Introduction to databases

STREAMLINED DATA INGESTION WITH PANDAS

Amany Mahfouz
Instructor

DataCamp

Relational Databases

- Data about entities is organized into tables

- Each row or record is an instance of an entity

- Each column has information about an attribute

- Tables can be linked to each other via unique keys

- Support more data, multiple simultaneous users, and data quality controls

- Data types are specified for each column

- SQL (Structured Query Language) to interact with databases

Common Relational Databases
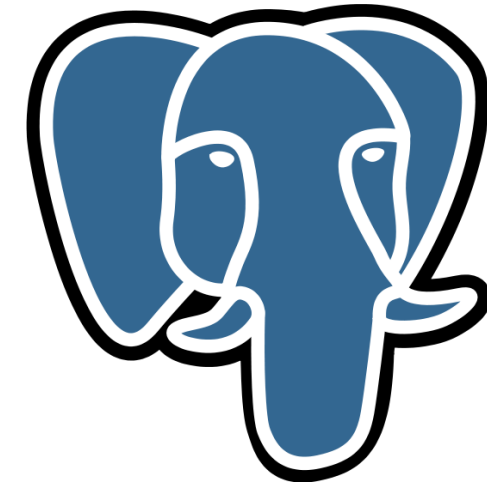
- SQLite databases are computer files

Connecting to Databases

- Two-step process:
  1. Create way to connect to database

  2. Query database

Creating a Database Engine

- `sqlalchemy` 's `create_engine()` makes an engine to handle database connections
  - Needs string URL of database to connect to

  - SQLite URL format: `sqlite:///filename.db`

Querying Databases

- `pd.read_sql(query, engine)` to load in data from a database

- Arguments

  - `query` : String containing SQL query to run or table to load

  - `engine` : Connection/database engine object

SQL Review: SELECT

- Used to query data from a database

- Basic syntax:

```sql
SELECT [column_names] FROM [table_name];
```

- To get all data in a table:

```sql
SELECT * FROM [table_name];
```

- Code style: keywords in ALL CAPS, semicolon (;) to end a statement

Getting Data from a Database

```python
# Load pandas and sqlalchemy's create_engine
import pandas as pd
from sqlalchemy import create_engine


# Create database engine to manage connections
engine = create_engine("sqlite:///data.db")


# Load entire weather table by table name
weather = pd.read_sql("weather", engine)
```

```
# Create database engine to manage connections
engine = create_engine("sqlite:///data.db")


# Load entire weather table with SQL
weather = pd.read_sql("SELECT * FROM weather", engine)


print(weather.head())
```

```
     station                      name  latitude  ...  prcp  snow  tavg tmax  tmin
0  USW00094728  NY CITY CENTRAL PARK, NY US  40.77898  ...  0.00   0.0        52    42
1  USW00094728  NY CITY CENTRAL PARK, NY US  40.77898  ...  0.00   0.0        48    39
2  USW00094728  NY CITY CENTRAL PARK, NY US  40.77898  ...  0.00   0.0        48    42
3  USW00094728  NY CITY CENTRAL PARK, NY US  40.77898  ...  0.00   0.0        51    40
4  USW00094728  NY CITY CENTRAL PARK, NY US  40.77898  ...  0.75   0.0        61    50


[5 rows x 13 columns]
```

Let's practice!

STREAMLINED DATA INGESTION WITH PANDAS

Refining imports with SQL queries

STREAMLINED DATA INGESTION WITH PANDAS

Amany Mahfouz

Instructor

SELECTing Columns

- `SELECT [column names] FROM [table name];`

- Example:

```
SELECT date, tavg
FROM weather;
```

WHERE Clauses

- Use a `WHERE` clause to selectively import records

```
SELECT [column_names]
  FROM [table_name]
 WHERE [condition];
```

Filtering by Numbers

- Compare numbers with mathematical operators
  - `=`
  - `>` and `>=`
  - `<` and `<=`
  - `<>` (not equal to)
  - Example:

```sql
SELECT *
FROM weather
WHERE tmax > 32;
```

