



**PLURALSIGHT**

# M&T Bank Data Academy

Week 5



**Tarek Atwan**  
Instructor, Pluralsight

*Proprietary and confidential*

# What is a Database

A database is a structured collection of data that is organized and stored in a computer system. It is designed to efficiently manage, store, and retrieve large amounts of data for various applications and users.

- **Data organization:** Databases use a structured format to organize data into tables, rows, and columns, allowing for efficient storage and retrieval.
- **Data integrity:** Databases enforce data integrity by implementing rules and constraints that ensure the accuracy and consistency of the stored data.
- **Data security:** Databases provide mechanisms for controlling access to the data, ensuring that only authorized users can view, modify, or delete the stored information.
- **Data concurrency:** Databases support multiple users accessing and modifying the data simultaneously, while maintaining data consistency and integrity.
- **Data recovery:** Databases implement backup and recovery mechanisms to protect against data loss and ensure the availability of the stored information.

# What is a Data Warehouse

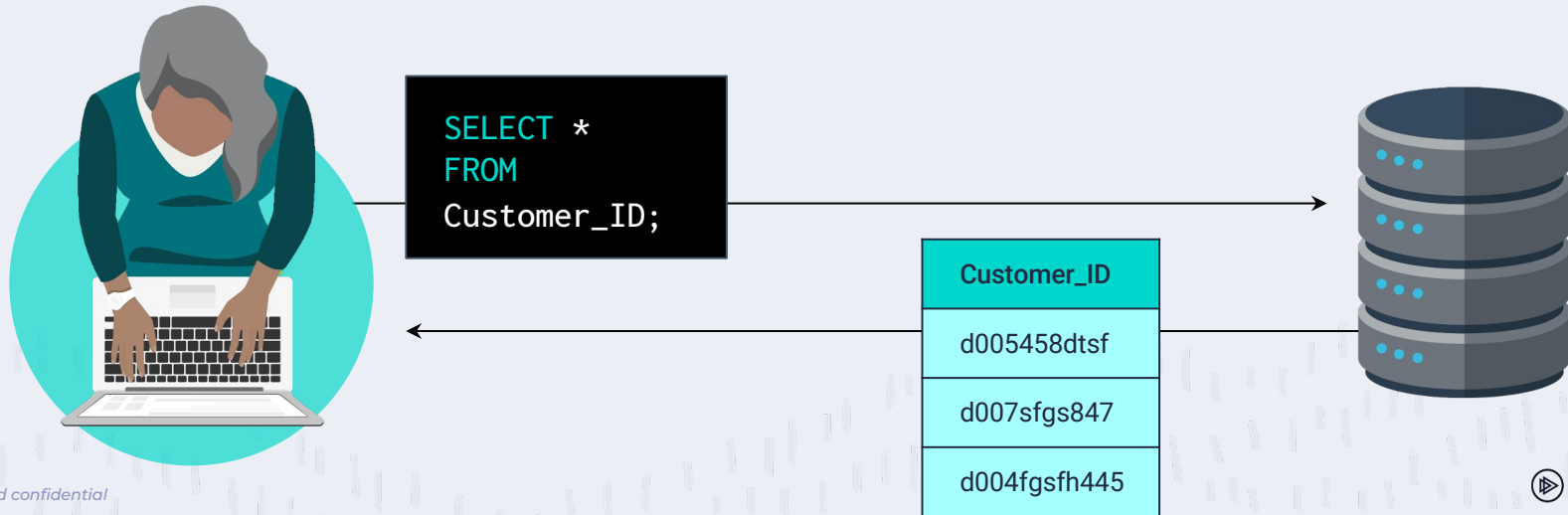
A data warehouse (DW) is a type of database that is specifically designed for analytical rather than transactional work. It collects and aggregates data from various sources, such as operational databases, external data feeds, and more, to provide a consolidated and integrated view of the organization's data.

- **Structured and filtered data:** Data warehouses store structured, filtered, and pre-processed data that is optimized for analytical queries and reporting.
- **Historical data:** Data warehouses often include historical data, allowing users to analyze trends and patterns over time.
- **Separation of operational and analytical workloads:** Data warehouses are separate from operational databases, ensuring that analytical queries do not impact the performance of transactional systems.
- **Support for complex queries and reporting:** Data warehouses are optimized for complex analytical queries and reporting, allowing users to gain insights from large volumes of data.

