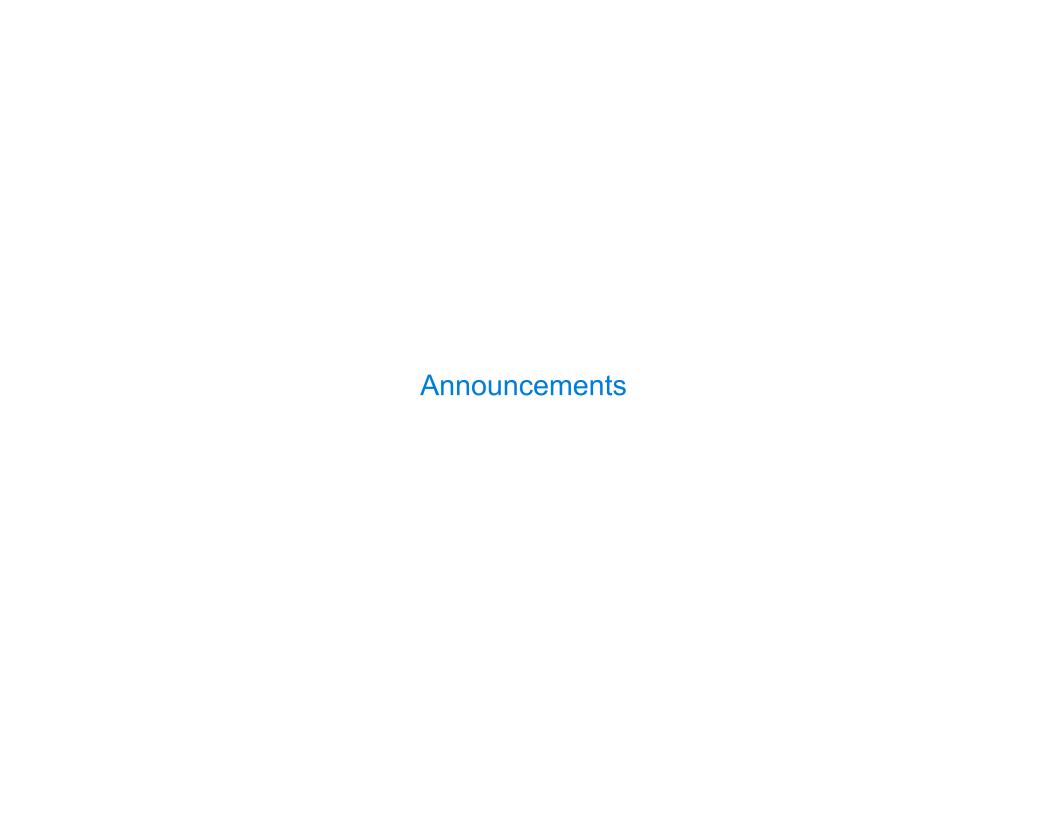
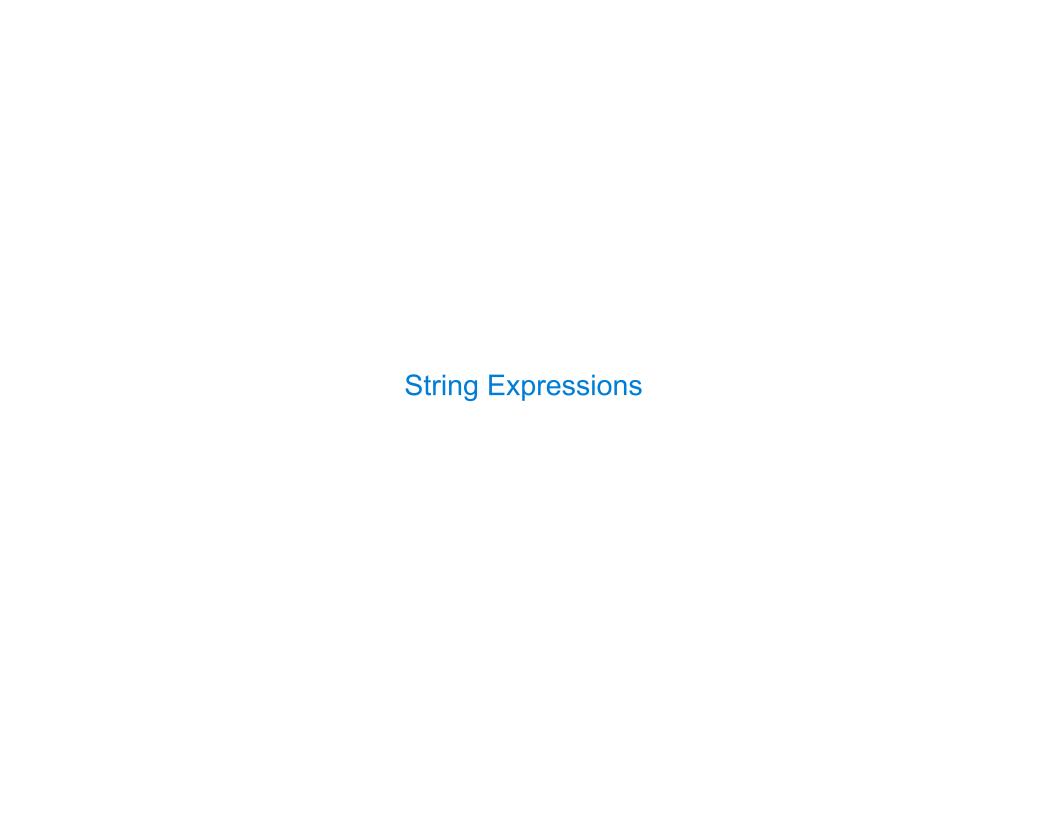
| Tables | | | |
|--------|--|--|--|
| | | | |





