SQL & Databases

1

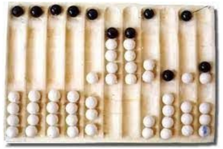
Database?

Let’s start with an overview

Data Collection and Storage

18,000 BC

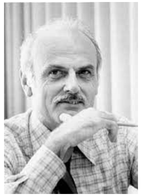
2,400 BC



...

1965

1970



1999

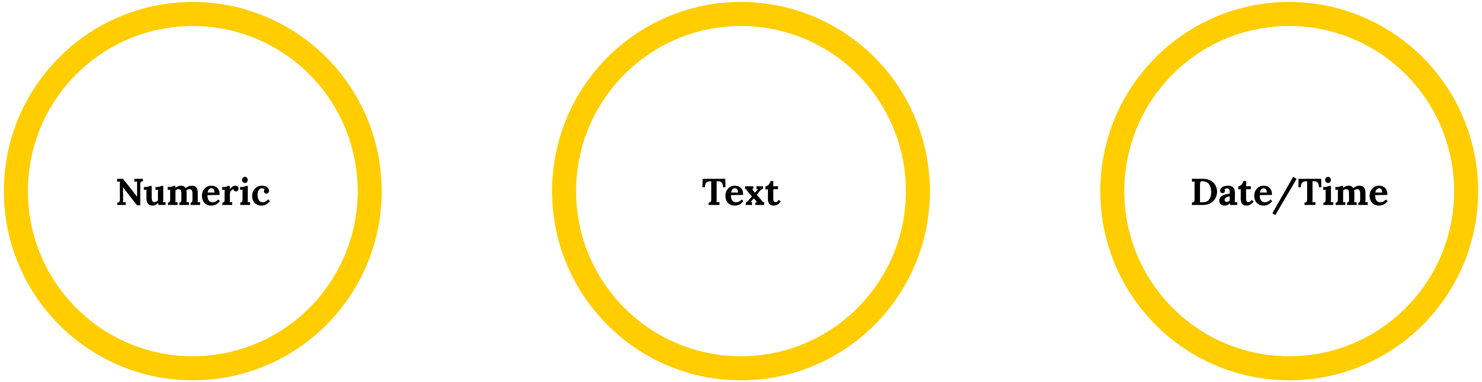
Now

Amazon S3

How does a database looks like?

Modelize one with a Google sheet

Type of Information



Numeric

Text

Date/Time

Type of Information

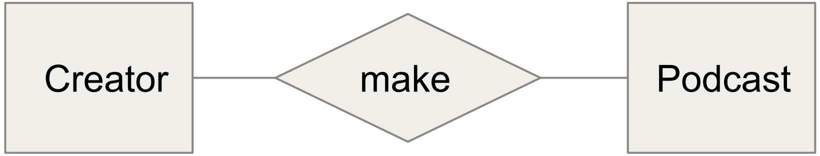
1, 1.2, 10

“Hello there”

22/09/2022,
22/09/2022
18:00

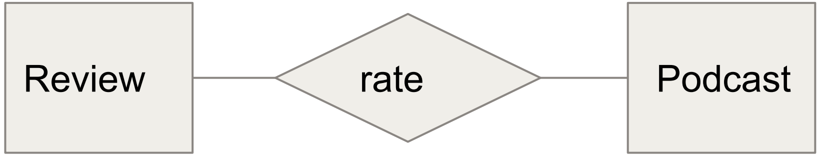
Create a Database

ER Diagram (Entity Relationship)



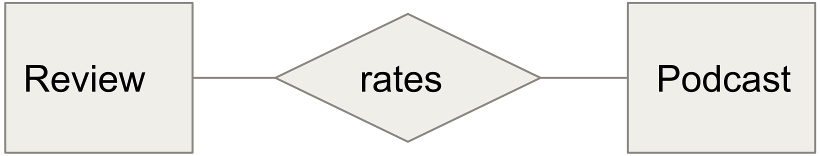
Create a Database

ER Diagram (Entity Relationship)



Create a Database

ER Diagram (Entity Relationship)

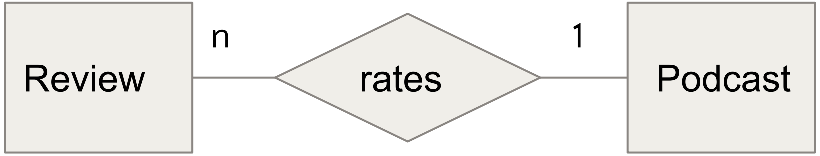


review
Content
Rating
Author
date

podcast
title
I_tunes url

Create a Database

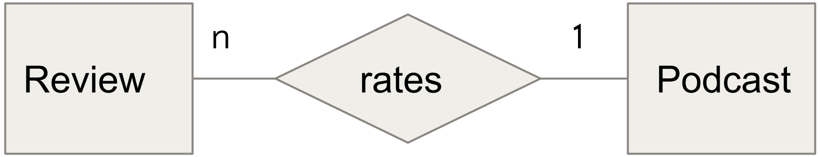
ER Diagram (Entity Relationship)



One Podcast can have several reviews **(n)**
One review for one podcast only **(1)**

Create a Database

ER Diagram (Entity Relationship)



1-N Relationship (or 1 to Many)

Create a Database

Relationships

1-1

- 1 book is only having one author (hypothesis in our case study)

1-N

- 1 book is having different reviews
- 1 review is just for one book

N-N :

- 1 book can be published by different publishers
- 1 publishers can published different books

Ids (let’s go back to the spreadsheet)

Primary Key
Id

Foreign Keys
Podcast_id, author_id

How to Create an ER Diagram ?

Let’s connect to an online tool

Presenting SQL Results

From the terminal

Data Studio

From an application

Database Tool

Relational VS Non Relational

Relational

- Pre-established relationships
- One row = one single data item
- Schema, primary keys, foreign keys
- Each record have the same structure

Non-Relational (Less Structured)

- Unstructured, structured, semi-structured items
- Store data in a non tabular form
-possibility to add a reference to the items
- transactions can be distributed across multiple servers

Relational VS Non Relational

Pros of a Relational Database

- Easy structure with categories
- Great for structured data
- Consistency of the data inputs
- Data points can be linked easily with SQL
- Easy to build complex queries
- Secure transactions, high reliability

Pros of a Non-Relational Database

- Data is not confined to a structured group
- Greater flexibility, more dynamic
- Rapid adaptation to changing requirements
- High performance
- Easy to store large amount of data with less structure

–

Relational VS Non Relational

Cons of a Relational Database

- No adaption
- High performance and scalability are only possible with expensive or custom-built hardware

Cons of a Non-Relational Database

- Manual Language
- Low reliability
- Data integrity and consistency not the priority

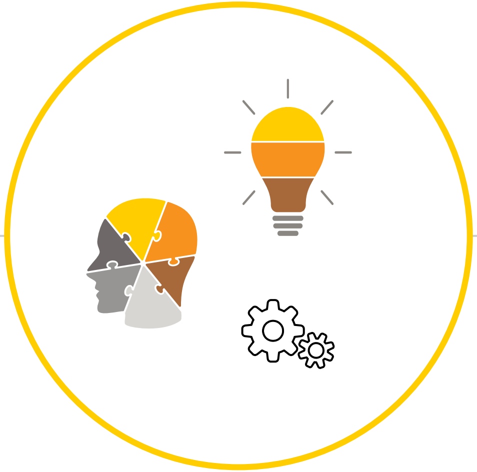
Relational VS Non Relational

SQL

ORACLE
MySQL
PostgreSQL
MariaDB

No-SQL

mongoDB
APACHE HBASE
redis
amazon DynamoDB



How to choose?

