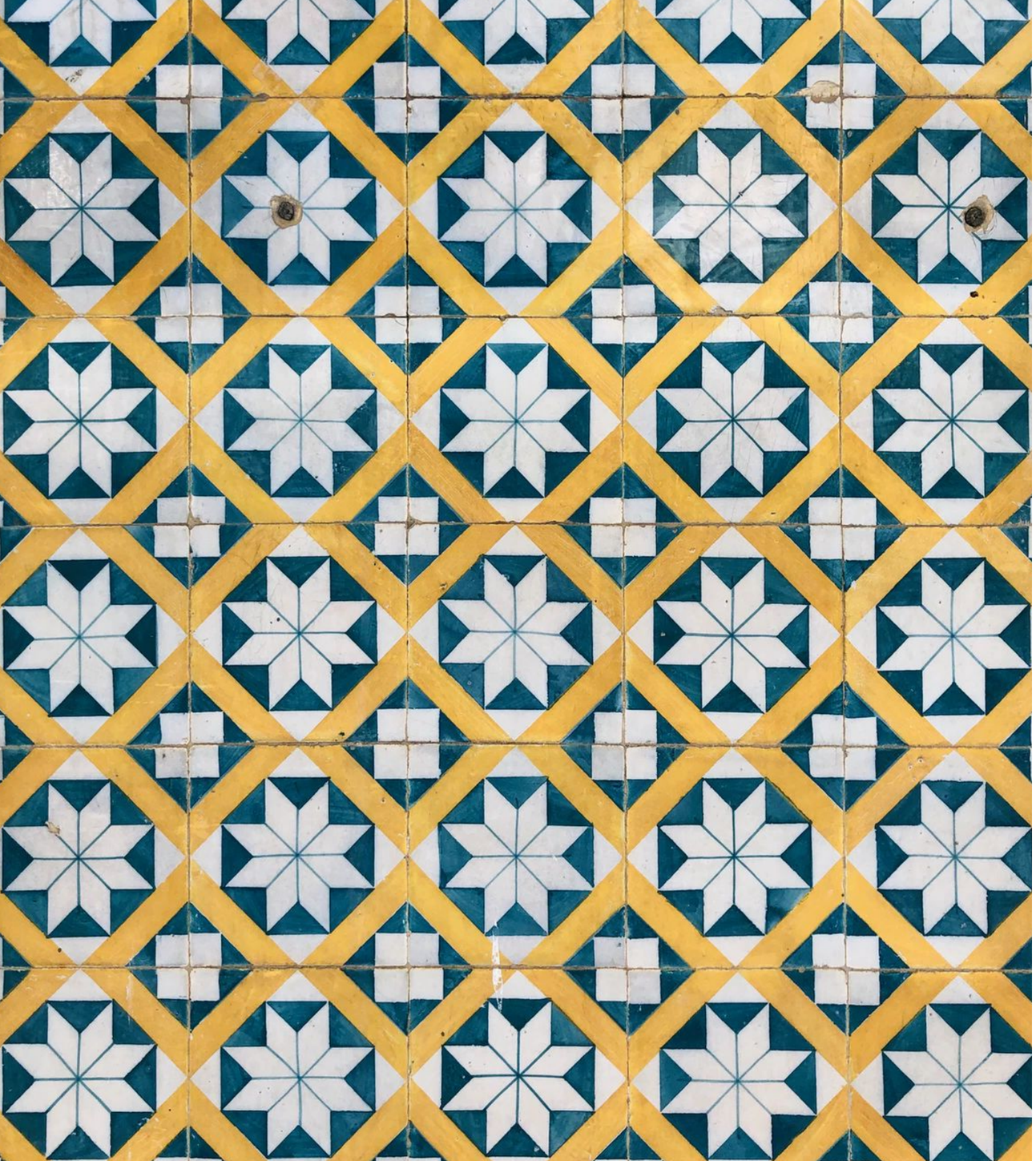


SQL 2
Creating Tables & Working with Multiple Tables

METIS®

Objectives

Create Tables

- Prerequisite: Know how to **query** tables in a database using SQL
- This lesson: Learn how to **create and modify** tables

Work with Multiple Tables

- Prerequisite: Know how to query data from a **single** table
- This lesson: Learn how to query data from **multiple** tables

