SQL 1
Introduction to Databases & Basic SQL Queries

METIS®

Objectives

Background

Understand what a database is and why SQL is useful

Tool

Write SQL queries within SQLite and DB Browser for SQLite

Application

- Use SQL to select, filter, sort and limit data
- Use SQL to aggregate and filter aggregated data

Chapter 1

Introduction to Databases

METIS®

What is a database?

It is a place to store data in an organized way

There are two types of databases: **relational** and **non-relational**

Relational databases

A relational database consists of many relations aka **tables**

Example table

Name	**Birthday**	**City**	**State**
Arthur	1/1/90	Austin	TX
Dora	2/2/00	Denver	CO
Peppa	3/3/10	Pittsburgh	PA

← 1 row of data

↑
1 column of data

Note: Each row of a table is unique and unordered

Relational databases

The structure of the table must be **predefined**

Example:

- All values in the name column must be a text data type
- All values in the birthday column must be a date data type

Name	Birthday	City	State
Arthur	1/1/90	Austin	TX
Dora	2/2/00	Denver	CO
Peppa	3/3/10	Pittsburgh	PA

Relational databases

To access data in a relational database, you must know **SQL** (structured query language)

In English: “Show me all of the data in the table.”

In SQL: `SELECT * FROM Table;` ← This is called a SQL query

Name	Birthday	City	State
Arthur	1/1/90	Austin	TX
Dora	2/2/00	Denver	CO
Peppa	3/3/10	Pittsburgh	PA

This lesson focuses on
relational databases and **SQL**,
but as a quick side note...

Non-relational databases

- Data is stored in **other formats** besides tables, such as dictionaries
- The structure of the database is not predefined, but **dynamic**
- Sometimes called **NoSQL** databases, which stands for “Not Only SQL”

Biggest takeaway: SQL ≠ NoSQL

Now back to

relational databases and **SQL**

Relational database software

A **Relational Database Management System (RDBMS)** is software used to manage a relational database.

Popular examples include:

- **SQLite** ← This is the RDBMS we will be focusing on in this lesson
- MySQL
- PostgreSQL
- Microsoft SQL Server
- Oracle

Biggest takeaway: SQL ≠ SQLite

Note: SQL syntax varies slightly for each RDBMS

Who knows SQL?

Roles	Tasks
Database Engineers Database Administrators (DBA)	- Figure out how to structure a database - Move data around in a database - Manage user permissions
Data Analysts Data Scientists	- Read data from a database aka query tables

