WEEK 3

SQL

TODAY

- More Labs
- More DB Concepts
  - Foreign Keys
  - Indexes
- Views
  - Joins (inner & left)
- Parent Id Concept

ALTER TABLE

- Command used to modify a database table after created
  - Can add columns
  - Drop columns
  - Modify columns
- Format

```
Alter table [tablename]
[Add] [columnname] OR
[Drop] COLUMN [columnname] OR
[alter/modify] COLUMN  [columnname] [datatype]
```

- Example

```
Alter table products
add product_description varchar(500) null;

alter table products
drop column product_description;
```

Lab
Add new description field to products table
Note: Pay attention to wording!!

DESTRUCTIVE DB COMMANDS: DROP & DELETE

DELETE TABLE DATA

- Can manually delete specific rows or all data in a table
  - delete from [tablename] where [filter]
  - delete from products

DROP TABLE STRUCTURE

- Manually deletes the entire table
- You must delete data from table before dropping
  - drop [tablename]
  - drop products;

FK CONSTRAINTS

- The Foreign Key Constraint maintains data and referential integrity
  - Stops users from accidentally deleting data from the parent table
  - Can also enforce updating parent table when cascade delete \ cascade update

- In most standard DBs alter table products

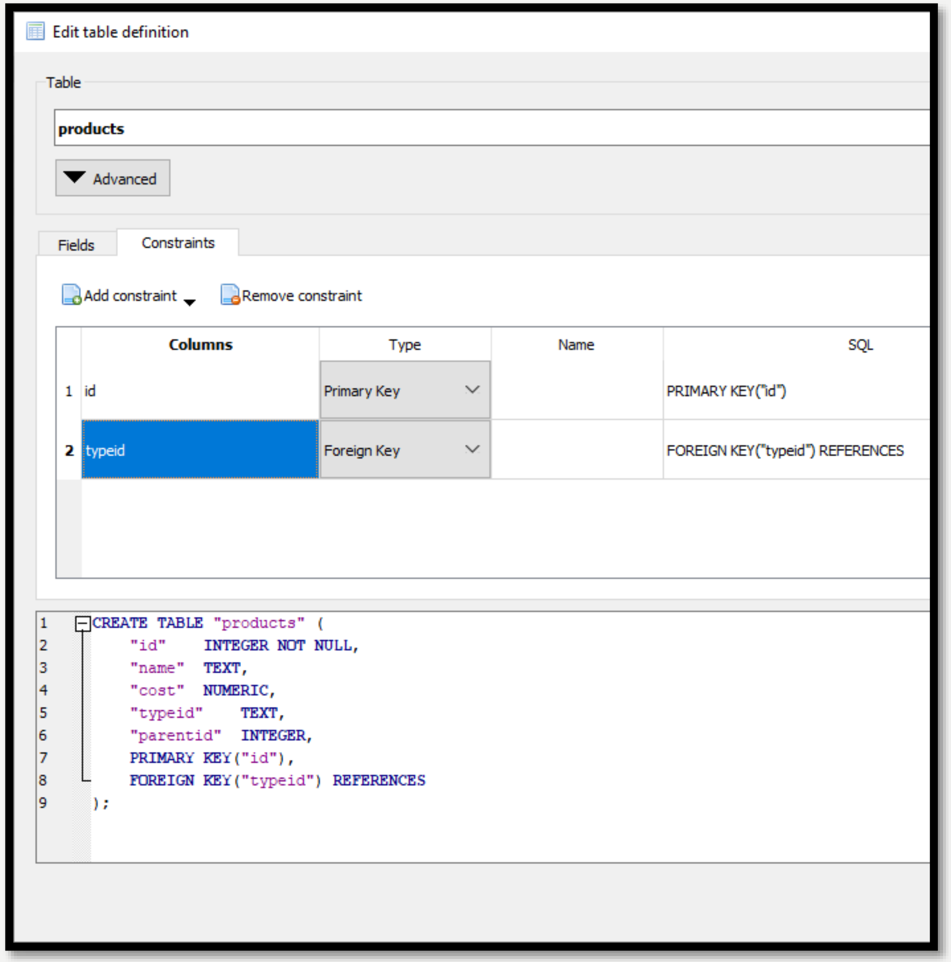
  add constraint fk_typeid

  Foreign key product(typeid) references type(id);