SQL in Python

- Prerequisite: SQL basics and Python basics
- This lesson: Write SQL queries within Python with SQLAlchemy

Chapter 1

Create Tables

METIS®

SQL Statement

General SQL code used to interact with a database

SELECT Statement
Retrieve data from a database

```
SELECT * FROM new_table;
```

CREATE Statement
Define an object, such as a database or table

```
CREATE DATABASE new_database;
```

Scenario

Create a database of what my friends ate in the past week

Steps

1. Create a database
2. Create a table
3. Insert data into the table

Step 1: Create a database

Typically, you would write a CREATE statement

CREATE DATABASE meals;

It is a little bit different in SQLite

```
my-computer$ sqlite3

sqlite> .database
main:

sqlite> .quit
```

```
my-computer$ sqlite3 meals.db

sqlite> .database
main: /Users/user/meals.db

sqlite> .tables

sqlite> .quit
```

Step 2: Create a table

Tables must be predefined

```
sqlite> .tables

sqlite> CREATE TABLE meal_details (
   ...> meal_id INTEGER,
   ...> name TEXT,
   ...> date TEXT,
   ...> meal TEXT,
   ...> description TEXT,
   ...> calories REAL);

sqlite> .tables
meal_details
```

INTEGER	1, 2, 3, 4, 5...
REAL	125.33, 9.999...
TEXT	‘hello world’, ‘2020-01-01’

Step 2: Create a table

A few more keywords: IF NOT EXISTS | NOT NULL | PRIMARY KEY

```
sqlite> .tables

sqlite> CREATE TABLE IF NOT EXISTS meal_details (
   ...> meal_id INTEGER NOT NULL,
   ...> name TEXT NOT NULL,
   ...> date TEXT,
   ...> meal TEXT NOT NULL,
   ...> description TEXT,
   ...> calories REAL,
   ...> PRIMARY KEY (meal_id));

sqlite> .tables
meal_details
```

Create Tables Summary

SQL Statements

- SELECT Statements
- CREATE Statements

Steps

1. Create a database
2. Create a table
    - Column names
    - Data types: INTEGER, REAL, TEXT
    - Other: PRIMARY KEY, NOT NULL, IF NOT EXISTS
3. Insert data into the table

Chapter 2

Modify Tables

METIS®

SQL Statements

General SQL code used to interact with a database

SELECT	**INSERT**	**DELETE**
CREATE	**UPDATE**	**DROP**

Empty table

```
sqlite> .tables

sqlite> CREATE TABLE IF NOT EXISTS meal_details (
   ...> meal_id INTEGER NOT NULL,
   ...> name TEXT NOT NULL,
   ...> date TEXT,
   ...> meal TEXT NOT NULL,
   ...> description TEXT,
   ...> calories REAL,
   ...> PRIMARY KEY (meal_id));

sqlite> .tables
meal_details
```

Insert a single row / multiple rows of data into a table

```
sqlite> .tables
meal_details

sqlite> .headers on

sqlite> INSERT INTO meal_details (meal_id, name,
date, meal, description, calories) VALUES
(1,'al','2020-01-01','lunch','pizza',285);

sqlite> SELECT * FROM meal_details;
meal_id,name,date,meal,description,calories
1,al,2020-01-01,lunch,pizza,285.0
```

```
INSERT INTO table_name
(column_list)

VALUES
(value_list_1),
(value_list_2),
...
(value_list_n);
```

Insert data from a flat file into a table

meal_data.csv

```
2,bo,2020-01-01,dinner,pasta,350
3,jo,2020-01-01,breakfast,eggs,78
```

```
sqlite> .tables
meal_details

sqlite> .mode csv
sqlite> .import meal_data.csv meal_details

sqlite> SELECT * FROM meal_details;
meal_id,name,date,meal,description,calories
1,al,2020-01-01,lunch,pizza,285.0
2,bo,2020-01-01,dinner,pasta,350.0
3,jo,2020-01-01,breakfast,eggs,78.0
```

