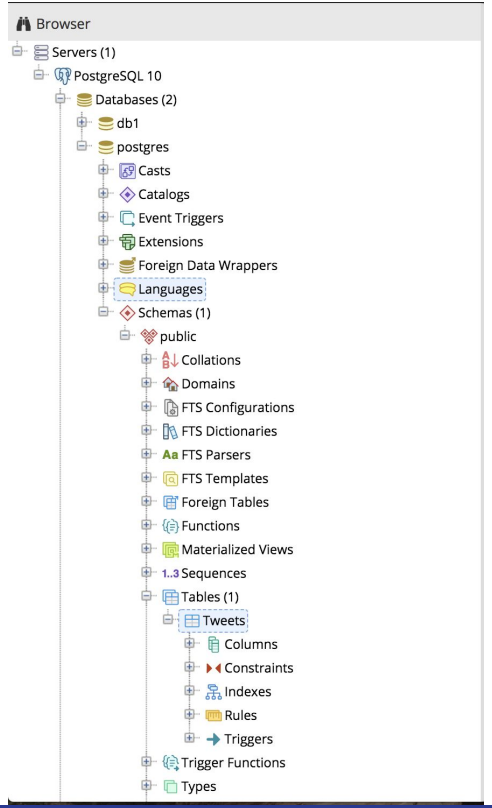
ALTER TABLE

Practice with Tweets

Download: https://www.kaggle.com/crowdflower/twitter-airline-sentiment

Demo!

Understanding our database



1. Create a New Database in PostgreSQL 10 Server, call it **‘db1’**
2. Scroll down to **‘Schemas’**. There should be a public one ready for use.
3. Open up the **‘public’** schema. Create a new Table called ‘**Tweets**’.
4. Add the relevant columns from our csv. ← DEMO!

Columns of Tweets

- Tables (1)
  - Tweets
    - Columns (15)
      - tweet_id
      - airline_sentiment
      - airline_sentiment_confidence
      - negativereason
      - negativereason_confidence
      - airline
      - airline_sentiment_gold
      - name
      - negativereason_gold
      - retweet_count
      - text
      - tweet_coord
      - tweet_created
      - tweet_location
      - user_timezone

Practicing SELECT

It’s time to start working with our Twitter Airline Sentiment database!

Exercise: Write a **query** to **select** the columns tweet_id, airline_sentiment, airline, and tweet_created

```
SELECT tweet_id, airline_sentiment, airline, tweet_created
FROM Public.”Tweets”;
```

WHERE ...

Only returns records for which the boolean/filter is true

> , >= , = , < , <= , <>

e.g.

SELECT * FROM houses

WHERE bedrooms = 3;

	price	sqft	lotsize	bedrooms
2	171967.681947	1329	14267	3
3	244162.229612	2110	11160	3
4	189831.097588	1629	13830	3

Practicing WHERE

1. Write a query to select the text of tweets where the airline is United
2. Write a query to select the retweet count and text of the negative tweets

```
SELECT text FROM Public.”Tweets” WHERE airline = “United”;
```

```
SELECT retweet_count, text FROM Public.”Tweets” WHERE
airline_sentiment = “negative”;
```

ORDER BY … (ASC/DESC)

- Sorts results in ASCending or DESCending order according to the column specified
- Ascending/smallest first is **default**, meaning if you want results in ascending order, you don't need to specify
- If you want results in decreasing order, y ne ORDER BY clause

E.g.

```
SELECT * FROM houses
ORDER BY sqft;
```

	price	sqft	lotsize	bedrooms
1	104855.329819	896	11622	2
2	171967.681947	1329	14267	3
4	189831.097588	1629	13830	3
3	244162.229612	2110	11160	3

