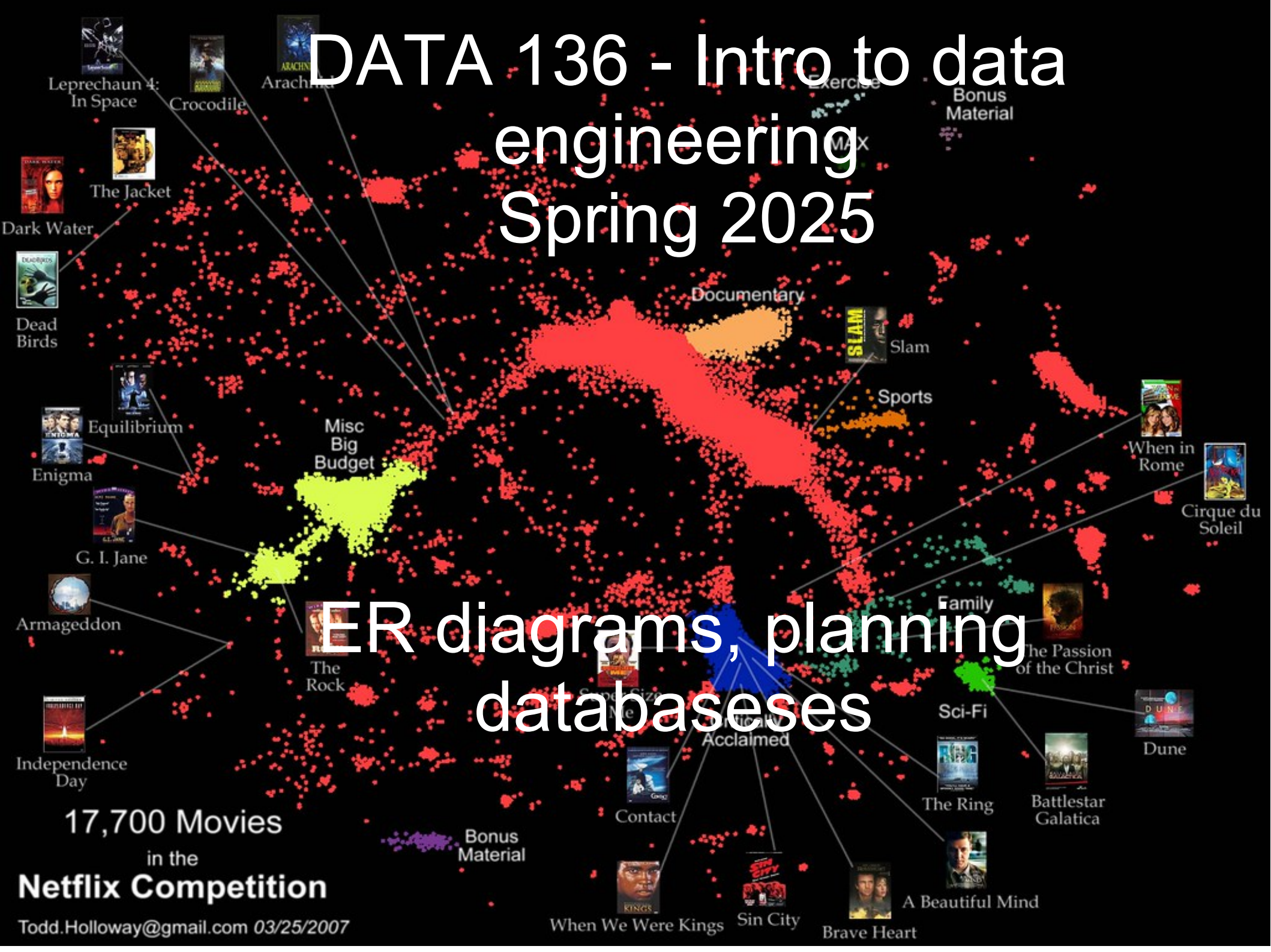