Update the table

```
sqlite> SELECT * FROM meal_details;
meal_id,name,date,meal,description,calories
1,al,2020-01-01,lunch,pizza,285.0
2,bo,2020-01-01,dinner,pasta,350.0
3,jo,2020-01-01,breakfast,eggs,78.0

sqlite> UPDATE meal_details SET calories = 500
WHERE description = ‘pizza’;

sqlite> SELECT * FROM meal_details;
meal_id,name,date,meal,description,calories
1,al,2020-01-01,lunch,pizza,500.0
2,bo,2020-01-01,dinner,pasta,350.0
3,jo,2020-01-01,breakfast,eggs,78.0
```

Delete data from the table

```
sqlite> SELECT * FROM meal_details;
meal_id,name,date,meal,description,calories
1,al,2020-01-01,lunch,pizza,500.0
2,bo,2020-01-01,dinner,pasta,350.0
3,jo,2020-01-01,breakfast,eggs,78.0

sqlite> DELETE FROM meal_details WHERE meal_id = 2;

sqlite> SELECT * FROM meal_details;
meal_id,name,date,meal,description,calories
1,al,2020-01-01,lunch,pizza,500.0
3,jo,2020-01-01,breakfast,eggs,78.0
```

Delete the whole table

```
sqlite> SELECT * FROM meal_details;
meal_id,name,date,meal,description,calories
1,al,2020-01-01,lunch,pizza,500.0
3,jo,2020-01-01,breakfast,eggs,78.0

sqlite> DROP TABLE meal_details;

sqlite> .tables

sqlite> DROP TABLE meal_details;
Error: no such table: meal_details

sqlite> DROP TABLE IF EXISTS meal_details;
```

Delete the whole database

Typically, you would write a DROP statement
DROP DATABASE meals;

It is a little bit different in SQLite

```
sqlite> .quit

my-computer$ rm meals.db
```

Modify Tables Summary

SQL Statements

- SELECT
- CREATE
- INSERT - type in data or import flat file
- UPDATE
- DELETE
- DROP

Chapter 3

JOIN Basics

METIS®

SQL query on a single table

```
SELECT *
FROM employee;
```

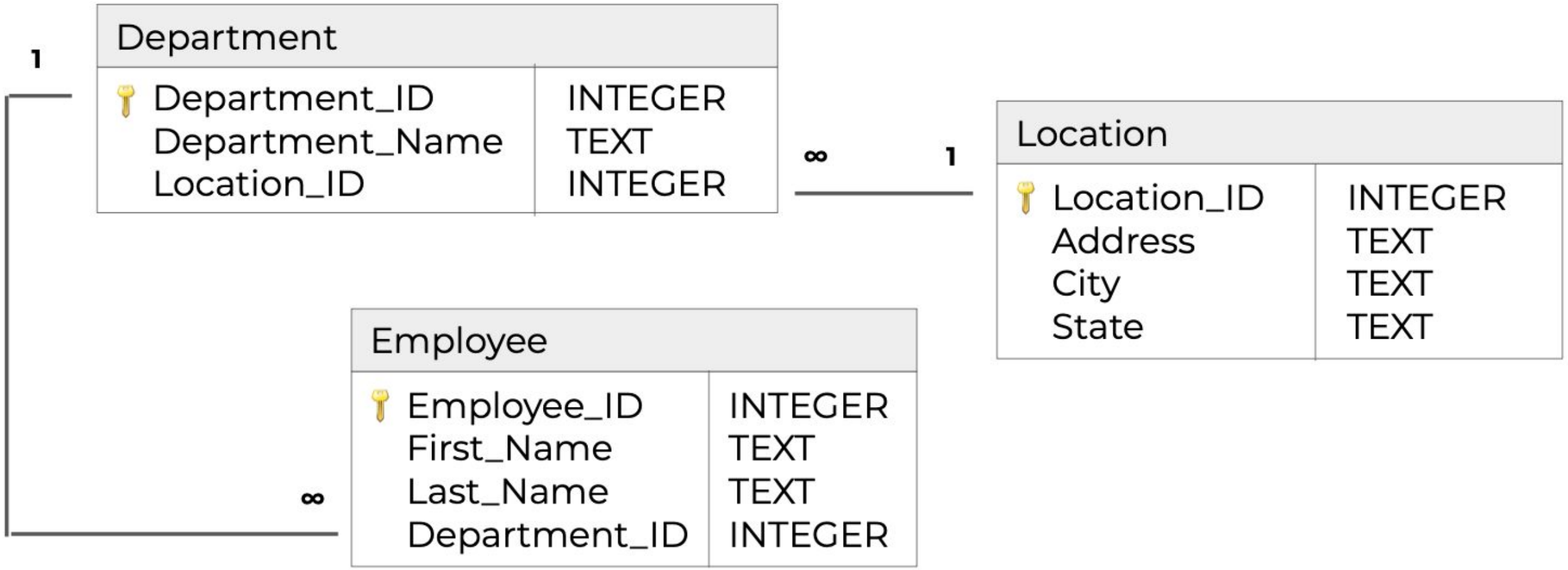
Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500
10003	Peppa	Peterson	500
10004	Sam	Scott	500
10005	Thomas	Thompson	600