String Expressions

String values can be combined to form longer strings



sqlite> SELECT "hello," || " world";
hello, world

Basic string manipulation is built into SQL, but differs from Python



sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;
low

Strings can be used to represent structured values, but doing so is rarely a good idea



Ice Cream Cones

Let's revisit the ice cream shop from DATA 8...

```
CREATE TABLE cones AS

SELECT 1 AS Id, 'strawberry' AS Flavor, 'pink' AS Color, 3.75 AS Price UNION

SELECT 2,'chocolate', 'light brown', 4.75 UNION

SELECT 3, 'chocolate', 'dark brown', 5.25 UNION

SELECT 4, 'strawberry', 'pink', 5.5 UNION

SELECT 5, 'bubblegum', 'pink', 4.75 UNION

SELECT 7, 'Vanilla', 'white', 4.20 UNION

SELECT 8, 'Mint Chocolate', 'green', 3.95 UNION

SELECT 9, 'Fancy Mint Chocolate', 'green', 5.95;

CREATE TABLE sales AS

SELECT 'Baskin' AS Cashier, 1 AS id, 2 AS cone_id UNION

SELECT 'Baskin', 3, 1 UNION

SELECT 'Baskin', 4, 2 UNION

SELECT 'Robin', 2, 3 UNION

SELECT 'Robin', 5, 2 UNION

SELECT 'Robin', 6, 1;
```

Filtering rows - where

Set of Table records (rows) that satisfy a condition

```
select [columns] from [table] where [condition] order by [order];
```

```
In [5]: cones.select(['Flavor', 'Price'])

Dut[5]: Flavor Price

strawberry 3.55

chocolate 4.75

chocolate 5.25

strawberry 5.25

bubblegum 4.75

chocolate 5.25
```

```
sqlite> select * from cones where Flavor = "chocolate";
ID|Flavor|Color|Price
2|chocolate|light brown|4.75
3|chocolate|dark brown|5.25
6|chocolate|dark brown|5.25
```

```
cones.where(cones["Price"] > 5)

ID Flavor Color Price
3 chocolate dark brown 5.25
4 strawberry pink 5.25
6 chocolate dark brown 5.25

SQL:

sqlite> select * from cones where Price > 5;
ID|Flavor|Color|Price
3|chocolate|dark brown|5.25
4|strawberry|pink|5.25
6|chocolate|dark brown|5.25
```