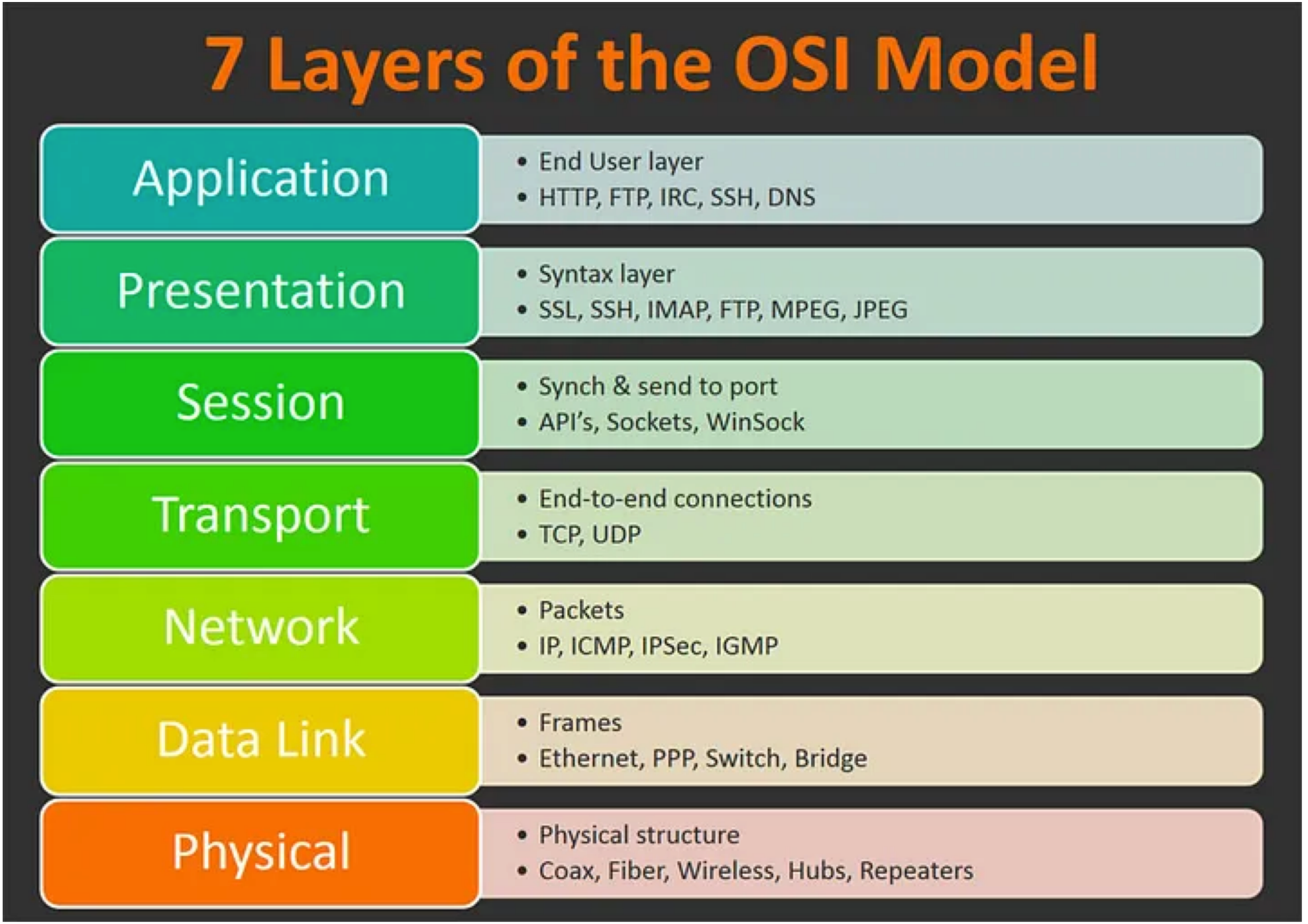

