TRY
SQL

Who is this course designed for?

This course will teach the basics of how to read, write, and program in the database language known as SQL.

Try SQL

Section 1: Understanding Databases

The Story

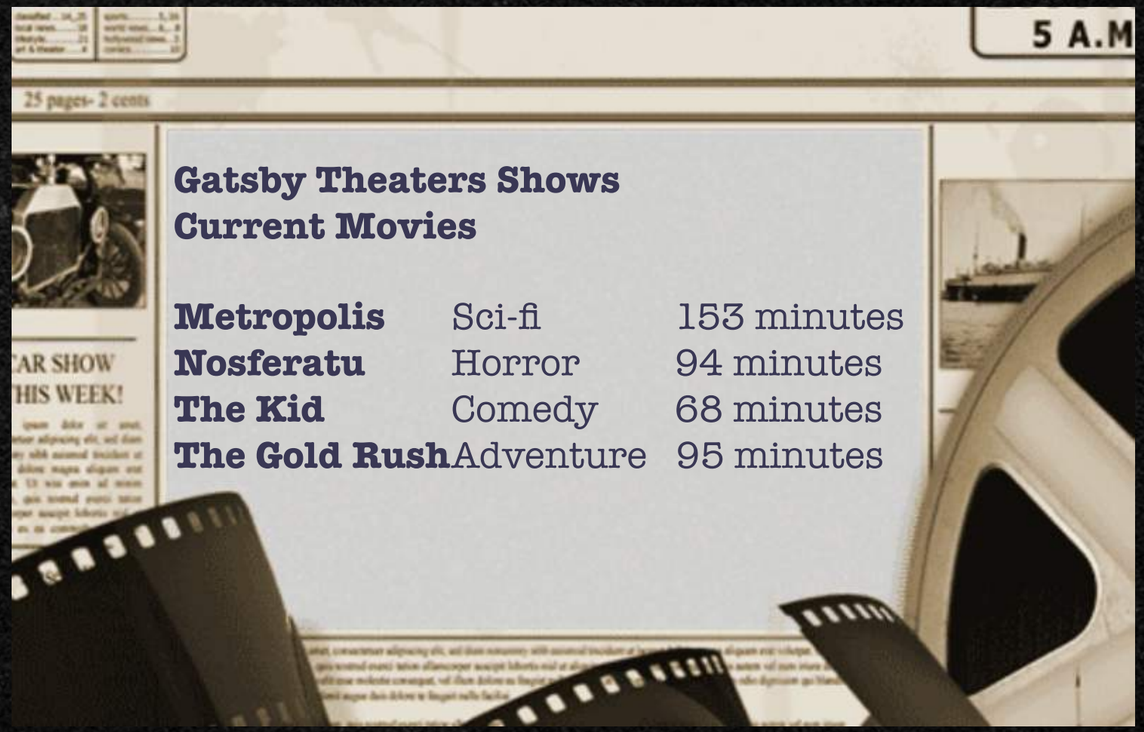
Where to begin?

We want to see a movie at Gatsby Theaters, but we're only interested in seeing a comedy.

How can we retrieve a listing of all movies shown at Gatsby Theaters?

Consulting a newspaper

Finding a list of movies

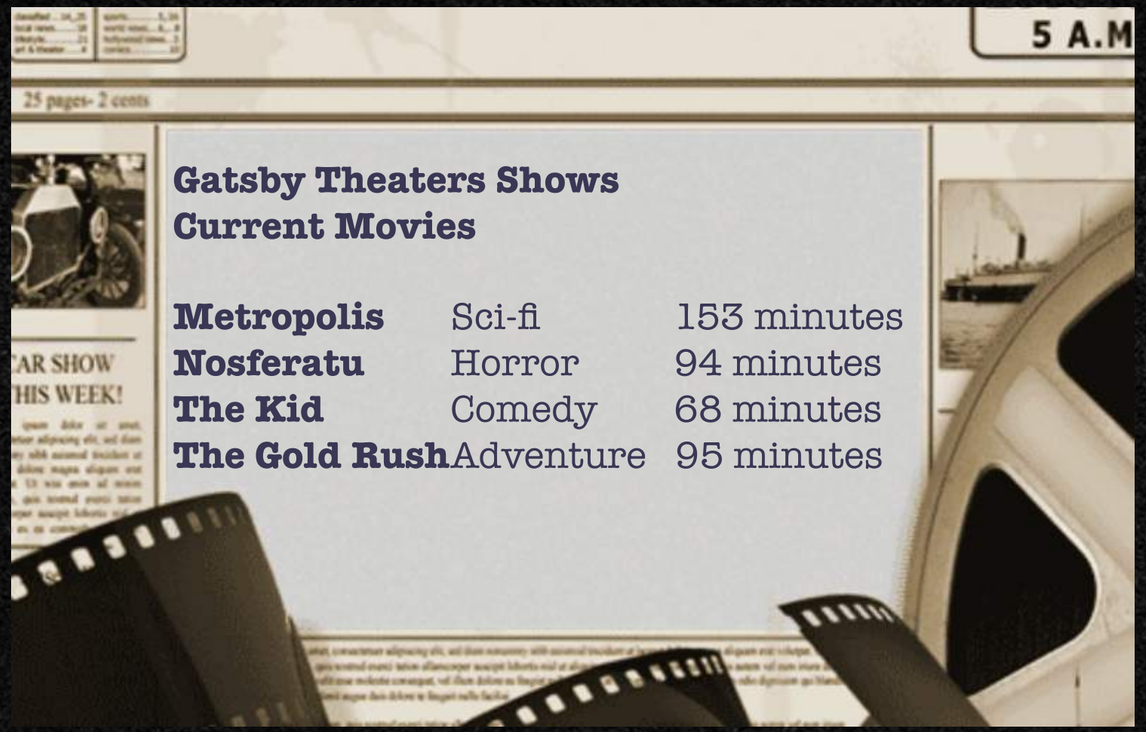


Reviewing a website

Finding a list of showtimes

Movie listings from the web

Movie listings from a newspaper



Why is the website updating?

It's updating because it's pulling this information from a database.

Requesting the latest showtimes from a website

Databases collect and organize data to allow for easy retrieval.

Databases can be used by lots of devices

Information from a database can be used by many different services.

What does this database look like?

Here is the Gatsby Theaters database.