SQL query on a single table

```
SELECT First_Name, Dpt_ID
FROM employee
WHERE Dpt_ID = 500
ORDER BY First_Name
LIMIT 2;
```

First_Name	Dpt_ID
Dora	500
Peppa	500

Example data model



Data from multiple tables

Department

Dpt_ID	Dpt_Name	Location_ID
400	Sales	101
500	Marketing	101
600	Technology	201

Location

Location_ID	Address	City	State
101	100 N. Michigan Ave	Chicago	IL
201	500 Market Street	SF	CA

Employee

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500
10003	Peppa	Peterson	500
10004	Sam	Scott	500
10005	Thomas	Thompson	600

Data from multiple tables

Department

Dpt_ID	Dpt_Name	Location_ID
400	Sales	101
500	Marketing	101
600	Technology	201

Employee

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500
10003	Peppa	Peterson	500
10004	Sam	Scott	500
10005	Thomas	Thompson	600

Data from multiple tables

Department

Dpt_ID	Dpt_Name	Location_ID
400	Sales	101
500	Marketing	101
600	Technology	201

Employee

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500
10003	Peppa	Peterson	500
10004	Sam	Scott	500
10005	Thomas	Thompson	600

SQL query on a multiple tables

```
SELECT *
FROM employee e JOIN department d
     ON e.dpt_id = d.dpt_id;
```

Emp_ID	First_Name	Last_Name	Dpt_ID	Dpt_ID	Dpt_Name	Location_ID
10001	Arthur	Andrews	400	400	Sales	101
10002	Dora	Davis	500	500	Marketing	101
10003	Peppa	Peterson	500	500	Marketing	101
10004	Sam	Scott	500	500	Marketing	101
10005	Thomas	Thompson	600	600	Technology	201

SQL query on a multiple tables

```
SELECT e.Emp_ID, e.First_Name, d.Dpt_Name
FROM employee e JOIN department d
      ON e.dpt_id = d.dpt_id;
```

Emp_ID	First_Name	Dpt_Name
10001	Arthur	Sales
10002	Dora	Marketing
10003	Peppa	Marketing
10004	Sam	Marketing
10005	Thomas	Technology

JOIN Summary

The JOIN in SQL allows you to query from multiple tables

- You must specify which columns match in the two tables
- You can provide an alias for each table

```
SELECT *
FROM employee e JOIN department d
     ON e.dpt_id = d.dpt_id;
```

Chapter 4

INNER / LEFT / RIGHT / OUTER JOIN

METIS®

Types of JOINs

- INNER
- LEFT
- RIGHT
- OUTER

Example tables

Movies

ID	Name	Movie
1	Alice	Coco
2	Henry	Frozen
3	Henry	Frozen II
4	Lily	Sing

Drinks

ID	Name	Drink
7	Lara	Coffee
8	Alice	Tea
9	Henry	Juice

INNER JOIN

```
SELECT *
FROM Movies m INNER JOIN Drinks d
     ON m.Name = d.Name;
```

ID	Name	Movie	ID	Name	Drink
1	Alice	Coco	8	Alice	Tea
2	Henry	Frozen	9	Henry	Juice
3	Henry	Frozen II	9	Henry	Juice

Movies

ID	Name	Movie
1	Alice	Coco
2	Henry	Frozen
3	Henry	Frozen II
4	Lily	Sing

Drinks

ID	Name	Drink
7	Lara	Coffee
8	Alice	Tea
9	Henry	Juice

LEFT JOIN

```
SELECT *
FROM Movies m LEFT JOIN Drinks d
     ON m.Name = d.Name;
```