DATA 136 - Intro to data engineering
Spring 2025
ER diagrams, planning databaseses
Leprechaun 4: In Space
Crocodile
Arachnid
Exercise
Bonus Material
The Jacket
Dark Water
Dead Birds
Documentary
Slam
Sports
Equilibrium
Misc Big Budget
Enigma
When in Rome
Cirque du Soleil
G. I. Jane
Armageddon
The Rock
Family
The Passion of the Christ
Sci-Fi
Dune
Independence Day
Acclaimed
Contact
The Ring
Battlestar Galatica
17,700 Movies
in the
Netflix Competition
Bonus Material
When We Were Kings
Sin City
Brave Heart
A Beautiful Mind
Todd.Holloway@gmail.com 03/25/2007

- A **data model** defines how different pieces of digital data are organized, defines how they relate to one another, and how these representations correspond to the properties of real-world entities.

- A **data dictionary** is essentially the same thing, but packaged as metadata.

Open Systems Interconnection



Data models

- Logical data model: A model that describes the semantics of the data, as represented by the particular data analysis technology.

- Physical data model : describes the physical means by which data are stored. This is concerned with partitions, CPUs, bits, data types, and so on.

- As history marches on, these drift apart from each other. We don't worry about physical representation (bolts and bytes) unless/until we encounter a relevant resource constraint.

Appendix A: Record Layout for Public Library System Data File, FY 2022 (PLS_FY22_AE_pud22i)

Note: See **Appendix G** for definitions of flag variables (F_*).

Variable name	Field length	Data type	Survey item	Description
				Data Source: Public Libraries Survey, Fiscal Year 2022 **Number of records = 9,248 (one record per observation)** **Number of fields per record = 192**
				IDENTIFICATION
STABR	02	A	†	Two-letter American National Standards Institute (ANSI) State Code. (See **Appendix D** for list of State Codes.)
FSCSKEY	06	A	150	Library identification code assigned by IMLS
LIBID	20	A	151	Library identification code assigned by the state. IMLS assigns the FSCSKEY to this field if the state did not assign a code.
LIBNAME	60	A	152	Name of library (administrative entity)
				STREET ADDRESS
ADDRESS	35	A	153	Street address of administrative entity
CITY	20	A	154	City or town (of street address) of administrative entity
ZIP	05	A	155	Standard five-digit postal zip code (of street address) of administrative entity.
ZIP4	04	A	†	Four-digit postal zip code extension (of street address) of administrative entity. M–Missing
				MAILING ADDRESS
ADDRES_M	35	A	157	Mailing address of administrative entity
CITY_M	20	A	158	City or town (of mailing address) of administrative entity
ZIP_M	05	A	159	Standard five-digit postal zip code (of mailing address) of administrative entity

Example: design a database to manage orders for a retailer

- Imagine that you need to write programs to interact with the retailer; you'll need to do stuff.

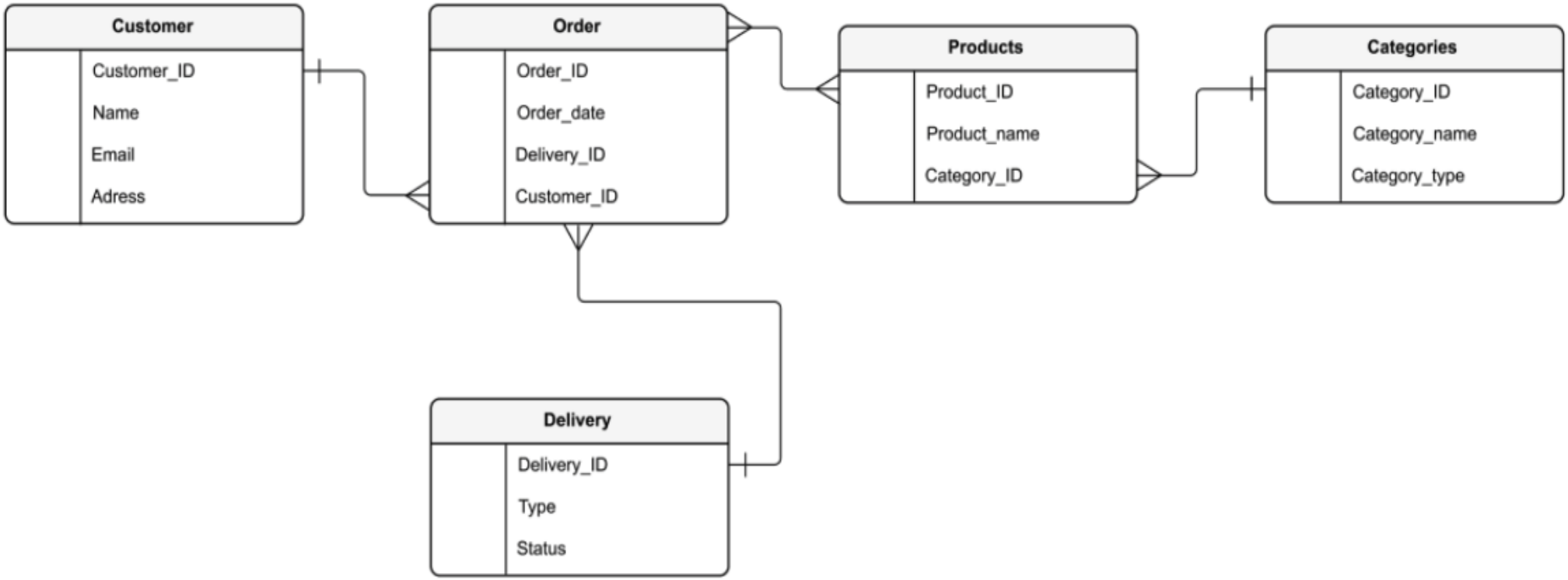
- What do you need to store? (Data about customers, data about products, data about orders..)