This is the task we will be focusing on in this lesson

Database Summary

A **database** is a place to store data in an organized way

A **relational database** puts data into many tables

To work with data in a relational database, you must know **SQL**

For this lesson, we will be using an **RDBMS** called **SQLite**

We will focus on reading data from a database, or writing **SQL queries**

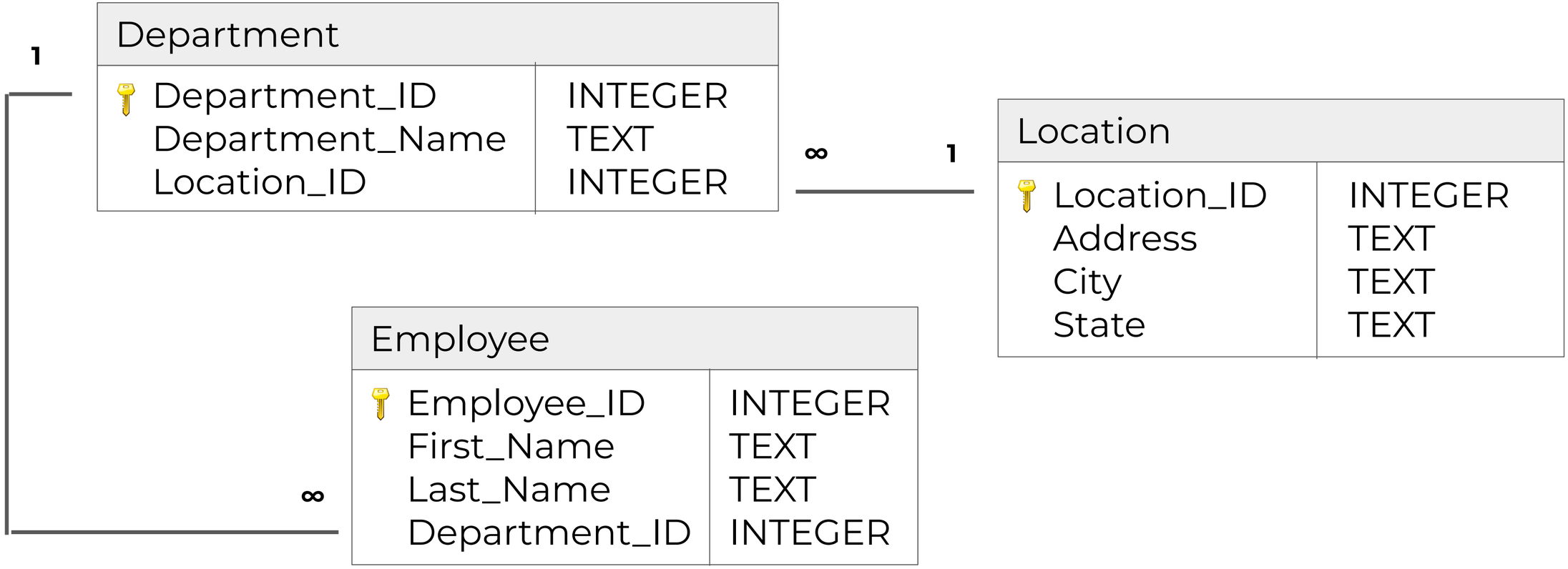
Chapter 2

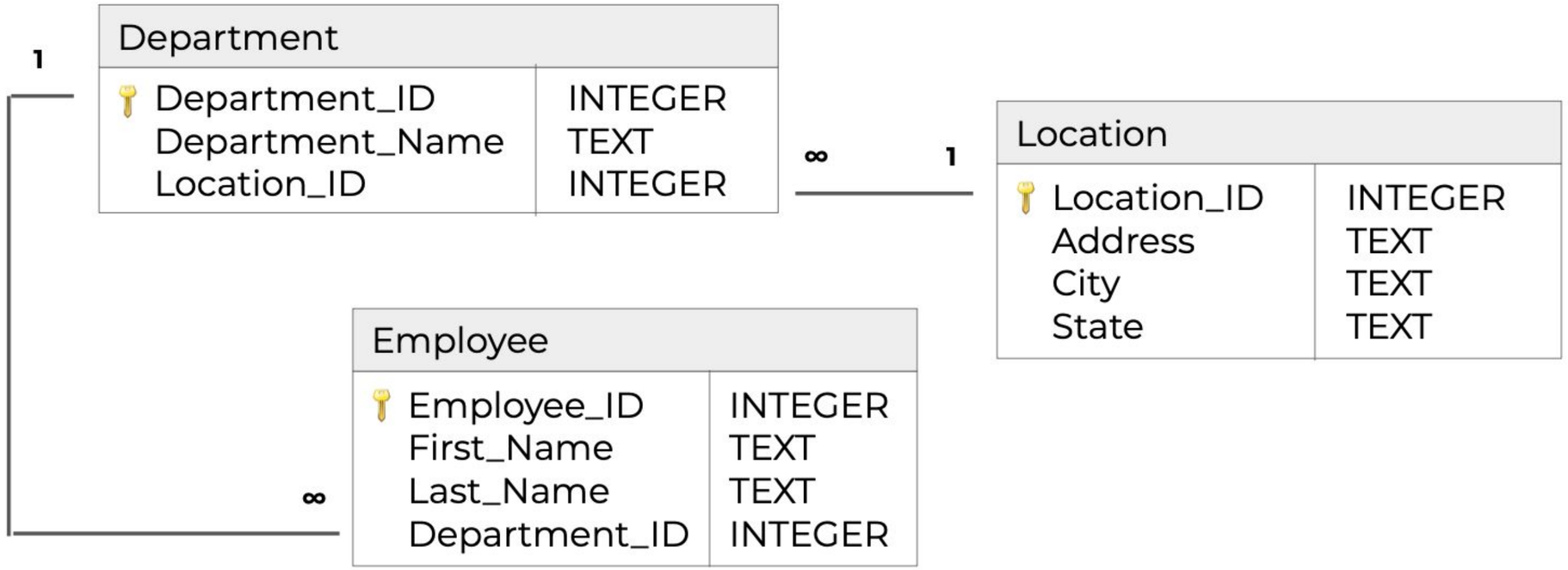
Data Models

METIS®

What is a data model?

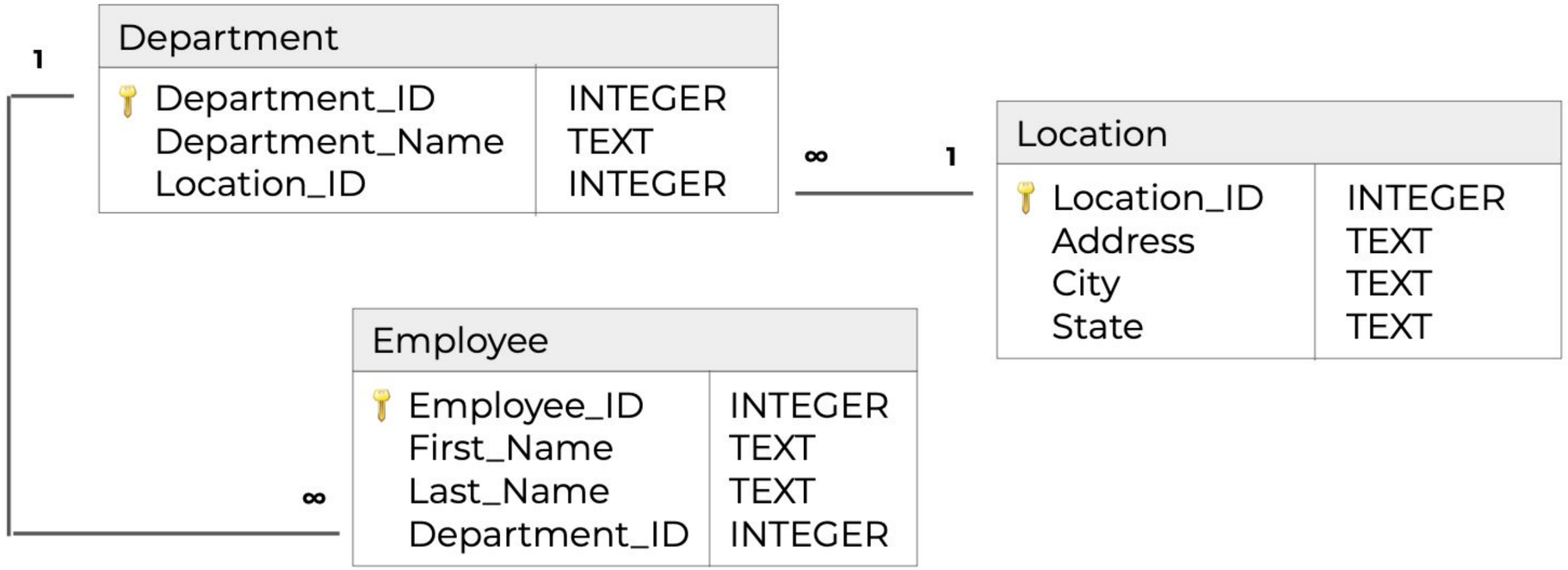
It is a visual representation of how the tables in a database are organized