Practicing ORDER BY

Exercise: Write a query that selects **airline sentiment** and **airline sentiment confidence** for all tweets about **US Airways**, sorted by **decreasing** highest confidence

```
SELECT airline_sentiment,
airline_sentiment_confidence
FROM Public.”Tweets”
WHERE airline = “US Airways”
ORDER BY airline_sentiment_confidence DESC;
```

Aggregates

Aggregate:

Def: A whole formed by combining several (typically disparate) elements

GROUP BY bedrooms

	price	sqft	lotsize	bedrooms
1	104855.329819	896	11622	2
2	171967.681947	1329	14267	3
3	244162.229612	2110	11160	3
4	189831.097588	1629	13830	3

	price	sqft	lotsize	bedrooms
1	104855.329819	896	11622	2

These are aggregated

	price	sqft	lotsize	bedrooms
2	171967.681947	1329	14267	3
3	244162.229612	2110	11160	3
4	189831.097588	1629	13830	3

SUM(price), SUM(sqft), SUM(lotsize)

	price	sqft	lotsize
bedrooms			
2	104855.329819	896	11622
3	605961.009146	5068	39257

GROUP BY ...

GROUP BY defaults to returning a count for each aggregated column but can also be specified with a different aggregate function.

SUM, COUNT, AVG, MIN, MAX

And you can write your own!

(P.S. You can only apply aggregate functions to non-grouped columns)

Practice GROUPING

Exercise: Write a query that returns a table with a row for each **airline** with the **count** of the tweets related to it.

```
SELECT airline, COUNT(*)
FROM Public.”Tweets”
GROUP BY airline;
```

HAVING …

Having has the same functionality as WHERE but is used in the aggregate, not final result

	price	sqft	lotsize	bedrooms
1	104855.329819	896	11622	2
2	171967.681947	1329	14267	3
3	244162.229612	2110	11160	3
4	189831.097588	1629	13830	3

These are aggregated

	price	sqft	lotsize	bedrooms
1	104855.329819	896	11622	2

<= HAVING bedrooms = 2

	price	sqft	lotsize
bedrooms			
2	104855.329819	896	11622
3	605961.009146	5068	39257

	price	sqft	lotsize	bedrooms
2	171967.681947	1329	14267	3
3	244162.229612	2110	11160	3
4	189831.097588	1629	13830	3

<= HAVING bedrooms = 3

Practice with GROUP BY… HAVING

Find the **average confidence** for each airline's various rating levels that have at least one retweet. Your table should have the airline name, sentiment, and the average confidence level.

```
SELECT airline, airline_sentiment,
AVG(airline_sentiment_confidence) as c
FROM Tweets
WHERE retweet_count > 0
GROUP BY airline, airline_sentiment
HAVING c > .7;
```

Joins

Inner Join

This is the **DEFAULT join.** Takes **only values that are in both tables.**

Table name:
employees

EmpID	Name
1	Jason
2	Alex
3	Ram
4	Ben
5	Cindy

Table name:
locations

EmpID	Location
1	San Jose
3	Washington D.C.
5	Seattle
7	Portland

SQL Syntax for Joins

The following SQL code matches IDs from table1 and table2 and gives you column2 from table1 and column3 from table2. (One row for each unique ID)

```
SELECT table1.ID, table1.column2, table2.column3

FROM table1

JOIN table2 ON table1.ID = table2.ID
```

Inner Join

Table name:
employees

EmpID	Name
1	Jason
2	Alex
3	Ram
4	Ben
5	Cindy

Table name:
locations

EmpID	Location
1	San Jose
3	Washington D.C.
5	Seattle
7	Portland

Exercise: What will **joining** employees and locations result in?

EmpID	Name	Location
1	Jason	San Jose
3	Ram	Washington D.C.
5	Cindy	Seattle

Practice with JOIN...ON

Table name:
employees

EmpID	Name
1	Jason
2	Alex
3	Ram
4	Ben
5	Cindy

Table name:
locations

EmpID	Location
1	San Jose
3	Washington D.C.
5	Seattle
7	Portland

Exercise: Write SQL that returns a table with EmpID, Name and Location**, joined on EmpID**

Answer:

```
SELECT employees.EmpID,
employees.Name, locations.Location
FROM employees
JOIN locations ON employees.EmpID =
locations.EmpID;
```

Outer Joins

Takes all values from one of the table's columns and matches them with whatever overlap there is in the other table.