database name

Gatsby Theaters

movie info

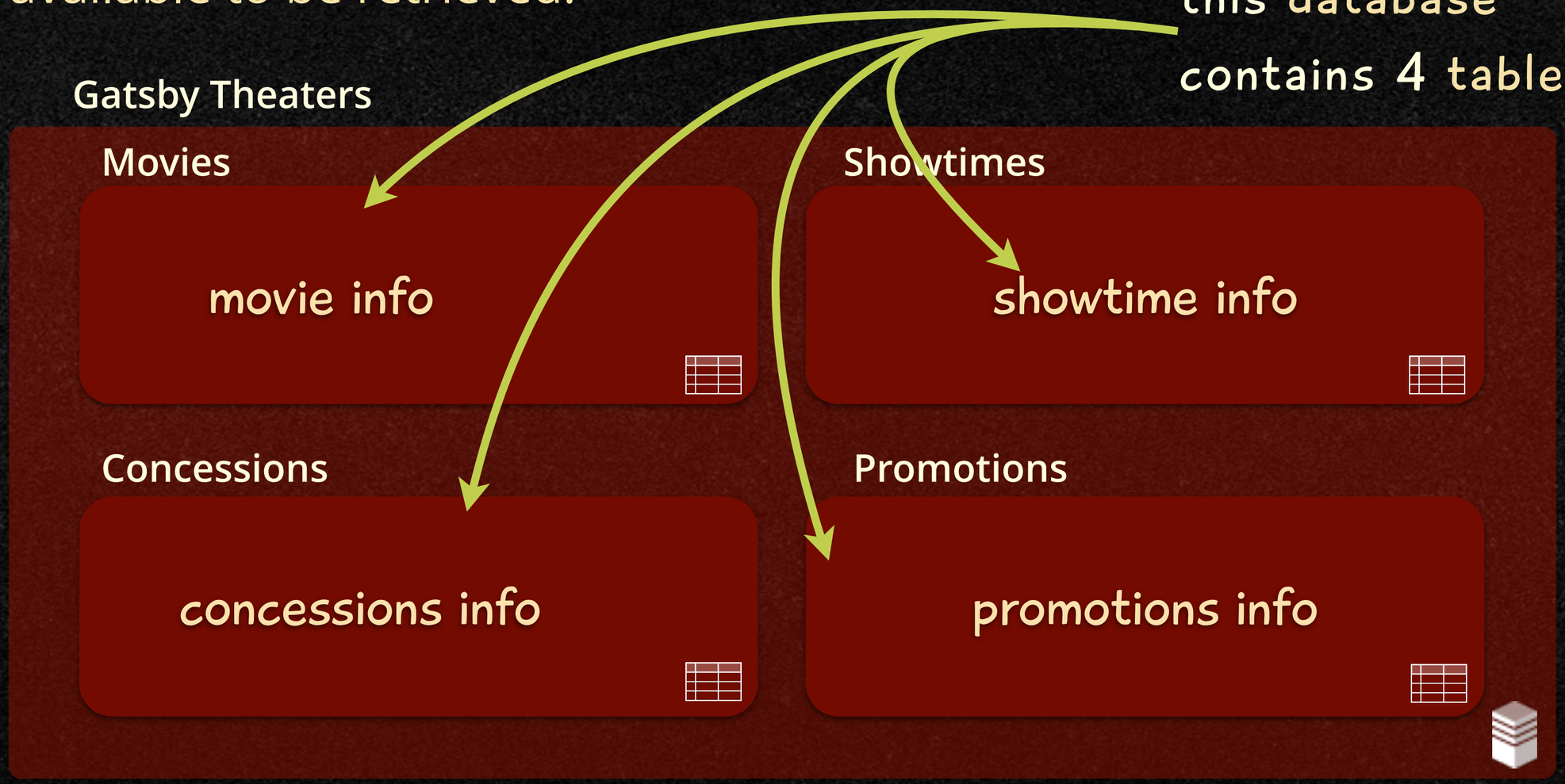
showtime info

concessions info

promotions info

Introducing database tables

The **data** is organized and stored into tables, which hold the data so it is available to be retrieved.



Retrieving movie information

So, in order to look up information on movies, we would need to go into the _____ **database** and look inside the _____ **table**.

Gatsby Theaters

Movies

movie info

Showtimes

showtime info

Concessions

concessions info

Promotions

promotions info

Retrieving movie information

So, in order to look up information on movies, we would need to go into the Gatsby Theaters **database** and look inside the Movies **table**.

What is inside of a table?

Inside the Movies table is information about the movies in a cell, or a field of data.

Movies

1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

Tables are also organized into rows

A row represents an entire data record within each table.

Rows

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

Tables have columns with descriptive names

Each column has a name describing the kind of data found in that column.

Columns → **id** **title** **genre** **duration**

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

Often tables will have a unique identifier

Unique identifier

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

Sometimes the unique identifier is labeled as the primary key.

First step: Moving through the columns

The columns are the first place to look inside of the table.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

The film title being searched for is "Nosferatu."

First step: Moving through the columns

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

First step: Moving through the columns

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

The `title` column contains the matching record.

But wait, the next step...

Second step: Moving through the rows

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

Reaching the final destination for the data

Most databases have fewer columns than rows, making it easier to search columns first.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

The destination of "Nosferatu" has now been reached!

The data search path

If the entire table search path is visualized, it would look something like this:

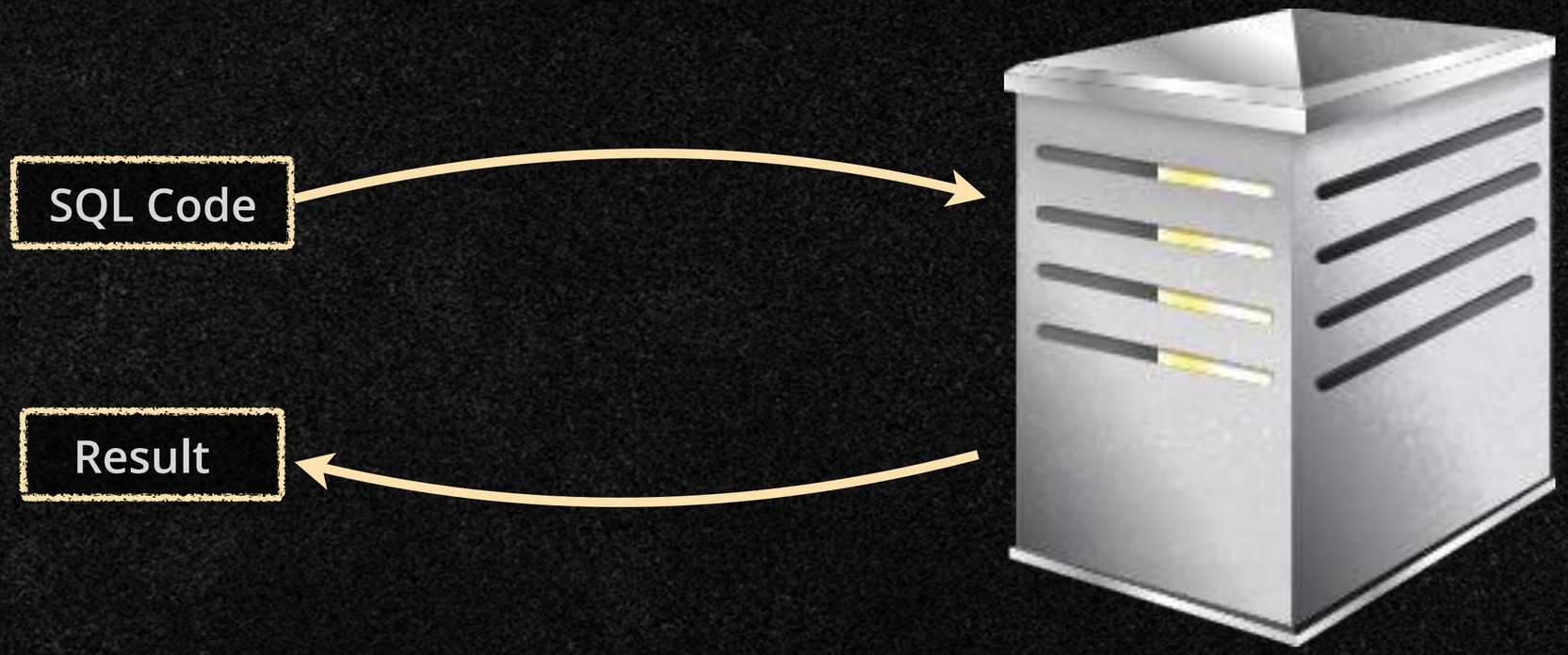
id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

