Loading Data into SQLite

```sql
> sqlite3 lecture04.db
sqlite> create table Purchase
    (pid int primary key,
     product text,
     price float,
     quantity int,
     month varchar(15));

sqlite3 -- download data.
sqlite> .mode list
sqlite> .import lec04-data.txt Purchase
```

MUST BE IN SAME MODE AS FILE TYPE

If the data is separated by commas, need to set
```sql
sqlite> .mode csv
```

Joins in SQL

**Retrieve all Japanese products that cost < $150**

```sql
SELECT P.pname, P.price
FROM Product as P, Company as C
WHERE P.manufacturer = C.cname
    AND C.country = 'Japan'
    AND C.price < 150
```

**Retrieve all USA companies that manufacture "gadget" products**

```sql
SELECT DISTINCT C.cname
FROM Product as P, Company as C
WHERE C.country = 'USA' AND C.category = 'gadget'
    AND P.manufacturer = C.cname
```
INNER joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
    AND manufacturer = cname
```

Product

<table>
<thead>
<tr>
<th>pname</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
</tbody>
</table>

Company

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
</tr>
</tbody>
</table>

INNER joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
    AND manufacturer = cname
```

Product

<table>
<thead>
<tr>
<th>pname</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
</tbody>
</table>

Company

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
</tr>
</tbody>
</table>

INNER joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
    AND manufacturer = cname
```

Product

<table>
<thead>
<tr>
<th>pname</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
</tbody>
</table>

Company

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
</tr>
</tbody>
</table>

INNER joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
    AND manufacturer = cname
```

Product

<table>
<thead>
<tr>
<th>pname</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
</tbody>
</table>

Company

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
</tr>
</tbody>
</table>

INNER joins

```
SELECT DISTINCT cname
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
    AND manufacturer = cname
```

Product

<table>
<thead>
<tr>
<th>pname</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
</tbody>
</table>

Company

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
</tr>
</tbody>
</table>
(Inner) joins

```
SELECT DISTINCT name
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
AND manufacturer = name
```

Product

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td>GizmoWorks</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
<td>Hitachi</td>
</tr>
</tbody>
</table>

Company

<table>
<thead>
<tr>
<th>name</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
</tr>
</tbody>
</table>

(Inner) joins

```
SELECT DISTINCT name
FROM Product, Company
WHERE country='USA' AND category = 'gadget'
AND manufacturer = name
```

This is called nested loop semantics since we are interpreting what a join means using a nested loop.

Another example

```
Product(pname, price, category, manufacturer)
Company(cname, country)
-- manufacturer is foreign key to Company

SELECT DISTINCT z.cname
FROM Product x, Company z
WHERE z.country = 'USA'
AND x.manufacturer = z.cname
AND x.category = 'gadget'
AND x.category = 'photography';
```

Does this work?
Another example

Product(pname, price, category, manufacturer)
Company(cname, country)

-- manufacturer is foreign key to Company

Find US companies that manufacture both 'gadgets' and 'photo' products

```sql
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname
AND x.category = 'gadget'
AND y.category = 'photo';
```

Self-Joins and Tuple Variables

Find US companies that manufacture both 'gadgets' and 'photo' products

- Joining Product with Company is insufficient: need to join Product, with Product, and with Company
- When a relation occurs twice in the FROM clause we call it a self-join; in that case we must use tuple variables (aka table aliases) (why?)

Self-joins

```sql
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;
```

Self-joins

```sql
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;
```

Self-joins

```sql
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;
```
SELECT DISTINCT z.cname
FROM Product x, Product y, Company z
WHERE z.country = 'USA'
AND x.category = 'gadget'
AND y.category = 'photo'
AND x.manufacturer = z.cname
AND y.manufacturer = z.cname;
Joins in SQL

- The join we have just seen is sometimes called an inner join – Each row in the result must come from both tables in the join
- Sometimes we want to include rows from only one of the two tables: outer join

Retrieve employees and their sales

```sql
SELECT *
FROM Employee E, Sales S
WHERE E.id = S.employeeID
```

Alternative syntax

```sql
SELECT *
FROM Employee E
INNER JOIN Sales S
ON E.id = S.employeeID
```
### Outer Join

<table>
<thead>
<tr>
<th>Employee(id, name)</th>
<th>Sales(employeeID, productId)</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>1</td>
<td>Joe</td>
</tr>
<tr>
<td>2</td>
<td>Jack</td>
</tr>
<tr>
<td>3</td>
<td>Jill</td>
</tr>
</tbody>
</table>

Retrieve employees and their sales

**SELECT** * FROM Employee E
**LEFT OUTER JOIN** Sales S
**ON** E.id = S.employeeID

### Outer joins

**Product(name, category)**
**Purchase(prodName, store)**

-- prodName is foreign key

```sql
SELECT Product.name, Purchase.store
FROM Product
**LEFT OUTER JOIN** Purchase
**ON** Product.name = Purchase.prodName
```

**CSE 414 - Autumn 2018**

**Name**  
Gizmo  
Camera  
OneClick  

**Category**  
gadget  
Photo  

**ProdName**  
Gizmo  
Camera  
OneClick  

**Store**  
Wiz  
Ritz  

**Output**  
Gizmo  
Wiz  

<table>
<thead>
<tr>
<th>Product</th>
<th>Purchase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

**Output**

<table>
<thead>
<tr>
<th>Name</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
</tbody>
</table>

CSE 414 - Autumn 2018
SELECT Product.name, Purchase.store 
FROM Product 
LEFT OUTER JOIN Purchase ON Product.name = Purchase.prodName

Output:
<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

SELECT Product.name, Purchase.store 
FROM Product 
FULL OUTER JOIN Purchase ON Product.name = Purchase.prodName

Output:
<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>Wiz</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
<td>NULL</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>Foo</td>
</tr>
</tbody>
</table>

Outer Joins

- Left outer join:  
  - Include tuples from tableA even if no match
- Right outer join:  
  - Include tuples from tableB even if no match
- Full outer join:  
  - Include tuples from both even if no match
- In all cases:  
  - Patch tuples without matches using NULL

Aggregates in SQL

Five basic aggregate operations in SQL:

- `select count(*) from Purchase`
- `select sum(quantity) from Purchase`
- `select avg(price) from Purchase`
- `select max(quantity) from Purchase`
- `select min(quantity) from Purchase`

Except count, all aggregations apply to a single attribute.

Aggregates and NULL Values

Null values are not used in aggregates

```
insert into Purchase
values(12, 'gadget', NULL, NULL, 'april')
```

Try the following:

```
select count(*) from Purchase
select count(quantity) from Purchase
select sum(quantity) from Purchase
select count(*) from Purchase
where quantity is not null;
```
COUNT applies to duplicates, unless otherwise stated:

```
SELECT count(product)
FROM Purchase
WHERE price > 4.99
```

same as `count(*)` if no nulls

We probably want:

```
SELECT count(DISTINCT product)
FROM Purchase
WHERE price > 4.99
```

More Examples

```
SELECT Sum(P.price * P.quantity)
FROM Purchase as P
```

```
SELECT Sum(P.price * P.quantity)
FROM Purchase as P
WHERE P.product = 'bagel'
```

What do they mean?