ID	Name	Movie	ID	Name	Drink
1	Alice	Coco	8	Alice	Tea
2	Henry	Frozen	9	Henry	Juice
3	Henry	Frozen II	9	Henry	Juice
4	Lily	Sing	NULL	NULL	NULL

Movies

ID	Name	Movie
1	Alice	Coco
2	Henry	Frozen
3	Henry	Frozen II
4	Lily	Sing

Drinks

ID	Name	Drink
7	Lara	Coffee
8	Alice	Tea
9	Henry	Juice

RIGHT JOIN

```
SELECT *
FROM Movies m RIGHT JOIN Drinks d
     ON m.Name = d.Name;
```

ID	Name	Movie	ID	Name	Drink
NULL	NULL	NULL	7	Lara	Coffee
1	Alice	Coco	8	Alice	Tea
2	Henry	Frozen	9	Henry	Juice
3	Henry	Frozen II	9	Henry	Juice

Movies

ID	Name	Movie
1	Alice	Coco
2	Henry	Frozen
3	Henry	Frozen II
4	Lily	Sing

Drinks

ID	Name	Drink
7	Lara	Coffee
8	Alice	Tea
9	Henry	Juice

OUTER JOIN

```
SELECT *
FROM Movies m OUTER JOIN Drinks d
     ON m.Name = d.Name;
```

ID	Name	Movie	ID	Name	Drink
NULL	NULL	NULL	7	Lara	Coffee
1	Alice	Coco	8	Alice	Tea
2	Henry	Frozen	9	Henry	Juice
3	Henry	Frozen II	9	Henry	Juice
4	Lily	Sing	NULL	NULL	NULL

Movies

ID	Name	Movie
1	Alice	Coco
2	Henry	Frozen
3	Henry	Frozen II
4	Lily	Sing

Drinks

ID	Name	Drink
7	Lara	Coffee
8	Alice	Tea
9	Henry	Juice

Types of JOINs

INNER

Only matching rows are returned & the default JOIN

LEFT

All rows of the first table are returned & more common than RIGHT JOIN

RIGHT

All rows of the second table are returned

OUTER

All rows of both tables are returned

NOTE: Regardless of type, joins return all matches, including possible repeats

Chapter 5

More on JOINs

METIS®

More on JOINs

1. Join multiple tables
2. Join on multiple columns
3. Self joins

1. Join multiple tables

Department

Dpt_ID	Dpt_Name	Location_ID
400	Sales	101
500	Marketing	101
600	Technology	201

Location

Location_ID	Address	City	State
101	100 N. Michigan Ave	Chicago	IL
201	500 Market Street	SF	CA

Employee

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500
10003	Peppa	Peterson	500
10004	Sam	Scott	500
10005	Thomas	Thompson	600

1. Join multiple tables

Department

Dpt_ID	Dpt_Name	Location_ID
400	Sales	101
500	Marketing	101
600	Technology	201

Location

Location_ID	Address	City	State
101	100 N. Michigan Ave	Chicago	IL
201	500 Market Street	SF	CA

Employee

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500
10003	Peppa	Peterson	500
10004	Sam	Scott	500
10005	Thomas	Thompson	600

1. Join multiple tables

Department

Dpt_ID	Dpt_Name	Location_ID
400	Sales	101
500	Marketing	101
600	Technology	201

Location

Location_ID	Address	City	State
101	100 N. Michigan Ave	Chicago	IL
201	500 Market Street	SF	CA

Employee

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500
10003	Peppa	Peterson	500
10004	Sam	Scott	500
10005	Thomas	Thompson	600

1. Join multiple tables

Department

Dpt_ID	Dpt_Name	Location_ID
400	Sales	101
500	Marketing	101
600	Technology	201

Location

Location_ID	Address	City	State
101	100 N. Michigan Ave	Chicago	IL
201	500 Market Street	SF	CA

Employee

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500
10003	Peppa	Peterson	500
10004	Sam	Scott	500
10005	Thomas	Thompson	600

1. Join multiple tables

For each employee, what is their department name and address?