# What is SQL

SQL (often pronounced "sequel") stands for Structured Query Language.

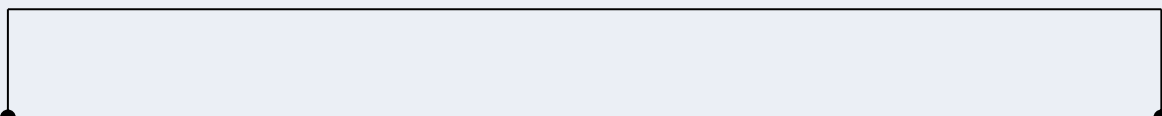
It is a powerful tool that enables programmers to create, populate, manipulate, and access databases. It also provides an easy method for dealing with server-side storage.



# What is SQL

Data using SQL is stored in tables on the server, much like spreadsheets you would create in Microsoft Excel.

This makes the data easy to visualize and search.



Customer_ID	Date_ID
d005458dtsf	6/26/2019
d007sfgs847	8/3/2018
d004fgsfh445	12/3/2018

Order_ID	Customer_ID	Date_ID
10001	d005458dtsf	6/26/2019
10002	d007sfgs847	8/3/2018
10003	d004fgsfh445	12/3/2018

# Using CRUD

**Create Read Update Delete** is a set of operations used with persistent storage.

Create	Create data in a table with the <b>INSERT</b> statement.
Read	Read data by using <b>SELECT</b> .
Update	Updated a table's data by using <b>UPDATE</b> .
Delete	Deleted data via <b>DELETE</b> .

# ANATOMY OF SQL QUERY (READ)

- SELECT** Used to specify the **columns** or expression you want to retrieve from the database
- FROM** Used to specify the **table** from which you want to retrieve the data
- WHERE** An optional clause used to filter the rows (data) returned
- ORDER BY** An optional clause used to sort the rows returned by the query based on one or more columns.



## Query Example

---

The **SELECT** clause can specify more than one column.

```
SELECT pet_type, pet_name  
FROM people  
WHERE pet_type = 'dog'  
AND pet_age < 5;
```

## Query Example

---

Data is filtered by using additional clauses such as **WHERE** and **AND**.

```
SELECT pet_type, pet_name  
FROM people  
WHERE pet_type = 'dog'  
AND pet_age < 5;
```

## Query Example

---

The **WHERE** clause will extract only the data that meets the condition specified. **AND** adds a second condition to the original clause, further refining the query.

```
SELECT pet_type, pet_name  
FROM people  
WHERE pet_type = 'dog'  
AND pet_age < 5;
```

## Query Example

---

Note that unlike in Python where comparisons are done with a double equals (`==`) sign, in SQL only a single equal sign is used.

```
SELECT pet_type, pet_name  
FROM people  
WHERE pet_type = 'dog'  
AND pet_age < 5;
```

## Wildcard: % and \_

---

Use wildcards to substitute zero, one, or multiple characters in a string. The keyword **LIKE** indicates the use of a wildcard.

```
SELECT *  
FROM actor  
WHERE last_name LIKE 'Will%';
```

## Wildcard: % and \_

---

The **%** will substitute zero, one, or multiple characters in a query.  
In this example, all of the following are matches: Will, Willa, and Willows.

```
SELECT *  
FROM actor  
WHERE last_name LIKE 'Will%';
```