Try SQL

Section 2: Introducing SQL

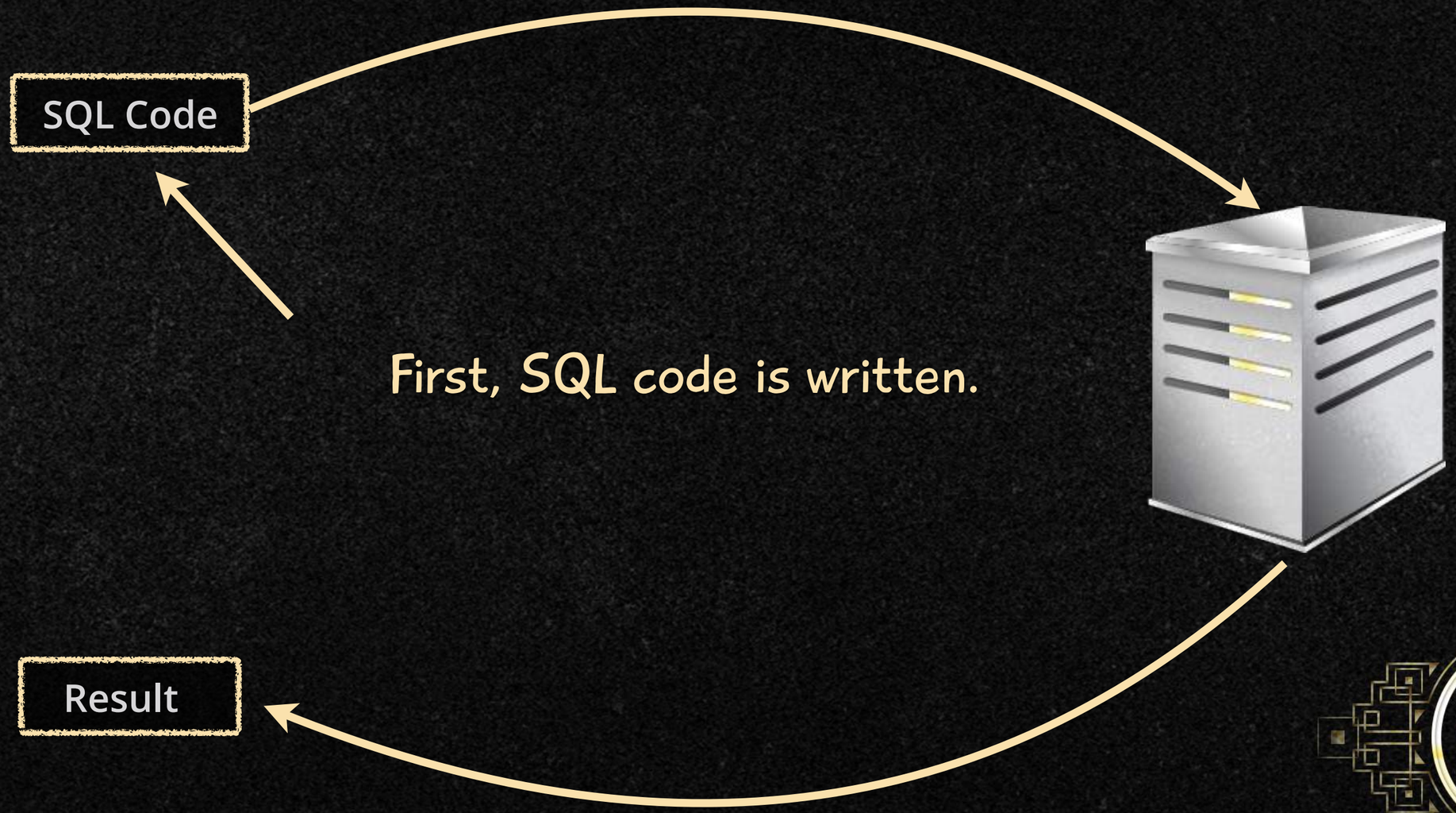
How do we speak to the database?

SQL is a programming language that allows interaction with **databases**.



Using SQL to work with the database

SQL is written as statements that ask the database to perform many things.



Using SQL to work with the database

SQL is written as statements that ask the database to perform many things.

SQL Code

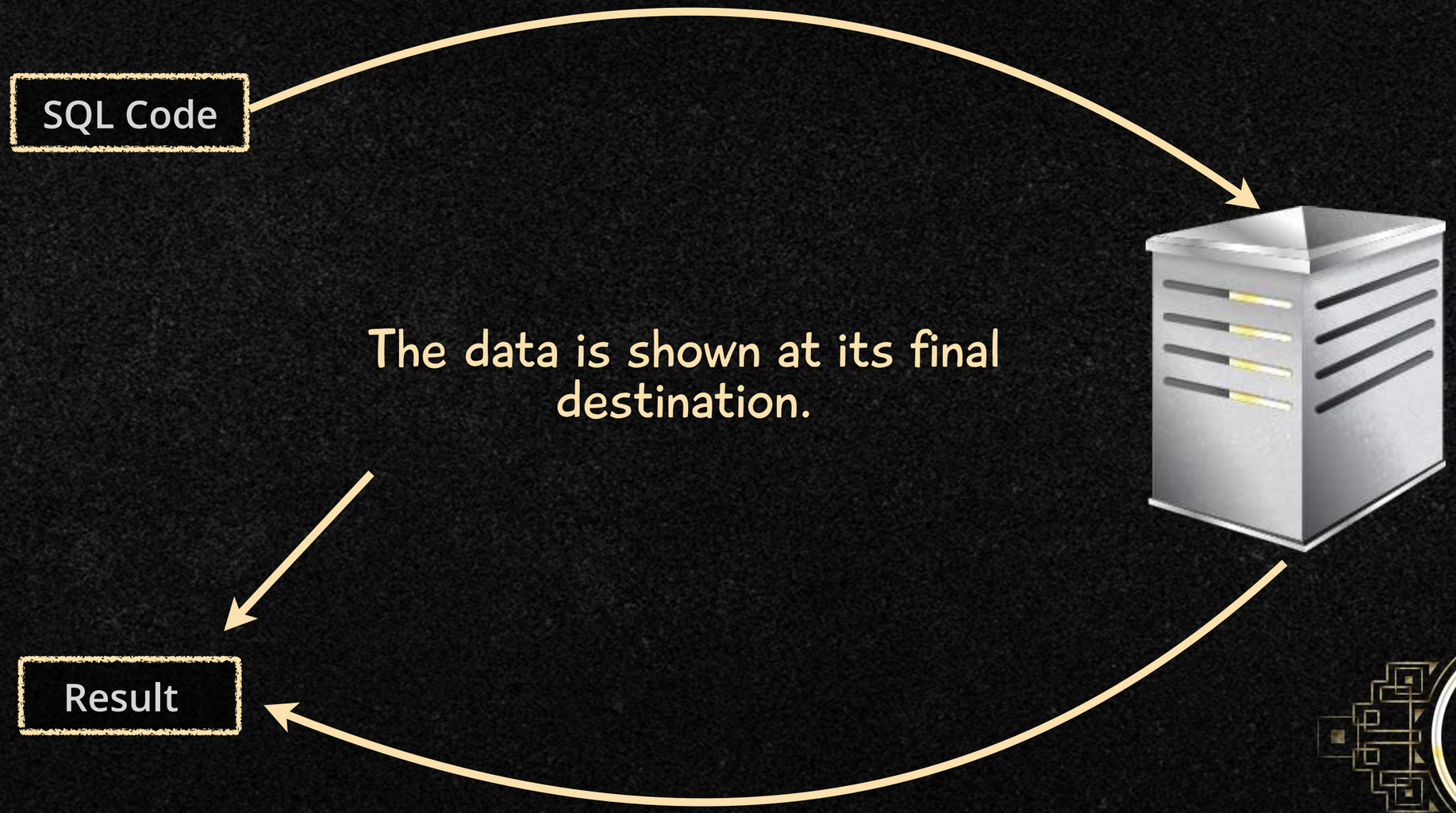
The database receives the SQL code and runs it.