1. Join multiple tables

```
SELECT *
FROM employee e
     LEFT JOIN department d ON e.dpt_id = d.dpt_id
     LEFT JOIN location l ON d.loc_id = l.loc_id;
```

Emp_ ID	First_ Name	Last_ Name	Dpt ID	Dpt ID	Dpt_ Name	Loc ID	Loc ID	Address	City	State
10001	Arthur	Andrews	400	400	Sales	101	101	100 N. Michigan	Chicago	IL
10002	Dora	Davis	500	500	Marketing	101	101	100 N. Michigan	Chicago	IL
10003	Peppa	Peterson	500	500	Marketing	101	101	100 N. Michigan	Chicago	IL
10004	Sam	Scott	500	500	Marketing	101	101	100 N. Michigan	Chicago	IL
10005	Thomas	Thompson	600	600	Technology	201	201	500 Market St	SF	CA

2. Join on multiple columns

Doctors

Name	Day	Location	Details
Arthur	Monday	Chicago	Check ups
Arthur	Sunday	Chicago	On call
Dora	Monday	Evanston	Surgery
Dora	Wednesday	Evanston	Surgery
Dora	Sunday	Chicago	On call

Rate

Day	Office	Rate
Monday	Chicago	90
Monday	Evanston	210
Wednesday	Evanston	210
Sunday	Chicago	20

2. Join on multiple columns

Doctors

Name	Day	Location	Details
Arthur	Monday	Chicago	Check ups
Arthur	Sunday	Chicago	On call
Dora	Monday	Evanston	Surgery
Dora	Wednesday	Evanston	Surgery
Dora	Sunday	Chicago	On call

Rate

Day	Office	Rate
Monday	Chicago	90
Monday	Evanston	210
Wednesday	Evanston	210
Sunday	Chicago	20

2. Join on multiple columns

Doctors

Name	Day	Location	Details
Arthur	Monday	Chicago	Check ups
Arthur	Sunday	Chicago	On call
Dora	Monday	Evanston	Surgery
Dora	Wednesday	Evanston	Surgery
Dora	Sunday	Chicago	On call

Rate

Day	Office	Rate
Monday	Chicago	90
Monday	Evanston	210
Wednesday	Evanston	210
Sunday	Chicago	20

2. Join on multiple columns

For each doctor, what is their rate depending on the day and location?

2. Join on multiple columns

```
SELECT *
FROM doctors d INNER JOIN rate r
     ON d.Day = r.Day AND d.Location = r.Office;
```

Name	Day	Location	Details	Day	Office	Rate
Arthur	Monday	Chicago	Check ups	Monday	Chicago	90
Arthur	Sunday	Chicago	On call	Sunday	Chicago	20
Dora	Monday	Evanston	Surgery	Monday	Evanston	210
Dora	Wednesday	Evanston	Surgery	Wednesday	Evanston	210
Dora	Sunday	Chicago	On call	Sunday	Chicago	20

3. Self join

Doctors

Name	Day	Location	Details
Arthur	Monday	Chicago	Check ups
Arthur	Sunday	Chicago	On call
Dora	Monday	Evanston	Surgery
Dora	Wednesday	Evanston	Surgery
Dora	Sunday	Chicago	On call
Peppa	Wednesday	Evanston	Check ups

3. Self join

Doctors

Name	Day	Location	Details
Arthur	Monday	Chicago	Check ups
Arthur	Sunday	Chicago	On call
Dora	Monday	Evanston	Surgery
Dora	Wednesday	Evanston	Surgery
Dora	Sunday	Chicago	On call
Peppa	Wednesday	Evanston	Check ups

3. Self join

Doctors

Name	Day	Location	Details
Arthur	Monday	Chicago	Check ups
Arthur	Sunday	Chicago	On call
Dora	Monday	Evanston	Surgery
Dora	Wednesday	Evanston	Surgery
Dora	Sunday	Chicago	On call
Peppa	Wednesday	Evanston	Check ups

3. Self join

For each doctor, who else is in the office when they are?