Filtering Text

- Match exact strings with the `=` sign and the text to match

- String matching is case-sensitive

- Example:

```
/* Get records about incidents in Brooklyn */
SELECT *
FROM hpd311calls
WHERE borough = 'BROOKLYN';
```

SQL and pandas

```python
# Load libraries
import pandas as pd
from sqlalchemy import create_engine

# Create database engine
engine = create_engine("sqlite:///data.db")

# Write query to get records from Brooklyn
query = """SELECT *
             FROM hpd311calls
            WHERE borough = 'BROOKLYN';"""

# Query the database
brooklyn_calls = pd.read_sql(query, engine)

print(brooklyn_calls.borough.unique())
```

```
['BROOKLYN']
```

Combining Conditions: AND

- WHERE clauses with AND return records that meet all conditions

```python
# Write query to get records about plumbing in the Bronx
and_query = """SELECT *
                 FROM hpd311calls
                WHERE borough = 'BRONX'
                  AND complaint_type = 'PLUMBING';"""

# Get calls about plumbing issues in the Bronx
bx_plumbing_calls = pd.read_sql(and_query, engine)


# Check record count
print(bx_plumbing_calls.shape)
```

```
(2016, 8)
```

Combining Conditions: OR

- WHERE clauses with OR return records that meet at least one condition

```python
# Write query to get records about water leaks or plumbing
or_query = """SELECT *
                FROM hpd311calls
               WHERE complaint_type = 'WATER LEAK'
                  OR complaint_type = 'PLUMBING';"""

# Get calls that are about plumbing or water leaks
leaks_or_plumbing = pd.read_sql(or_query, engine)


# Check record count
print(leaks_or_plumbing.shape)
```

```
(10684, 8)
```

Let's practice!

More complex SQL queries

STREAMLINED DATA INGESTION WITH PANDAS

Amany Mahfouz
Instructor

Getting DISTINCT Values

- Get unique values for one or more columns with `SELECT DISTINCT`

- Syntax:

```
SELECT DISTINCT [column names] FROM [table];
```

- Remove duplicate records:

```
SELECT DISTINCT * FROM [table];
```

```
/* Get unique street addresses and boroughs */
SELECT DISTINCT incident_address,
       borough
  FROM hpd311calls;
```

Aggregate Functions

- Query a database directly for descriptive statistics

- Aggregate functions
  - `SUM`
  - `AVG`
  - `MAX`
  - `MIN`
  - `COUNT`

Aggregate Functions

- `SUM` , `AVG` , `MAX` , `MIN`
  - Each takes a single column name

```sql
SELECT AVG(tmax) FROM weather;
```

- `COUNT`
  - Get number of rows that meet query conditions

```sql
SELECT COUNT(*) FROM [table_name];
```

  - Get number of unique values in a column

```sql
SELECT COUNT(DISTINCT [column_names]) FROM [table_name];
```

GROUP BY

- Aggregate functions calculate a single summary statistic by default

- Summarize data by categories with `GROUP BY` statements

- Remember to also select the column you're grouping by!

```sql
/* Get counts of plumbing calls by borough */
SELECT borough,
       COUNT(*)
  FROM hpd311calls
 WHERE complaint_type = 'PLUMBING'
 GROUP BY borough;
```

Counting by Groups

```python
# Create database engine
engine = create_engine("sqlite:///data.db")


# Write query to get plumbing call counts by borough
query = """SELECT borough, COUNT(*)
             FROM hpd311calls
            WHERE complaint_type = 'PLUMBING'
            GROUP BY borough;"""


# Query databse and create data frame
plumbing_call_counts = pd.read_sql(query, engine)
```

Counting by Groups

```
print(plumbing_call_counts)
```

```
          borough  COUNT(*)
0           BRONX      2016
1        BROOKLYN      2702
2       MANHATTAN      1413
3          QUEENS       808
4   STATEN ISLAND       178
```

Let's practice!