1

SQL Syntax

Introduction to SQL

Interacting with the Database

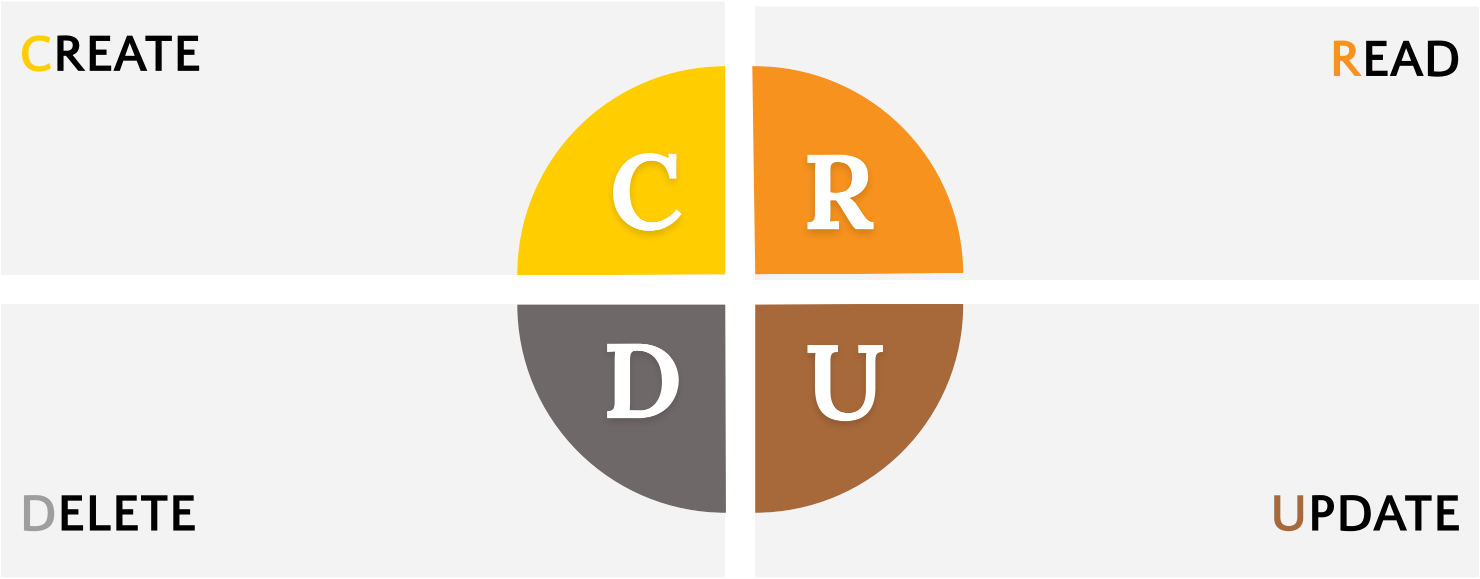
DBeaver

Free multi-platform database tool

Support DB such as:

MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, Teradata, Firebird, Apache Hive, Phoenix, Presto

CRUD



CREATE

READ

DELETE

UPDATE

CREATE

Create a new row in the DB

```
INSERT INTO table (column1 , column2,
column3 ... )
VALUES (value1 , value2, value3 ... )
```

Column 1	Column 2	....
Hello	8	
Bye	4	
Value 1	**Value 2**	**...**

READ

- **SELECT** is the most common sql operator
- Used to get (read) data from a table

```
SELECT *
FROM table1;
```

```
SELECT column1, column 2
FROM table1;
```

Later, we go back to query creation in more details

UPDATE

Change the data of one or more records in a table

```
UPDATE table_name
SET column_name = value , column_name = value ...
WHERE condition;
```

DELETE

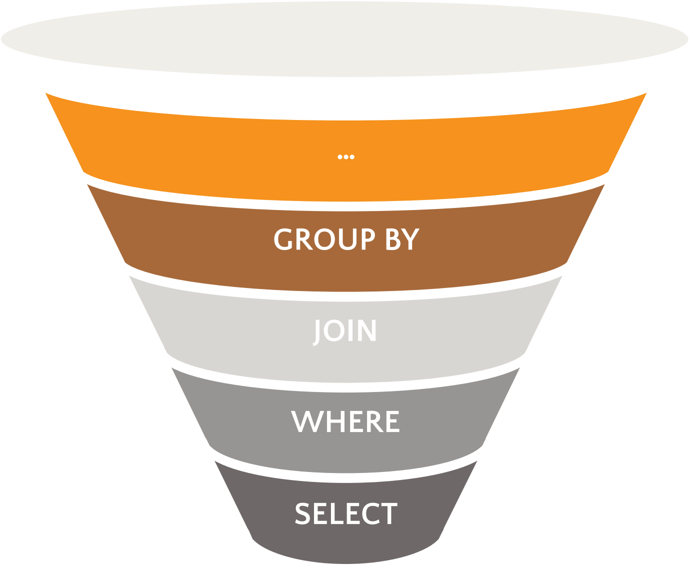
Removes one or more records from the table

```
DELETE FROM table
WHERE condition;
```

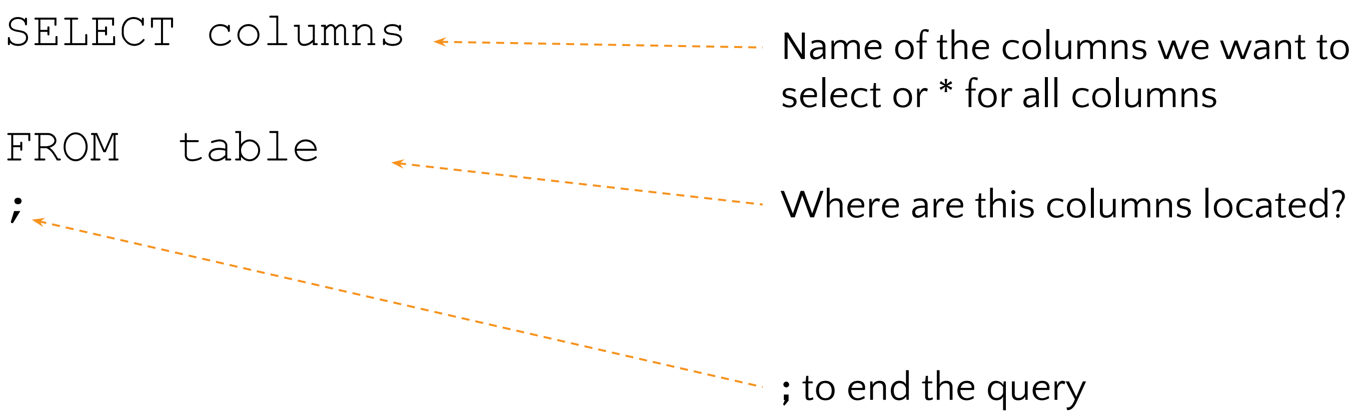
Simple SQL Queries

SELECT and its friends

Building Queries



SELECT - FROM



```
SELECT columns

FROM  table
;
```

Name of the columns we want to select or * for all columns

Where are this columns located?

; to end the query

SELECT - FROM

Famous SQL query

Looking how our table is looking like

```
SELECT *

FROM  podcasts

LIMIT 10

;
```

All the columns

We query the table named ‘podcasts’

We just want the top 10 rows

SELECT - FROM

Famous SQL query

Looking how our table is looking like

```
SELECT podcast_id, title

FROM   podcasts

;
```

The columns

We query the table named ‘podcasts’

SELECT - FROM

Famous SQL query

Looking how our table is looking like

```
SELECT podcasts.podcast_id,
podcasts.title

FROM  podcasts

;
```

The columns are from the podcast table

We query the table named ‘podcasts’

SELECT - FROM

Famous SQL query

Looking how our table is looking like

```
SELECT podcasts.podcast_id AS
podcast_number,
podcasts.title

FROM  podcasts

;
```

We can give a nickname to our column name

We query the table named ‘podcasts’

SELECT - FROM

```
SELECT title, rating * 2

FROM  reviews


;
```

We can perform arithmetic operations with sql (+, -, *, / and %)

```
SELECT 3 * 2
;
```

WHERE FOR CONDITION

```
SELECT columns

FROM   table

WHERE condition
;
```

We only select the row meeting some conditions

WHERE FOR CONDITION

```
SELECT *

FROM  reviews

WHERE rating >= 4
;
```

We only select the row meeting some conditions

WHERE FOR CONDITION

Different predicates

- ➔ `=, <>, >, >=, <, <=`
- ➔ `IN, BETWEEN, LIKE, IS NULL, IS NOT NULL`

WHERE FOR CONDITION

Different predicates

- IN: `WHERE categories.category IN ('Art', 'Music')`
- BETWEEN: `WHERE runs.run_at BETWEEN 2021-05-10 AND 2021-05-10`
- LIKE: `WHERE reviews.content LIKE ‘%interesting%’`
- IS NULL: `WHERE reviews.title IS NULL`
- IS NOT NULL: `WHERE reviews.title IS NOT NULL`