Result

TRY SQL

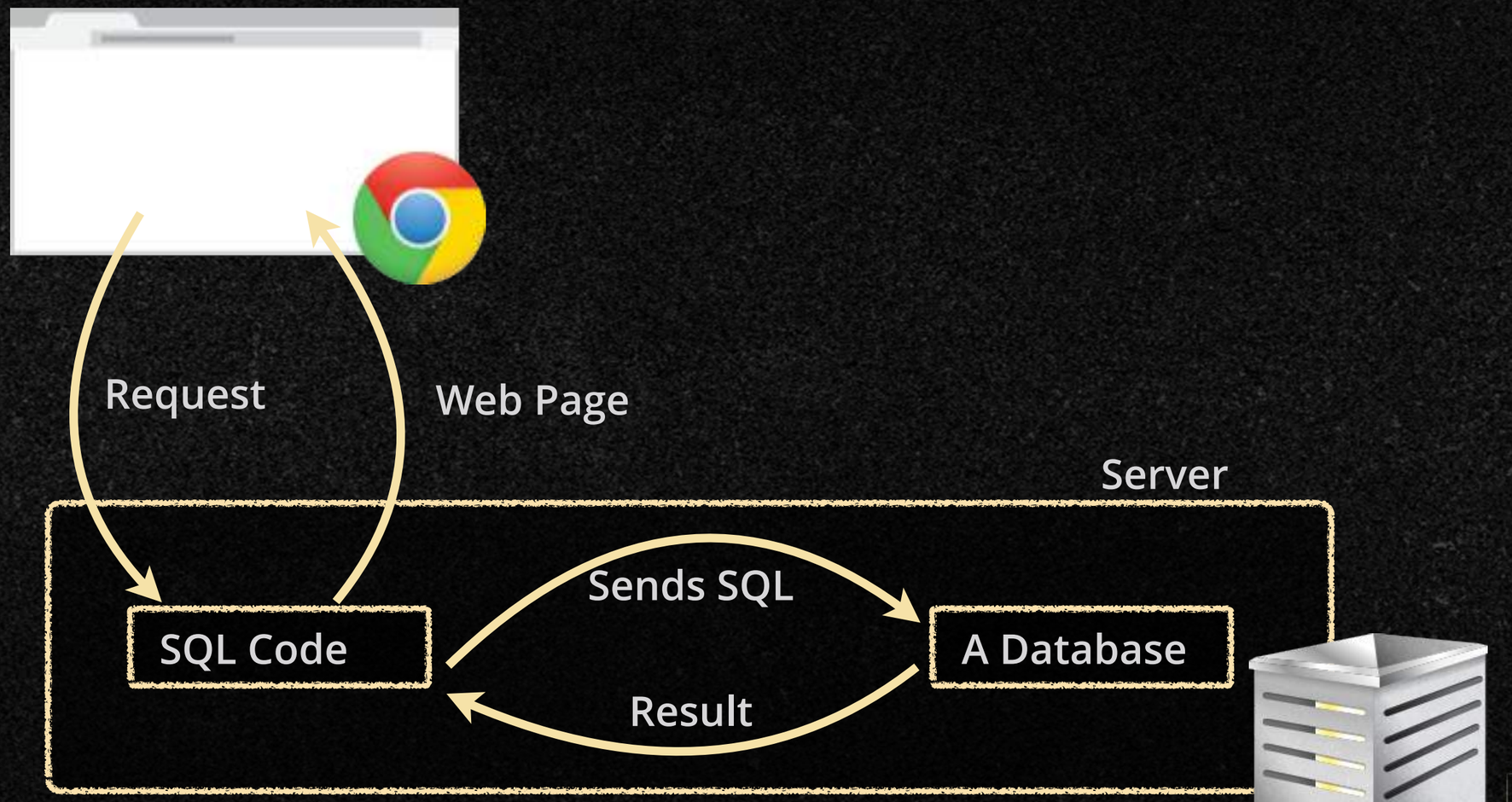
Using SQL to work with the database

SQL is written as statements that ask the database to perform many things.



Back to the original example

What would the browser do to get showtimes?



Now, let's see how to use SQL

Here are some things SQL will help solve.

So now, look back at the first issue...

Retrieve a list of just the titles of all movies shown at the Gatsby Theaters.

But as always, one answer can lead to more questions...

- Retrieve a list of all movies with id, title, genre, and duration of the movie.
- Retrieve just the title of the movie with id of 2.
- Retrieve all movies with the title of "The Kid."

Retrieving data from our database

A SELECT statement is needed to get movie information.

SELECT Recipe

```
SELECT ________
FROM __________;
```

The column or columns to be searched

All SQL statements must end in a semicolon

The table the data is in

Using our SELECT recipe

Retrieving a list of all movie titles using the SELECT statement

Table

Movies

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies;
```

By choosing to select a single column, the result will contain only this column's data from all the rows.

Using our SELECT recipe

Retrieving a list of all movie titles using the SELECT statement

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies;
```

The database will return these values.

Metropolis
Nosferatu
The Kid
The Gold Rush

Even more SQL queries

Retrieving a list of all movies showing the id, title, genre, and duration

```
SELECT title, genre
FROM movies;
```

Returns both title and genre

```
SELECT id, title, genre, duration
FROM movies;
```

Returns id, title, genre and duration

Returns same result

```
SELECT *
FROM movies;
```

Returns all columns

The asterisk means to retrieve ALL matching results.

Filtering table data

The WHERE clause can be used to filter out data and match the requested data.

WHERE Recipe

```
SELECT ______
FROM ________
WHERE _______;
```

The match case for the WHERE clause

Filters out other information and will return only the match that is requested

Adding WHERE clause to return smaller set of data

Retrieving the movie title with id of 2.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 2;
```

Filtering the Movies by id of 2.

Adding WHERE clause to return smaller set of data

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 2;
```

Before retrieving any data, all columns with a name of `title` are selected.

Adding WHERE clause to return smaller set of data

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 2;
```

Is this a correct row?

Adding WHERE clause to return smaller set of data

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 2;
```

This is a correct row.

Adding WHERE clause to return smaller set of data

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 2;
```

The last two rows are also searched for matching conditions, but there are none.

Adding WHERE clause to return smaller set of data

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 2;
```

Since the column `title` is the only column that is queried, that's the only data returned.

Nosferatu

Using the WHERE clause to return a string of data

Retrieving all movies with the title of “The Kid.”

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT *
FROM movies
WHERE title = 'The Kid';
```

Filtering all movies by `title` of “The Kid.”

Using the WHERE clause to return a string of data

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT *
FROM movies
WHERE title = 'The Kid';
```

Since data for all movies is searched, all records are at first considered for the result.

Using the WHERE clause to return a string of data

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT *
FROM movies
WHERE title = 'The Kid';
```

However, if focus is placed on the column `title` as the first filter...

Using the WHERE clause to return a string of data

Single quotation marks are used around a string value.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT *
FROM movies
WHERE title = 'The Kid';
```

These are used for selecting whole string character values.

Try SQL

Section 3: Guiding Data Criteria

What if we want to retrieve ordered data?

Retrieve the title of the movies sorted by duration.

How can data be retrieved in a specially ordered way?

We want to retrieve all titles of the movies, sorted by duration, and return these values for all of the films in the table.

Sorting data

The ORDER BY clause can be used to sort data in a specific way.

ORDER BY Recipe

```
SELECT ______
FROM ________
ORDER BY _______;
```

The column to be ordered

Sorts data by columns based on how it is to be viewed

Retrieving data using the ORDER BY clause

Retrieve the title of the movies sorted by duration.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
ORDER BY duration;
```

The `title` column is where we are pulling data from. Results can be displayed in ascending or descending order.

TRY SQL

Retrieving data using the ORDER BY clause

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
ORDER BY duration;
```

Also, as a reminder, sorting the `title` column focuses on which `duration` is lowest.

Sorting using ORDER BY in ascending order

The result set defaults to displaying data in ASC order, which is known as ascending.

id	title	genre	duration
3	The Kid	Comedy	68
2	Nosferatu	Horror	94
4	The Gold Rush	Adventure	95
1	Metropolis	Sci-Fi	153

```
SELECT title
FROM movies
ORDER BY duration;
```

The duration is growing as more results are added to the query.

Sorting using ORDER BY in descending order

The result set displays data in DESC order, which is known as descending.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
4	The Gold Rush	Adventure	95
2	Nosferatu	Horror	94
3	The Kid	Comedy	68

```
SELECT title
FROM movies
ORDER BY duration DESC;
```

This is how descending (DESC) order can be returned from the result set, shown as the duration gets lower.

Filtering data using comparison operators

Find films that have a duration of more than 100 minutes.

How do we use current data to compare against itself to find the desired results?

We want to retrieve all films that have a duration of more than 100 minutes. How can we use SQL to find the results?

Comparing data using comparison operators

Comparison operators can also be used to specify data results.

Comparison Operator Recipe

```
SELECT _____
FROM _______
WHERE ______ > ______;
```

Used to signify that all items to the left are greater than what is to the right

Multiple comparison fields are used to compare records in the query.

Using comparison operators

Find films that have a duration equal to 100 minutes.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT *
FROM movies
WHERE duration = 100;
```

The query retrieves no records that provide a true comparison.

Using comparison operators

Using the greater-than symbol.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT *
FROM movies
WHERE duration > 100;
```

Since the duration time for "Metropolis" is compared to all movie durations, only one record is retrieved.

More comparison operators in SQL

Here are more comparison operators that can be used to filter multiple conditions.

```
SELECT *
FROM movies
WHERE duration < 100;
```

This query will retrieve all records with a duration of less than 100 minutes.

```
SELECT *
FROM movies
WHERE duration >= 94;
```

This query will retrieve all records where the duration is greater than or equal to 94 minutes.

```
SELECT *
FROM movies
WHERE duration <= 95;
```

This query will retrieve all records where the duration is less than or equal to 95 minutes.

Using the NOT EQUAL comparison operator

Find films that do not have a genre of Horror.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT *
FROM movies
WHERE genre <> 'Horror';
```

The Not Equal To sign (`<>` or `!=`) is used to compare all records that do not have a genre of `Horror`, as shown.

Matching records with multiple conditions

Retrieve records that have an id of 1 and genre of Comedy.

How do we retrieve the data with specific matching conditions?

We want to retrieve all records that have both an id of 1 and have the genre of Comedy. How would we do this with SQL?

Showing matching criteria using AND operator

Using the AND operator to filter multiple conditions.

AND Recipe

```
SELECT ______
FROM ________
WHERE _______
AND _______;
```

Multiple matching criteria to display matching records in the query

Uses multiple criteria to display matching records in a table

Using the AND operator to select precise data

Retrieve records that have an id of 1 and genre of Comedy.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 1
AND genre = 'Comedy';
```

All conditions of the SELECT statement must be met. If not, the search, or query, will return no records, as shown.