Loading multiple tables with joins

STREAMLINED DATA INGESTION WITH PANDAS

Amany Mahfouz
Instructor

DataCamp

Keys

- Database records have unique identifiers, or keys

unique_key	created_date	agency	complaint_type	incident_zip	incident_address	community_board	borough
Filter	Filter	Fi...	Filter	Filter	Filter	Filter	Filter
38070822	01/01/2018	HPD	HEAT/HOT WATER	10468	2786 JEROME AVEN...	07 BRONX	BRONX
38065299	01/01/2018	HPD	PLUMBING	10003	323 EAST 12 STRE...	03 MANHATTAN	MANHA...
38066653	01/01/2018	HPD	HEAT/HOT WATER	10452	1235 GRAND CONC...	04 BRONX	BRONX
38070264	01/01/2018	HPD	HEAT/HOT WATER	10032	656 WEST 171 STR...	12 MANHATTAN	MANHA...
38072466	01/01/2018	HPD	HEAT/HOT WATER	11213	1030 PARK PLACE	08 BROOKLYN	BROOK...

Keys

- Database records have unique identifiers, or keys

course_code	course_title	department	faculty	credits	level	term	instructor_id
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
ANAT212	Molecular Me...	Anatomy and ...	Medicine	3	Undergra...	Spring	199157185
ANAT214	Systemic Hu...	Anatomy and ...	Medicine	3	Undergra...	Fall	199157185
ANAT261	Introduction t...	Anatomy and ...	Medicine	4	Undergra...	Fall	175319914
ANAT69D1	Cell and Deve...	Anatomy and ...	Medicine	3	Graduate	Fall	193644393
ANTH201	Prehistoric Ar...	Anthropology	Arts	3	Undergra...	Spring	182832540

Keys

- Database records have unique identifiers, or keys

course_code	course_title	department	faculty	credits	level	term	instructor_id
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
ANAT212	Molecular Me...	Anatomy and ...	Medicine	3	Undergra...	Spring	199157185
ANAT214	Systemic Hu...	Anatomy and ...	Medicine	3	Undergra...	Fall	199157185
ANAT261	Introduction t...	Anatomy and ...	Medicine	4	Undergra...	Fall	175319914
ANAT69D1	Cell and Deve...	Anatomy and ...	Medicine	3	Graduate	Fall	193644393
ANTH201	Prehistoric Ar...	Anthropology	Arts	3	Undergra...	Spring	182832540

Keys

id	name	title	degree_institution	faculty
Filter	Filter	Filter	Filter	Filter
112013131	Isabel Morales	Assistant Professor	Complutense Uni...	Science
184601849	Abraham Waldman	Assistant Professor	LMU Munich	Science
182832540	Henry Jones	Professor	University of Chi...	Arts
146661411	Anna Ng	Associate Professor	Queen's University	Arts
199157185	Vanessa Hawkins	Professor	Queen's University	Medicine

Keys

course_code	course_title	department	faculty	credits	level	term	instructor_id	name
ANAT212	Molecular Mechanisms of Cell Function	Anatomy and Cell Biology	Medicine	3	Undergraduate	Spring	199157185	Vanessa Hawkins
ANAT214	Systemic Human Anatomy	Anatomy and Cell Biology	Medicine	3	Undergraduate	Fall	199157185	Vanessa Hawkins
ANAT261	Introduction to Dynamic Histology	Anatomy and Cell Biology	Medicine	4	Undergraduate	Fall	175319914	Arun Agarwal
ANAT69D1	Cell and Developmental Biology	Anatomy and Cell Biology	Medicine	3	Graduate	Fall	193644393	Alexis Cook
ANTH201	Prehistoric Archaology	Anthropology	Arts	3	Undergraduate	Spring	182832540	Henry Jones