WHERE FOR CONDITION

DETAILS ABOUT LIKE OPERATOR

%: replace anything:

- `WHERE title like ‘%d’` : find values that end with d
- `WHERE title like ‘%d%’` : find values that have d somewhere
- `WHERE title like ‘%d ’`: find values that end with d

_: replace any single character:

- `WHERE title like ‘_d%’` : find values that have d in second position
- `WHERE title like ‘d_%’` : find values that have start with d and have at least 1 letter after
- `WHERE title like ‘d__%’` : find values that have start with d and have at least 2 letters after

WHERE FOR CONDITION

LOGICAL OPERATORS

```
-   OR : (condition 1 OR condition 2)
-   AND : ( condition 1 AND condition 2)
-   NOT  : (NOT condition 1)
```

WHERE FOR CONDITION

LOGICAL OPERATORS

c1	*c2*	AND	OR
T	T	T	T
T	F	F	T
F	T	F	T
F	F	F	F

WHERE FOR CONDITION

LOGICAL OPERATORS

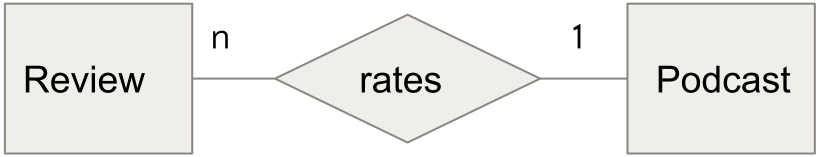
- OR : (condition 1 OR condition 2)
- AND : (condition 1 AND condition 2)
- NOT : (NOT condition 1)

We can also combine all this operators to create complex conditions

SQL JOINS

One table is cool, multiple tables are more fun

Introduction to Joins



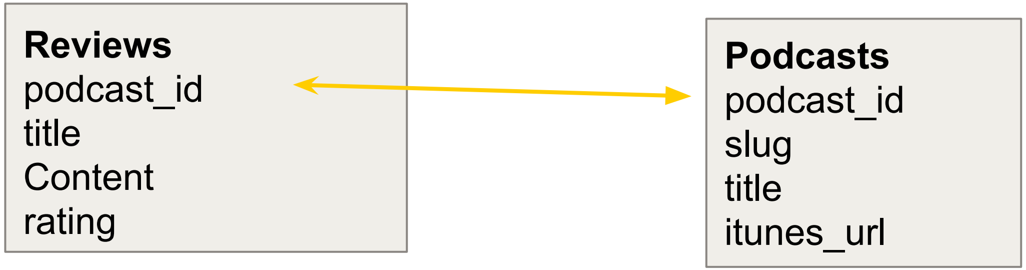
Review

n

rates

1

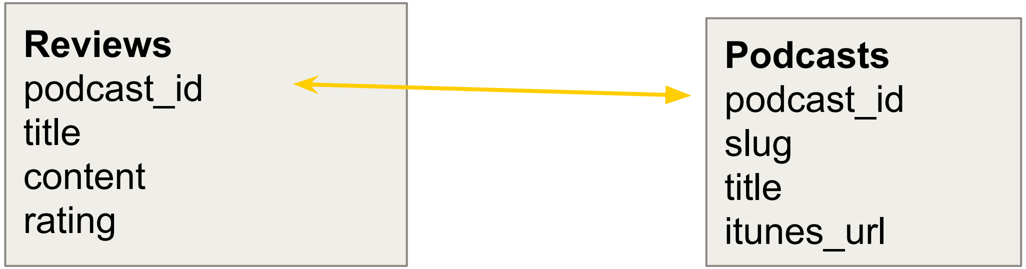
Podcast



Reviews
podcast_id
title
Content
rating

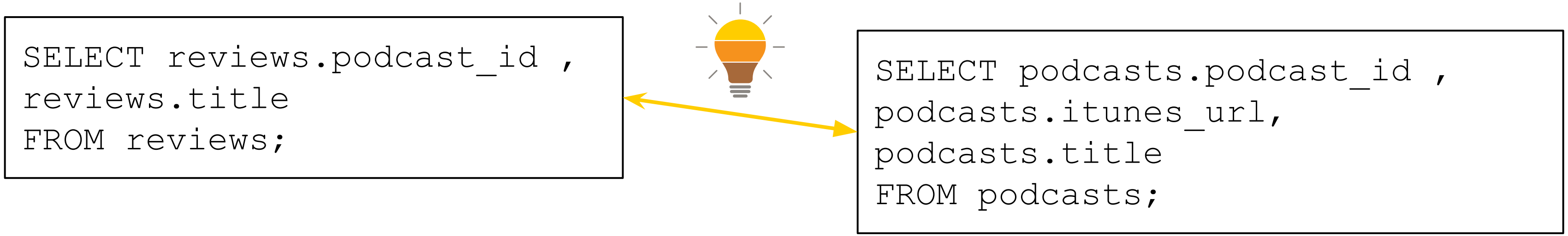
Podcasts
podcast_id
slug
title
itunes_url

Introduction to Joins



Reviews
podcast_id
title
content
rating

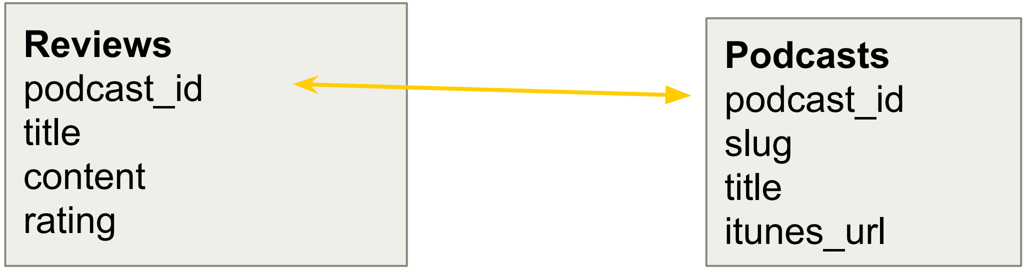
Podcasts
podcast_id
slug
title
itunes_url



```
SELECT reviews.podcast_id ,
reviews.title
FROM reviews;
```

```
SELECT podcasts.podcast_id ,
podcasts.itunes_url,
podcasts.title
FROM podcasts;
```

Introduction to Joins

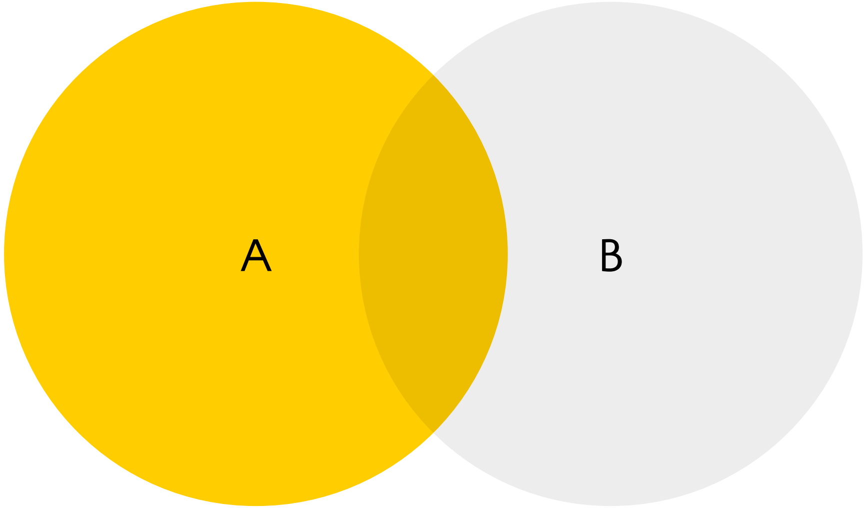


Reviews
podcast_id
title
content
rating

Podcasts
podcast_id
slug
title
itunes_url

```
SELECT podcast.podcast_id,
podcasts.title
reviews.rating
FROM podcasts
JOIN reviews ON reviews.podcast_id = podcasts.podcast_id;
```

