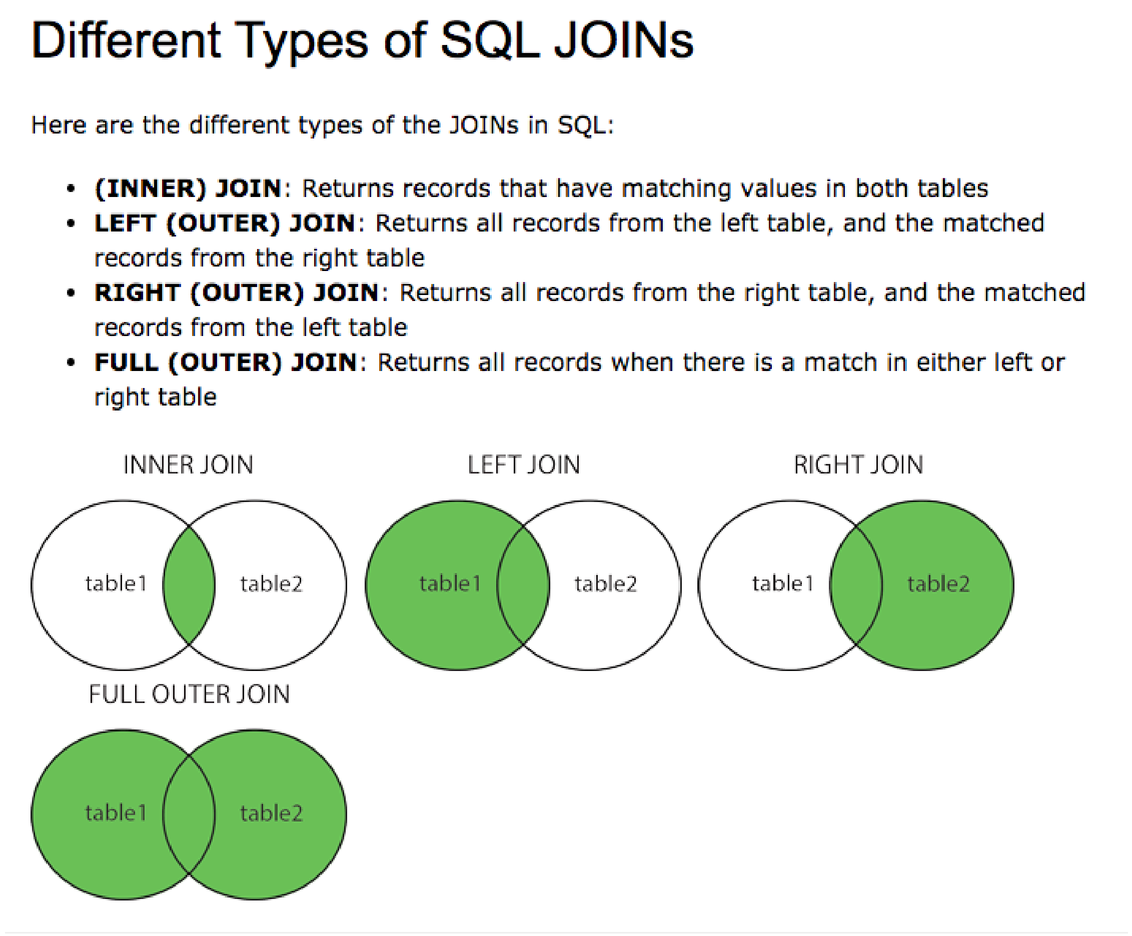
JOINS

// A JOIN clause is used to combine rows from two or more tables, based on a related column between them.



Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

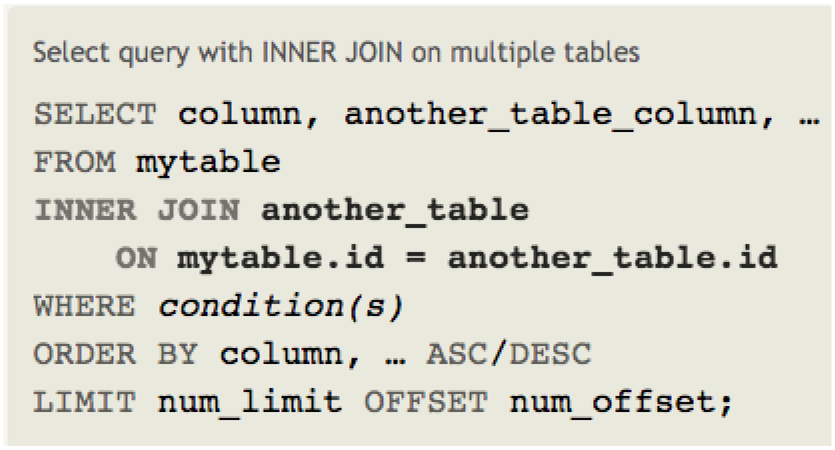
- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table

Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity

Database normalization is useful because it minimizes duplicate data in any single table, and allows for data in the database to grow independently of each other (ie. Types of car engines can grow independent of each type of car).