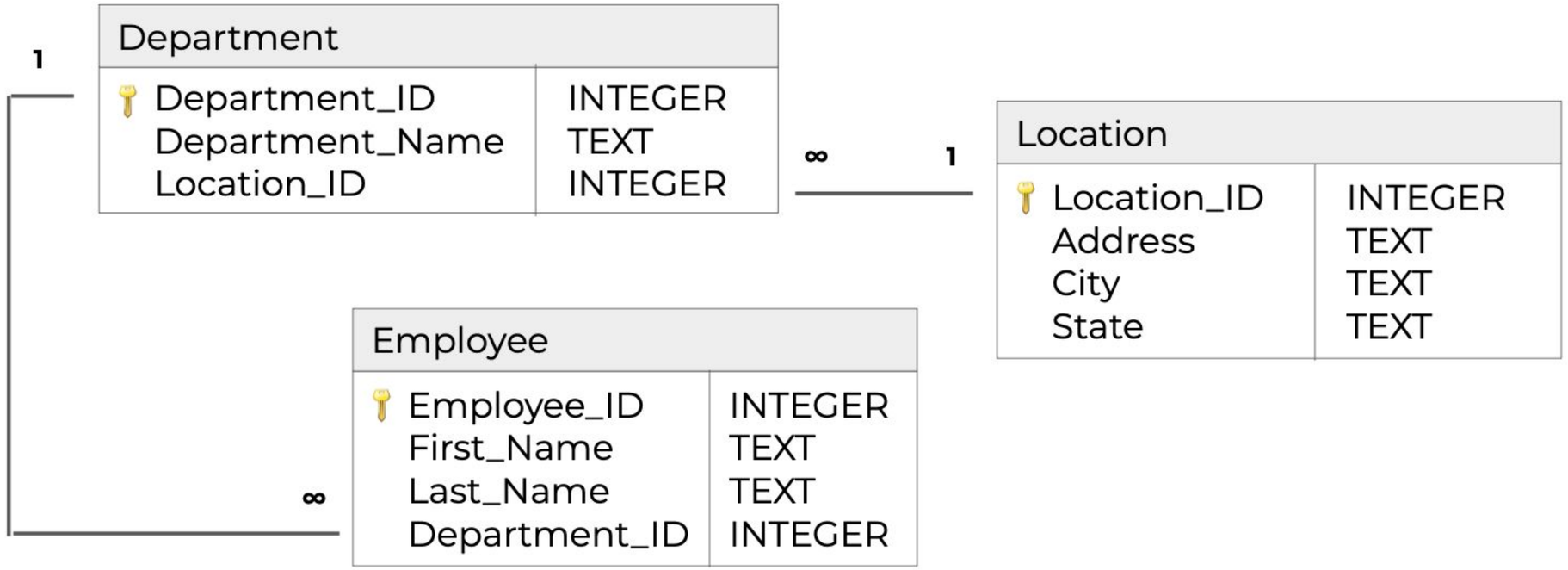
What does this data model represent?

This is a data model of the employee database at a company.