LEFT JOIN



LEFT JOIN

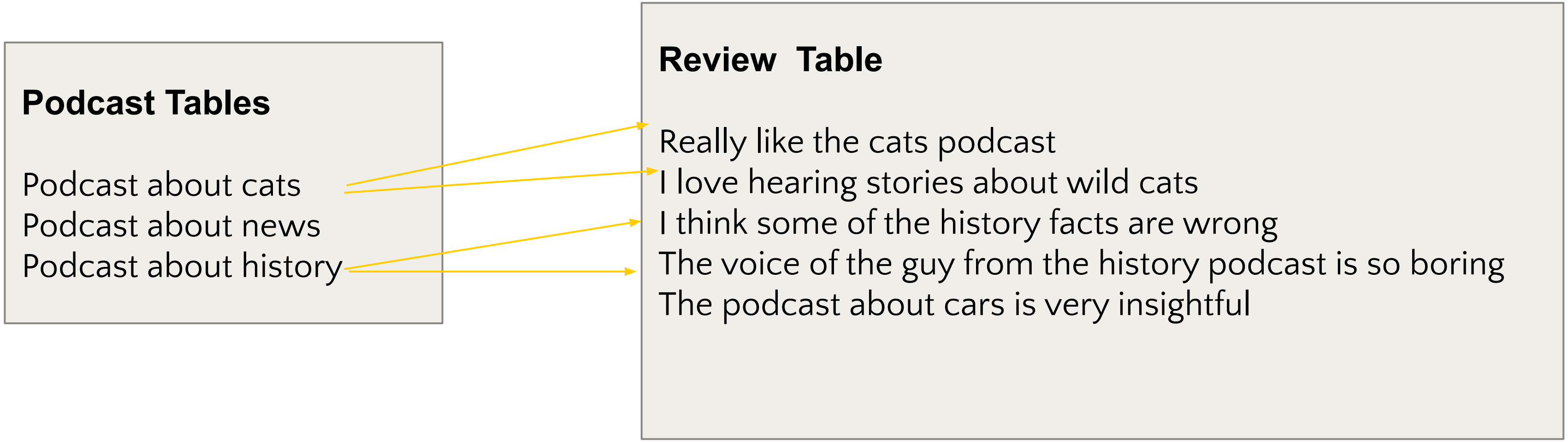
Podcast Tables

Podcast about cats
Podcast about news
Podcast about history

Review Table

Really like the cats podcast
I love hearing stories about wild cats
I think some of the history facts are wrong
The voice of the guy from the history podcast is so boring
The podcast about cars is very insightful

LEFT JOIN



Podcast Tables

Podcast about cats
Podcast about news
Podcast about history

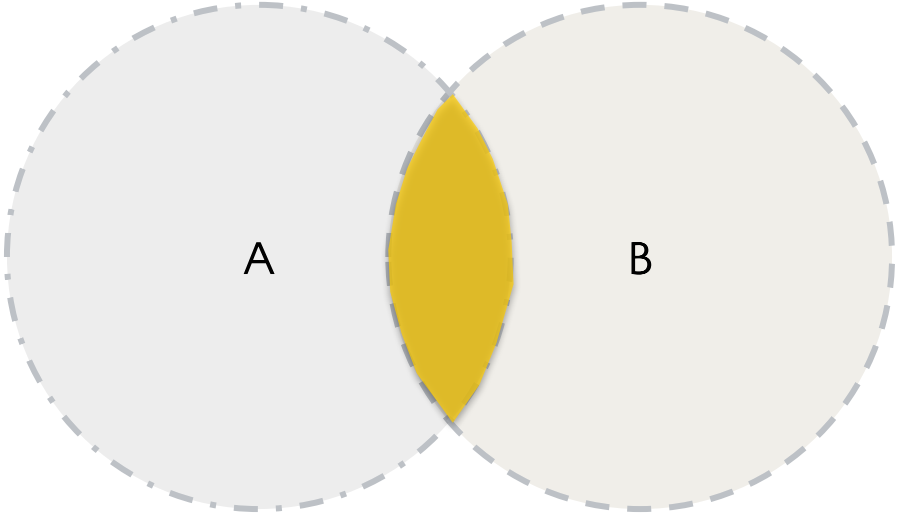
Review Table

Really like the cats podcast
I love hearing stories about wild cats
I think some of the history facts are wrong
The voice of the guy from the history podcast is so boring
The podcast about cars is very insightful

LEFT JOIN

Podcast about cats	Really like the cats podcast
Podcast about cats	I love hearing stories about wild cats
Podcast about news	**NULL**
Podcast about history	I think some of the history facts are wrong
Podcast about history	The voice of the guy from the history podcast is so boring

INNER JOIN



INNER JOIN

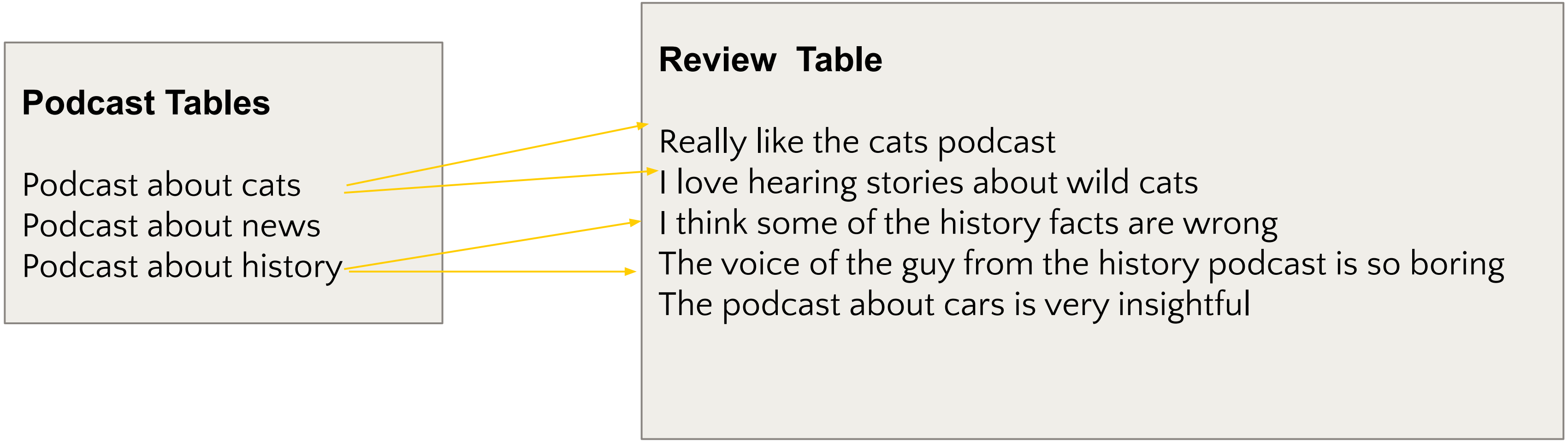
Podcast Tables

Podcast about cats
Podcast about news
Podcast about history

Review Table

Really like the cats podcast
I love hearing stories about wild cats
I think some of the history facts are wrong
The voice of the guy from the history podcast is so boring
The podcast about cars is very insightful

INNER JOIN



Podcast Tables

Podcast about cats
Podcast about news
Podcast about history

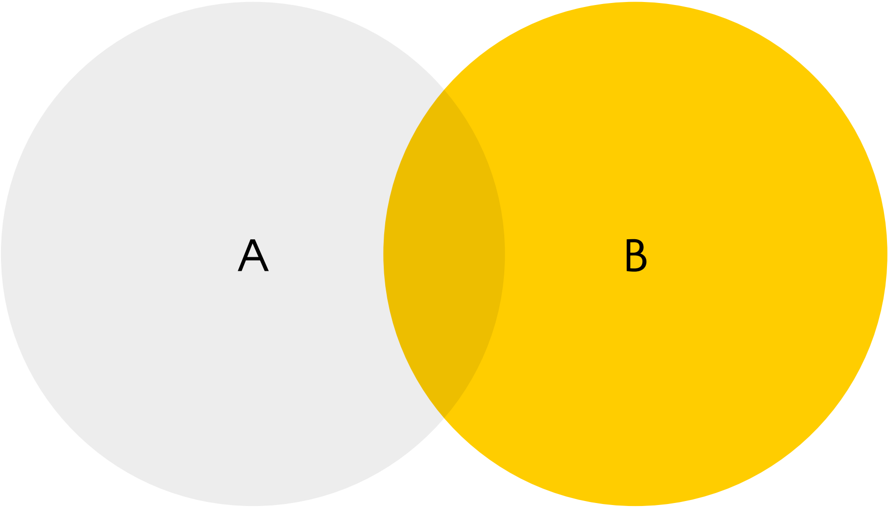
Review Table

Really like the cats podcast
I love hearing stories about wild cats
I think some of the history facts are wrong
The voice of the guy from the history podcast is so boring
The podcast about cars is very insightful