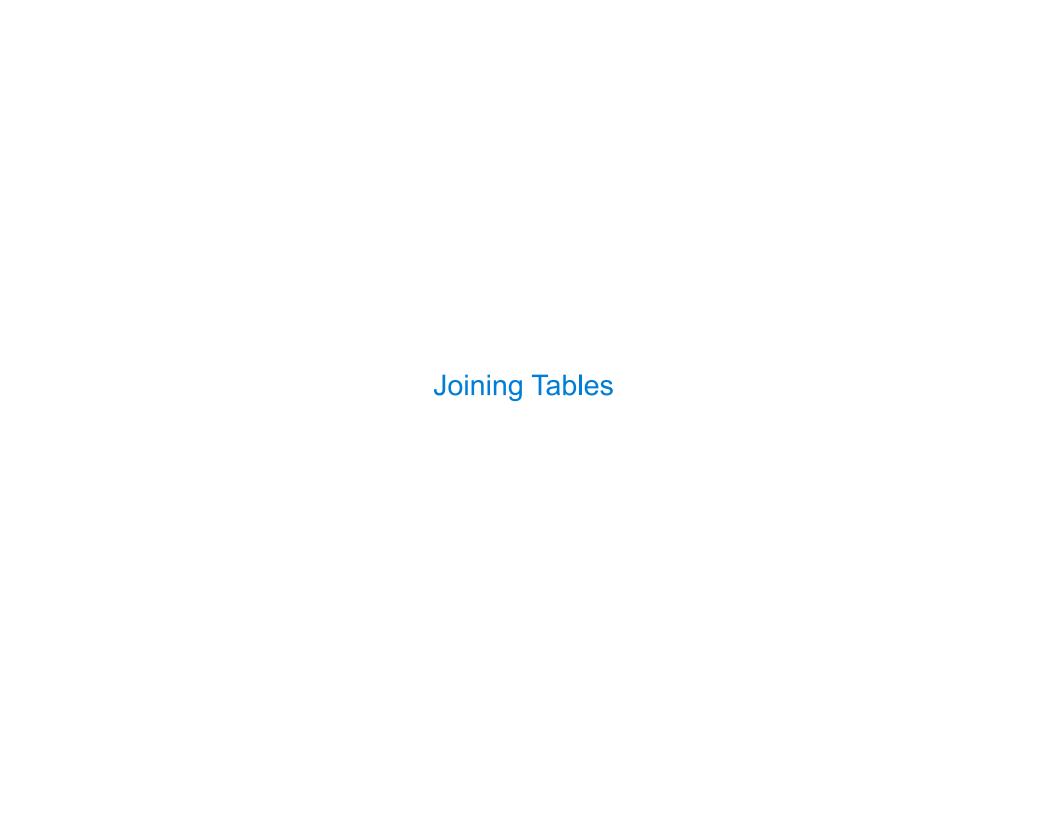
SQL Operators for predicate

 use the WHERE clause in the SQL statements such as <u>SELECT</u>, <u>UPDATE</u> and <u>DELETE</u> to filter rows that do not meet a specified condition

Approximate Matching: LIKE [Docs]

- LIKE compares text to a pattern
- Case-Insensitive by default. Means 'a' and 'A' are the same.
- Allows "wildcards" that match any character.
- % means "zero or more" characters at this "spot" in the pattern
- Examples:

```
'abc' LIKE 'abc' → true
'abc' LIKE 'a%' → true
'abc' LIKE '%b%' → true -shortcut for "does abc contain b?"
'b' LIKE '%b%' → true
'abc' LIKE 'c' → false
```



Ice Cream Cones and Sales

Two tables are joined by a comma to yield all combinations of a row from each

SELECT * FROM sales, cones;

Joins combine two tables
A "cross product" or full join gives all
combinations, e.g. M*N total rows.
This is often not useful!

Instead we often make inner join where we "combine" rows only on some logical identifier, like an "id"

[sqlite> select * from sales, cones; Baskin|1|1|strawberry|pink|3.55 Baskin|1|2|chocolate|light brown|4.75 Baskin|1|3|chocolate|dark brown|5.25 Baskin|1|4|strawberry|pink|5.25 Baskin|1|5|bubblegum|pink|4.75 Baskin|1|6|chocolate|dark brown|5.25 Baskin|3|1|strawberry|pink|3.55 Baskin|3|2|chocolate|light brown|4.75 Baskin|3|3|chocolate|dark brown|5.25 Baskin|3|4|strawberry|pink|5.25 Baskin|3|5|bubblegum|pink|4.75 Baskin|3|6|chocolate|dark brown|5.25 Baskin | 4 | 1 | strawberry | pink | 3.55 Baskin | 4 | 2 | chocolate | light brown | 4.75 Baskin|4|3|chocolate|dark brown|5.25 Baskin | 4 | 4 | strawberry | pink | 5.25 Baskin | 4 | 5 | bubblegum | pink | 4.75 Baskin|4|6|chocolate|dark brown|5.25 Robin | 2 | 1 | strawberry | pink | 3.55 Robin|2|2|chocolate|light brown|4.75 Robin | 2 | 3 | chocolate | dark brown | 5.25 Robin | 2 | 4 | strawberry | pink | 5.25 Robin | 2 | 5 | bubblegum | pink | 4.75 Robin|2|6|chocolate|dark brown|5.25 Robin|5|1|strawberry|pink|3.55 Robin|5|2|chocolate|light brown|4.75 Robin|5|3|chocolate|dark brown|5.25 Robin|5|4|strawberry|pink|5.25 Robin|5|5|bubblegum|pink|4.75 Robin|5|6|chocolate|dark brown|5.25 Robin | 6 | 1 | strawberry | pink | 3.55 Robin|6|2|chocolate|light brown|4.75 Robin|6|3|chocolate|dark brown|5.25 Robin | 6 | 4 | strawberry | pink | 5.25 Robin|6|5|bubblegum|pink|4.75

Inner Joins on Two Tables

How do we know which sale goes with which cone?

Often this is called a "foreign key" or a reference to an object in another table.

When column names conflict we write: table_name.column_name in a query. We can "alias" table names in the FROM expression as well.

```
SELECT * FROM sales s, cones c WHERE s.cone_id = c.id;
```

```
sqlite> SELECT * FROM cones, sales WHERE cone_id=cones.id;
Id|Flavor|Color|Price|Cashier|id|cone_id
1|strawberry|pink|3.55|Baskin|3|1
1|strawberry|pink|3.55|Robin|6|1
2|chocolate|light brown|4.75|Baskin|1|2
2|chocolate|light brown|4.75|Baskin|4|2
2|chocolate|light brown|4.75|Robin|5|2
3|chocolate|dark brown|5.25|Robin|2|3
```