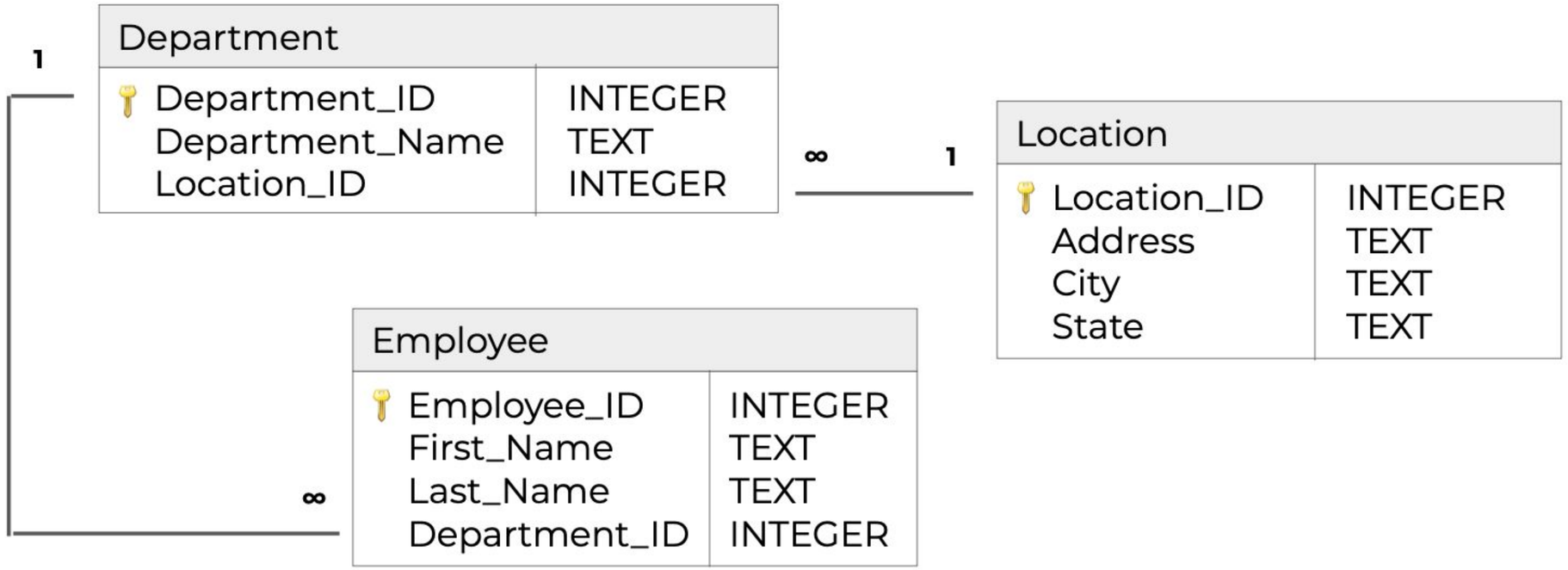
How many tables are in this employee database?

There are three tables - Employee, Department and Location.



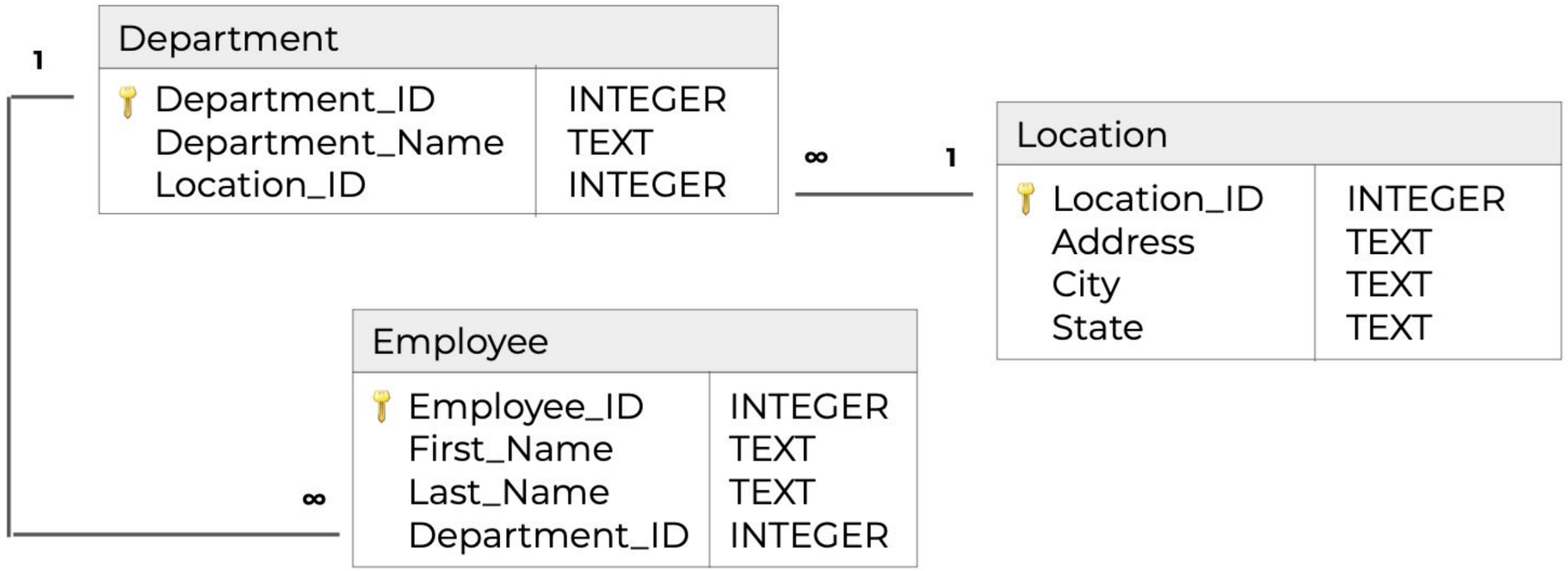
What is the additional information in the Employee table?

There are four columns in the table with data types of integer and text.