INNER JOIN

Podcast about cats	Really like the cats podcast
Podcast about cats	I love hearing stories about wild cats
Podcast about history	I think some of the history facts are wrong
Podcast about history	The voice of the guy from the history podcast is so boring

RIGHT JOIN



RIGHT JOIN

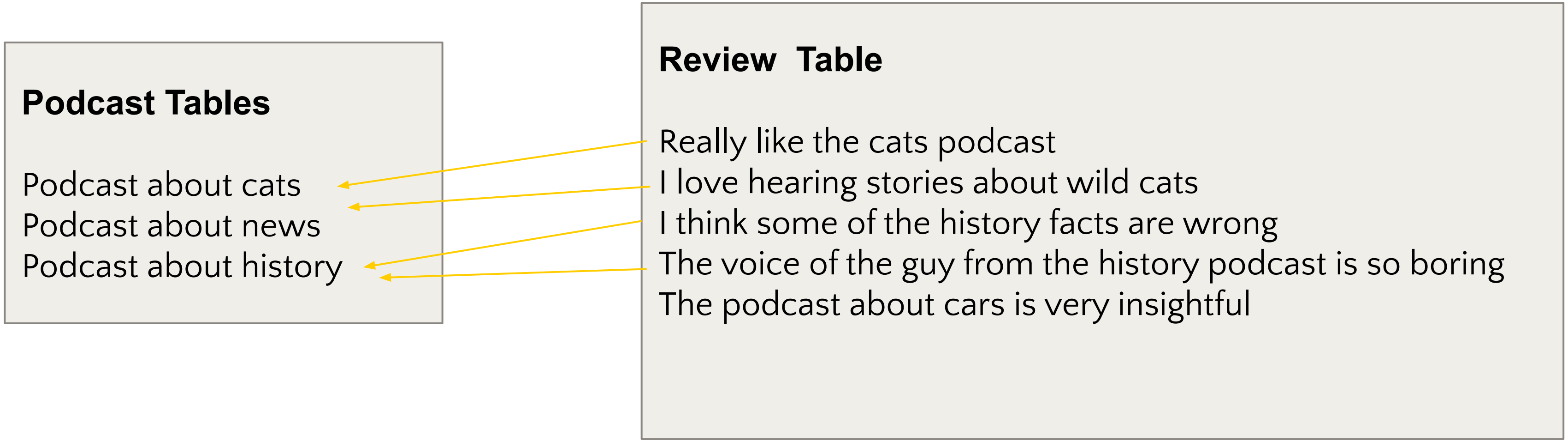
Podcast Tables

Podcast about cats
Podcast about news
Podcast about history

Review Table

Really like the cats podcast
I love hearing stories about wild cats
I think some of the history facts are wrong
The voice of the guy from the history podcast is so boring
The podcast about cars is very insightful

RIGHT JOIN



Podcast Tables

Podcast about cats
Podcast about news
Podcast about history

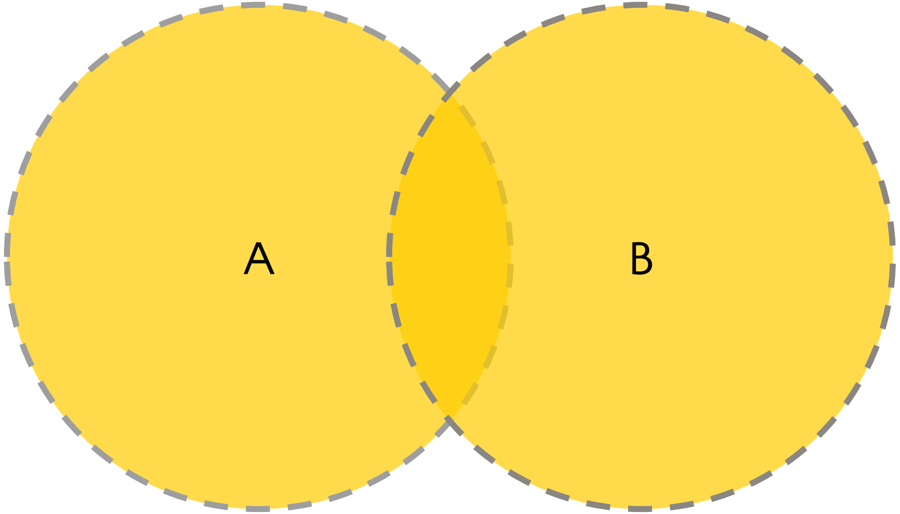
Review Table

Really like the cats podcast
I love hearing stories about wild cats
I think some of the history facts are wrong
The voice of the guy from the history podcast is so boring
The podcast about cars is very insightful

RIGHT JOIN

Podcast about cats	Really like the cats podcast
Podcast about cats	I love hearing stories about wild cats
Podcast about history	I think some of the history facts are wrong
Podcast about history	The voice of the guy from the history podcast is so boring
NULL	The podcast about cars is very insightful

FULL OUTER JOIN



FULL OUTER JOIN

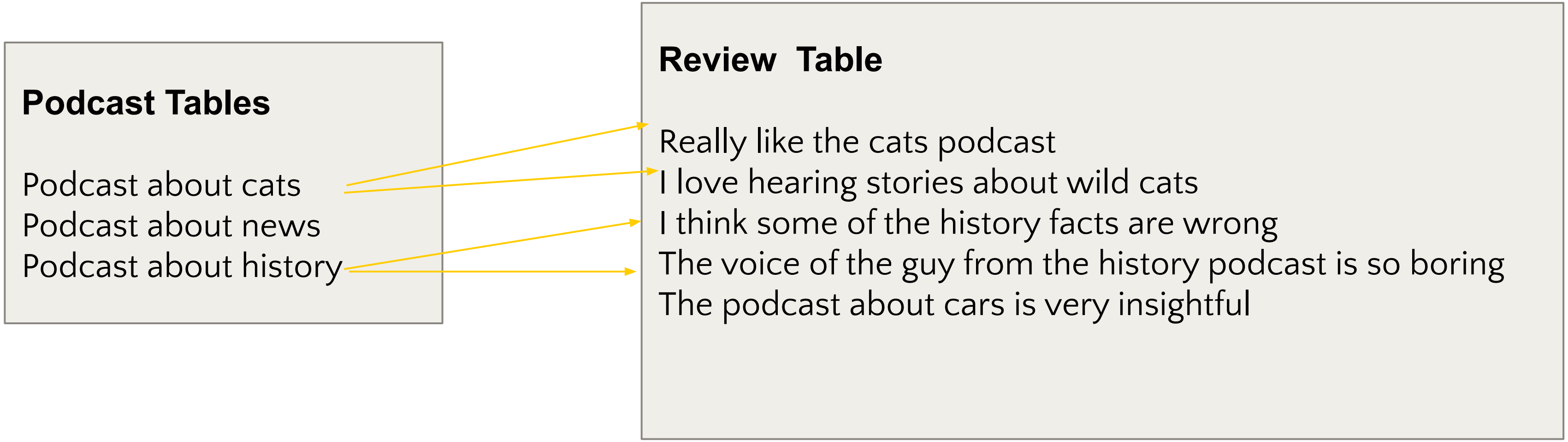
Podcast Tables

Podcast about cats
Podcast about news
Podcast about history

Review Table

Really like the cats podcast
I love hearing stories about wild cats
I think some of the history facts are wrong
The voice of the guy from the history podcast is so boring
The podcast about cars is very insightful