Tables that share information about a single entity need to have a **primary key** that identifies that entity **uniquely** across the database. One common primary key type is an auto-incrementing integer (because they are space efficient), but it can also be a string, hashed value, so long as it is unique.

Using the **JOIN** clause in a query, we can combine row data across two separate tables using this unique key. The first of the joins that we will introduce is the **INNER JOIN**.


Select query with INNER JOIN on multiple tables

```
SELECT column, another_table_column, …
FROM mytable
INNER JOIN another_table
    ON mytable.id = another_table.id
WHERE condition(s)
ORDER BY column, … ASC/DESC
LIMIT num_limit OFFSET num_offset;
```

The **INNER JOIN** is a process that matches rows from the first table and the second table which have the same key (as defined by the ON constraint) to create a result row with the combined columns from both tables. After the tables are joined, the other clauses we learned previously are then applied.

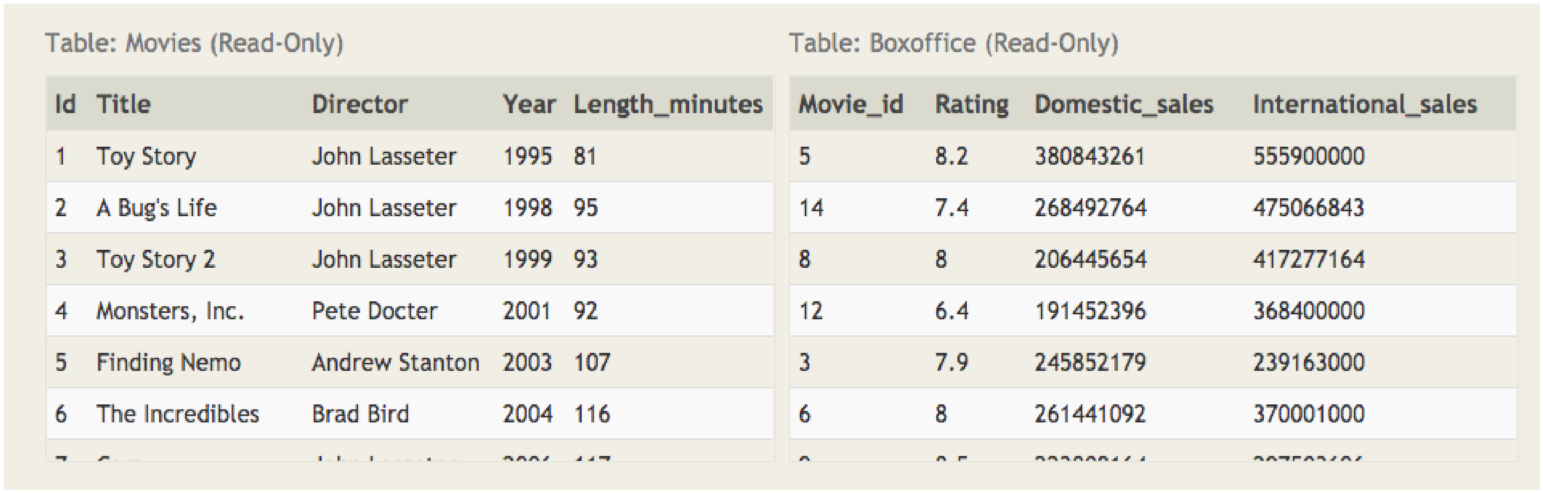


Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000
[illegible]	[illegible]	[illegible]	[illegible]

