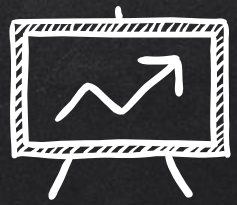


Databases and SQL with Pandas

A "Super Python" Talk by Nicholas A. Del Grosso

Today's Goals

1. Understand the Benefits of Tables as Data Structures
   - 5-minute Geek Out Session
   - 5-minutes of Pulling Myself Together
2. Review Pandas as a Tool for working with Tabular data.
   - 20-minute guided, hands-on exercise
3. Become Familiar with the Concept of a "Relational" Database.
   - Demonstration of Normalization
4. Understand where Database software fits into all this.
   - 10-Minute Live Coding Demo: SQLite3
5. Learn some SQL and how to use it with Pandas.
   - 10-minute Live Coding Demo
   - 30-minute Exercises

Table: Experiment

Subject	Gender	Session	Date	Trial	Condition	Drink	Reaction Time
Nick	M	1	5/5/16	1	Exp	Beer	100.2
Nick	M	1	5/5/16	2	Ctrl	Water	83.4
Nick	M	1	5/5/16	3	Ctrl	Water	95.2
Nick	M	1	5/5/16	4	Exp	Beer	78.0
Nick	M	1	5/5/16	5	Exp	Water	104.3
Nick	M	1	5/5/16	6	Ctrl	Water	87.9

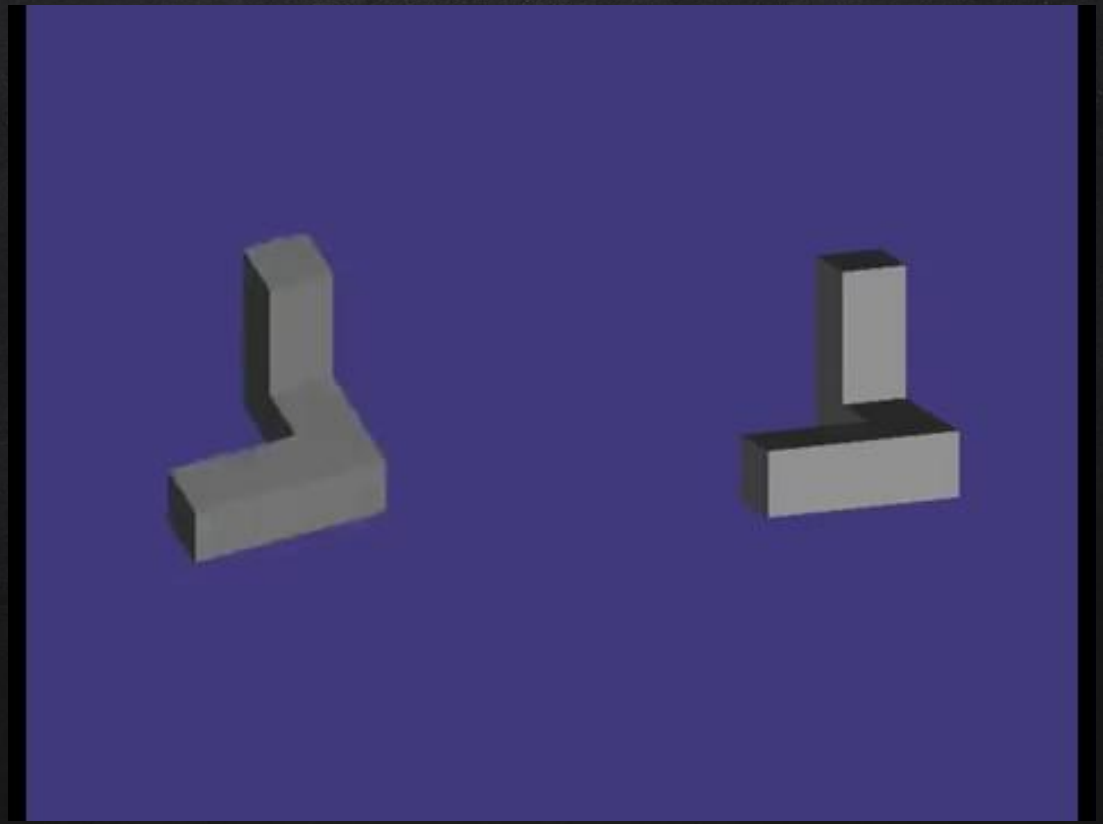
Algorithmic Concepts: Querying a Table

- How many total Trials were there, across the entire Experiment?
- How many Subjects are in the Experiment?
- What was the mean Reaction Time?
  - ...for each Condition?
  - ...for each Subject?

  ....for each Subject, for each Condition?

Subject	Gender	Session	Date	Trial	Condition	Stimulus	Reaction Time
Nick	M	1	5/5/16	1	Exp	Beer	100.2
Nick	M	1	5/5/16	2	Ctrl	Water	83.4

Pandas Review: Loading and Querying Tabular Data



Problems with the Table: Redundancy and Large File Size

Solution: “Normalization” and the “Relational Database”

Experiment

Subject	Gender	Session	Date	Trial	Condition	Stimulus	Reaction Time
Nick	M	1	5/5/16	1	Exp	Beer	100.2
Nick	M	1	5/5/16	2	Ctrl	Water	83.4
Nick	M	1	5/5/16	3	Ctrl	Water	95.2
Nick	M	1	5/6/16	4	Exp	Bear	78.0
Nick	M	1	5/5/16	5	Exp	Water	104.3
Nick	M	1	5/5/16	6	Ctrl		87.9

