1 SQL Queries

- Join Queries
- Unions and Aliases
- Nested Queries
INNER Join

- finds the intersection between two tables
- can explicitly list the relationship or use INNER JOIN or use NATURAL JOIN

```sql
1  SELECT DISTINCT surname, firstname, customer.cust_id
2    FROM customer, orders
3    WHERE customer.cust_id = orders.cust_id;
4
5  SELECT DISTINCT surname, firstname, customer.cust_id
6    FROM customer
7    INNER JOIN orders USING (cust_id);
8
9  SELECT DISTINCT surname, firstname, customer.cust_id
10    FROM customer
11    NATURAL JOIN orders;
```
ON Clause

- use `ON` when attributes don’t have the same name
- use `WHERE` to limit the rows of the output using additional conditions

```
1 SELECT juice_type.juice_type
2   FROM juice INNER JOIN juice_type
3   ON juice.juice_type=juice_type.juice_type_id
4 WHERE juice.juice_id=100;

6 SELECT juice_id FROM orders INNER JOIN items
7   ON orders.order_id=items.order_id AND orders.cust_id=items.cust_id
8 WHERE orders.cust_id=20 AND orders.order_id=1;
```
LEFT and RIGHT Outer Join

- outputs all rows from left side of the join, supplying NULL when there is no match from the right side
- list all the countries and customers who live in that country:

```sql
1 SELECT country, surname, firstname, cust_id
2 FROM countries LEFT JOIN customer USING (country_id);
```

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Australia</td>
<td>Stribling</td>
<td>Michelle</td>
<td>646</td>
</tr>
<tr>
<td>2</td>
<td>Australia</td>
<td>Skerry</td>
<td>Samantha</td>
<td>647</td>
</tr>
<tr>
<td>3</td>
<td>Australia</td>
<td>Cassisi</td>
<td>Betty</td>
<td>648</td>
</tr>
<tr>
<td>4</td>
<td>Australia</td>
<td>Krennan</td>
<td>Jim</td>
<td>649</td>
</tr>
<tr>
<td>5</td>
<td>Australia</td>
<td>Woodburne</td>
<td>Lynette</td>
<td>650</td>
</tr>
<tr>
<td>6</td>
<td>Austria</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>7</td>
<td>Azerbaijan</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>8</td>
<td>Bahamas</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>

- RIGHT: outputs all rows from the right side of the join, supplying NULL when there is no match from the left side
More Fun with Outer Join

- find the customers who have never placed an order:

```
1 SELECT surname, firstname, orders.cust_id
2 FROM customer LEFT JOIN orders USING (cust_id)
3 WHERE orders.cust_id IS NULL;
```

```
<table>
<thead>
<tr>
<th>surname</th>
<th>firstname</th>
<th>cust_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorrenti</td>
<td>Caitlyn</td>
<td>NULL</td>
</tr>
<tr>
<td>Mockridge</td>
<td>Megan</td>
<td>NULL</td>
</tr>
<tr>
<td>Krennan</td>
<td>Samantha</td>
<td>NULL</td>
</tr>
<tr>
<td>Dimitria</td>
<td>Melissa</td>
<td>NULL</td>
</tr>
<tr>
<td>Oaton</td>
<td>Mark</td>
<td>NULL</td>
</tr>
<tr>
<td>Cassisi</td>
<td>Joshua</td>
<td>NULL</td>
</tr>
</tbody>
</table>
```
User Variables

- save the result of a calculation to use later
- names of customers who bought the most expensive juice:

```sql
1   SELECT @max_cost:=max(cost) FROM inventory;
2     
3   SELECT customer.cust_id, surname, firstname
4     FROM customer INNER JOIN items USING (cust_id)
5       INNER JOIN inventory USING (juice_id)
6     WHERE cost = @max_cost;
7     
8     | cust_id | surname  | firstname |
9     |---------|----------|-----------|
10    | 32      | Archibald| Joshua    |
11    | 33      | Galti    | Lynette   |
12    | 44      | Mellili  | Michelle  |
13    | 54      | Woodestock| George   |
14    | 71      | Mellaseca| Lynette   |
15    |         |          |           |
16    |         |          |           |
```