	Name
1	Jason
2	Alex
3	Ram
4	Ben
5	Cindy

Left

	Location
1	San Jose
3	Washington D.C.
5	Seattle
7	Portland

Right

	Name	Location
1	Jason	San Jose
2	Alex	NaN
3	Ram	Washington D.C.
4	Ben	NaN
5	Cindy	Seattle

Implicit Join

Takes and returns all specified overlapping between two tables. Each row represents the unique combinations of relations from the two sets. In this case, the cross product.

(technically full outer join)

t1:

A int	B int
7	0
2	8

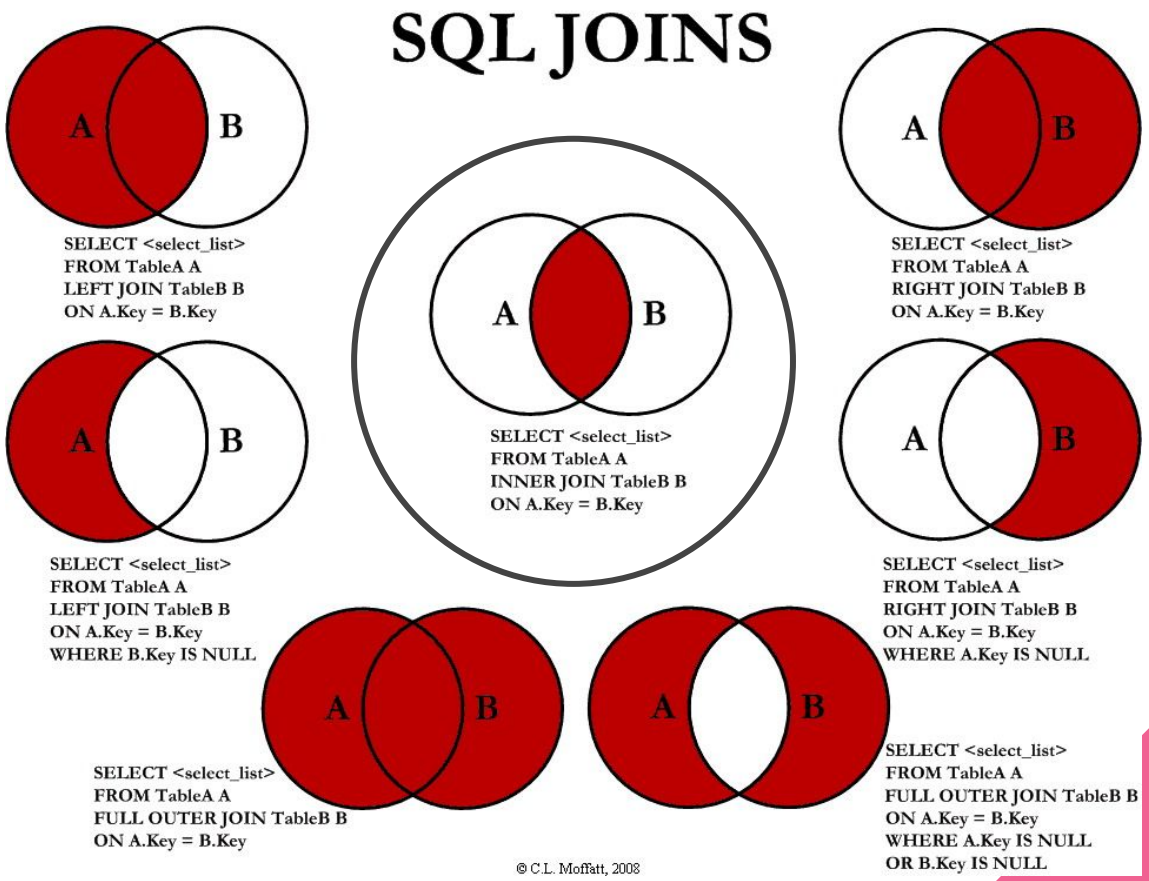
t2:

X int	Y int	Z int
2	5	4
8	3	9

```
SELECT *

FROM t1, t2;
```

X int	Y int	Z int	A int	B int
2	5	4	7	0
2	5	4	2	8
8	3	9	7	0
8	3	9	2	8



https://www.codeproject.com/KB/database/Visual_SQL_Joins/Visual_SQL_JOINS_orig.jpg

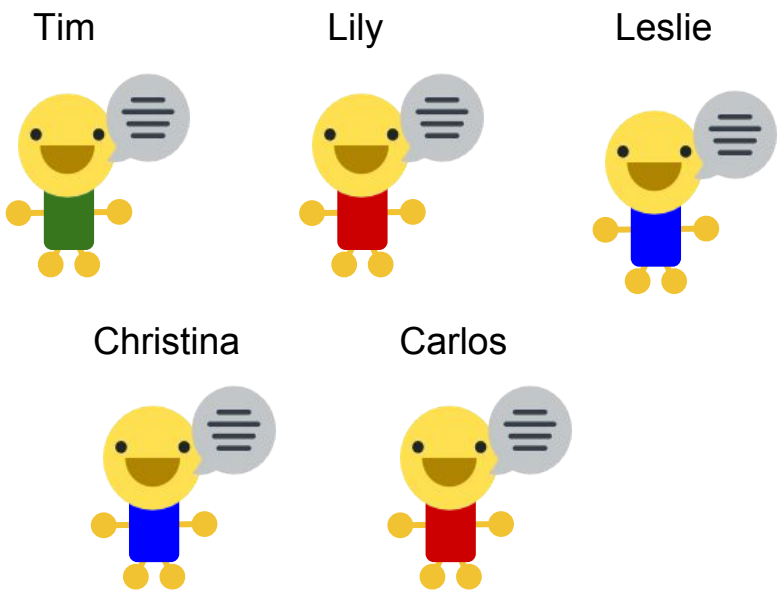
Project Introduction

Instant Runoff Voting (IRV)

- What is IRV? → A different voting method on electing officials to public office
  - When is it used?
  - Australia → House of Representatives
  - India → President
  - Ireland → President
  - United States → ?