- https://www.w3schools.com/sql/sql_foreignkey.asp

ADDING FK TO ALL TABLES: SQLITE

SQLite does not allow addition of FKs, so you have to drop your tables and then recreate with FKs



products

	Columns	Type	Name	SQL
1	id	Primary Key		PRIMARY KEY("id")
2	typeid	Foreign Key		FOREIGN KEY("typeid") REFERENCES

```
CREATE TABLE "products" (
    "id"    INTEGER NOT NULL,
    "name"  TEXT,
    "cost"  NUMERIC,
    "typeid"    TEXT,
    "parentid"  INTEGER,
    PRIMARY KEY("id"),
    FOREIGN KEY("typeid") REFERENCES
);
```

```
CREATE TABLE "products" (
        "id"  INTEGER NOT NULL,
        "name"     TEXT,
        "cost"      NUMERIC,
        "typeid"    TEXT,
        "parentid"       INTEGER,
        PRIMARY KEY("id"),
        FOREIGN KEY("typeid") REFERENCES
type(id)
);
```

LAB

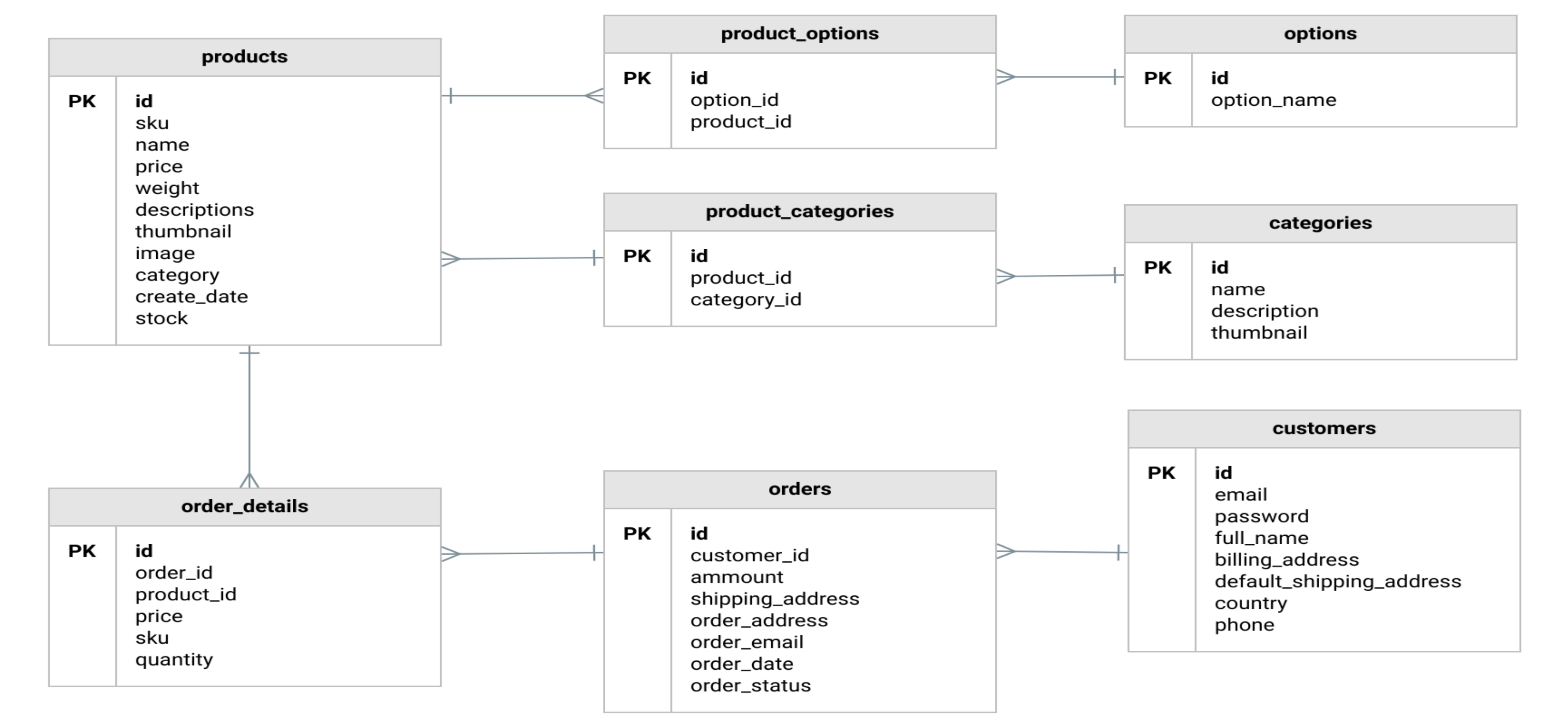
- Export data from the products table
  - Right-click on table, choose export to csv
- Manually Delete data from the products table
  - delete from [tablename]
- Manually Drop the products table
  - drop [tablename]
- Manually recreate products table

```
CREATE TABLE "products" (
        "id"            INTEGER NOT NULL,
        "name"          TEXT,
        "cost"          NUMERIC,
        "typeid"        TEXT,
        "parentid"      INTEGER,
        PRIMARY KEY("id"),
        FOREIGN KEY("typeid") REFERENCES type(id)
);
```