- What functions do you want to specify? We can write down what the functions should do before we start hacking...

Application Programming Interface

- Code that does stuff
- API codifies communication between the user or program using the API and the engine producing the API results.
- Documentation and examples show you what inputs are required, permitted, how can the behavior of the program be modified.
- We can talk about bash, pandas, pytorch, amazon S3, instagram, podcast publishers...
- If you aren't running the server running the API, usually not free -> authentication

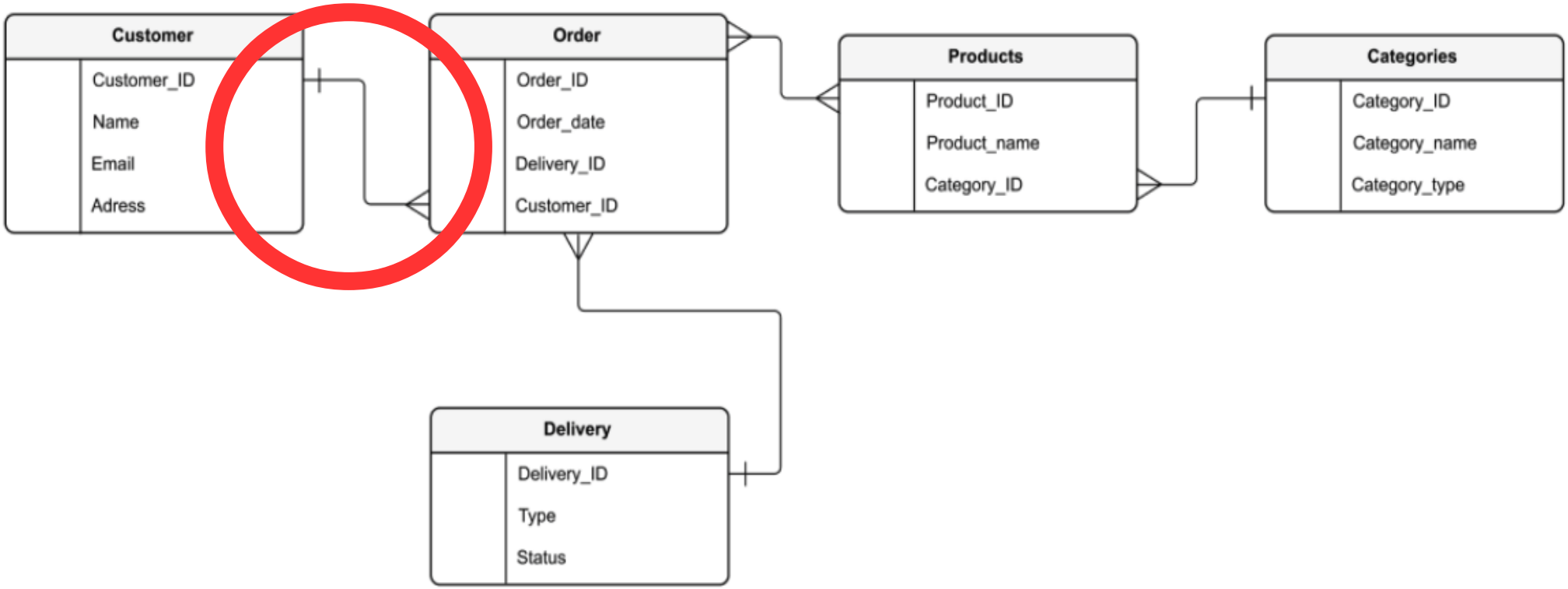
Entity-relation diagrams



Entities (rectangles): entities represent real-world objects or concepts, such as customers, products, employees, interactions...
Attributes: (rows) properties of entities
Relationships: represent associations or connections between entities.
Can be one-to-one, one-to-many, many-to-one, or many-to-many

