Get clickers today!
CREATE TABLE Product (  
id INT,
name VARCHAR(50) NOT NULL,
company_ID VARCHAR(50) NOT NULL,
units_sold INT DEFAULT 0,
PRIMARY KEY (id),
FOREIGN KEY (company_id) REFERENCES Company (id)  
);
Can name key constraints

CREATE TABLE Product (
  id INT,
  name VARCHAR(50) NOT NULL,
  company_ID VARCHAR(50) NOT NULL,
  units_sold INT DEFAULT 0,
  CONSTRAINT pk_productid PRIMARY KEY (id),
  CONSTRAINT fk_companyid FOREIGN KEY (company_id) REFERENCES Company (id)
);
Other key constraints

Companies wouldn’t give multiple of their own products the same name.

```sql
CREATE TABLE Product ( 
    id INT, 
    name VARCHAR(50) NOT NULL, 
    company_ID VARCHAR(50) NOT NULL, 
    units_sold INT DEFAULT 0, 
    CONSTRAINT pk_productid PRIMARY KEY (id), 
    CONSTRAINT un_nameco UNIQUE (name, company_ID), 
    CONSTRAINT fk_companyid FOREIGN KEY (company_id) REFERENCES Company (id) 
);
```

Combination of fields must be UNIQUE. But, unlike PRIMARY KEY, allows them to be NULL.
Why would we want UNIQUE?

Remember: Tables/relations can have multiple keys. Only one key is primary.

One common usage: Some advocate that all tables use a synthetic key, even though this breaks BCNF. The synthetic key would be PRIMARY KEY. The natural key(s) would be UNIQUE.
Adding constraints

ALTER TABLE Product
ADD PRIMARY KEY (id);

ALTER TABLE Product
ADD CONSTRAINT pk_productid PRIMARY KEY (id);

ALTER TABLE Product
MODIFY name VARCHAR(50) NOT NULL;
Deleting constraints

Most:

\[
\text{ALTER TABLE Product DROP CONSTRAINT pk_productid;}
\]

Some:

\[
\text{ALTER TABLE Product DROP PRIMARY KEY;}
\]

Some:

N/A
General constraint for a single attribute

CREATE TABLE Product (  
  ...
  price DECIMAL (10,2) CHECK (price > 0),
  ...
);  

CREATE TABLE Person (  
  ...
  CONSTRAINT chk_gender CHECK (gender IN (‘M’, ‘F’)),
  ...
);
General constraint for multiple attributes

CREATE TABLE Student (  
    ...  
    CONSTRAINT chk_dates CHECK (matriculation_date < graduation_date),  
    ...  
);  

Must be attributes of same table.
Which can constrain to finite set?

A. Lookup table
B. CHECK
C. Both

Correct answer: C. Both

Response Counter:

- Lookup table: 33%
- CHECK: 33%
- Both: 33%
Which can constrain to infinite range?

A. Lookup table
B. CHECK
C. Both
Lookup table vs. CHECK on finite sets

Lookup table more maintainable:
• Valid set can be user-maintained
• Can use constraint on multiple attributes without duplicating data

Lookup table more usable:
• Can grow with additional attributes
• Data usable in query – e.g., summarize other data for each value

Note: Having lots of little lookup tables is not a concern.

CHECK is generally faster.

Use CHECK only for small, unshared sets that won’t change.
Lookup table vs. CHECK – examples to discuss

- Rice colleges
- State abbreviations
- ZIP codes
- SQL keywords
- Client type
- Genders
Constraining attributes from multiple tables

• Might signal that attributes should be put in the same table.

• Most straight-forward solution: CHECK constraint in one table that uses a user-defined function that accesses other tables.
Constraints can be misused

CHECK sometimes misused to ensure consistency of redundant columns, instead of calculating some data
Activity – Add & verify constraints

13-constraints.ipynb