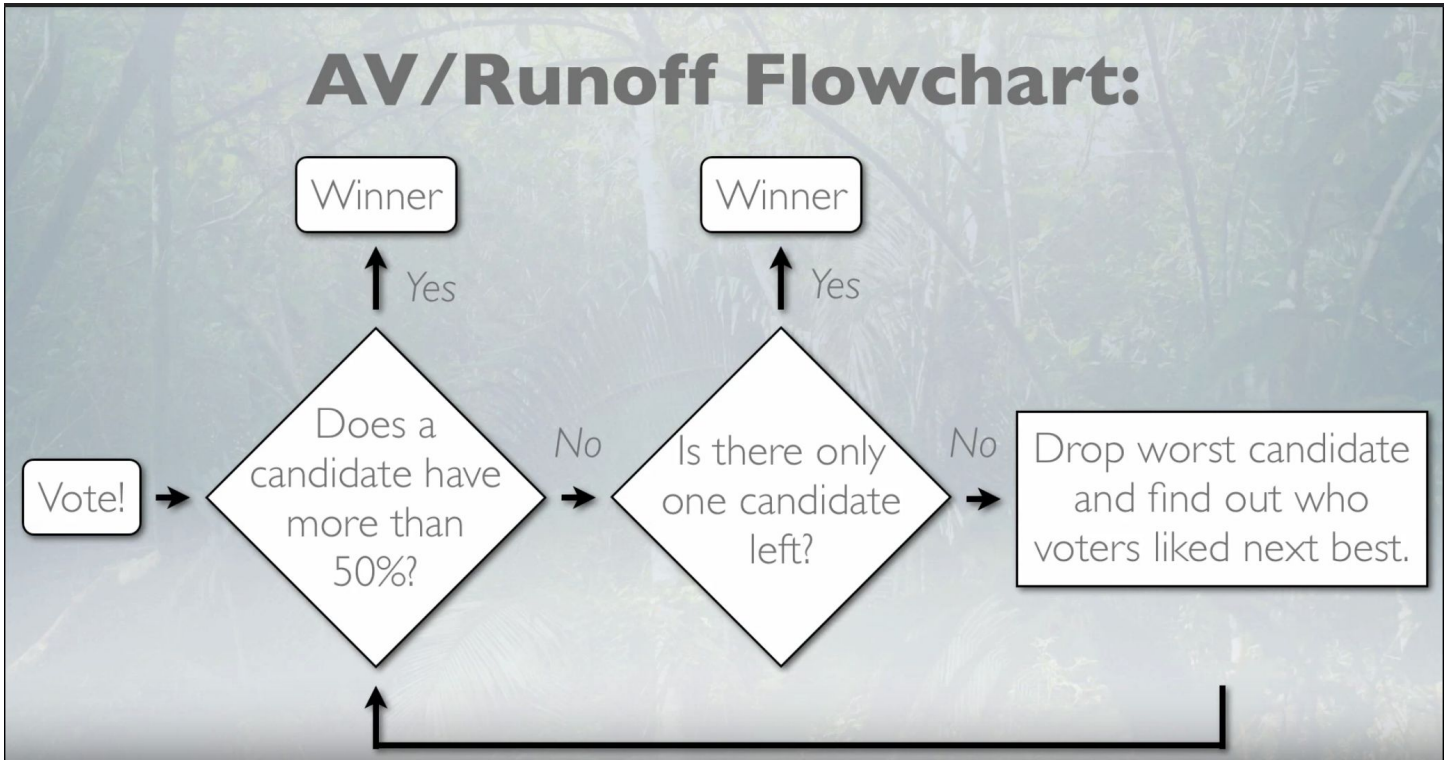
- United States
  - Academy of Motion Picture Arts and Sciences → Best Motion Picture
  - Maine
  - Oakland & San Francisco
- Also known as:
  - alternative vote
  - ranked-choice voting

Instant Runoff Voting (IRV) Simulation



- ➔ Fair, progressive city of Tofutown
- ➔ We must elect a new Mayor of our town, but we want to make sure that there is a fairer process to support smaller parties
- ➔ Your town has now trusted you with the city's full votes (10,000 of them)
- ➔ Your task: find the winner of the election. Your final table should output the name of the winner of the election.

Flowchart



From: “The Alternative Vote Explained:
https://www.youtube.com/watch?v=3Y3jE3B8HsE

Example

Highest Count

40%

2nd Count

35%

Highest Count

15%

Example

Highest Count

40%

Plurality Winner

2nd Count

35%

Highest Count

15%

Example

Highest Count

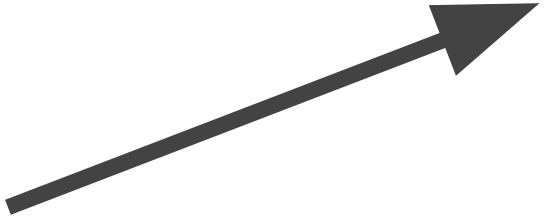
40%

2nd Count

35%

Highest Count

15%



Example

Highest Count

40%

2nd Count

35%

15%

IRV Winner

Highest Count

Who is the Next Mayor of Tofutown?