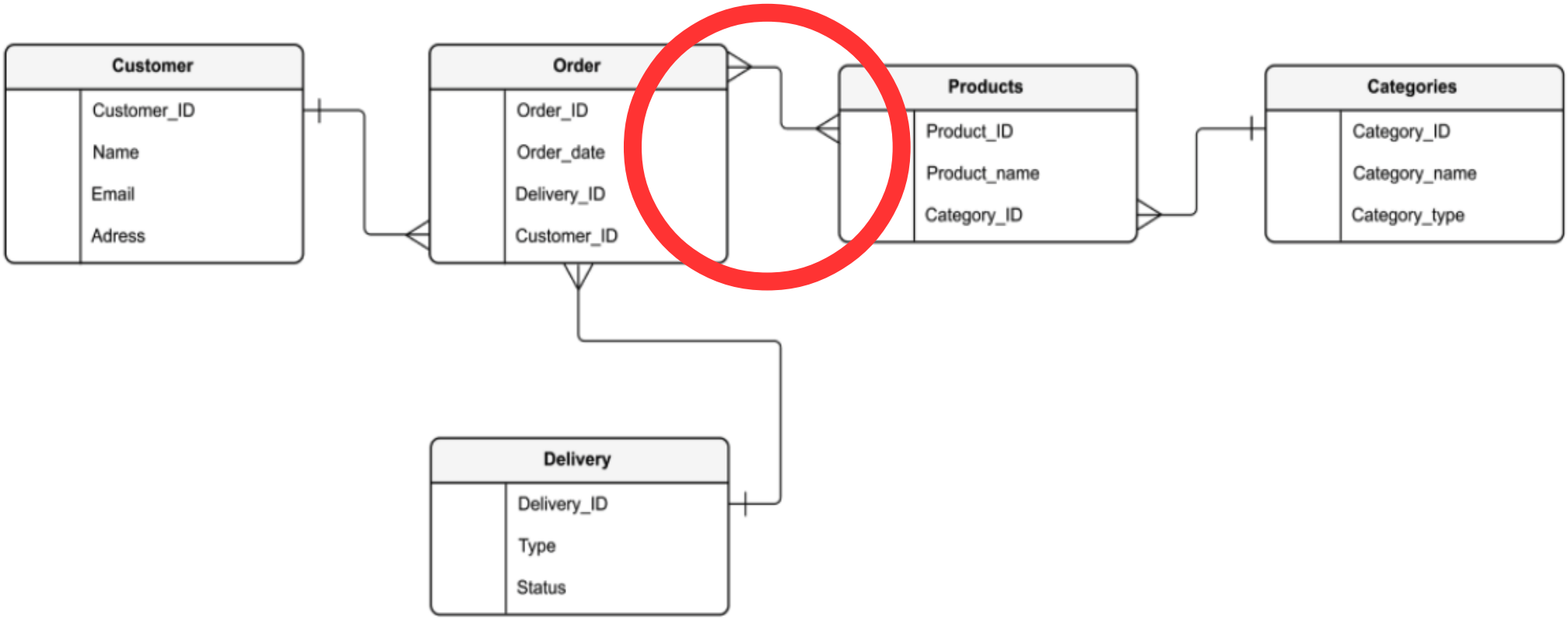
Entity-relation diagrams

One-to-many relationship



Entity-relation diagrams

Many-to-many relationship



Many-to-many

OrderID	Product	CustomerID	Qty
1	Insect repellent	1	1
1	Bug deflector	1	1
2	Diffraction grating	2	25
3	Earbuds	1	6

- Which fields can be determined from which other fields?

Many-to-many

OrderID	Product	CustomerID	Qty
1	Insect repellent	1	1
1	Bug deflector	1	1
2	Diffraction grating	2	25
3	Earbuds	1	6

- Where is the primary key?

Many-to-many

OrderID	Product	CustomerID	Qty
1	Insect repellent	1	1
1	Bug deflector	1	1
2	Diffraction grating	2	25
3	Earbuds	1	6

- Suppose I need to organize products by product category...

Many-to-many

OrderID	Product	CustomerID	Qty
1	Insect repellent	1	1
1	Bug deflector	1	1
2	Diffraction grating	2	25
3	Earbuds	1	6

Product	Category
Bug Deflector	Cooking
Insect Repellent	Cooking
Earbuds	Halloween Costumes
Diffraction grating	Mental Health
Shame hat	Books

Why not keep category in the orders table?

OrderID	Product	CustomerID	Qty	Category
1	Insect repellent	1	1	Cooking
1	Bug deflector	1	1	Cooking
2	Diffraction grating	2	25	Mental Health
3	Earbuds	1	6	Halloween

Why not keep category in the orders table?