## Wildcard: % and \_

---

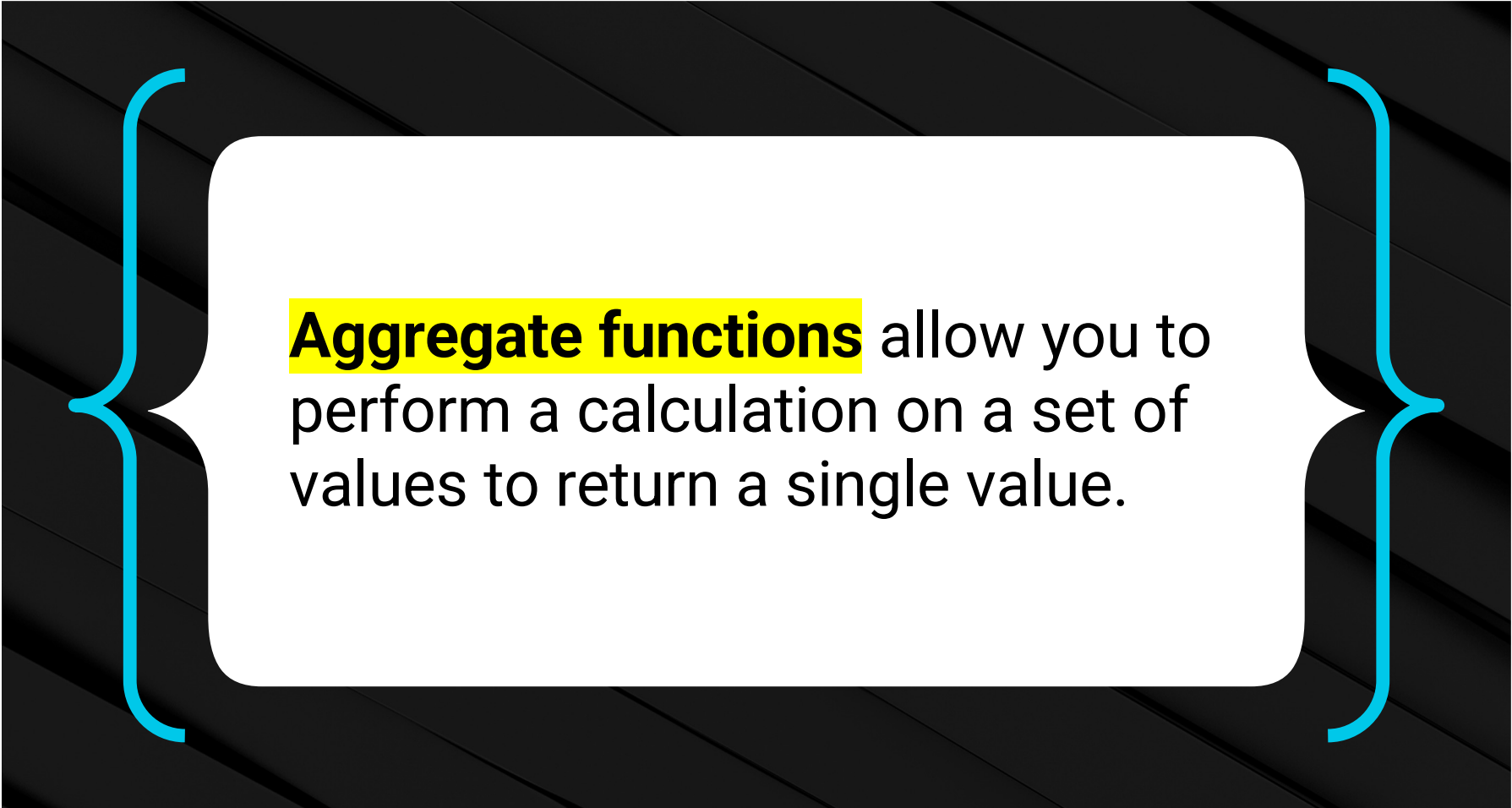
The `_` will substitute only **one** character in a query.

`_an` returns all actors whose first name contains three letters, the second and third of which are `an`.

```
SELECT *  
FROM actor  
WHERE first_name LIKE '_an';
```

# Gregarious Aggregates





**Aggregate functions** allow you to perform a calculation on a set of values to return a single value.

# Aggregate Functions

---

The most commonly used aggregate functions are:

AVG	Calculates the average of a set of values
COUNT	Counts the rows in a specific table or view
MIN	Returns the minimum value in a set of values
MAX	Returns the maximum value in a set of values
SUM	Calculates the sum of a set of values

# Aggregate Functions

---

Aggregate functions are often used with:

01

The **GROUP BY** clause

02

The **HAVING** clause

03

The **SELECT** statement

# Order By Aggregates

---

The **ORDER BY** function:



Is added towards the end of a query.



Returns in an ascending order by default.



Can also return in a descending order by adding **DESC**.



Can limit the return by adding **LIMIT**.