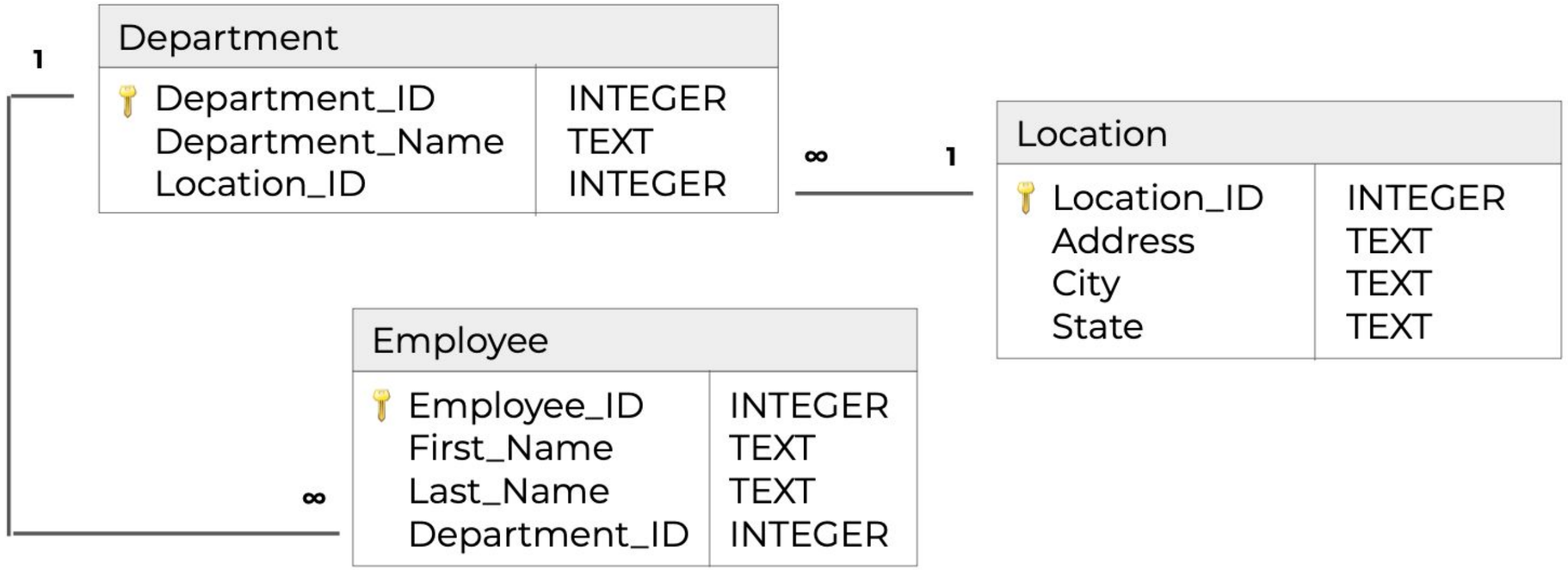
What does the key icon represent?

It is the **primary key**, or the unique identifier, of the table.



How come Department_ID is in multiple tables?

The Department_ID in the Department table links to the Department_ID in the Employee table, and is called the **foreign key**.



What do the lines / 1 / ∞ symbols represent?

The line shows the link between primary and foreign keys in two tables, and the 1 and ∞ represent a **one-to-many relationship**.

Data Model Summary

A **data model** is a visual representation of how a database is organized

Each box in a data model represents a table along with the **table name**, **column names** and corresponding **data types** of the table

Columns in a table are marked as a **primary key** if they uniquely identify a row of data and a **foreign key** if they link to other tables

One row in a table can **link** to one or more rows in other tables

It is useful to **review a data model** before writing SQL queries

Chapter 3

SQLite Setup

METIS®

Key Terms

SQL
The language for working with relational databases

SQLite
Basic relational database management system (RDBMS) software for writing SQL code

DB Browser for SQLite
Application that provides a more user-friendly interface for writing SQL code

What is SQLite?

There are many RDBMS options for writing SQL code

Some are proprietary:

- Microsoft SQL Server
- Oracle

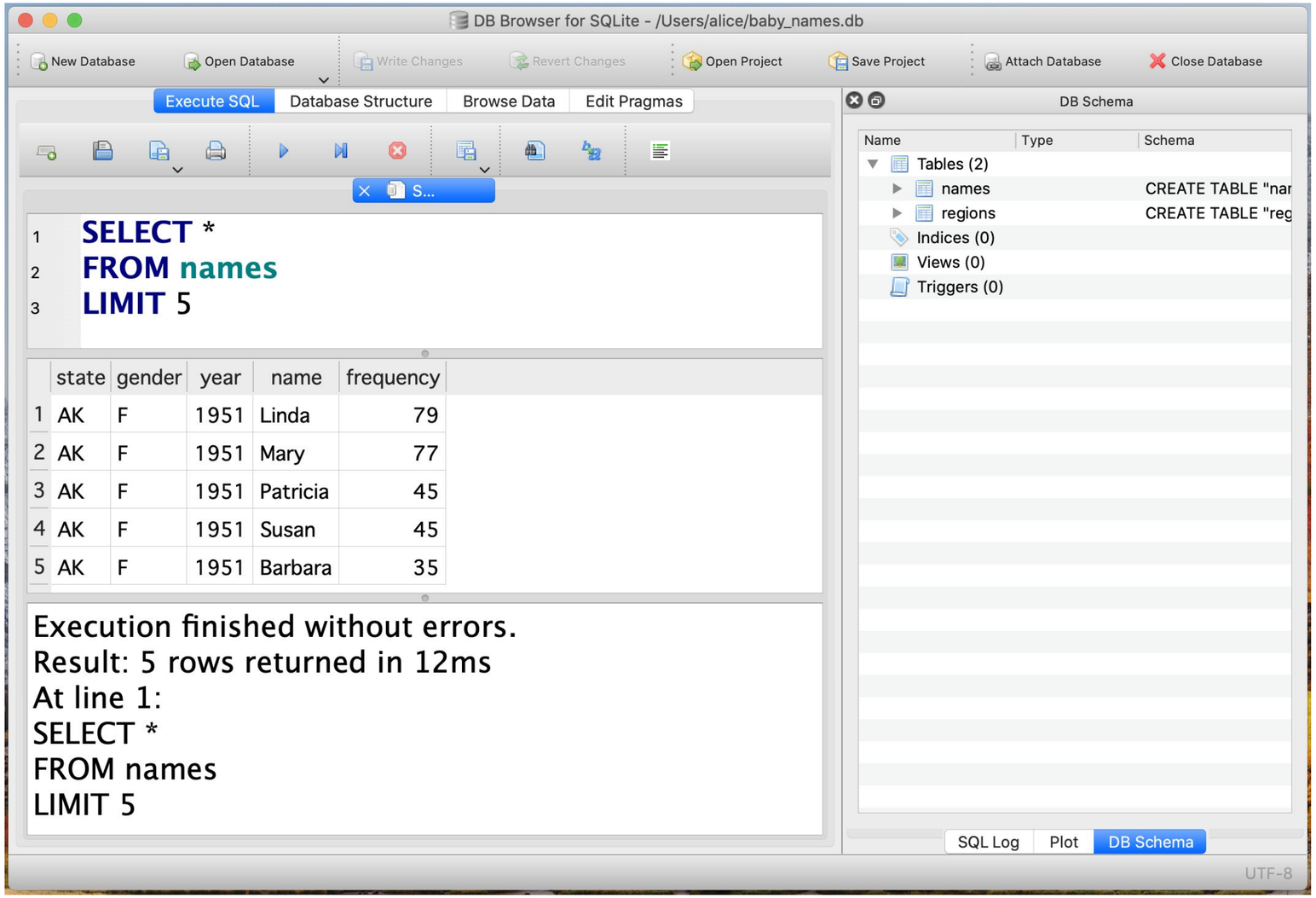
Some are open source:

- MySQL
- PostgreSQL
- SQLite ← lightweight and easy to install, great for beginners

SQLite in a Terminal Window

```
my-computer$ sqlite3 baby_names.db

sqlite> .tables
names    regions

sqlite> .headers on

sqlite> select * from names limit 3;
state|gender|year|name|frequency
AK|F|1951|Linda|79
AK|F|1951|Mary|77
AK|F|1951|Patricia|45

sqlite> .quit
```

DB Browser for SQLite (v3.12.0)

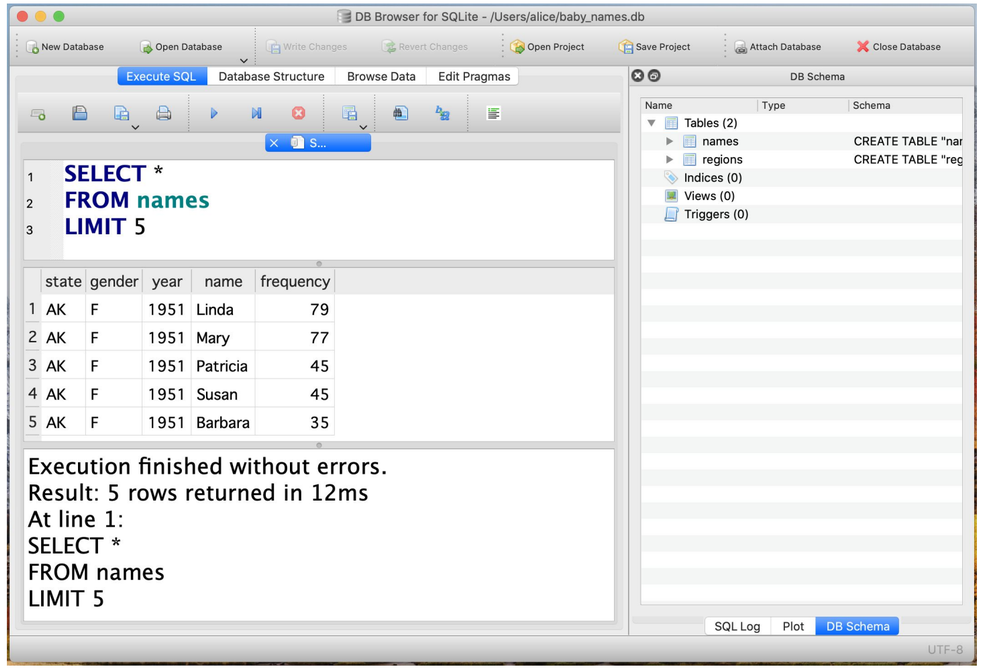


SQLite Summary

SQLite

```
my-computer$ sqlite3 baby_names.db

sqlite> .tables
names     regions

sqlite> .headers on

sqlite> select * from names limit 3;
state|gender|year|name|frequency
AK|F|1951|Linda|79
AK|F|1951|Mary|77
AK|F|1951|Patricia|45

sqlite> .quit
```

DB Browser for SQLite



Chapter 4

SQL Basics: SELECT, WHERE, ORDER BY, LIMIT

METIS®

The Simplest SQL Query

Let's say you are looking at the employees database and you want to see all of the data in the employee table.

```
SELECT *
FROM employee;
```