Using the AND operator to select precise data

id	title	genre	duration
1	Metropolis	Comedy	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 1
AND genre = 'Comedy';
```

However, if the genre of "Metropolis" were changed to Comedy, a better display of the AND operator concepts is shown.

Using the AND operator to select precise data

id	title	genre	duration
1	Metropolis	Comedy	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 1
AND genre = 'Comedy';
```

The WHERE and AND conditions must both be true.

Using the AND operator to select precise data

id	title	genre	duration
1	Metropolis	Comedy	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 1
AND genre = 'Comedy';
```

Even though two records both contain Comedy, only one has an id of 1.

Matching records with different conditions

Retrieve records that have an id of 1 or genre of Comedy.

How do we retrieve the data with specific matching conditions?

We want to retrieve all records that have either an id of 1 or have the genre of Comedy. What conditions must be met to complete this with SQL?

Showing matching criteria using OR operator

The OR operator can be used to filter multiple conditions.

OR Recipe

```
SELECT ______
FROM ________
WHERE _______
OR _______;
```

Multiple matching criteria to display matching records in the query

The OR operator is included to help cover more range with our SELECT statement.

Using the OR criteria condition

Retrieve records that have an id of 1 or genre of Comedy.

id	title	genre	duration
1	Metropolis	Comedy	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 1
OR genre = 'Comedy';
```

Either the WHERE condition or the OR condition can be true.

Using the OR criteria condition

id	title	genre	duration
1	Metropolis	Comedy	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
SELECT title
FROM movies
WHERE id = 1
OR genre = 'Comedy';
```

Both records hold true since the result set can contain at least an id of 1 or a genre of Comedy.

Try SQL

Section 1: Adding Data

Handling data using SQL

Using SQL to direct the data to our database

Okay, now we can access the data. But how do we add data to the current table?

We want to add a new movie called “The Circus” to the Movies table.
It’s a Comedy with a duration of 71 minutes.

Adding data to the table

To add movie information, we need to use an INSERT statement.

The table the data is in

INSERT Recipe

```
INSERT INTO ______ (________)
VALUES (______________);
```

The column or columns we want to add to

The actual values or data to add, separated by commas

Adding data using SQL

Using the INSERT INTO statement to add data to the Movies table

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

The task is to add a new movie called "The Circus" to the table `movies`. It is a `Comedy` with a `duration` of 71 minutes.

Adding data using SQL

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
INSERT INTO movies (id, title, genre, duration)
VALUES (5, 'The Circus', 'Comedy', 71);
```

The INSERT INTO statement signals that a new row will be added.

Adding data using SQL

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
INSERT INTO movies (id, title, genre, duration)
VALUES (5, 'The Circus', 'Comedy', 71);
```

Here, the `movies` table specifies into which table the new data will be added.

Specifying which columns to insert data into

Column names are specified to tell our statement where to insert the values.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
INSERT INTO movies (id, title, genre, duration)
VALUES (5, 'The Circus', 'Comedy', 71);
```

The VALUES keyword inside the INSERT statement

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
INSERT INTO movies (id, title, genre, duration)
VALUES (5, 'The Circus', 'Comedy', 71);
```

The VALUES keyword signifies where to begin placing the values to be entered.

The actual data to be added to the table

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
INSERT INTO movies (id, title, genre, duration)
VALUES (5, 'The Circus', 'Comedy', 71);
```

... and here are the VALUES. Remember the quotes from earlier?

The result of the INSERT statement

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Comedy	71

```
INSERT INTO movies (id, title, genre, duration)
VALUES (5, 'The Circus', 'Comedy', 71);
```

So, let's execute the INSERT INTO statement and see that there are now 5 rows.

A different way to insert data into the table

```
INSERT INTO movies (id, title, genre, duration)
VALUES (5, 'The Circus', 'Comedy', 71);
```

Same

The same goal could have been accomplished without having the columns, only because data is being inserted into all the columns.

```
INSERT INTO movies
VALUES (5, 'The Circus', 'Comedy', 71);
```

This is a quicker way to write code.

We don’t have to insert into every column

```
INSERT INTO movies (id, title)
VALUES (5, 'The Circus');
```

5	The Circus		

```
INSERT INTO movies (title, duration)
VALUES ('The Fly',80);
```

6	The Fly		80

Notice the id was set, even though it wasn’t explicitly done. Why is this?

Understanding primary keys

SQL can have a primary key for a table, which is unique.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Comedy	71
6	The Fly		80

This field will never be blank or empty.

```
INSERT INTO movies (title, duration)
VALUES ('The Fly',80);
```

Understanding auto-increment

SQL can automatically increment the primary key for a table for new rows.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Comedy	71
6	The Fly		80

This is now a new row with an `id` of 6.

```
INSERT INTO movies (title, duration)
VALUES ('The Fly',80);
```

Understanding NULL / empty fields

SQL can also add data where there is no data for a field.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Comedy	71
6	The Fly		80

A NULL value is used to represent a field with missing data.

A NULL value is used as a placeholder for unknown data.

```
INSERT INTO movies (title, duration)
VALUES ('The Fly',80);
```

Try SQL

Section 2: Changing Current Data

Changing the data using SQL

Using SQL to change the data in the database

Great — we can now add new data to the database.

But how do we update existing data?

We want to change the genre for the film “The Circus” from a Comedy to a Romance.

Updating data in the table

To change the movie data, an UPDATE statement is needed.

The table the data is in

UPDATE Recipe

```
UPDATE _______
SET _______ = _______
(WHERE clause)
```

The value needed to change the column to

This sets the column to be changed, followed by the column name

The WHERE condition to help pinpoint where the change will take place

Changing current data using SQL

Using the UPDATE statement to change data in the Movies table

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Comedy	71

So how can the genre of "The Circus" be changed?

Updating existing data

Use the UPDATE statement to change an existing row.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Comedy	71

```sql
UPDATE movies
SET genre = 'Romance'
WHERE id = 5;
```

Updating the Movies table

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Comedy	71

```
UPDATE movies
SET genre = 'Romance'
WHERE id = 5;
```

State the table movies to pinpoint where this UPDATE will be made.

The SET keyword in our UPDATE statement

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Comedy	71

```
UPDATE movies
SET genre = 'Romance'
WHERE id = 5;
```

Let's use the SET keyword to set the column genre to a value of Romance...

Using WHERE to specify the UPDATE statement

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Comedy	71

```
UPDATE movies
SET genre = 'Romance'
WHERE id = 5;
```

... WHERE the id is 5.

The record has been successfully changed

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Romance	71

```
UPDATE movies
SET genre = 'Romance'
WHERE id = 5;
```

The UPDATE statement successfully changed the genre from Comedy to Romance for the movie "The Circus."

Changing multiple fields at once

Making multiple changes to an existing row

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Comedy	70

```
UPDATE movies
SET genre = 'Comedy', duration = 70
WHERE id = 5;
```

Separate changes by adding a comma and add each entry to the end of the line.

Changing multiple rows at once

Making multiple changes to different rows

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Romance	68
4	The Gold Rush	Adventure	95
5	The Circus	Romance	70

```
UPDATE movies
SET genre = 'Romance'
WHERE id = 3 OR id = 5;
```

To accomplish this, simply add an OR clause to the WHERE statement.

Try SQL

Section 3: Removing Data

Removing data using SQL

Using SQL to remove data from the database

We can add or update data in the database.

But how do we remove existing data?

We want to remove the film “The Circus” from our database.

Removing data in a table

Use a DELETE statement to remove movie data

The table that contains the data to be removed

DELETE Recipe

```
DELETE FROM _____ (WHERE clause);
```

The WHERE condition to help pinpoint where the change will take place

Deleting data using SQL

Using the DELETE statement to remove a record from the Movies table

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Romance	71

How can a movie be removed from a table using SQL? We DELETE it, of course!

Removing a record from the Movies table

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Romance	71

```
DELETE FROM movies WHERE id = 5;
```

The DELETE statement removes rows from the table based on which conditions are specified by the WHERE clause.