FULL OUTER JOIN



Podcast Tables

Podcast about cats
Podcast about news
Podcast about history

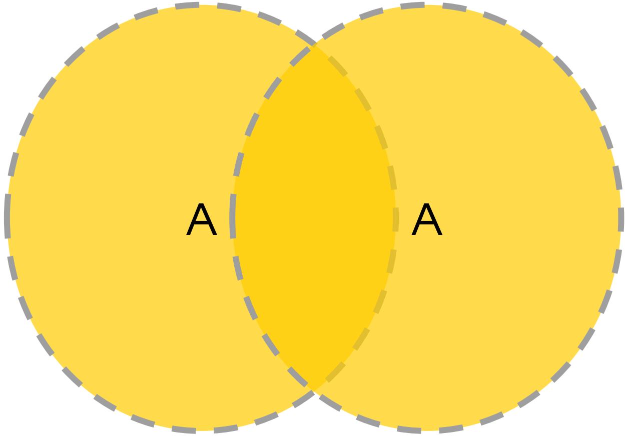
Review Table

Really like the cats podcast
I love hearing stories about wild cats
I think some of the history facts are wrong
The voice of the guy from the history podcast is so boring
The podcast about cars is very insightful

FULL OUTER JOIN

Podcast about cats	Really like the cats podcast
Podcast about cats	I love hearing stories about wild cats
Podcast about news	**NULL**
Podcast about history	I think some of the history facts are wrong
Podcast about history	The voice of the guy from the history podcast is so boring
NULL	The podcast about cars is very insightful

SELF JOIN



Id	Employee_name	Supervisor_id
1	Jane	3
2	Petra	5
3	Lili	2

Id	Employee_name	Supervisor_name
1	Jane	Lili
2	Petra	Anna
3	Lili	Petra

SELF JOIN

```
SELECT employees.id,
employees.employee_name,
supervisor.employee_name as
supervisor_name

FROM employees as employee
JOIN employees as supervisor
ON employees.supervisor_id =
supervisor.id
;
```

Id	Employee_name	Supervisor_id
1	Jane	3
2	Petra	5
3	Lili	2

Id	Employee_name	Supervisor_name
1	Jane	Lilie
2	Petra	Anna
3	Lili	Petra

CROSS JOIN

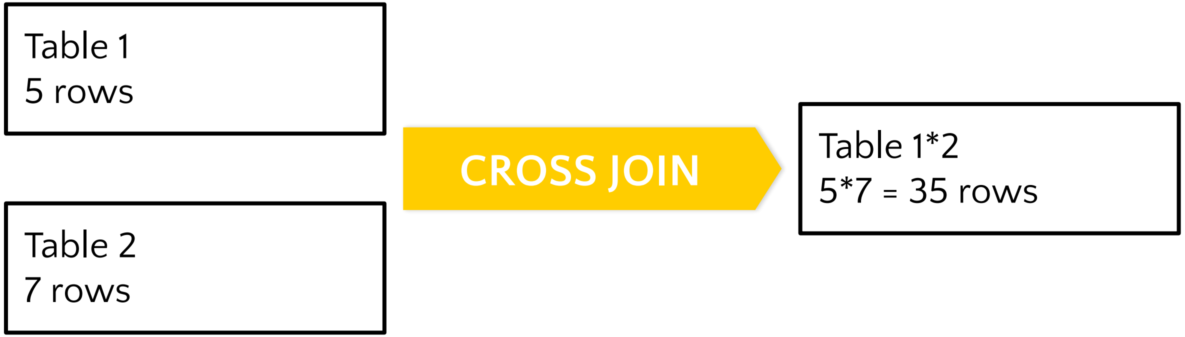


Table 1
5 rows

Table 2
7 rows

CROSS JOIN

Table 1*2
5*7 = 35 rows

CROSS JOIN

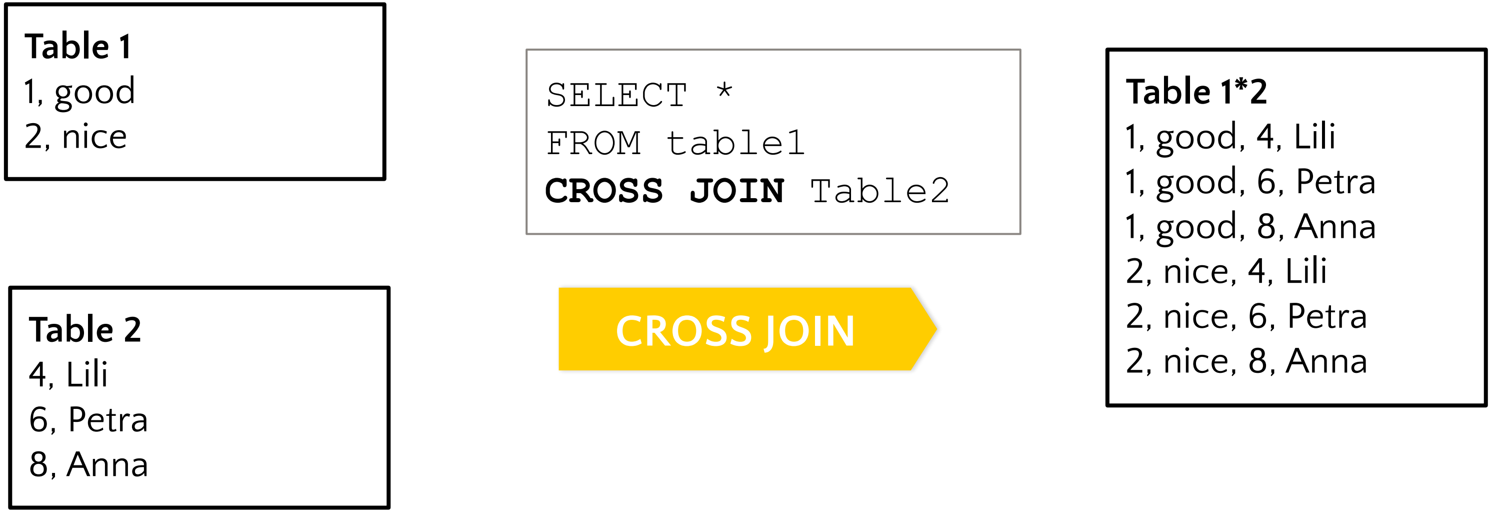


Table 1
1, good
2, nice

Table 2
4, Lili
6, Petra
8, Anna

```
SELECT *
FROM table1
CROSS JOIN Table2
```

CROSS JOIN

Table 1*2
1, good, 4, Lili
1, good, 6, Petra
1, good, 8, Anna
2, nice, 4, Lili
2, nice, 6, Petra
2, nice, 8, Anna

UNION

Title	Rating
5/5	5
Interesting	5

Title	Rating
Very bad	1
Boring	1

UNION →

Title	Rating
5/5	5
Interesting	5
Very bad	1
Boring	1

UNION

Title	Rating
5/5	5
Interesting	5

Title	Rating
Very bad	1
Boring	1

```
SELECT title,
Rating
FROM ratings
WHERE rating=5

UNION

SELECT title,
Rating
FROM ratings
WHERE rating=1
```

SQL Function

Let’s add some spice to our queries

SQL functions - Basic Ones

Operations on the columns:

- LEFT: `LEFT('Hello Everyone', 3)` -> ’Hel’
- RIGHT `RIGHT('Hello Everyone', 3)` -> ’one’
- TRIM `TRIM('   Hello Everyone ')` -> ’Hello Everyone’
- LEN `LEN('Hello')` -> 5
- LOWER `LOWER('Hello')` -> ‘hello’
- DATEDIFF `DATEDIFF(year, '2017/01/25', '2011/01/25') AS DateDiff` -> 6
- Many others (month, dateadd , upper..)