- Import your CSV into products table
  - Click on the table
  - Choose import table from file
  - Choose the file
  - Click columns in first row
  - Import

DATABASE NORMALIZATION

Databases contain numerous discrete tables to avoid data redundancy



products	
PK	**id** sku name price weight descriptions thumbnail image category create_date stock

product_options	
PK	**id** option_id product_id

options	
PK	**id** option_name

product_categories	
PK	**id** product_id category_id

categories	
PK	**id** name description thumbnail

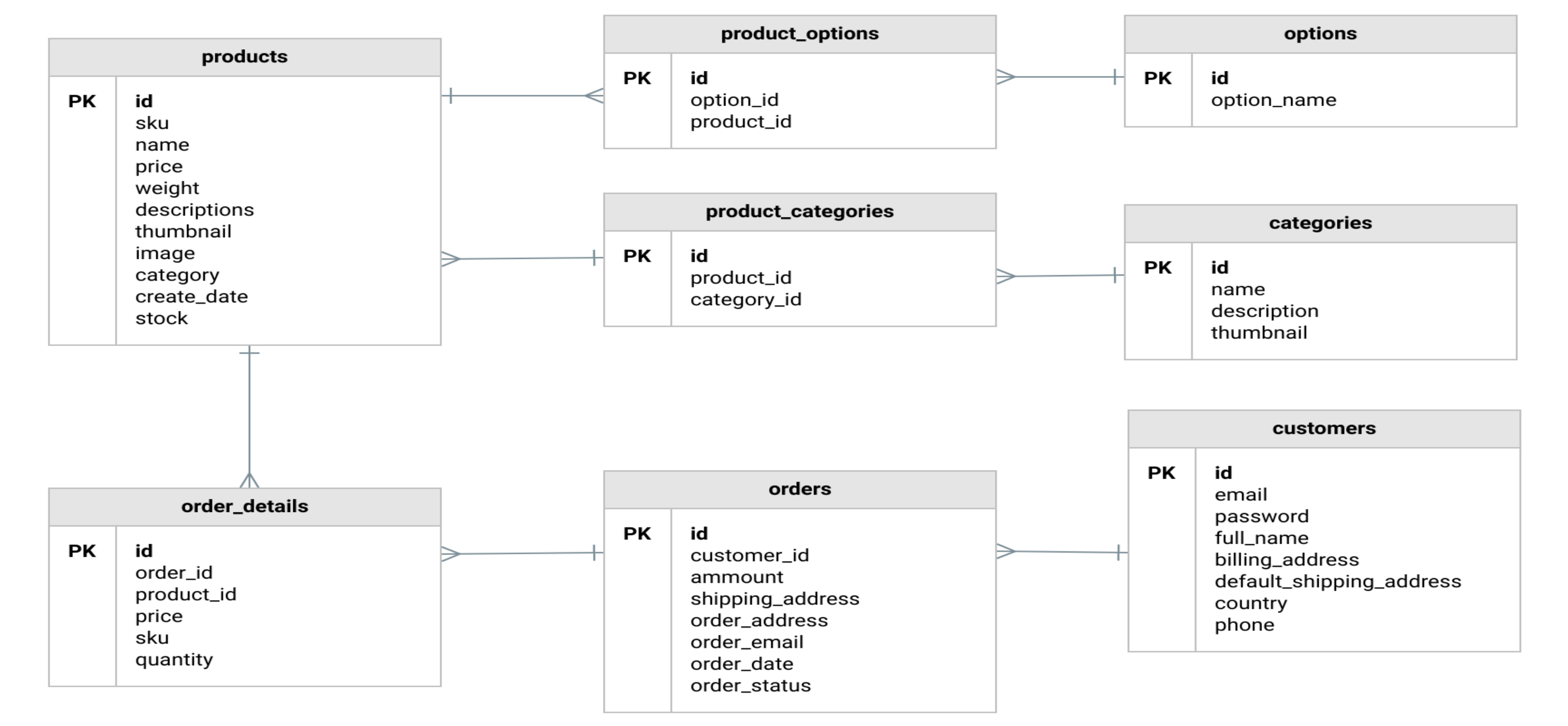
customers	
PK	**id** email password full_name billing_address default_shipping_address country phone

order_details	
PK	**id** order_id product_id price sku quantity

orders	
PK	**id** customer_id ammount shipping_address order_address order_email order_date order_status

DB RELATIONSHIPS

- Tables eventually connect back to each other through foreign keys



products	
PK	**id** sku name price weight descriptions thumbnail image category create_date stock

product_options	
PK	**id** option_id product_id

options	
PK	**id** option_name

product_categories	
PK	**id** product_id category_id

categories	
PK	**id** name description thumbnail

customers	
PK	**id** email password full_name billing_address default_shipping_address country phone

