Introduction to Databases

INTRODUCTION TO DATABASES IN PYTHON

Jason Myers

Co-Author of Essential SQLAlchemy and Software Engineer

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

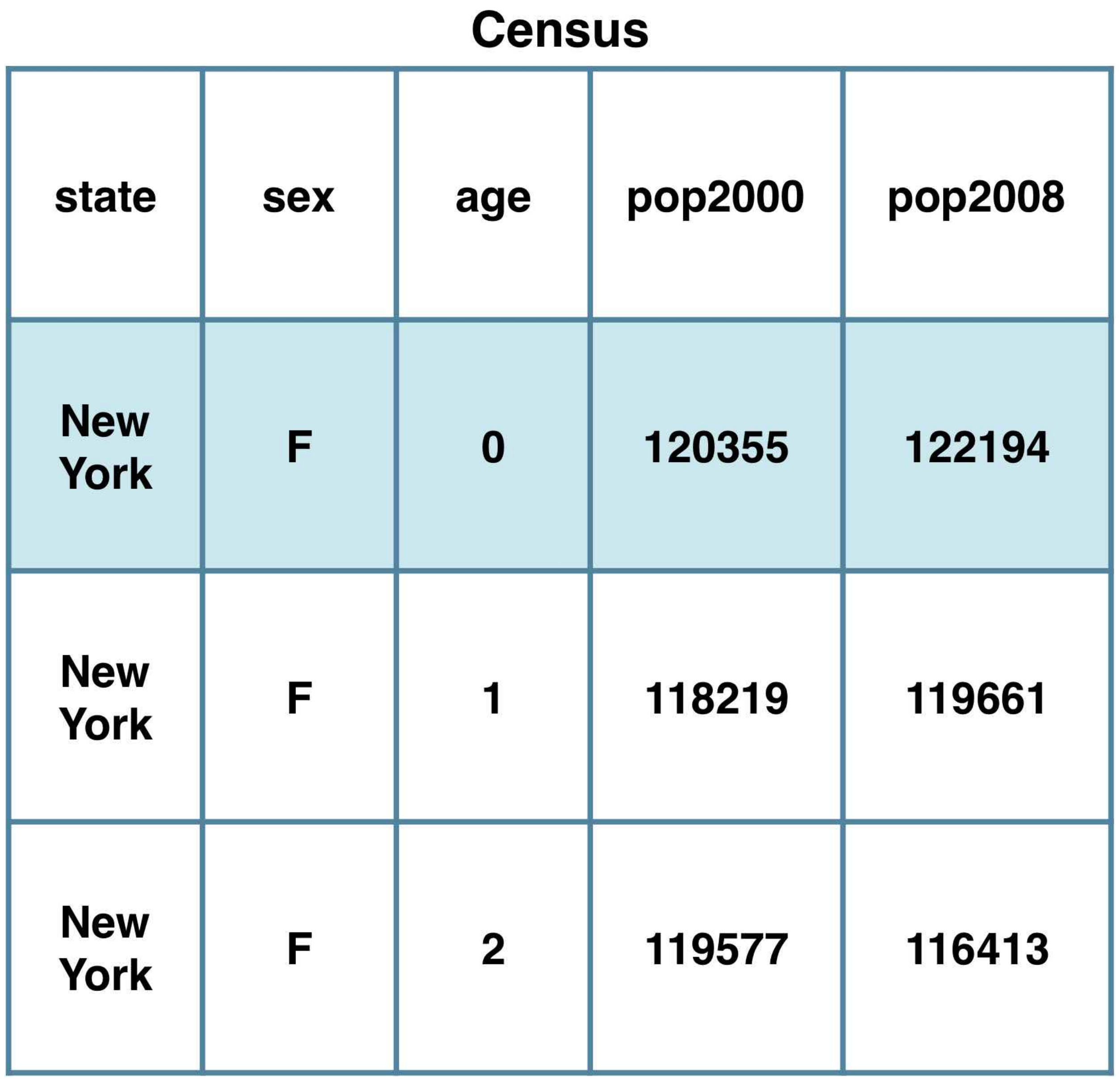
name	abbreviation	type
New York	NY	state
Washington DC	DC	capitol
Washington	WA	state