**NOTE:** Use the **ROUND** function to round up the number after the the decimal.



## Pro Tip:

---

One important point about AGGREGATE FUNCTIONS:  
**The returned data is in random order!**

# Introduction to Subqueries

# Subqueries

---

A subquery is nested inside a larger query. Subqueries occur in:

01

The **SELECT** statement

02

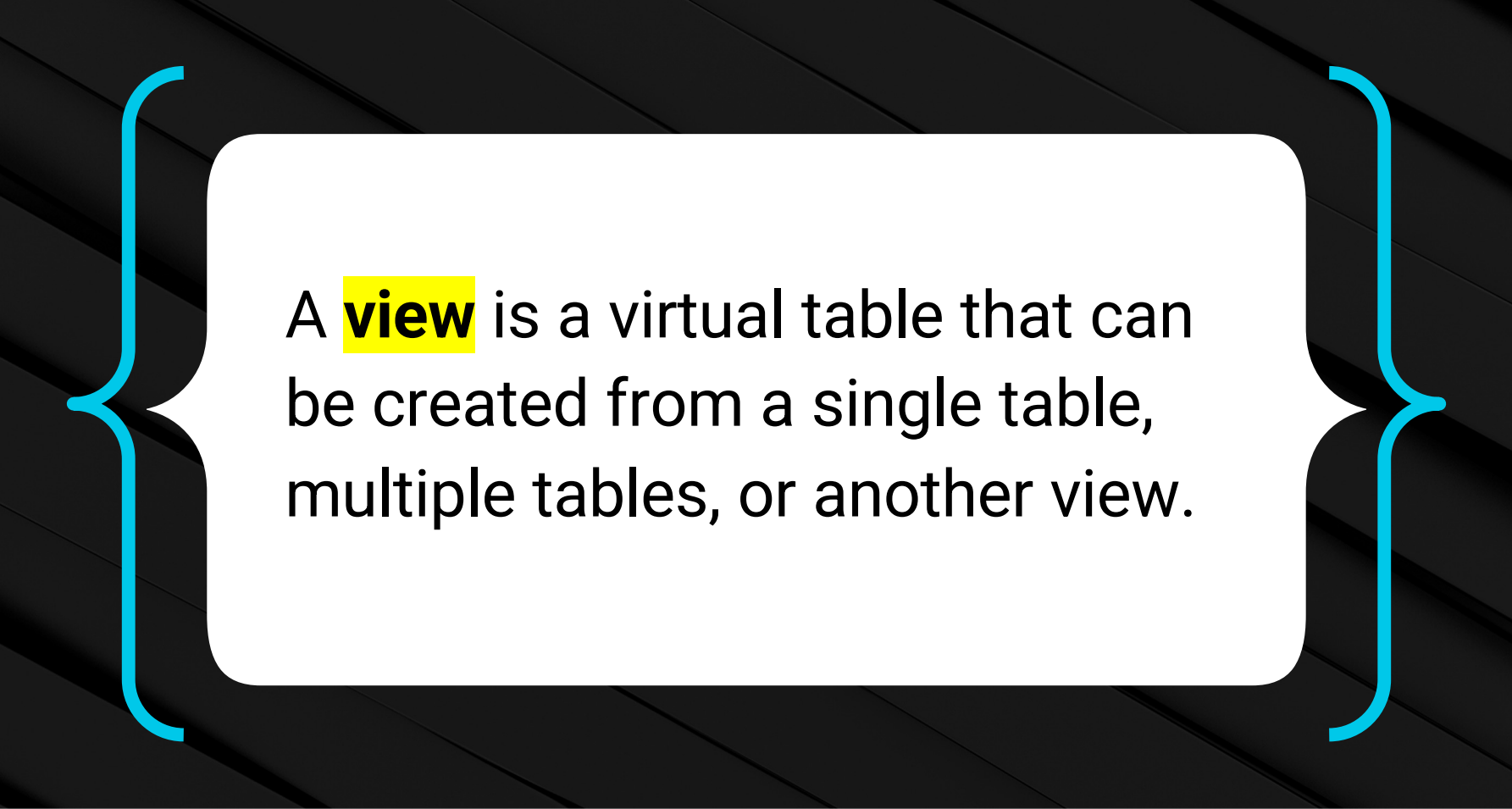
The **FROM** clause

03

The **WHERE** clause

# SQL Views





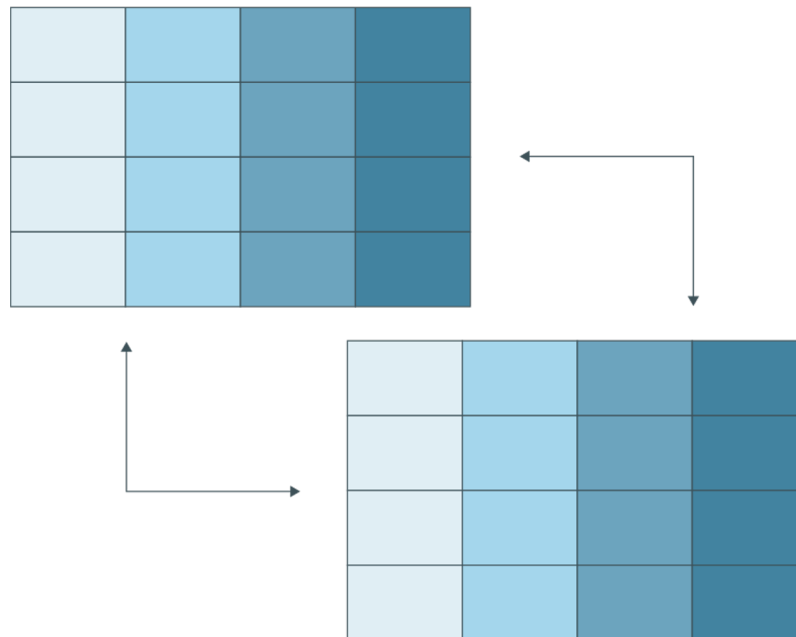
A **view** is a virtual table that can be created from a single table, multiple tables, or another view.

# SQL Views

---

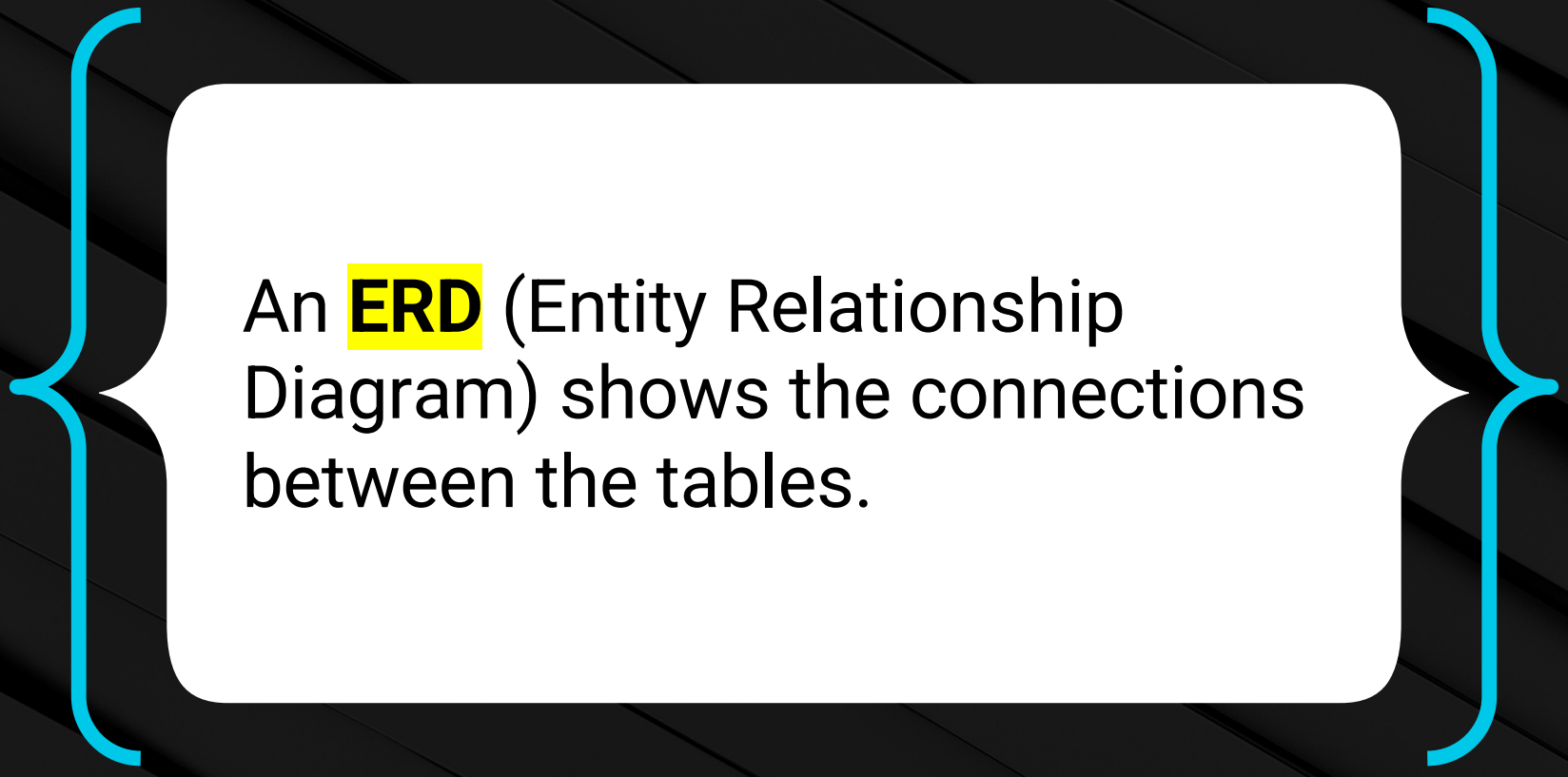
Views are created by using the `CREATE VIEW` statement.