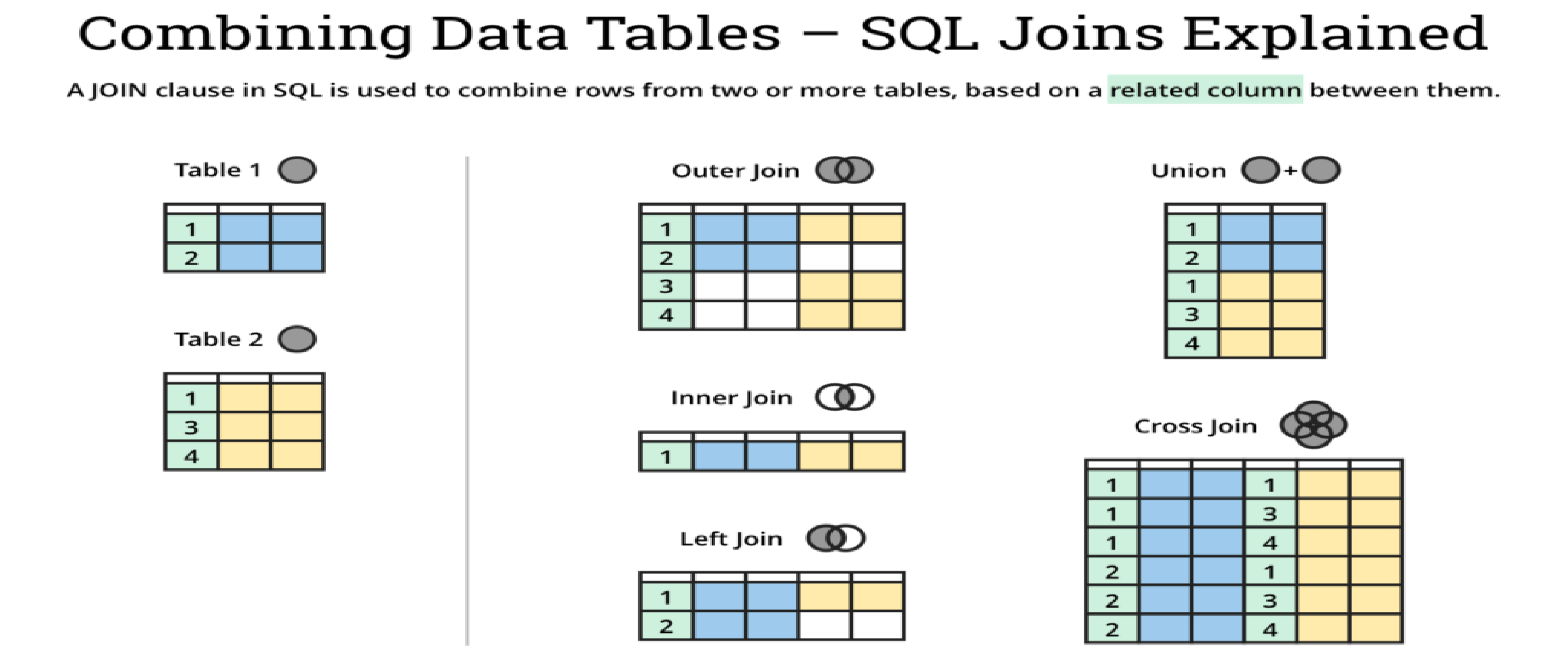
order_details	
PK	**id** order_id product_id price sku quantity

orders	
PK	**id** customer_id ammount shipping_address order_address order_email order_date order_status

- https://dataschool.com/how-to-teach-people-sql/inner-join-animated/

DB RELATIONSHIPS

- We use SQL Queries to reflect the relationships between the different tables
- Joins are queries that connect tables together through ids
- And different joins bring out different data connections
- Joins are what are used to generate reports