Daniel Zappala
CS 360 Internet Programming
UNION Clause

- combine the results of two or more queries
- list the three oldest and three newest customers:

```sql
(SELECT cust_id, surname, firstname
 FROM customer ORDER BY cust_id LIMIT 3)
UNION
(SELECT cust_id, surname, firstname
 FROM customer ORDER BY cust_id DESC LIMIT 3);
```

<table>
<thead>
<tr>
<th>cust_id</th>
<th>surname</th>
<th>firstname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Rosenthal</td>
<td>Joshua</td>
</tr>
<tr>
<td>2</td>
<td>Serrong</td>
<td>Martin</td>
</tr>
<tr>
<td>3</td>
<td>Leramonth</td>
<td>Jacob</td>
</tr>
<tr>
<td>650</td>
<td>Woodburne</td>
<td>Lynette</td>
</tr>
<tr>
<td>649</td>
<td>Krennan</td>
<td>Jim</td>
</tr>
<tr>
<td>648</td>
<td>Cassisi</td>
<td>Betty</td>
</tr>
</tbody>
</table>

Daniel Zappala | CS 360 Internet Programming
- shorthand for a table name, to save some typing

```sql
1 SELECT * FROM inventory i, juice j
2 WHERE i.juice_id = 183 AND i.juice_id = j.juice_id;
```

- find two customers with the same surname:

```sql
1 SELECT c1.cust_id, c2.cust_id FROM customer c1, customer c2
2 WHERE c1.surname = c2.surname AND c1.cust_id != c2.cust_id;
```
More Fun with Aliases

- bookmark table
  - id
  - url
  - tag

select all bookmarks with both the “blog” and “baseball” tags:

1. SELECT DISTINCT b1.bookmark FROM bookmarks b1, bookmarks b2
2. WHERE b1.id != b2.id AND b1.tag = "blog" AND b2.tag = "baseball";
Attribute Aliases

1. SELECT surname AS s, firstname AS f FROM customer
2. WHERE surname = "Krennan" ORDER BY s, f;

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Krennan</td>
<td>Andrew</td>
</tr>
<tr>
<td>8</td>
<td>Krennan</td>
<td>Betty</td>
</tr>
<tr>
<td>9</td>
<td>Krennan</td>
<td>Caitlyn</td>
</tr>
<tr>
<td>10</td>
<td>Krennan</td>
<td>Caitlyn</td>
</tr>
<tr>
<td>11</td>
<td>Krennan</td>
<td>Dimitria</td>
</tr>
</tbody>
</table>
Introduction

- useful when you need to combine several queries
- note, next two examples could use a compound WHERE clause

1 # name of juiceries in the Margaret River region
2 SELECT juicery_name FROM juicery WHERE region_id = (SELECT region_id FROM region
3     WHERE region_name = "Margaret River");
4
5 # name of region that makes juice #17
6 SELECT region_name FROM region WHERE region_id = (SELECT region_id FROM juicery
7     WHERE juicery_id = (SELECT juicery_id FROM juice WHERE juice_id = 17));
Needed Nested Queries

- find the customer who has made the single largest purchase of a juice

```
1 SELECT DISTINCT customer.cust_id FROM customer
2     INNER JOIN items USING (cust_id)
3 WHERE price = (SELECT MAX(price) FROM items);
```
IN Clause

```sql
1  # find bookmarks with blog and baseball tag
2  SELECT id FROM bookmarks
3     WHERE tag="blog" AND bookmark_id IN
4         (SELECT id FROM bookmarks WHERE tag="baseball");
5
6  # find juices purchased by customers who placed six or more orders
7  SELECT DISTINCT juice_id FROM items WHERE cust_id IN
8     (SELECT customer.cust_id FROM customer
9         INNER JOIN orders USING (cust_id)
10        GROUP BY cust_id HAVING count(order_id) >= 6);
```
print results from outer query only if inner query returns results

select the regions that have at least 35 juiceries:

```
1 SELECT region_name FROM region WHERE EXISTS
2 (SELECT * FROM juicery WHERE region.region_id = juicery.region_id
3 GROUP BY region_id HAVING count(*) > 35);
```