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500
10003	Peppa	Peterson	500

Side Notes

SQL is case insensitive

Whitespace is optional

The semicolon is required

SELECT

Specifies what column(s) to return

```
SELECT First_Name, Last_Name
FROM employee;
```

First_Name	Last_Name
Arthur	Andrews
Dora	Davis
Peppa	Peterson

FROM

Specifies what table(s) to read from

```
SELECT *
FROM employee;
```

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500
10003	Peppa	Peterson	500

WHERE

Filters rows of data

```
SELECT *
FROM employee
WHERE First_Name = ‘Arthur’;
```

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400

Side Notes

Text must be in quotes

Can filter on multiple fields using AND and OR

Other logical operators such as > or < are allowed

WHERE

AND and less than or equal to

```
SELECT *
FROM employee
WHERE First_Name = ‘Arthur’ AND Dpt_ID <= 400;
```

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400

ORDER BY

Sorts columns of data

```
SELECT *
FROM employee
ORDER BY Emp_ID DESC;
```

Emp_ID	First_Name	Last_Name	Dpt_ID
10003	Peppa	Peterson	500
10002	Dora	Davis	500
10001	Arthur	Andrews	400

Side Notes

The default order is ASC

Can sort on multiple fields

LIMIT

Specifies how many rows to return

```
SELECT *
FROM employee
LIMIT 2;
```

Emp_ID	First_Name	Last_Name	Dpt_ID
10001	Arthur	Andrews	400
10002	Dora	Davis	500

Put It All Together

```
SELECT First_Name, Dpt_ID
FROM employee
WHERE Dpt_ID >= 400
ORDER BY First_Name
LIMIT 2;
```

First_Name	Dpt_ID
Arthur	400
Dora	500

Side Notes

These are 5 of the 7 basic clauses

They must always be written in this order

SELECT is required, the remaining are optional

Chapter 5

Aggregations, DISTINCT and GROUP BY

METIS®

The employee table

```
SELECT *
FROM employee;
```

Emp_ID	First_Name	Last_Name	Date	Sale
10001	Arthur	Andrews	7/1/20	5000
10002	Dora	Davis	7/1/20	100
10002	Dora	Davis	7/2/20	299
10002	Dora	Davis	7/5/20	7050
10003	Peppa	Peterson	7/3/20	3000
10003	Peppa	Peterson	7/4/20	2999

COUNT

To count the number of rows in a table

```
SELECT COUNT(*)
FROM employee;
```

COUNT(*)
6

COUNT

A few modifications to COUNT(*)

```
SELECT COUNT(Emp_ID)
FROM employee;
```

COUNT(*)
6

```
SELECT COUNT(Emp_ID) as cnt
FROM employee;
```

cnt
6

DISTINCT

To find the unique values in a column

```
SELECT DISTINCT Emp_ID
FROM employee;
```

Emp_ID
10001
10002
10003

COUNT(DISTINCT)

To find the number of unique values in a column

```
SELECT COUNT(DISTINCT Emp_ID) as num_emp
FROM employee;
```

num_emp
3

Aggregations

There are five aggregation options in SQL

```
SELECT  COUNT(Sale), SUM(Sale),
        MIN(Sale), MAX(Sale), AVG(Sale)
FROM    employee;
```

COUNT(Sale)	SUM(Sale)	MIN(Sale)	MAX(Sale)	AVG(Sale)
6	18448	100	7050	3075

Basic SQL Clauses

```
SELECT
FROM
WHERE
ORDER BY
LIMIT
```

Let's add one more

```
SELECT
FROM
WHERE
GROUP BY
ORDER BY
LIMIT
```

GROUP BY

Count the number of sales
for all employees

```
SELECT COUNT(Sale)
FROM employee;
```

COUNT(*)
6

Count the number of sales
for each employee

```
SELECT Emp_ID, COUNT(Sale)
FROM employee
GROUP BY Emp_ID;
```

Emp_ID	COUNT(Sale)
10001	1
10002	3
10003	2

GROUP BY

Count the number of sales **for each** employee

```
SELECT Emp_ID, COUNT(Sale)
FROM employee
GROUP BY Emp_ID;
```

Emp_ID	COUNT(Sale)
10001	1
10002	3
10003	2

GROUP BY Steps

1. How to group rows of data?
By Employee ID

2. Once rows are groups, what aggregation should be returned?
Number of sales

GROUP BY

Original employee table

Emp_ID	First_Name	Last_Name	Date	Sale
10001	Arthur	Andrews	7/1/20	5000
10002	Dora	Davis	7/1/20	100
10002	Dora	Davis	7/2/20	299
10002	Dora	Davis	7/5/20	7050
10003	Peppa	Peterson	7/3/20	3000
10003	Peppa	Peterson	7/4/20	2999

GROUP BY

How to group the rows of data?

Emp_ID	First_Name	Last_Name	Date	Sale
10001	Arthur	Andrews	7/1/20	5000
10002	Dora	Davis	7/1/20	100
10002	Dora	Davis	7/2/20	299
10002	Dora	Davis	7/5/20	7050
10003	Peppa	Peterson	7/3/20	3000
10003	Peppa	Peterson	7/4/20	2999

```
SELECT Emp_ID, COUNT(Sale)
FROM employee
GROUP BY Emp_ID;
```

GROUP BY

Once rows are grouped, what aggregation should be returned?