SQL functions

More complex ones but very practical

CASE WHEN

```
SELECT reviews.title,
CASE WHEN reviews.rating >= 4 THEN  ‘good podcast’
WHEN reviews.rating = 3 THEN ‘ok podcast’
ELSE  ‘bad podcast’
END  AS category_rating
From reviews;
```

SQL functions - Aggregation

```
SELECT COUNT(*)
FROM Podcasts
```

SQL functions - Aggregation

```
SELECT COUNT(*)
FROM Podcasts
```

```
SELECT SUM(rating)
FROM reviews
```

SQL functions - Aggregation

```
SELECT COUNT(*)
FROM Podcasts
```

```
SELECT SUM(rating)
FROM reviews
```

```
SELECT AVG(rating)
FROM reviews
```

SQL functions - Aggregation

```
SELECT COUNT(*)
FROM Podcasts
```

```
SELECT SUM(rating)
FROM reviews
```

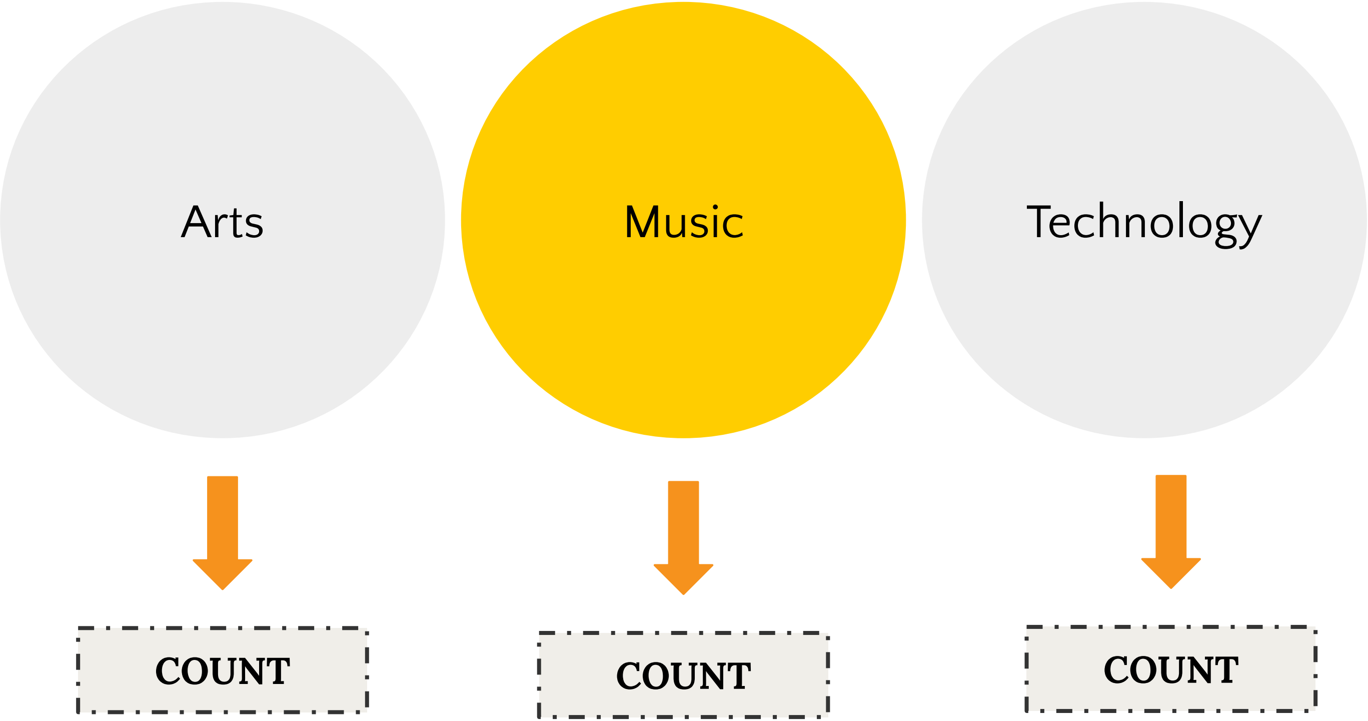
```
SELECT AVG(rating)
FROM reviews
```

```
SELECT MAX(rating)
FROM reviews
```

```
SELECT MIN(rating)
FROM reviews
```

SQL functions - Aggregation

And Group by



SQL functions - Aggregation

And Group by

```
SELECT categories.category,
COUNT(*) AS podcast_count
FROM categories
GROUP BY categories.category
```

category	podcast_count
Arts	65
Music	77
Technology	30

SQL functions - Aggregation

And Group by

AVERAGE and GROUP BY

What is the average rating of each podcast?

```
SELECT podcasts.title, AVG(reviews.rating) as average_rating
FROM podcasts
JOIN reviews on reviews.podcast_id = podcasts.podcast_id
GROUP BY podcasts.title
ORDER BY average_rating DESC
```

SQL functions - Aggregation

And Group by

What if we add some others metrics …

```
SELECT podcasts.title,
AVG(reviews.rating) as average_rating,
MIN(reviews.rating) as lowest_score,
MAX(reviews.rating) as highest_score
FROM podcasts
JOIN reviews on reviews.podcast_id = podcasts.podcast_id
GROUP BY podcasts.title
ORDER BY average_rating DESC
```

SQL functions - Aggregation

And Group by

What if we combine what we know…

```
SELECT CASE WHEN reviews.rating >= 4 THEN  ‘good podcast’
WHEN reviews.rating = 3 THEN ‘ok podcast’
ELSE  ‘bad podcast’
END  AS category_rating,
COUNT(*)
FROM reviews on reviews.podcast_id = podcasts.podcast_id
GROUP BY category_rating
```

SQL functions - Aggregation

And Group by

How to filter the results of group by?

```
SELECT podcasts.title,
AVG(reviews.rating) as average_rating
FROM podcasts
JOIN reviews on reviews.podcast_id = podcasts.podcast_id
GROUP BY podcasts.title
HAVING average_rating >= 4
```

Here we select the podcast whose average rating is greater than 4

SQL functions - Aggregation

And Group by

HAVING AND WHERE ?

```
SELECT …
HAVING average_rating >= 4
```

title	average_rating
Podcast 1	3.1
Podcast 2	4.5
Podcast 3	4.9

HAVING

title	average_rating
Podcast 2	4.5
Podcast 3	4.9

SQL functions - Aggregation

And Group by

HAVING AND WHERE ?

```
SELECT podcasts.title,
AVG(reviews.rating) as average_rating
FROM podcasts
JOIN reviews on reviews.podcast_id = podcasts.podcast_id
WHERE reviews.rating >=2
GROUP BY podcasts.title
```

Here we perform the average ONLY for podcast whose review is greater than 2

SQL functions - Aggregation

And Group by

HAVING AND WHERE ?

```
SELECT …
WHERE reviews.rating >=2
GROUP BY podcasts.title
```

title	rating
Podcast 1	4
Podcast 1	5
Podcast 1	1
Podcast 2	4

WHERE

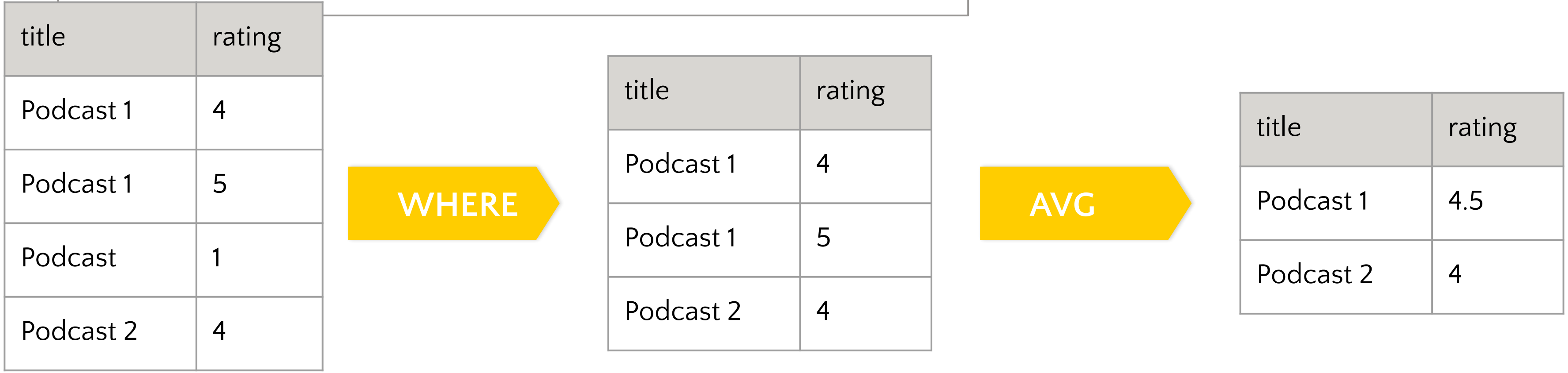
title	rating
Podcast 1	4
Podcast 1	5
Podcast 2	4

SQL functions - Aggregation

And Group by

HAVING AND WHERE ?

```
SELECT …
WHERE reviews.rating >=2
GROUP BY podcasts.title
```