Combining Data Tables – SQL Joins Explained

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

Table 1

1	🟦	🟦
2	🟦	🟦

Table 2

1	🟨	🟨
3	🟨	🟨
4	🟨	🟨

Outer Join

1	🟦	🟦	🟨	🟨
2	🟦	🟦		
3			🟨	🟨
4			🟨	🟨

Inner Join

1	🟦	🟦	🟨	🟨

Left Join

1	🟦	🟦	🟨	🟨
2	🟦	🟦		

Union +

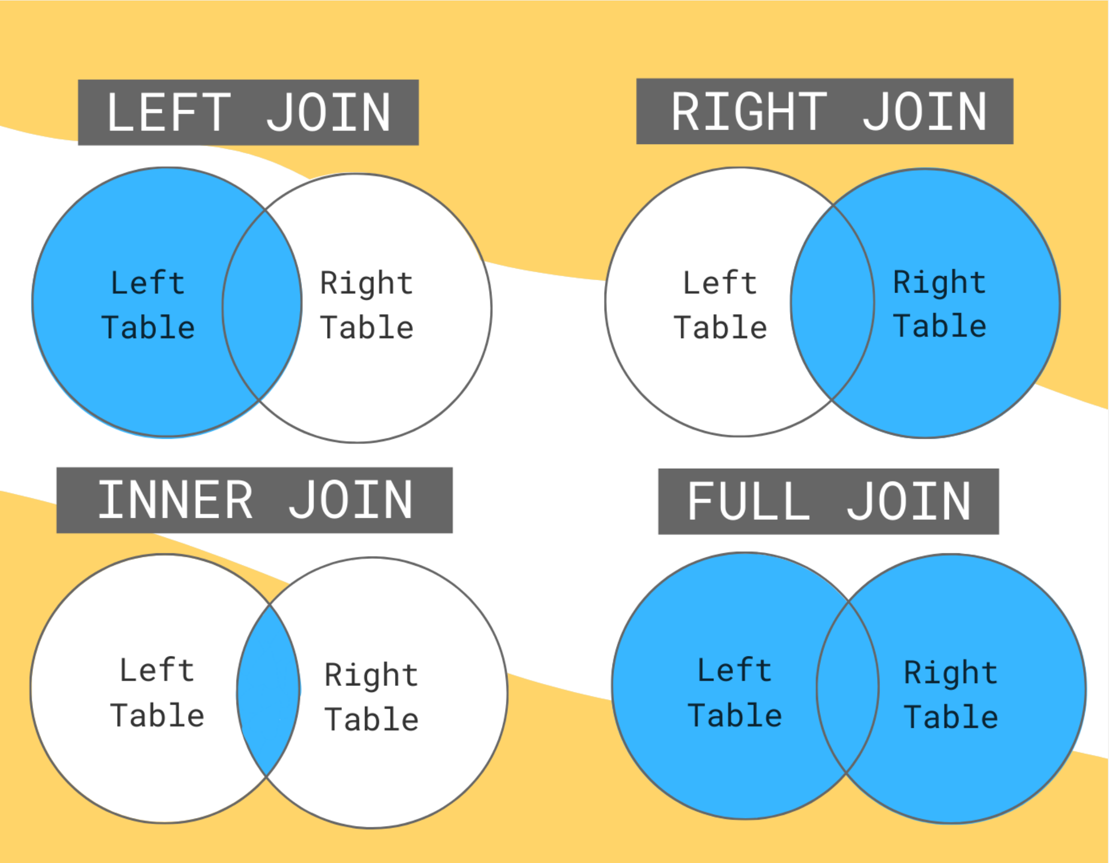
1	🟦	🟦
2	🟦	🟦
1	🟨	🟨
3	🟨	🟨
4	🟨	🟨

Cross Join

1	🟦	🟦	1	🟨	🟨
1	🟦	🟦	3	🟨	🟨
1	🟦	🟦	4	🟨	🟨
2	🟦	🟦	1	🟨	🟨
2	🟦	🟦	3	🟨	🟨
2	🟦	🟦	4	🟨	🟨

- https://dataschool.com/how-to-teach-people-sql/inner-join-animated/

JOIN TYPES



https://learnsql.com/blog/learn-and-practice-sql-joins/

```
SELECT product.name, type.name
FROM product
INNER JOIN type
ON
product.typeid = type. id

SELECT
product.name, type.name
FROM product
RIGHT JOIN type
ON
product.typeid = type. id

SELECT
product.name, type.name
FROM product
LEFT JOIN type
ON
product.typeid = type. id
```

LAB: EXPAND OUR DB MODEL

- Create a users table
  - Fields: id, username, email, firstname, lastname
- Create an orders table
  - Fields: id, userid, productid, amount

VIEWS

- A stored query
- Reusable
- Great for reporting
- Only retrieves data
- Great for complex queries
- Sometimes faster than a unique query*

To Create a View

- CREATE VIEW [viewname] as
  [Standard SQL i.e. select * from dogs]

To Drop\Delete a View

- drop view [viewname]