Joining Tables

station	name	latitude	longitude	elevation	date	month	awnd	prcp	snow	...	tmax	tmin
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	...	Filter	Filter
USW00094728	NY CITY CEN...	40.77898	-73.96925	42.7	01/01/2018	January	7.83	0	0		19	7
USW00094728	NY CITY CEN...	40.77898	-73.96925	42.7	01/02/2018	January	8.05	0	0		26	13
USW00094728	NY CITY CEN...	40.77898	-73.96925	42.7	01/03/2018	January	3.13	0	0		30	16
USW00094728	NY CITY CEN...	40.77898	-73.96925	42.7	01/04/2018	January	12.53	0.76	9.8		29	19
USW00094728	NY CITY CEN...	40.77898	-73.96925	42.7	01/05/2018	January	12.97	0	0		19	9
USW00094728	NY CITY CEN...	40.77898	-73.96925	42.7	01/06/2018	January	10.96	0	0		13	6
USW00094728	NY CITY CEN...	40.77898	-73.96925	42.7	01/07/2018	January	6.49	0	0		18	5

unique_key	created_date	agency	complaint_type	incident_zip	incident_address	community_board	borough
Filter	Filter	Fi...	Filter	Filter	Filter	Filter	Filter
38070822	01/01/2018	HPD	HEAT/HOT WATER	10468	2786 JEROME AVEN...	07 BRONX	BRONX
38065299	01/01/2018	HPD	PLUMBING	10003	323 EAST 12 STRE...	03 MANHATTAN	MANHATTAN
38066653	01/01/2018	HPD	HEAT/HOT WATER	10452	1235 GRAND CONC...	04 BRONX	BRONX
38070264	01/01/2018	HPD	HEAT/HOT WATER	10032	656 WEST 171 STR...	12 MANHATTAN	MANHATTAN
38072466	01/01/2018	HPD	HEAT/HOT WATER	11213	1030 PARK PLACE	08 BROOKLYN	BROOKLYN

Joining Tables

```sql
SELECT *
  FROM hpd311calls
```

Joining Tables

```
SELECT *
  FROM hpd311calls
       JOIN weather
       ON hpd311calls.created_date = weather.date;
```

- Use dot notation (`table.column`) when working with multiple tables

- Default join only returns records whose key values appear in both tables

- Make sure join keys are the same data type or nothing will match

Joining and Filtering

```sql
/* Get only heat/hot water calls and join in weather data */
SELECT *
  FROM hpd311calls
       JOIN weather
       ON hpd311calls.created_date = weather.date
 WHERE hpd311calls.complaint_type = 'HEAT/HOT WATER';
```

Joining and Aggregating

```sql
/* Get call counts by borough */
SELECT hpd311calls.borough,
       COUNT(*)
  FROM hpd311calls
 GROUP BY hpd311calls.borough;
```

Joining and Aggregating

```sql
/* Get call counts by borough
   and join in population and housing counts */
SELECT hpd311calls.borough,
       COUNT(*),
       boro_census.total_population,
       boro_census.housing_units
  FROM hpd311calls
 GROUP BY hpd311calls.borough
```

Joining and Aggregating

```sql
/* Get call counts by borough
   and join in population and housing counts */
SELECT hpd311calls.borough,
       COUNT(*),
       boro_census.total_population,
       boro_census.housing_units
  FROM hpd311calls
       JOIN boro_census
       ON hpd311calls.borough = boro_census.borough
 GROUP BY hpd311calls.borough;
```

```
query = """SELECT hpd311calls.borough,
                  COUNT(*),
                  boro_census.total_population,
                  boro_census.housing_units
             FROM hpd311calls
                  JOIN boro_census
                  ON hpd311calls.borough = boro_census.borough
           GROUP BY hpd311calls.borough;"""


call_counts = pd.read_sql(query, engine)
print(call_counts)
```

```
        borough  COUNT(*)  total_population  housing_units
0         BRONX     29874           1455846         524488
1      BROOKLYN     31722           2635121        1028383
2     MANHATTAN     20196           1653877         872645
3        QUEENS     11384           2339280         850422
4  STATEN ISLAND      1322            475948         179179
```

Review

- SQL order of keywords
  - `SELECT`
  - `FROM`
  - `JOIN`
  - `WHERE`
  - `GROUP BY`

Let's practice!

STREAMLINED DATA INGESTION WITH PANDAS