Deleting data

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Romance	71

```
DELETE FROM movies WHERE id = 5;
```

The FROM keyword specifies the exact table we will perform this command on...

Using WHERE to specify data removal

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95
5	The Circus	Romance	71

```
DELETE FROM movies WHERE id = 5;
```

... and the row is specified with the WHERE clause.

Removing the row from our table

The row has been removed completely from the Movies table.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
DELETE FROM movies WHERE id = 5;
```

What would have happened if the WHERE clause was not included?

Removing the row from our table

The row has been removed completely from the Movies table.

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
DELETE FROM movies                    ;
```

What would have happened if the WHERE clause was not included?

Deleting all data from a table

Completely removing all data from a table

id	title	genre	duration
1	Metropolis	Sci-Fi	153
2	Nosferatu	Horror	94
3	The Kid	Comedy	68
4	The Gold Rush	Adventure	95

```
DELETE FROM movies;
```

Lucky for us that we didn't do this, right?

Removing multiple rows from a table

Including a less-than comparison operator in the WHERE statement

id	title	genre	duration
1	Metropolis	Sci-Fi	153
~~2~~	~~Nosferatu~~	~~Horror~~	~~94~~
~~3~~	~~The Kid~~	~~Comedy~~	~~68~~
~~4~~	~~The Gold Rush~~	~~Adventure~~	~~95~~

```
DELETE FROM movies WHERE duration < 100;
```

This statement would remove all records that have a duration of less than 100.

Try SQL

Section 1: Creating and Removing Databases and Tables

Using SQL to create new databases

Using SQL to create a new database

We can retrieve and manipulate our data. How do we create a completely new database?

We want to add a new theater to the list of databases.

Remember the first database?

Gatsby Theaters Database

We need to create similar tables in a new database.

Gatsby Theaters

Movies

movie info

Showtimes

showtime info

Concessions

concessions info

Promotions

promotions info

Creating a new theater database

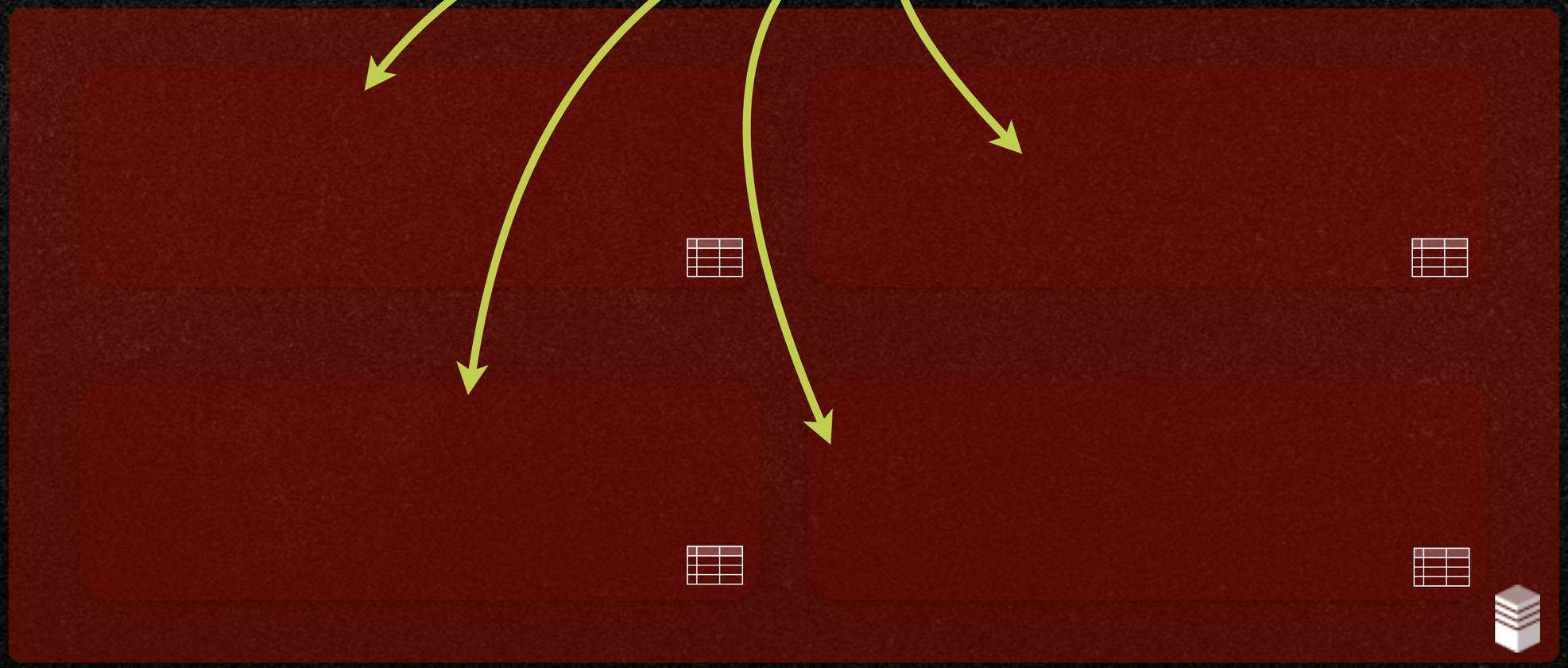
Finding out what is needed to create a completely new database

First, let's start out with an empty database.

Chaplin Theaters

What goes inside of the new database

This is the way the new database will look, but there aren't any tables.



Creating databases using SQL

SQL can be used to create and maintain databases and their tables

The name of the database

CREATE DATABASE Recipe

```
CREATE DATABASE ____________;
```

The statement to create the database

Using the CREATE DATABASE statement

```
CREATE DATABASE Chaplin Theaters;
```

Empty database created... no tables yet!

Using SQL to remove databases

Using SQL to remove a database

What if we realized we do not need a new theater. How would we remove the database?

We want to see what it would take to remove a database from the list of databases.

Dropping databases using SQL

SQL can be used to remove entire databases and their tables

The name of the database

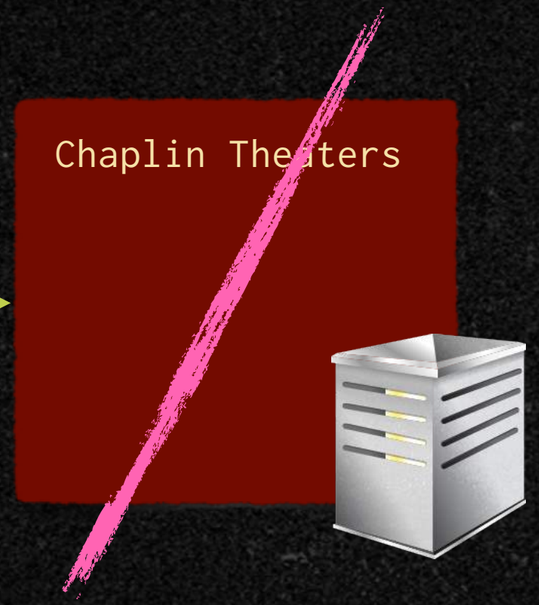
DROP DATABASE Recipe

```
DROP DATABASE ____________;
```

The statement to remove the database

Using the DROP DATABASE statement

```
DROP DATABASE Chaplin Theaters;
```



Be warned: Once this command is run, the database and ALL its data will no longer be available!

Using the DROP DATABASE statement

```
DROP DATABASE Chaplin Theaters;
```

The new (empty) database would be completely removed from the server!

Using SQL to create new tables

Using SQL to create a new table

How can we create a completely new table inside of the new database?

We want to create a Movies table inside the Chaplin Theaters database.

Using SQL to create new tables

CREATE TABLE Recipe

```
CREATE TABLE _________
(
column_name1
column_name2
column_name3
...
);
```

The name of the table

Set of parentheses describing the columns of the table

Using SQL to create new tables

CREATE TABLE Recipe

```
CREATE TABLE
(
column_name1 datatype,
column_name2 datatype,
column_name3 datatype,
…
);
```

Normally a description of the data contained in that column

Describes what kind of value a column can contain

Chaplin Theaters database

Adding a new table to the new database

This is the first table to be added to the database.

Chaplin Theaters

Movies

movie info

showtime info

concessions info

promotions info

Choosing how the columns will look

Chaplin Theaters data for movies

id	title	genre	duration
1	Don Juan	Romance	110
2	Peter Pan	Adventure	105
3	The Lost World	Fantasy	106
4	Robin Hood	Adventure	143

How can the identity of the columns and their data types be created in the database?