Subjects

Name	Gender
Nick	M
Anna	F

Conditions

Condition	Drink
Exp	Beer
Ctrl	Water

Sessions

Session	Date
1	5/5/16
1	5/5/16

Trials

Trial	Reaction Time
1	100.2
2	83.4
3	95.2
4	78.0
5	104.3
6	87.9

Subjects

ID	Name	Gender
1	Nick	M
2	Anna	F

Conditions

ID	Condition	Drink
1	Exp	Beer
2	Ctrl	Water

Sessions

ID	Subject	Session	Date
1	1	1	5/5/16
2	2	1	5/5/16

Trials

ID	Session	Condition	Trial	Reaction Time
1	1	1	1	100.2
2	1	2	2	83.4
3	1	2	3	95.2
4	1	1	4	78.0
5	1	1	5	104.3
6	1	2	6	87.9

Querying a Relational Database: Make a Single Table First through The "Join"

- "Which Participant was in Session 1?"
  - Join Sessions and Subjects,
  - Take Only Name and Session

Session	Name
1	Nick
2	Anna

- "Which Stimuli Were Used in Each Session?"
  - Join Conditions and Sessions and Trials,
  - Take only Session and Stimulus

Session	Drink
1	Beer
1	Water
2	Beer

Quick Demo: Join in Pandas

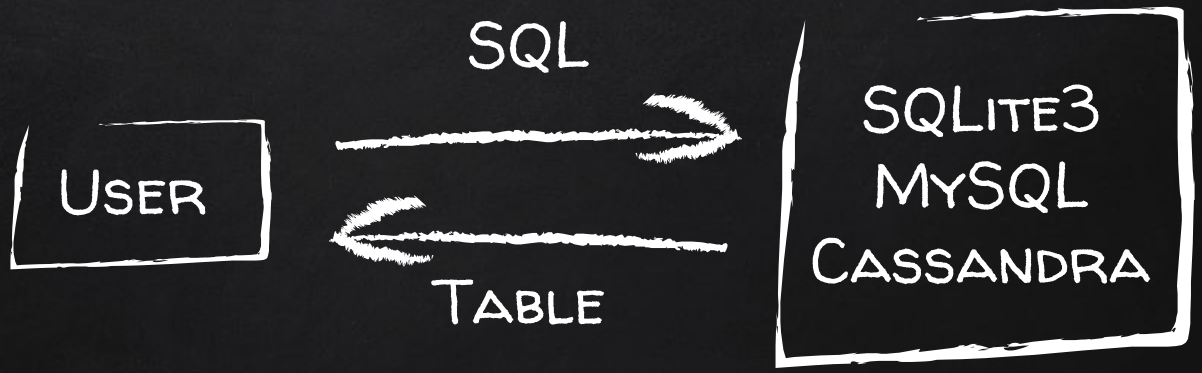
```
pd.merge(conditions, trials, left_on = 'ID',  right_on = 'Condition')

trials.join(conditions, on='Condition')
```

Review Summary: Normalization

- Normalization for Storage: 1 Table → Multiple Tables
  - Reduces File Size
  - Reduces Redundancy
  - Reduces Errors
  - Increases Read / Write Speeds
  - Can Form a More "Natural" Organization Schema
- Querying through Joins: Multiple Tables → 1 Table

SQL

SQL is the Language Used to TAlk to Relational Database Software



SQL: Structured Query Language

Create Tables

- CREATE TABLE
  Define a new table
- DROP TABLE
  Remove a table

Query and Update Columns

- SELECT
  Retrieve columns from a table or view
- INSERT INTO
  Create new rows in a table
- COMMIT
  Save changes to the database
- UPDATE
  Update rows of a table

Filter, Group, and Process the Rows

- WHERE
  To retrieve specific information from a table excluding other irrelevant data
- DISTINCT
  To return only distinct (different) values in a column
- BETWEEN..AND..
  To select values within a range

SQL Querying Examples

```
SELECT Name, Gender FROM Subjects;
```

Name	Gender
Nick	M
Anna	F

```
SELECT * FROM Conditions;
```

ID	Condition	Drink
1	Exp	Beer
2	Ctrl	Water

```
SELECT ReactionTime FROM Trials LIMIT 2;
```

Reaction Time
100.2
83.4

SQL Querying Examples: Filtering, Aggregating, and Joining

```
SELECT Name FROM Subjects
WHERE Gender = “M”;
```

Name
Nick

```
SELECT Condition, avg(ReactionTime) FROM Trials
GROUP BY Condition;
```

Condition	avg(ReactionTime)
1	78.486
2	92.112

```
SELECT Drink, avg(ReactionTime) FROM Trials
JOIN Trials ON Conditions.ID = Trials.Condition
GROUP BY Condition;
```

Drink	avg(RT)
Beer	78.486
Water	92.112

A Couple SQLite-Specific Commands

Get all Table Names:

```
SELECT name FROM SQLITE_MASTER;
```

Get Column Names from a Table:

```
PRAGMA table_info(<TableName>);
```

Demo: SQL Querying in Python

Pandas uses SQLAlchemy to connect to Databases like SQLite3

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///my_folder/my_data.db')
conn = engine.connect()
```

```
import pandas as pd
query = "SELECT Angle, Correct, Matching FROM Trials;"
df = pd.read_sql(query, conn)
```

```
import seaborn as sns
sns.factorplot(x='Angle', y='Correct', hue='Matching',
data=df)
```

30 Minutes Hands-On:
(Thank You For Your Attention!)

- How Many Subjects are in this Study?
- Did Subjects Take Longer to Respond to Matching or Nonmatching Stimuli?
- Was there a ReactionTime – Stimulus Rotation Relationship?
- What was the mean Subject Age?
- Was there an Effect of Subject Sex on:
  - Reaction Time
  - Accuracy (Correctness)