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

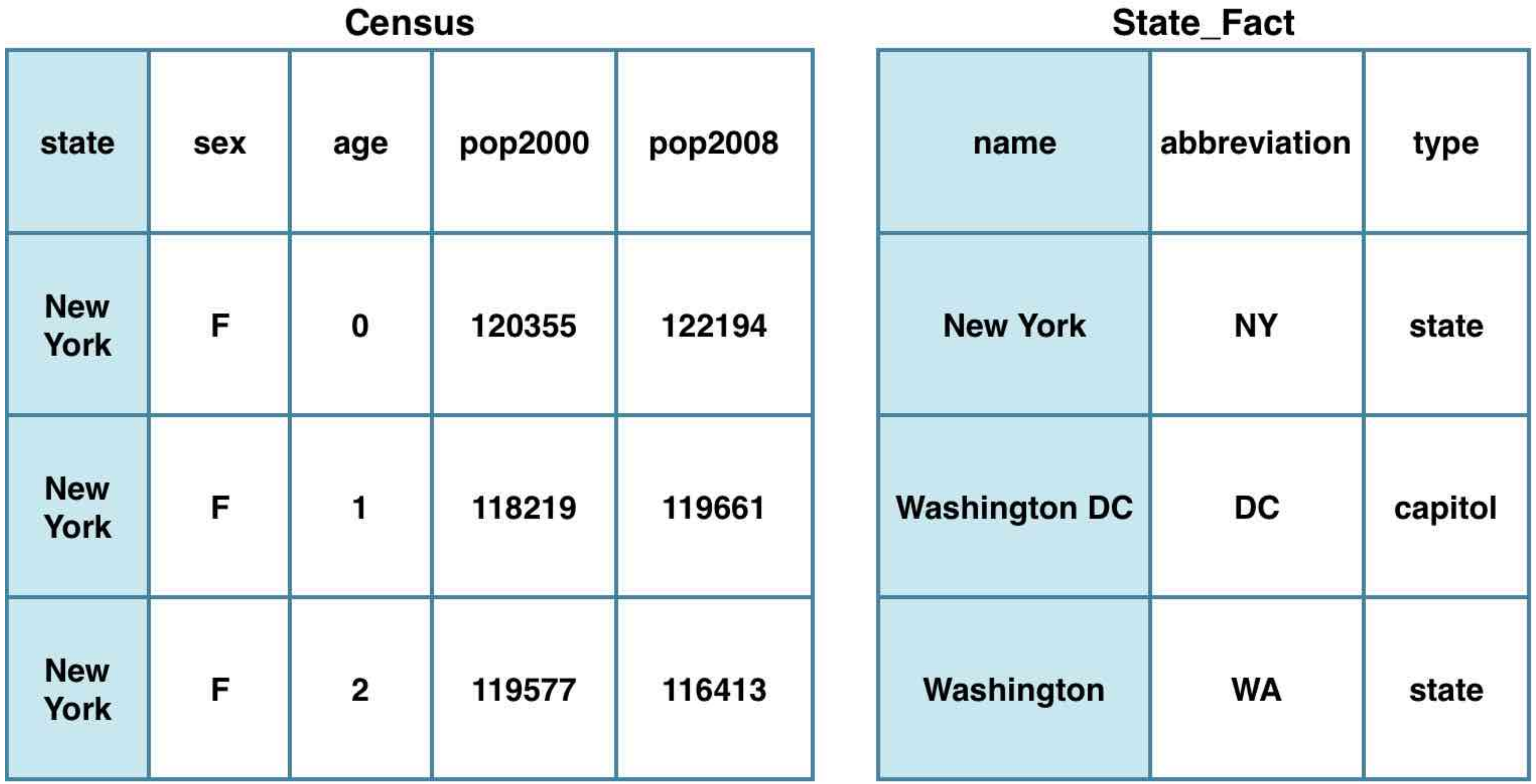
Table consist of columns and rows



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!

INTRODUCTION TO DATABASES IN PYTHON

Connecting to your database

INTRODUCTION TO DATABASES IN PYTHON

Jason Myers

Co-Author of Essential SQLAlchemy and Software Engineer

Meet SQLAlchemy

- Two main pieces
    - Core (Relational Model focused)
    - ORM (User Data Model focused)

There are many types of databases

- SQLite
- PostgreSQL
- MySQL
- Microsoft SQL Server
- Oracle SQL
- Many more

Connecting to a database

```python
from sqlalchemy import create_engine

engine = create_engine('sqlite:///census_nyc.sqlite')

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

A word on connection strings

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

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:

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///census_nyc.sqlite')
print(engine.table_names())
```

```
['census', 'state_fact']
```

Reflection

Reflection reads database and builds SQLAlchemy `Table` objects

```
from sqlalchemy import MetaData, Table
metadata = MetaData()
census = Table('census', metadata, autoload=True,
  autoload_with=engine)
print(repr(census))
```

```
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 DATABASES IN PYTHON

Introduction to SQL queries

INTRODUCTION TO DATABASES IN PYTHON

Jason Myers

Co-Author of Essential SQLAlchemy and Software Engineer

SQL Statements

- Select, insert, update, and delete data
- Create and alter data

Basic SQL querying

```
SELECT column_name FROM table_name
```

- `SELECT pop2008 FROM People`
- `SELECT * FROM People`

Basic SQL querying

```
from sqlalchemy import create_engine

engine = create_engine('sqlite:///census_nyc.sqlite')

connection = engine.connect()

stmt = 'SELECT * FROM people'

result_proxy = connection.execute(stmt)

results = result_proxy.fetchall()
```

ResultProxy vs ResultSet

```
result_proxy = connection.execute(stmt)


results = result_proxy.fetchall()
```

- `result_proxy` is a `ResultProxy`
- `results` is a `ResultSet`

Handling ResultSets

```
first_row = results[0]
print(first_row)
```

```
('Illinois', 'M', 0, 89600, 95012)
```

```
print(first_row.keys())
```

```
['state', 'sex', 'age', 'pop2000', 'pop2008']
```

```
print(first_row.state)
```

```
'Illinois'
```

SQLAlchemy to build queries

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

SQLAlchemy querying

```
from sqlalchemy import Table, MetaData
metadata = MetaData()

census = Table('census', metadata, autoload=True,
  autoload_with=engine)

stmt = select([census])

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

```
stmt = select([census])

print(stmt)
```

```
'SELECT * from CENSUS'
```

Let's practice!

INTRODUCTION TO DATABASES IN PYTHON

Congratulations!

INTRODUCTION TO DATABASES IN PYTHON

Jason Myers

Co-Author of Essential SQLAlchemy and
Software Engineer

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!

INTRODUCTION TO DATABASES IN PYTHON