Introduction to Databases

A database consists of tables

Census

state	sex	age	pop2000	pop2008
New York	F	0	120355	122194
New York	F	1	118219	119661
New York	F	2	119577	116413

State_Fact

name	abbreviation	type
New York	NY	state
Washington DC	DC	capitol
Washington	WA	state

Table consist of columns and rows

Census

state	sex	age	pop2000	pop2008
New York	F	0	120355	122194
New York	F	1	118219	119661
New York	F	2	119577	116413

Tables can be related

Census

state	sex	age	pop2000	pop2008
New York	F	0	120355	122194
New York	F	1	118219	119661
New York	F	2	119577	116413

State_Fact

name	abbreviation	type
New York	NY	state
Washington DC	DC	capitol
Washington	WA	state

Let’s practice!

Connecting to a Database

Meet SQLAlchemy

- Two Main Pieces
    - Core (Relational Model focused)
    - ORM (User Data Model focused)

There are many types of databases

- SQLite
- PostgreSQL
- MySQL
- MS SQL
- Oracle
- Many more

Connecting to a database

```
In [1]: from sqlalchemy import create_engine

In [2]: engine = create_engine('sqlite:///census_nyc.sqlite')

In [3]: connection = engine.connect()
```

- Engine: common interface to the database from SQLAlchemy
- Connection string: All the details required to find the database (and login, if necessary)

A word on connection strings

- `'sqlite:///census_nyc.sqlite'`

  Driver+Dialect Filename

What’s in your database?

- Before querying your database, you’ll want to know what is in it: what the tables are, for example:

```
In [1]: from sqlalchemy import create_engine

In [2]: engine = create_engine('sqlite:///census_nyc.sqlite')

In [3]: print(engine.table_names())
Out[3]: ['census', 'state_fact']
```

Reflection

- Reflection reads database and builds SQLAlchemy Table objects

```
In [1]: from sqlalchemy import MetaData, Table

In [2]: metadata = MetaData()

In [3]: census = Table('census', metadata, autoload=True,
autoload_with=engine)

In [4]: print(repr(census))
Out[4]:
Table('census', MetaData(bind=None), Column('state',
VARCHAR(length=30), table=<census>), Column('sex',
VARCHAR(length=1), table=<census>), Column('age', INTEGER(),
table=<census>), Column('pop2000', INTEGER(), table=<census>),
Column('pop2008', INTEGER(), table=<census>), schema=None)
```

Let’s practice!

Introduction to SQL Queries

SQL Statements

- Select, Insert, Update & Delete data
- Create & Alter data

Basic SQL querying

- `SELECT column_name FROM table_name`
- `SELECT pop2008 FROM People`
- `SELECT * FROM People`

Basic SQL querying

```
In [1]: from sqlalchemy import create_engine

In [2]: engine = create_engine('sqlite:///census_nyc.sqlite')

In [3]: connection = engine.connect()

In [4]: stmt = 'SELECT * FROM people'

In [5]: result_proxy = connection.execute(stmt)

In [6]  results = result_proxy.fetchall()
```

ResultProxy vs ResultSet

```
In [5]: result_proxy = connection.execute(stmt)

In [6]: results = result_proxy.fetchall()
```

- ResultProxy
- ResultSet

Handling ResultSets

```
In [1]: first_row = results[0]

In [2]: print(first_row)
Out[2]: ('Illinois', 'M', 0, 89600, 95012)

In [4]: print(first_row.keys())
Out[4]: ['state', 'sex', 'age', 'pop2000', 'pop2008']

In [6]: print(first_row.state)
Out[6]: 'Illinois'
```

SQLAlchemy to Build Queries

- Provides a Pythonic way to build SQL statements
- Hides differences between backend database types

SQLAlchemy querying

```
In [4]: from sqlalchemy import Table, MetaData

In [5]: metadata = MetaData()

In [6]: census = Table('census', metadata, autoload=True,
autoload_with=engine)

In [7]: stmt = select([census])

In [8]: results = connection.execute(stmt).fetchall()
```

SQLAlchemy Select Statement

- Requires a list of one or more Tables or Columns
- Using a table will select all the columns in it

```
In [9]: stmt = select([census])

In [10]: print(stmt)
Out[10]: 'SELECT * from CENSUS'
```

Let’s practice!

Congratulations!

You already

- Know about the relational model
- Can make basic SQL queries

Coming up next...

- Beef up your SQL querying skills
- Learn how to extract all types of useful information from your databases using SQLAlchemy
- Learn how to create and write to relational databases
- Deep dive into the US census dataset!

See you in the next chapter!