Explanation of creating columns in a table

4 separate columns of data

id	title	genre	duration
1	Don Juan	Romance	110
2	Peter Pan	Adventure	105
3	The Lost World	Fantasy	106
4	Robin Hood	Adventure	143

numbers | characters | characters | numbers

Explanation of creating columns in a table

id	title	genre	duration
1	Don Juan	Romance	110
2	Peter Pan	Adventure	105
3	The Lost World	Fantasy	106
4	Robin Hood	Adventure	143

First column of data

The first grows incrementally as more rows are added.

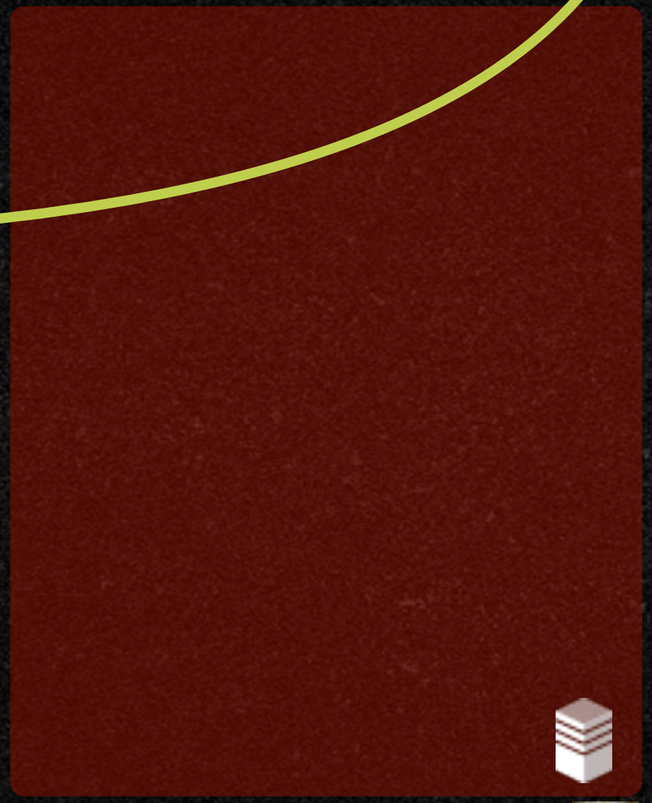
numbers

id	title	genre	duration
1	Don Juan	Romance	110
2	Peter Pan	Adventure	105
3	The Lost World	Fantasy	106
4	Robin Hood	Adventure	143

Creating the first column

This column will hold numbers. Therefore, it will use the data type of INT.

```
CREATE TABLE movies
(
id          int,
column_name datatype,
column_name datatype,
column_name datatype
);
```



Looking at the next column

This is where the names of the films are given, which can be letters and/or numbers.

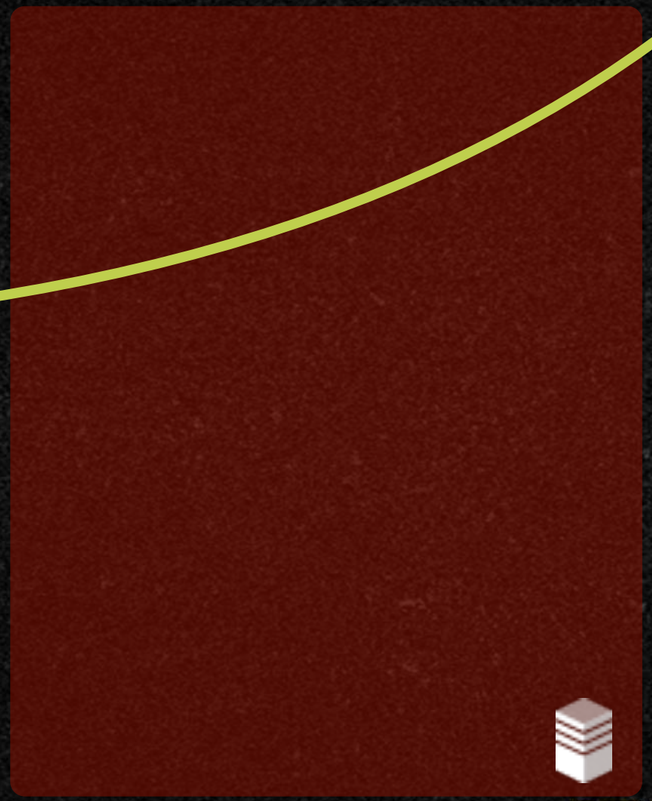
characters

id	title	genre	duration
1	Don Juan	Romance	110
2	Peter Pan	Adventure	105
3	The Lost World	Fantasy	106
4	Robin Hood	Adventure	143

More column formation

This column will hold letters or numbers, so it will use the data type of VARCHAR.

```
CREATE TABLE movies
(
id            int,
title         varchar(50),
column_name   datatype,
column_name   datatype
);
```



Creating columns in a table

This column is where the type of film is given, which are letters.

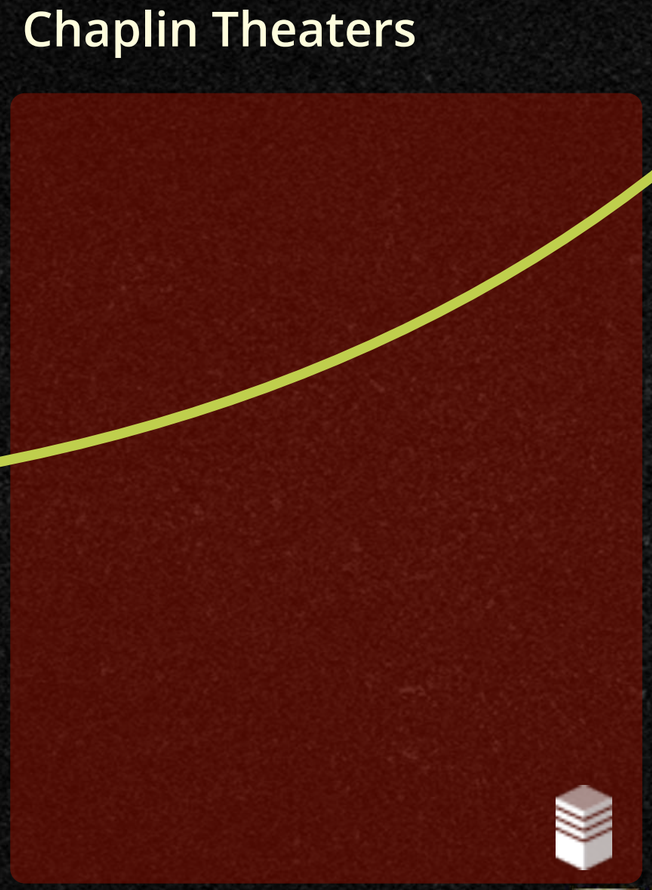
characters

id	title	genre	duration
1	Don Juan	Romance	110
2	Peter Pan	Adventure	105
3	The Lost World	Fantasy	106
4	Robin Hood	Adventure	143

Creating columns from different data

This column will hold letters or numbers, so it will also use the data type of VARCHAR.

```
CREATE TABLE movies
(
id            int,
title         varchar(50),
genre         varchar(15),
column_name   datatype
);
```



Last column

This column appears to hold only numbers.

numbers

id	title	genre	duration
1	Don Juan	Romance	110
2	Peter Pan	Adventure	105
3	The Lost World	Fantasy	106
4	Robin Hood	Adventure	143