Views are created from a single table, multiple tables, or another view.





How many people have rented  
the film Blanket Beverly?



An **ERD** (Entity Relationship Diagram) shows the connections between the tables.

# Entity Relationship Diagram

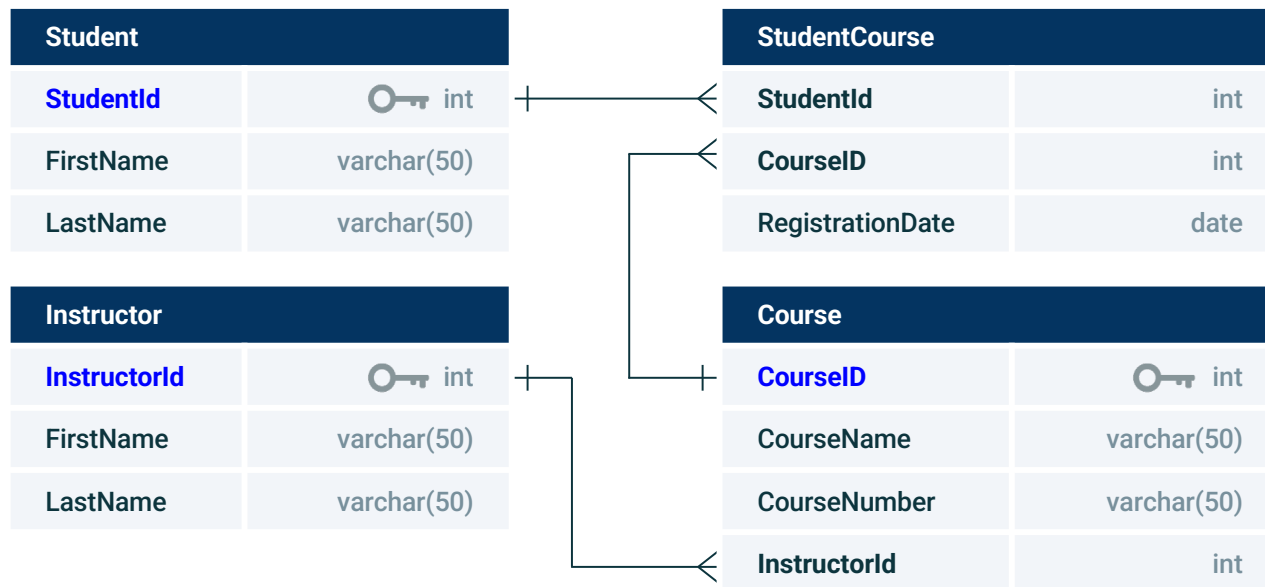
It's called an Entity Relationship Diagram because it shows:



The entities in your data model



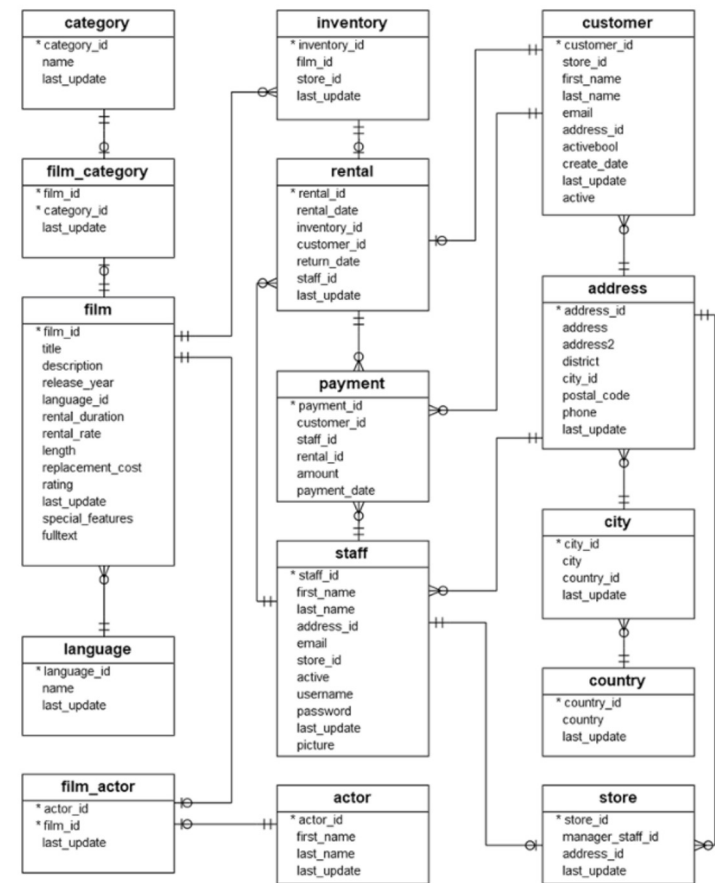
The relationship between entities



# Entity Relationship Diagram

The schema makes it easier to identify the tables we need as well as the keys we will use to link our subqueries.

DVD Rental ER Model



# Foreign Keys

# Foreign Keys

Foreign Keys reference the primary key of another table.

Can have a different name. It does not have to be unique.

Primary Key

	A	B
1	family_id	family
2	1	Smiths
3	2	Jones

Primary Key

Foreign Key

	A	B	C
1	child_id	family_id	children
2	11	1	Chris
3	22	1	Abby
4	33	1	Suzy



# Data Relationships

Relationships Link Tables/Entities.

Types of relationships:

01

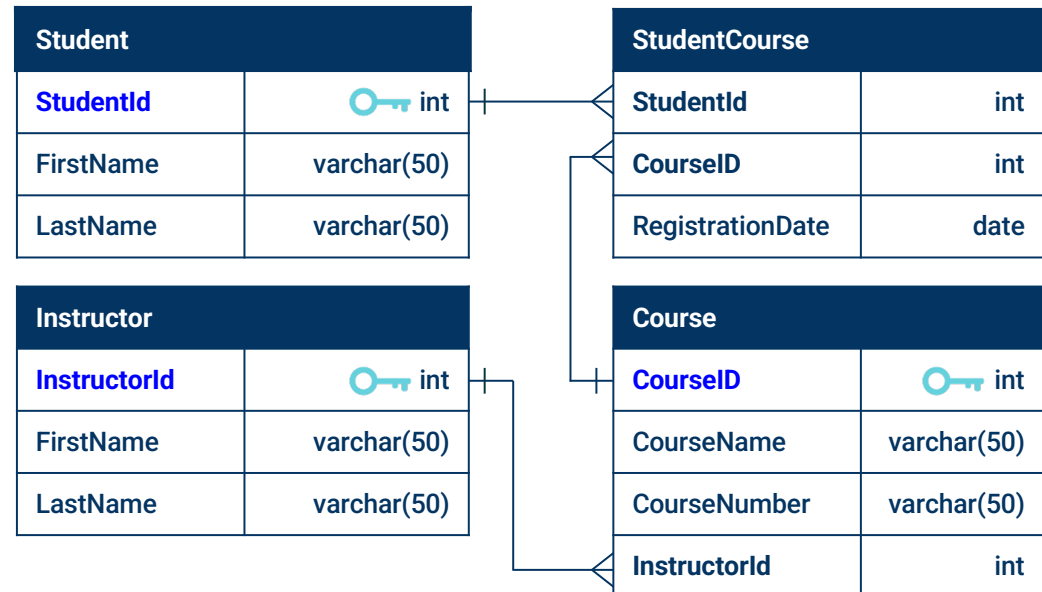
One-to-One

02

One-to-Many

03

Many-to-Many



## One-to-One Relationship

---

Each item in one column is linked to only one other item from the other column.