Emp_ID	First_Name	Last_Name	Date	Sale
10001	Arthur	Andrews	7/1/20	**5000**
10002	Dora	Davis	7/1/20	**100**
10002	Dora	Davis	7/2/20	**299**
10002	Dora	Davis	7/5/20	**7050**
10003	Peppa	Peterson	7/3/20	**3000**
10003	Peppa	Peterson	7/4/20	**2999**

```
SELECT Emp_ID, COUNT(Sale)
FROM employee
GROUP BY Emp_ID;
```

Emp_ID	COUNT(Sale)
10001	1
10002	3
10003	2

Summary

Aggregations

- Summarize all of the rows of data in a table in some way
- COUNT | SUM | MIN | MAX | AVG

DISTINCT

- Show unique values of a column of data

GROUP BY

1. Pick one or more columns to group the data
2. Pick one or more aggregations to return

Chapter 6

Operators and HAVING

METIS®

The employee table

```
SELECT *
FROM employee;
```

Emp_ID	First_Name	Last_Name	Date	Sale
10001	Arthur	Andrews	7/1/20	5000
10002	Dora	Davis	NULL	100
10002	Dora	Davis	7/2/20	299
10002	Dora	Davis	7/5/20	7050
10003	Peppa	Peterson	7/3/20	3000
10003	Peppa	Peterson	7/4/20	2999

Operators

Arithmetic Operators

- +
- -
- *
- /
- %

Logical Operators

- AND
- OR
- BETWEEN
- IN
- LIKE

Comparison Operators

- =
- > < >= <=
- != <>

NULL Operators

- IS
- IS NOT

Arithmetic operators

```
SELECT Emp_ID, Sale - 100 AS after_fee
FROM employee;
```

Emp_ID	after_fee
10001	4900
10002	0
10002	199
10002	6950
10003	2900
10003	2899

All arithmetic operators

+ - * / %

AND / OR

```
SELECT *
FROM employee
WHERE First_Name = ‘Dora’ AND Sale >= 299;
```

Emp_ID	First_Name	Last_Name	Date	Sale
10002	Dora	Davis	7/2/20	299
10002	Dora	Davis	7/5/20	7050

BETWEEN

```
SELECT *
FROM employee
WHERE Sale BETWEEN 1000 AND 6000;
```

Emp_ID	First_Name	Last_Name	Date	Sale
10001	Arthur	Andrews	7/1/20	5000
10003	Peppa	Peterson	7/3/20	3000
10003	Peppa	Peterson	7/4/20	2999

IN

```
SELECT *
FROM employee
WHERE First_Name IN (‘Arthur’, ‘Peppa’);
```

Emp_ID	First_Name	Last_Name	Date	Sale
10001	Arthur	Andrews	7/1/20	5000
10003	Peppa	Peterson	7/3/20	3000
10003	Peppa	Peterson	7/4/20	2999

LIKE

```
SELECT *
FROM employee
WHERE Last_Name LIKE ‘P%’;
```

Emp_ID	First_Name	Last_Name	Date	Sale
10003	Peppa	Peterson	7/3/20	3000
10003	Peppa	Peterson	7/4/20	2999

Text is **case sensitive**

Not equal

```
SELECT *
FROM employee
WHERE First_Name !=
‘Peppa’;
```

An alternative **<>**

Emp_ID	First_Name	Last_Name	Date	Sale
10001	Arthur	Andrews	7/1/20	5000
10002	Dora	Davis	NULL	100
10002	Dora	Davis	7/2/20	299
10002	Dora	Davis	7/5/20	7050

NULL

```
SELECT *
FROM employee
WHERE Date IS NULL;
```

The opposite is
WHERE Date IS NOT NULL

Emp_ID	First_Name	Last_Name	Date	Sale
10002	Dora	Davis	NULL	100

Operators

Arithmetic Operators

- +
- -
- *
- /
- %

Logical Operators

- AND
- OR
- BETWEEN
- IN
- LIKE

Comparison Operators

- =
- > < >= <=
- != <>

NULL Operators

- IS
- IS NOT

Basic SQL Clauses

```
SELECT
FROM
WHERE
GROUP BY
ORDER BY
LIMIT
```

Let's add one more

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
LIMIT
```

GROUP BY Review

Count the number of sales **for each** employee

```
SELECT Emp_ID, COUNT(Sale)
FROM employee
GROUP BY Emp_ID;
```

Emp_ID	COUNT(Sale)
10001	1
10002	3
10003	2

HAVING

Which employees had more than 1 sale? Filter on the GROUP BY result.

```
SELECT Emp_ID, COUNT(Sale)
FROM employee
GROUP BY Emp_ID
HAVING COUNT(Sale) > 1;
```

Emp_ID	COUNT(Sale)
10002	3
10003	2

WHERE vs HAVING

```
SELECT
FROM
WHERE     -- filters on entire table
GROUP BY
HAVING      -- filters on GROUP BY
results
ORDER BY
LIMIT
```

-- is a comment

Operators and HAVING Summary

Operators

- Arithmetic
- Comparison
- Logical
- NULL

HAVING

- Must follow a GROUP BY clause
- Allows for filtering on aggregations

Reference Slides

METIS®

Names database