Adding the last column to the table

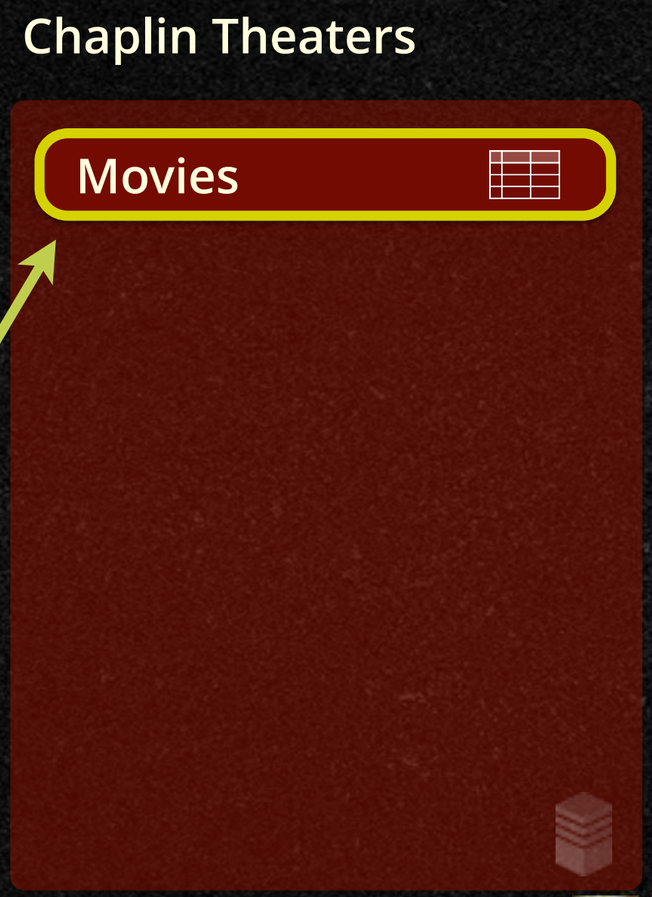
This column will hold numbers, so it will use the data type of INT.

```
CREATE TABLE movies
(
id              int,
title           varchar(20),
genre           varchar(100),
duration        int
);
```

Adding a new table to the database

```
CREATE TABLE movies
(
id          int,
title       varchar(20),
genre       varchar(100),
duration    int
);
```

The new `movies` table has been created in the Chaplin Theaters database.



So what's inside of the new table?

We would see an empty table about the Movies.

Movies

id	title	genre	duration

Looks like it's time to add, or INSERT, some data into the newly created table, which we learned how to do earlier.

Using SQL to remove a table

Removing a table using SQL

How do we remove an existing table from the database?

We want to know how to remove the Movies table from the Chaplin Theaters database.

Removing tables using SQL

Using SQL to remove a table from a database

The name of the table to be removed

DROP TABLE Recipe

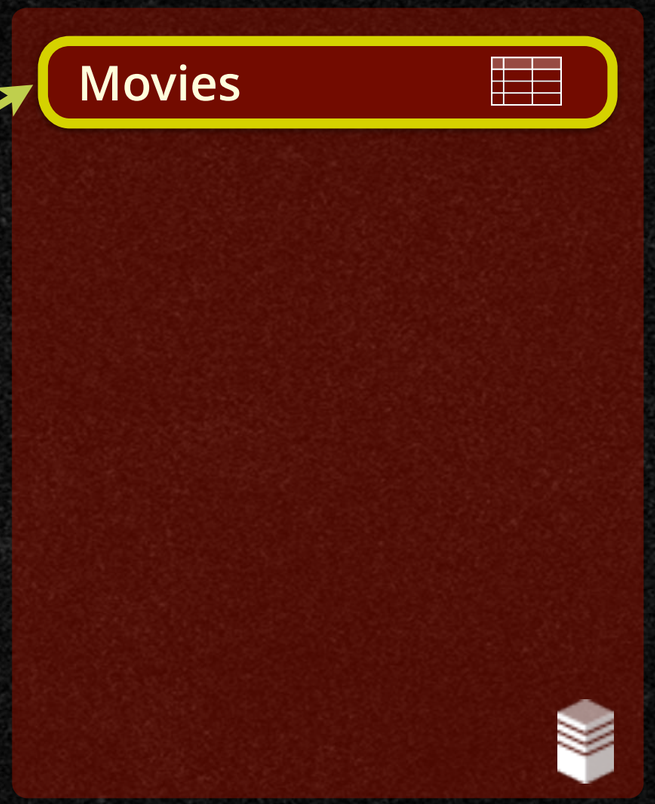
```
DROP TABLE ___________;
```

The statement to remove a table

Deleting tables

```
DROP TABLE movies;
```

Chaplin Theaters

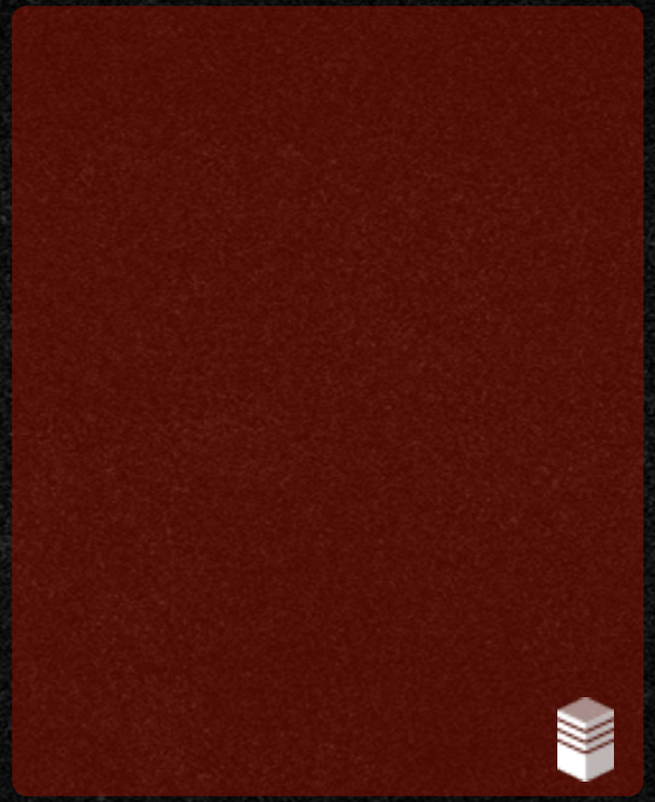


To remove the `movies` table from the Chaplin Theaters database, this is what would happen.

Deleting tables

```
DROP TABLE movies;
```

Chaplin Theaters



It would no longer be available to use by the Chaplin Theaters database.

Try SQL

Section 2: Manipulating Tables

Using SQL to change table structure

Using SQL to alter a column inside of a table

How can we add, change, or even drop a column in an existing table?

We want to add a new column called Ratings to the Movies table.

Using the ALTER TABLE command

The ALTER TABLE command is used to add, modify, and remove columns in a table.

ALTER TABLE ADD Recipe

```
ALTER TABLE ________
ADD COLUMN __________ _______;
```

The table to make changes to

The data type of the new column

The ADD clause allows a column to be added to a table

The name of the column

Adding a column to a table

Movies

id	title	genre	duration
1	Don Juan	Romance	110
2	Peter Pan	Adventure	105
3	The Lost World	Fantasy	106
4	Robin Hood	Adventure	143

```
ALTER TABLE movies
ADD COLUMN ratings int;
```

Let's ADD the column of ratings to the movies table.

It should be able to store numbers from 1 to 10 with the int data type.

Adding a column to a table

Movies

id	title	genre	duration	ratings
1	Don Juan	Romance	110	
2	Peter Pan	Adventure	105	
3	The Lost World	Fantasy	106	
4	Robin Hood	Adventure	143	

```
ALTER TABLE movies
ADD COLUMN ratings int;
```

Remember how to UPDATE data?

Adding some data to the ratings column

```
UPDATE movies
SET ratings = 8
WHERE title = 'Don Juan';
```

```
UPDATE movies
SET ratings = 9
WHERE title = 'Peter Pan';
```

```
UPDATE movies
SET ratings = 7
WHERE title = 'The Lost World';
```

```
UPDATE movies
SET ratings = 7
WHERE title = 'Robin Hood';
```

Again, the WHERE clause is used to help pinpoint the places where data can be added.

New column now has data

Movies

id	title	genre	duration	ratings
1	Don Juan	Romance	110	8
2	Peter Pan	Adventure	105	9
3	The Lost World	Fantasy	106	7
4	Robin Hood	Adventure	143	7

The new column now has data that can be used in the database.

Removing a column from a table

Movies

id	title	genre	duration	ratings
1	Don Juan	Romance	110	8
2	Peter Pan	Adventure	105	9
3	The Lost World	Fantasy	106	7
4	Robin Hood	Adventure	143	7

```
ALTER TABLE movies
DROP COLUMN ratings;
```

If the column is no longer needed, it can be removed with the DROP clause.

Removing a column from a table

Movies

id	title	genre	duration
1	Don Juan	Romance	110
2	Peter Pan	Adventure	105
3	The Lost World	Fantasy	106
4	Robin Hood	Adventure	143

```
ALTER TABLE movies
DROP COLUMN ratings;
```

The column has been removed from the table.