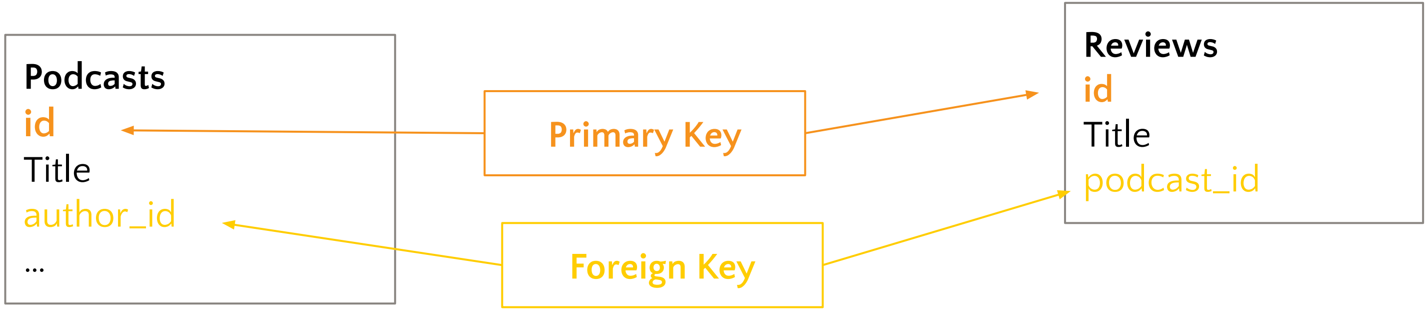
title	rating
Podcast 1	4
Podcast 1	5
Podcast	1
Podcast 2	4

title	rating
Podcast 1	4
Podcast 1	5
Podcast 2	4

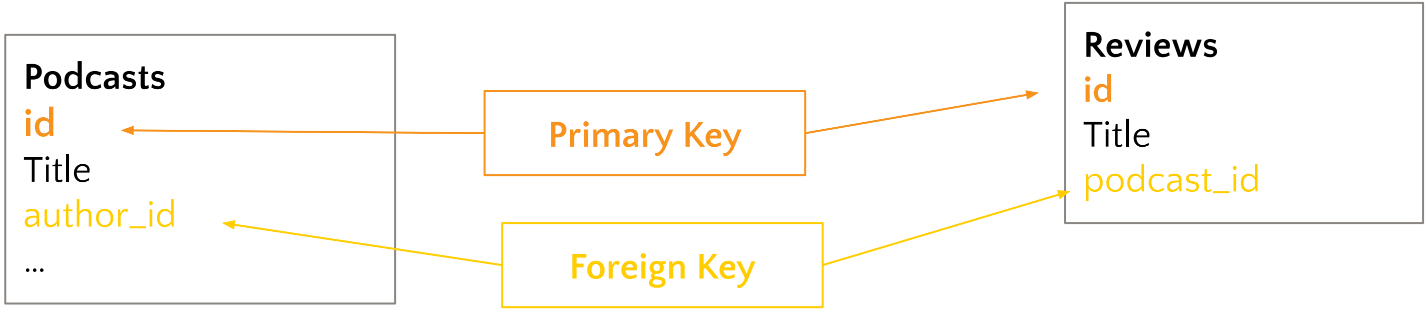
title	rating
Podcast 1	4.5
Podcast 2	4

A small digression on Keys and Tables Creation

Primary and Foreign Keys



Primary and Foreign Keys


Podcasts
id
Title
author_id
…

Primary Key

Foreign Key

Reviews
id
Title
podcast_id

Create a Table

```
CREATE TABLE Podcasts (
    Podcast_id int NOT NULL AUTO_INCREMENT,
    Title varchar(255) NOT NULL,
    Content varchar(255),
    Created_at DATE,
    PRIMARY KEY (Podcast_id)
);
```

Indexes

- Index are used to query the data faster
- More crucial for nosql DB
- We cannot see the indexes

```
CREATE INDEX idx_reviews
ON reviews (title, content,
rating, author_id, created_at,
podcast_id);
```

CHECK

GOAL: Add constraints to a column

At the Table Creation

```
CREATE TABLE reviews (
    Podcast id int NOT NULL AUTO_INCREMENT,
    Title varchar(255) NOT NULL,
    rating int CHECK (ratinge>=18),
    Content varchar(255),
    Created_at DATE
);
```

CHECK

GOAL: Add constraints to a column

At the Table Creation

```
CREATE TABLE reviews (
    Podcast id int NOT NULL AUTO_INCREMENT,
    Title varchar(255) NOT NULL,
    rating int ,
    Content varchar(255),
    Created_at DATE
    CONSTRAINT ct_rating CHECK (rating>=-2)
);
```

CHECK

GOAL: Add constraints to a column

After the Table Creation

```
ALTER TABLE reviews
ADD  CHECK (rating>-2)
;
```

```
ALTER TABLE reviews
ADD CONSTRAINTS ct_ratings  CHECK (rating>-2)
;
```

CHECK

GOAL: Add constraints to a column

Error message if violated

```
SQL Error [19]: [SQLITE_CONSTRAINT]  Abort due to constraint
violation (NOT NULL constraint failed: reviews.podcast_id)
```

DEFAULT

At the Table Creation

```
CREATE TABLE reviews (
    Podcast_id int NOT NULL AUTO_INCREMENT,
    Title varchar(255) NOT NULL,
    rating int ,
    Content varchar(255),
    Created_at DATE DEFAULT GETDATE()
);
```

DEFAULT

After the Table Creation

```
ALTER TABLE reviews
ALTER created_at SET DEFAULT GETDATE() ;
```

Nested Queries

Also called servers CTE

CTE

Common

Table

Expression

temporary named result set

CTE

Common

Table

Expression

temporary named result set

CREATING TABLE 1

```
SELECT *
FROM  TABLE 1
```

CTE

```
 with reviews_per_podcast  as
(select podcast_id,
 count(*) as number_of_reviews
 from reviews
 group by podcast_id)

 SELECT avg(number_of_reviews)
 from reviews_per_podcast;
```

CREATING TABLE 1

SELECT *
FROM TABLE 1

CTE

We want a table which shows for each review the average rating for the author

```
WITH average AS (
SELECT author_id,
AVG(rating) AS author_average
FROM  reviews
GROUP BY author_id
)
```

CREATING TABLE 1

```
SELECT r.title,
r.rating as podcast_rating_individual,
a.author_average
FROM reviews r
JOIN average a ON a.author_id = r.author_id
```

SELECT *
FROM TABLE 1

CTE

We want a table which shows for each review the average rating for the author

```
WITH average AS (
SELECT author_id,
AVG(rating) AS author_average
FROM  reviews
GROUP BY author_id
)

SELECT r.title,
r.rating as podcast_rating_individual,
a.author_average
FROM reviews r
JOIN average a ON a.author_id = r.author_id
```

CREATING TABLE 1

SELECT *
FROM TABLE 1

Multiple CTE

CREATING TABLE 1

CREATING TABLE 2

SELECT FROM TABLE1, TABLE 2

Multiple CTE

CREATING TABLE 1

```
WITH table1 AS () ,
```

CREATING TABLE 2

```
table2 AS ()
```

SELECT FROM TABLE1, TABLE 2

```
SELECT *
FROM TABLE1
JOIN TABLE2 on ..
```

Multiple CTE

CREATING TABLE 1

```
WITH table1 AS () ,
```

CREATING TABLE 2

```
table2 AS (SELECT .. FROM
table1)
```

SELECT FROM TABLE1, TABLE 2

```
SELECT *
FROM TABLE1
JOIN TABLE2 on ..
```

Multiple CTE

```
WITH rating_per_podcast AS (
            SELECT p.podcast_id,
                Avg(rating) AS rating_podcast
         FROM   podcasts p
                JOIN reviews r
                  ON r.podcast_id = p.podcast_id
         GROUP  BY author_id),
     Rating_per_category AS (
SELECT p.title,
       c.category,
       rating_for_this_category,
       rating_podcast
FROM   podcasts p
       JOIN categories c
         ON c.podcast_id = p.podcast_id
       JOIN rating_per_category rpc
         ON rpc.category = c.category
       JOIN rating_per_podcast rpp

         ON rpp.podcast_id = p.podcast_id;
```