OrderID	Product	CustomerID	Qty	Category
1	Insect repellent	1	1	Cooking
1	Bug deflector	1	1	Cooking
2	Diffraction grating	2	25	Mental Health
3	Earbuds	1	6	Halloween

Where / how to store category if there are no (historical) orders?

Updating table: should category ever split, update requires multiple rows.

Why not keep category in the orders table?

OrderID	Product	CustomerID	Qty	Category
1	Insect repellent	1	1	Cooking
1	Bug deflector	1	1	Cooking
2	Diffraction grating	2	25	Mental Health
3	Earbuds	1	6	Halloween

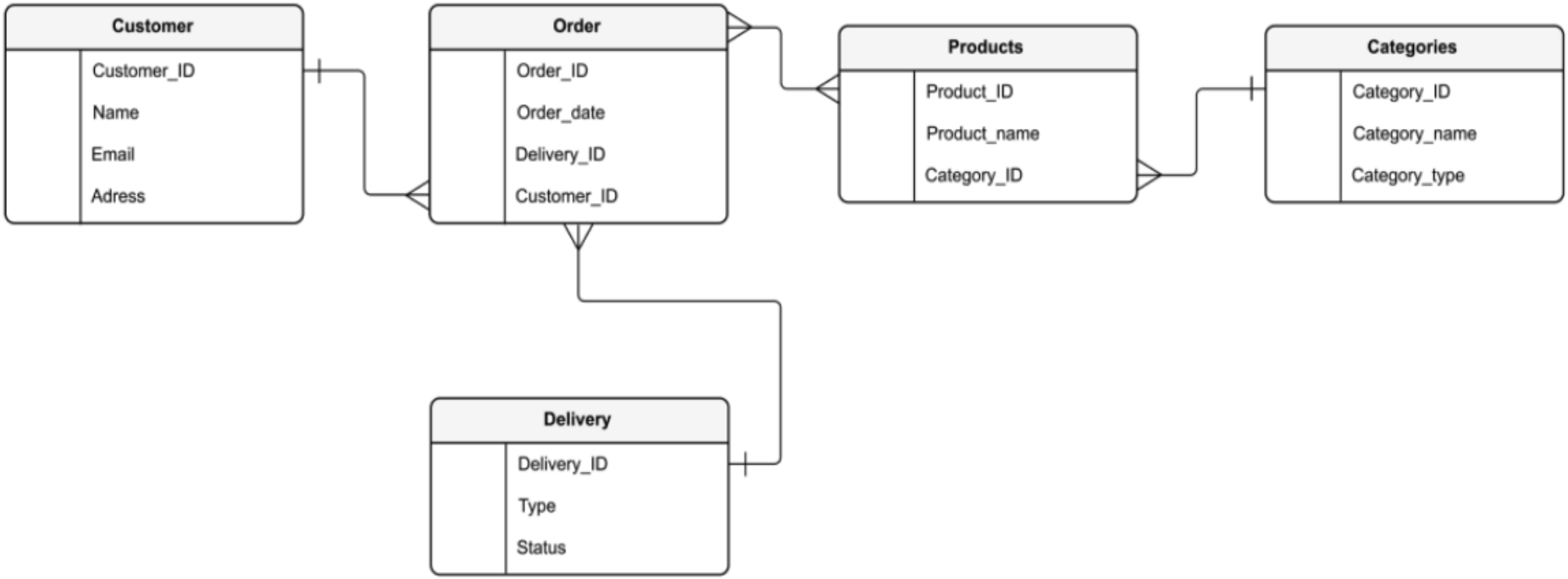
Functionally determined:

A determines B

When B is distinct, A is always distinct.