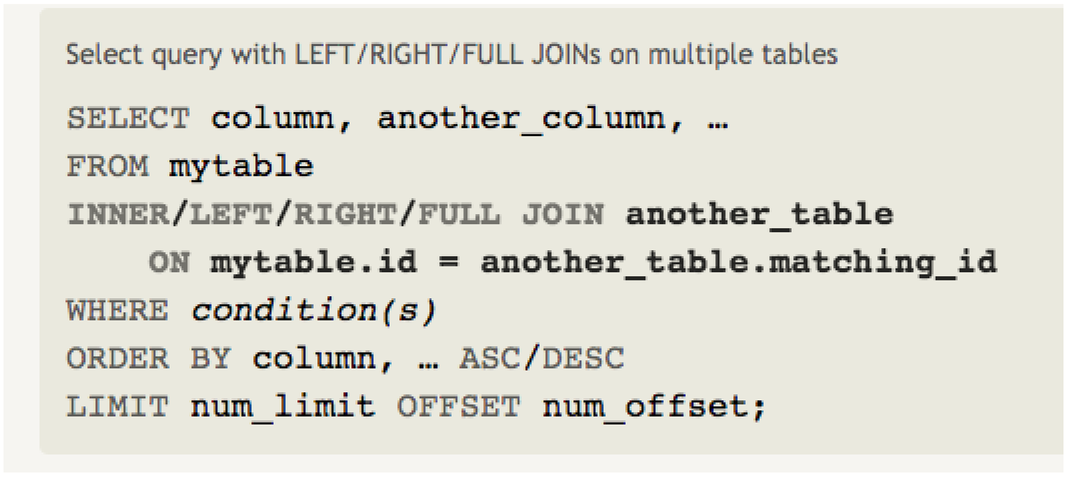
Find the domestic and international sales of each movie

```
SELECT title, domestic_sales, international_sales
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id;
```

Show the sales numbers for each movie that did better internationally than nationally.

```
SELECT title, domestic_sales, international_sales
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id
    where boxoffice.International_sales > boxoffice.Domestic_sales;
```

List all movies based on rating in order
SELECT title, rating
FROM movies
  JOIN boxoffice
    ON movies.id = boxoffice.movie_id
    ORDER BY Boxoffice.Rating DESC;



Select query with LEFT/RIGHT/FULL JOINs on multiple tables

```
SELECT column, another_column, …
FROM mytable
INNER/LEFT/RIGHT/FULL JOIN another_table
    ON mytable.id = another_table.matching_id
WHERE condition(s)
ORDER BY column, … ASC/DESC
LIMIT num_limit OFFSET num_offset;
```

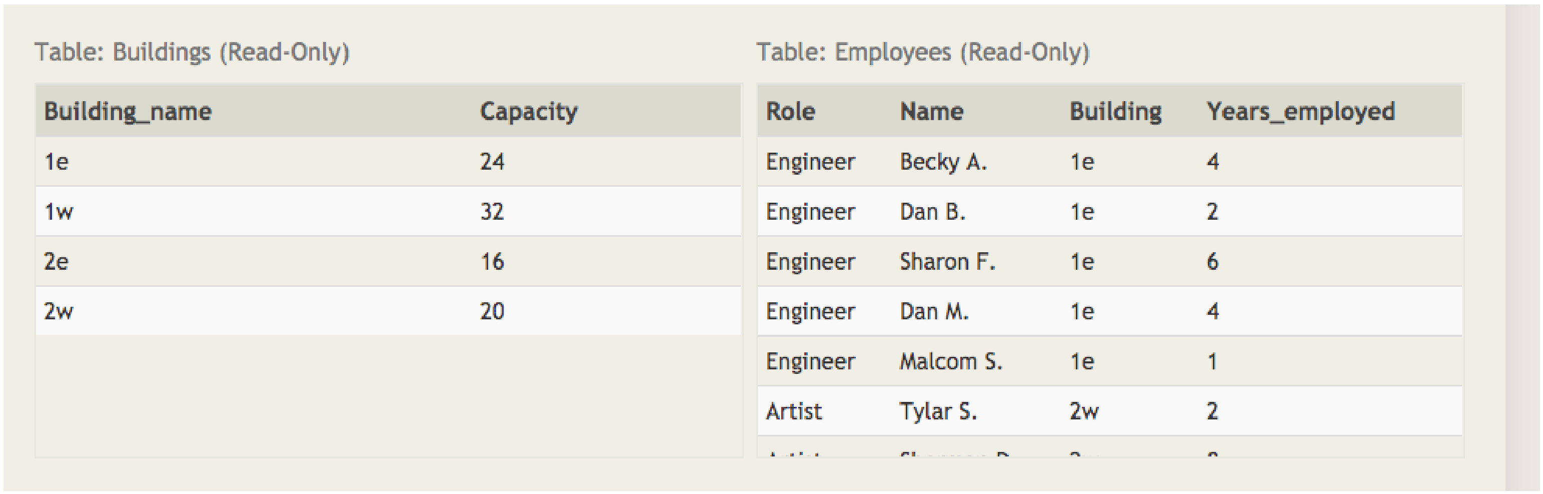


Table: Buildings (Read-Only)

Building_name	Capacity
1e	24
1w	32
2e	16
2w	20

Table: Employees (Read-Only)

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2

List all Buildings and distinct employee roles in each building (including empty buildings)
SELECT DISTINCT building_name, role
FROM Buildings
Left JOIN Employees
ON Buildings.Building_name = Employees.Building;