3. Self join

```
SELECT *
FROM   doctors t1, doctors t2
WHERE  t1.Day = t2.Day AND t1.Location = t2.Location
       AND t1.Name <> t2.Name
ORDER BY Name;
```

Name	Day	Location	Details	Name	Day	Location	Details
Arthur	Sunday	Chicago	On call	Dora	Sunday	Chicago	On call
Dora	Sunday	Chicago	On call	Arthur	Sunday	Chicago	On call
Dora	Wednesday	Evanston	Surgery	Peppa	Wednesday	Evanston	Check ups
Peppa	Wednesday	Evanston	Check ups	Dora	Wednesday	Evanston	Surgery

JOIN Summary

```
-- 1. Join multiple tables
SELECT *
FROM employee e
     LEFT JOIN department d ON e.dpt_id = d.dpt_id
     LEFT JOIN location l ON d.loc_id = l.loc_id;
```

```
-- 2. Join on multiple columns
SELECT *
FROM doctors d INNER JOIN rate r
     ON d.Day = r.Day AND d.Location = r.Office;
```

JOIN Summary

```
-- 3. Self join
SELECT *
FROM   doctors t1, doctors t2
WHERE  t1.Day = t2.Day
       AND t1.Location = t2.Location
       AND t1.Name <> t2.Name
ORDER BY Name;
```

Chapter 6

UNION

METIS®

Ways to pull data from multiple tables

Name	Movie	Genre
...	...	...
...	...	....

Name	Drink	Color
...	...	...
...	...	....

This can be done with a JOIN

Ways to pull data from multiple tables

Name	Movie	Genre
...	...	...
...	...	....

Name	Movie	Genre
...	...	...
...	...	....

This can be done with a UNION

UNION

Doctors_Chicago

Name	Details
Arthur	Surgery
Dora	Surgery
Peppa	Check ups
Sam	Check ups

Doctors_Evanston

Name	Details
Dora	Surgery
Peppa	Surgery
Thomas	Check ups

UNION

Doctors_Chicago

Name	Details
Arthur	Surgery
Dora	Surgery
Peppa	Check ups
Sam	Check ups

Doctors_Evanston

Name	Details
Dora	Surgery
Peppa	Surgery
Thomas	Check ups

UNION

Doctors_Chicago

Name	Details
Arthur	Surgery
Dora	Surgery
Peppa	Check ups
Sam	Check ups

Doctors_Evanston

Name	Details
Dora	Surgery
Peppa	Surgery
Thomas	Check ups

UNION

```
SELECT * FROM Doctors_Chicago
UNION
SELECT * FROM Doctors_Evanston;
```

Name	Details
Arthur	Surgery
Dora	Surgery
Peppa	Check ups
Peppa	Surgery
Sam	Check ups
Thomas	Check ups

UNION ALL

```
SELECT * FROM Doctors_Chicago
UNION ALL
SELECT * FROM Doctors_Evanston;
```

Name	Details
Arthur	Surgery
Dora	Surgery
Peppa	Check ups
Sam	Check ups
Dora	Surgery
Peppa	Surgery
Thomas	Check ups

UNION ALL

```
SELECT * FROM Doctors_Chicago
UNION ALL
SELECT * FROM Doctors_Evanston;
```

Name	Details
Arthur	Surgery
Dora	Surgery
Peppa	Check ups
Sam	Check ups
Dora	Surgery
Peppa	Surgery
Thomas	Check ups

JOIN vs UNION

Different ways to combine multiple tables

JOIN

Name	Movie	Genre
...	...	...
...	...	....

Name	Drink	Color
...	...	...
...	...	....

UNION

Name	Movie	Genre
...	...	...
...	...	....

Name	Movie	Genre
...	...	...
...	...	....