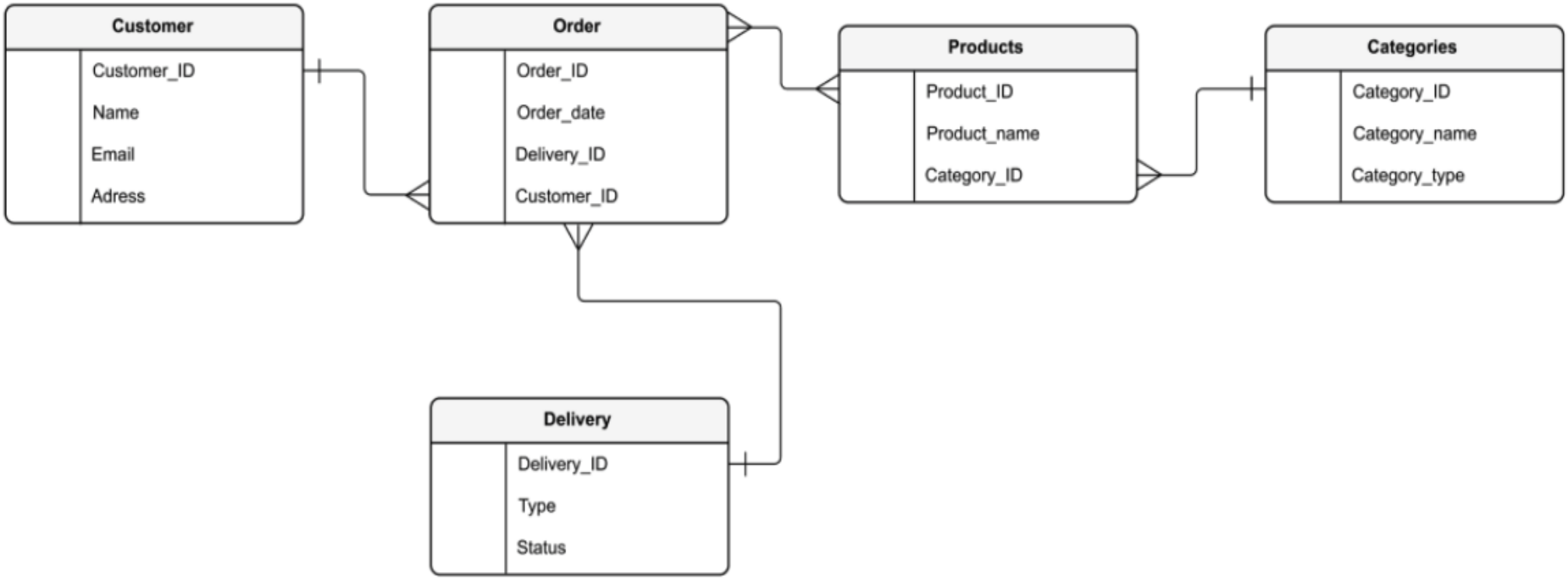
Order number functionally determines customer number:
Different customers always have different order numbers

Entity-relation diagrams



Why no one-to-one relationships in this diagram?

Entity-relation diagrams



Presence of foreign key constraint does not tell you if relationship is many-to-many, one-to-many, or one-to-one.

Ok, let's make a schema

This schema is a means to an end; the schema defines the relational DB relationships; the relational DB will store all the data for our API.

```
CREATE TABLE Orders (
    OrderID int,
    CustomerID int,
    Product varchar(255),
    ShippingInst varchar(255),
    Qty int
);
```

```
CREATE TABLE Customers (
     CustomerID int,
     LastName varchar(255),
     FirstName varchar(255),
     Address varchar(255),
     City varchar(255)
);
```

```
CREATE TABLE Product (
        Product varchar(255),
        Price float,
        Barcode int
);
```

```
CREATE TABLE Category (
        Product varchar(255),
        Category varchar(255),
)
```

Ok, let's make a schema

This schema is a means to an end; the schema defines the relational DB relationships; the relational DB will store all the data for our API.

```
CREATE TABLE Orders (
    OrderID int,
    CustomerID  int,
    Product  varchar(255),
    ShippingInst varchar(255),
    Qty  int
);
```

```
CREATE TABLE Customers (
     CustomerID  int,
     LastName varchar(255),
     FirstName  varchar(255),
    Address varchar(255),
    City  varchar(255)
);
```

```
CREATE TABLE Product (
        Product  varchar(255),
        Price float,
        Barcode int
);
```

```
CREATE TABLE Category (
        Product  varchar(255),
        Category varchar(255),
)
```