ID	Name	Social Security
1	Homer	111111111
2	Marge	222222222
3	Lisa	333333333
4	Bart	444444444
5	Maggie	555555555

Here, each person in the Simpsons family can have only one social security number.

Each social security number can be assigned only to one person.

## One-to-Many Relationship

---

This example has two tables. The first table lists only addresses.

The second table lists each person's Social Security number and address. As before, one Social Security number is unique to one individual.

ID	Address	ID	Name	Social Security	AddressID
11	742 Evergreen Terrace	1	Homer	111111111	11
12	221B Baker Street	2	Marge	222222222	11
		3	Lisa	333333333	11
		4	Bart	444444444	11
		5	Maggie	555555555	11
		6	Sherlock	112233445	12
		7	Watson	223344556	12

## One-to-Many Relationship

---

- Each address can be associated with multiple people.
- Each person has an address.
- The two tables, joined, would look like this.

ID	Address	ID	Name	Social Security	AddressID
11	742 Evergreen Terrace	1	Homer	111111111	11
12	221B Baker Street	2	Marge	222222222	11
		3	Lisa	333333333	11
		4	Bart	444444444	11
		5	Maggie	555555555	11
		6	Sherlock	112233445	12
		7	Watson	223344556	12

## Many-to-Many Relationship

---

- Each child can have more than one parent.
- Each parent can have more than one child.

ID	Child	ID	Parent
1	Bart	11	Homer
2	Lisa	12	Marge
3	Maggie		

## Many-to-Many Relationship

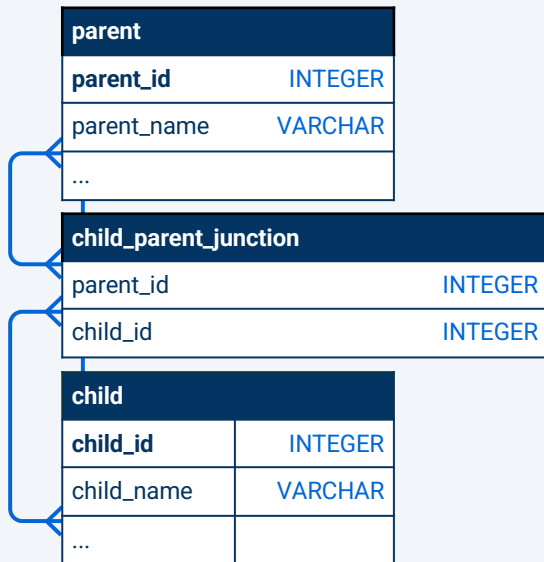
---

- Each child can have more than one parent.
- Each parent can have more than one child.
- The two tables are joined in a **junction table**.

ChildID	Child	ParentID	Parent
1	Bart	11	Homer
1	Bart	12	Marge
2	Lisa	11	Homer
2	Lisa	12	Marge
3	Maggie	11	Homer
3	Maggie	12	Marge

# Junction Table

The junction table contains many parent\_id's and many child\_id's.



	parent_id integer	child_id integer
1	11	1
2	11	2
3	11	3
4	12	1
5	12	2
6	12	3

Join child and parent  
table to junction table ↓

	parent_name character varying (255)	child_name character varying (255)
1	Homer	Bart
2	Homer	Lisa
3	Homer	Maggie
4	Marge	Bart
5	Marge	Lisa
6	Marge	Maggie

# SQL Joins

INNER JOIN	Returns records that have matching values in both tables.
LEFT JOIN	Returns all records from the left table and the matched records from the right table.
RIGHT JOIN	Returns all records from the right table and the matched records from the left table.
CROSS JOIN	Returns records that match every row of the left table with every row of the right table. This type of join has the potential to make very large tables.
FULL OUTER JOIN	Places null values within the columns that do not match between the two tables, after an inner join is performed.