UNION Summary

Combines multiple tables along all of its columns

UNION

Remove duplicate rows

UNION ALL

Keeps all rows, including duplicates

Chapter 7

SQL and Python

METIS®

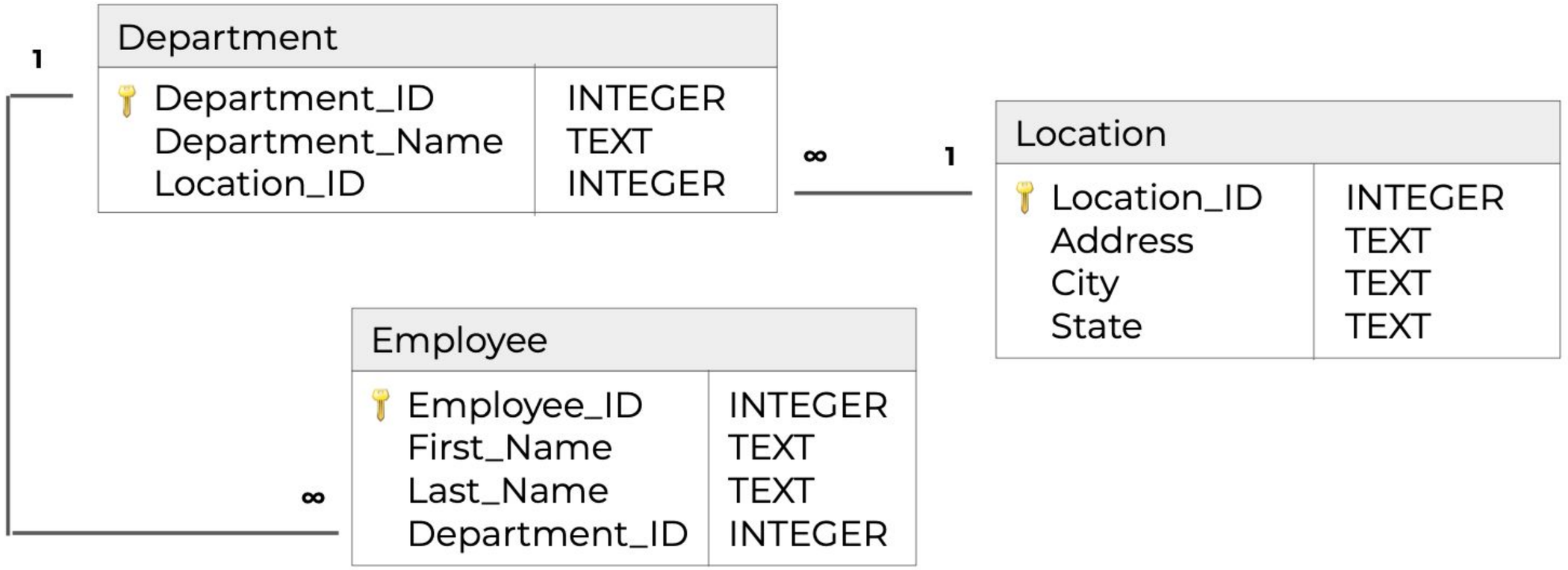
Data Science Workflow with SQL and Python

1. Define the problem
2. Go to the data warehouse
3. Query the database
4. Continue analysis in Python

1. Define the problem

Create a model that will predict which employees are likely to leave the company

2. Go to the data warehouse



3. Query the database

```
SELECT   e.employee_name, p.salary, o.manager
FROM    Employees e LEFT JOIN
        Payslips p ON e.emp_id = p.emp_id
        Organization ON e.mgr_id = o.mgr_id
WHERE   p.year > 2000;
```

3. Query the database

Left Within One Year?	Salary	Starting Month	Manager ID	Amount of Turnover for Job in 5 Years
Yes	$50K	March	102	3
No	$55K	January	527	0
…	…	…	…	…

4. Continue analysis in Python

- Connect Python to a database
- Execute SQL queries within Python
- Save outputs as pandas dataframes
- Continue more machine learning focused tasks in Python

Data Science Workflow with SQL and Python

1. Define the problem
2. Go to the data warehouse
3. Query the database
4. Continue analysis in Python

} Connect Python to a database

SQLAlchemy

- Python toolkit
- Maps database objects to Python objects

```
# install via Anaconda
!conda install -c anaconda sqlalchemy


# import
from sqlalchemy import create_engine
```

Connect to a database

```
# connect to a local database
engine = create_engine("sqlite:///my_database.db")

# engine contains details about the database
engine.table_names()
```

```
['employees','payslips,'organization']
```

Write SQL queries in a Python environment

```
import pandas as pd


df = pd.read_sql('SELECT * FROM employees;', engine)
df
```

Emp_ID	First Name	Last Name	Dpt_ID	Mgr_ID
101	Henry	Harper	202	195
102	Lily	Little	205	151
...	...	...	...	...

Summary

Data Science Workflow with SQL and Python

1. Define the problem
2. Go to the data warehouse
3. Query the database
4. Continue analysis in Python

SQLAlchemy

- Write SQL code in Python
- Maps database objects to Python objects