Local Definitions, Recursive Queries

Local Tables

- SQL provides a way to create (essentially) a temporary table for use in one `select`.
- Analogous to the `let` expression in Scheme.
- Here, `foreigner` is a one-column table local to this statement.

```sql
with foreigner(person) as
  (select "Martin" union
   select "Christina" union
   select "Johanna"
  )
select child from people, foreigner
where people.parent = foreigner.person;
```

What does this do?

<table>
<thead>
<tr>
<th>people parent</th>
<th>child</th>
</tr>
</thead>
<tbody>
<tr>
<td>Martin</td>
<td>George</td>
</tr>
<tr>
<td>Christina</td>
<td>George</td>
</tr>
<tr>
<td>George</td>
<td>Martin F</td>
</tr>
<tr>
<td>Johanna</td>
<td>Martin F</td>
</tr>
<tr>
<td>George N</td>
<td>Ann</td>
</tr>
<tr>
<td>George N</td>
<td>John</td>
</tr>
<tr>
<td>Martin F</td>
<td>George N</td>
</tr>
<tr>
<td>Martin F</td>
<td>Robert</td>
</tr>
<tr>
<td>Martin F</td>
<td>Donald</td>
</tr>
<tr>
<td>Donald</td>
<td>Peter</td>
</tr>
</tbody>
</table>

Example: Ancestry Relationships

- What does the program on the left do?
- (distinct removes duplicate rows.)

```sql
with kin(first, second) as
  (select a.child, b.child
   from people as a, people as b
   where a.parent = b.parent
   and a.child != b.child )
select distinct kin.second, child
from people, kin
where kin.first = parent;
```

Recursion, Yet Again

- As with Python, Scheme, and streams, (limited) recursion is possible in SQL using the `with` clause.

```sql
with table_name(column_names) as
  (select ...
   union -- Base case
   select ...
   from ..., table_name, ...
  )
select ...
```

- The recursively defined table must appear only once in the `from` clause of the last `select` in the `with` clause.
- Because of these restrictions, no mutual recursions or tree recursions are allowed.

Example: Integers

- Define the table `ints` to contain integers from 1–30:

```sql
create table ints as
with ints(n) as
  (select 1 union
   select n+1 from ints where n<=30
  )
select n from ints;
```

Defining Ancestor Recursively

- An ancestor is a parent or an ancestor of a parent.

```sql
with related(ancestor, descendant) as
  (select parent, child from people
   where descendant = parent
  )
select ancestor from related where descendant = "Paul";
```

Example: Ancestors

- Ancestors are parents or ancestors of parents.
A Famous Number

- There is a famous story about the "interesting" number 1729, the first of the "taxicab numbers."
- Given our table `ints` (numbers up to 50) how do we find them?