SQL Basics

- Based on the relational algebra.
- Nonprocedural language – what to be done not how
- Simple, powerful language
- Used for both
  - data manipulation
  - data definition

- Can be used as
  - interactive query language (ad hoc queries)  
    *(This is how we will use it)*
  - embedded in another programming language (Java, PHP, Python)
  - client/server language
  - internet data access language
SQL Standards

Originally developed at IBM in the 70s

Called SEQUEL (Structured English Query Language)

Part of NIST until 1996

Latest version SQL:2011

Many cross-platform incompatibilities
SQL Statements

- SQL statements begin with
  - SELECT
  - INSERT
  - DELETE
  - UPDATE
  - PROCEDURE
- COMMIT and ROLLBACK for transactions
SQL statements

Each SQL statement must begin with one of those Keywords and end with a semicolon.

The statements are independent.

Examples:
SELECT fields FROM table;

UPDATE table SET attribute WHERE condition;

INSERT INTO table VALUES (values);
SELECT statement

*Project onto all fields*
SELECT *
FROM Employee

*Project onto some fields*
SELECT lastName, firstName
FROM Employee
Ordering Results

You can sort the results of your select statement by adding an ORDER BY clause with optional ASC or DESC.

```sql
SELECT lastName, firstName
FROM Employee
ORDER BY lastName;
```

<table>
<thead>
<tr>
<th>lastName</th>
<th>firstName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fivozinsky</td>
<td>Bruce</td>
</tr>
<tr>
<td>Fortune</td>
<td>Julian</td>
</tr>
<tr>
<td>Threat</td>
<td>Ayisha</td>
</tr>
<tr>
<td>Toulouse</td>
<td>Jie</td>
</tr>
<tr>
<td>Uno</td>
<td>Jane</td>
</tr>
</tbody>
</table>

```sql
SELECT firstName, lastName
FROM Employee
ORDER BY firstName DESC;
```

<table>
<thead>
<tr>
<th>firstName</th>
<th>lastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Julian</td>
<td>Fortune</td>
</tr>
<tr>
<td>Jie</td>
<td>Toulouse</td>
</tr>
<tr>
<td>Jane</td>
<td>Uno</td>
</tr>
<tr>
<td>Bruce</td>
<td>Fivozinsky</td>
</tr>
<tr>
<td>Ayisha</td>
<td>Threat</td>
</tr>
</tbody>
</table>
DISTINCT

To remove duplicate rows from the result table (just like with relational algebra)

You can use DISTINCT or DISTINCTROW

SELECT accountId
FROM rental;

SELECT DISTINCT accountId
FROM rental;

SELECT DISTINCT accountId
FROM rental;
WHERE clause

The WHERE clause of the SELECT statement restricts the rows in the results table (like selection in relational algebra)

```
SELECT *
FROM Rental
WHERE accountId=101;
```
AND and OR in WHERE

SELECT *
FROM Rental
WHERE accountId=101 AND videoId>1000;

< (less than)
> (greater than)
= (equal)
>= (greater than or equal to)
<= (less than or equal to)
AND and OR in WHERE

SELECT *
FROM Customer
WHERE firstName='Jane' AND city='Apopka';
COUNT and Aliases

One of the many functions provided by SQL is count. This is used to count the number of rows in the result table. You can also use aliases for columns (and tables).

```
SELECT COUNT(*) AS outstandingRentals
FROM Rental;
```

```
SELECT COUNT(dateDue) AS outstandingRentalsDue
FROM Rental;
```

NULLS are not counted

outstandingRentals 10

outstandingRentalsDue 9

NULLS are not counted
Table aliases will be more useful when we have multiple Tables in an SQL statement as we will when we join two tables.

SELECT *
FROM Rental r
WHERE r.accountId=101 AND r.videoId>1000;
Joins

A regular INNER JOIN in with SQL can be written as follows:

```
SELECT *
FROM Employee e, Timecard t
WHERE e.ssn=t.ssn;
```

or

```
SELECT *
FROM Employee e INNER JOIN Timecard t
ON e.ssn=t.ssn;
```