Ok, let's make a schema

This schema is a means to an end; the schema defines the relational DB relationships; the relational DB will store all the data for our API.

```
CREATE TABLE Orders (
    OrderID int,
    CustomerID  int FOREIGN KEY references Customers,
    Product  varchar(255) FOREIGN KEY references Product,
    ShippingInst varchar(255),
    Qty  int
);
```

```
CREATE TABLE Customers (
     CustomerID,
     LastName varchar(255),
     FirstName  varchar(255),
    Address varchar(255),
    City  varchar(255)
);
```

```
CREATE TABLE Product (
        Product  varchar(255),
        Price float,
        Barcode int
);
```

```
CREATE TABLE Category (
        Product  varchar(255)
FOREIGN KEY references Product,
        Category varchar(255),
)
```

```
CREATE TABLE book (
  isbn varchar(8),
  title varchar(256),
  author varchar(256),
  year int,
  PRIMARY KEY(ISBN))

CREATE TABLE inventory (
  id int,
  inv_isbn varchar(8),
  year_acquired int,
  borrowed bool,
  PRIMARY KEY(id),
  FOREIGN KEY (inv_isbn) REFERENCES book(isbn))'
```

Draw the ER diagram for these two tables

```
CREATE TABLE book (
  isbn varchar(8),
  title varchar(256),
  author varchar(256),
  year int,
  PRIMARY KEY(ISBN))

CREATE TABLE inventory (
  id int,
  inv_isbn varchar(8),
  year_acquired int,
  borrowed bool,
  PRIMARY KEY(id),
  FOREIGN KEY (inv_isbn) REFERENCES book(isbn))'
```

Explain this SQL query in English

```
SELECT count(*)
FROM inventory
WHERE borrowed;
```

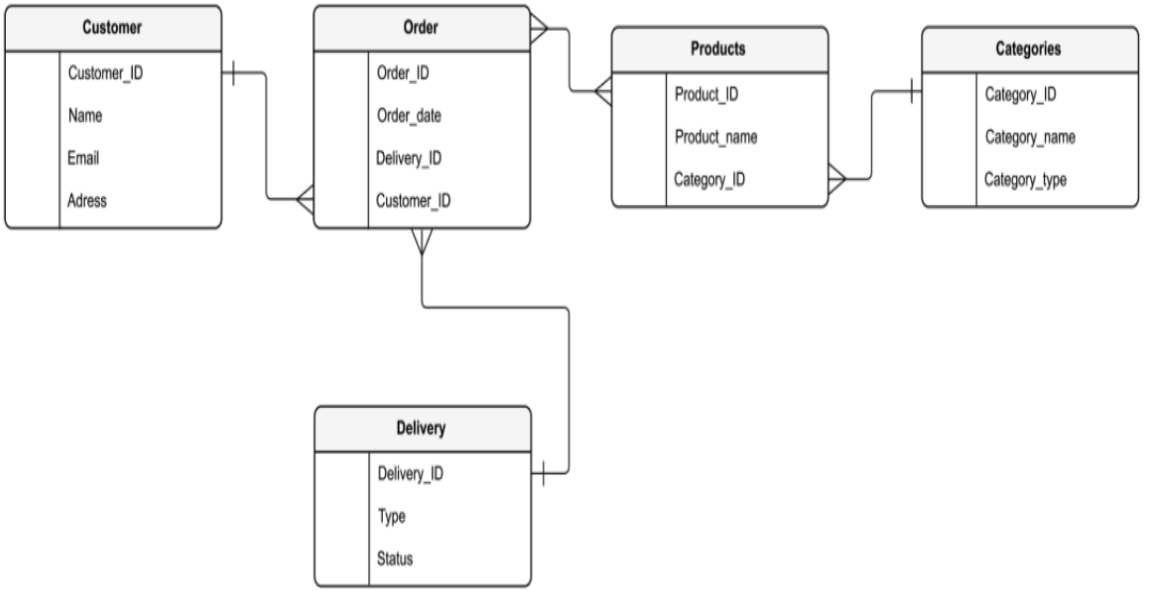
```
CREATE TABLE book (
  isbn varchar(8),
  title varchar(256),
  author varchar(256),
  year int,
  PRIMARY KEY(ISBN))

CREATE TABLE inventory (
  id int,
  inv_isbn varchar(8),
  year_acquired int,
  borrowed bool,
  PRIMARY KEY(id),
  FOREIGN KEY (inv_isbn) REFERENCES book(isbn))'
```

Explain this SQL query in English

```
SELECT count(*)
FROM book, inventory
WHERE book.isbn = inventory.inv_isbn AND
      borrowed AND
      book.author = 'Morrison, Toni'
```

Now, what do we need for our API?



Specifying your own API

What are “actions” that your application needs to implement?

```
CreateUser(firstName, lastName, address) -> userid

CreateOrder(userid, productid, qty) -> orderid

CreateInvoice(orderid)

...

AddProduct(...)

DeleteUser(...)

DeleteOrder(...)

CloseOrder(...)
```