Lecture 3: Continuing SQL

Wednesday, January 28, 2015
Agenda for today

- Class Demos: SQL Developer, SQL*Plus and creating textbook schemas
- Continue SQL: Chapter 3 in Murach textbook
Recall terminology:

- Relation
- Attribute
- Record
- Primary key
- Foreign key
- DML versus DDL
- Clauses of a SELECT statement
A SELECT statement that retrieves a calculated value

```sql
SELECT invoice_id, invoice_total, 
       (credit_total + payment_total) AS total_credits 
FROM invoices 
WHERE invoice_id = 17
```

<table>
<thead>
<tr>
<th>INVOICE_ID</th>
<th>INVOICE_TOTAL</th>
<th>TOTAL_CREDITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
<td>356.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>356.48</td>
</tr>
</tbody>
</table>
A SELECT statement that retrieves all invoices between given dates

```
SELECT invoice_number, invoice_date, invoice_total
FROM invoices
WHERE invoice_date BETWEEN '01-MAY-2014' AND '31-MAY-2014'
ORDER BY invoice_date
```

(70 rows selected)
Basic Pattern Matching in SQL

```
SELECT vendor_id, vendor_name, vendor_address1
FROM vendors
WHERE vendor_name LIKE '%%Consultants%%'
```

```
SELECT invoice_number, invoice_date, invoice_total
FROM invoices
WHERE invoice_number LIKE 'P-%'
```
Column specifications that use calculated values

An arithmetic expression that calculates balance_due

```
SELECT invoice_number,
    invoice_total - payment_total - credit_total
AS balance_due
```

A string expression that derives full_name

```
SELECT first_name || ' ' || last_name AS full_name
```
A SELECT statement that retrieves rows with null values

```
SELECT *
FROM null_sample
WHERE invoice_total IS NULL
```

![Table with null values](image)

A SELECT statement that retrieves rows without null values

```
SELECT *
FROM null_sample
WHERE invoice_total IS NOT NULL
```

![Table with non-null values](image)
The expanded syntax of the ORDER BY clause

ORDER BY expression [ASC|DESC]
[ , expression [ASC|DESC]] ...

An ORDER BY clause that sorts by one column

SELECT vendor_name,
    vendor_city || ', ' || vendor_state || ' ' ||
    vendor_zip_code AS address
FROM vendors
ORDER BY vendor_name

<table>
<thead>
<tr>
<th>VENDOR_NAME</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ASC Signs</td>
<td>Fresno, CA 93703</td>
</tr>
<tr>
<td>2 AT&amp;T</td>
<td>Phoenix, AZ 85062</td>
</tr>
<tr>
<td>3 Abbey Office Furnishings</td>
<td>Fresno, CA 93722</td>
</tr>
</tbody>
</table>
An ORDER BY clause that sorts by one column in descending sequence

```sql
SELECT vendor_name,
       vendor_city || ', ' || vendor_state || ' ' || vendor_zip_code AS address
FROM vendors
ORDER BY vendor_name DESC
```

<table>
<thead>
<tr>
<th>VENDOR_NAME</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zylka Design</td>
<td>Fresno, CA 93711</td>
</tr>
<tr>
<td>Zip Print &amp; Copy Center</td>
<td>Fresno, CA 93777</td>
</tr>
<tr>
<td>Zee Medical Service Co</td>
<td>Washington, IA 52353</td>
</tr>
</tbody>
</table>
An ORDER BY clause that sorts by three columns

```sql
SELECT vendor_name,
    vendor_city || ', ' || vendor_state || ' ' ||
    vendor_zip_code AS address
FROM vendors
ORDER BY vendor_state, vendor_city, vendor_name
```

<table>
<thead>
<tr>
<th>VENDOR_NAME</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 AT&amp;T</td>
<td>Phoenix, AZ 85062</td>
</tr>
<tr>
<td>2 Computer Library</td>
<td>Phoenix, AZ 85023</td>
</tr>
<tr>
<td>3 Wells Fargo Bank</td>
<td>Phoenix, AZ 85038</td>
</tr>
<tr>
<td>4 Aztek Label</td>
<td>Anaheim, CA 92807</td>
</tr>
<tr>
<td>5 Blue Shield of California</td>
<td>Anaheim, CA 92850</td>
</tr>
<tr>
<td>6 Diversified Printing &amp; Pub</td>
<td>Brea, CA 92621</td>
</tr>
<tr>
<td>7 ASC Signs</td>
<td>Fresno, CA 93703</td>
</tr>
</tbody>
</table>
An ORDER BY clause that uses an expression

```
SELECT vendor_name, 
    vendor_city || ', ' || vendor_state || ' ' || 
    vendor_zip_code AS address 
FROM vendors 
ORDER BY vendor_contact_last_name  
    || vendor_contact_first_name
```

<table>
<thead>
<tr>
<th>VENDOR_NAME</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Dristas Groom &amp; McCormick</td>
<td>Fresno, CA 93720</td>
</tr>
<tr>
<td>2 Internal Revenue Service</td>
<td>Fresno, CA 93888</td>
</tr>
<tr>
<td>3 US Postal Service</td>
<td>Madison, WI 53707</td>
</tr>
<tr>
<td>4 Yale Industrial Trucks-Fresno</td>
<td>Fresno, CA 93706</td>
</tr>
</tbody>
</table>
An ORDER BY clause that uses column positions

```sql
SELECT vendor_name,
    vendor_city || ' ', ' ' || vendor_state || ' ' ||
    vendor_zip_code AS address
FROM vendors
ORDER BY 2, 1
```

```
<table>
<thead>
<tr>
<th>VENDOR_NAME</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Aztek Label</td>
<td>Anaheim, CA 92807</td>
</tr>
<tr>
<td>2 Blue Shield of California</td>
<td>Anaheim, CA 92850</td>
</tr>
<tr>
<td>3 Malloy Lithographing Inc</td>
<td>Ann Arbor, MI 48106</td>
</tr>
<tr>
<td>4 Data Reproductions Corp</td>
<td>Auburn Hills, MI 48326</td>
</tr>
</tbody>
</table>
```
An expression in the WHERE clause

```
SELECT invoice_number, invoice_date, invoice_total, payment_total, credit_total, 
      invoice_total - payment_total - credit_total
AS balance_due
FROM invoices
WHERE invoice_total - payment_total - credit_total > 0
ORDER BY invoice_date
```
Joining two tables using the explicit join syntax

```sql
SELECT vendor_name, invoice_number, invoice_date, invoice_total
FROM vendors JOIN invoices
    ON vendors.vendor_id = invoices.vendor_id
WHERE invoice_total >= 500
ORDER BY vendor_name, invoice_total DESC
```
Joining two tables using the implicit join syntax

```
SELECT vendor_name, invoice_number, invoice_date, invoice_total
FROM vendors, invoices
WHERE vendors.vendor_id = invoices.vendor_id
AND invoice_total >= 500
ORDER BY vendor_name, invoice_total DESC
```