SELECT Statement

Basic Syntax

- The **SELECT** statement is used to execute a query that retrieves data from a Database Object.

Basic Syntax

```
SELECT column1, column2
      FROM table_name
```

- The **SELECT** statement shown above retrieves the columns you specify in the SELECT clause from the base table specified in the FROM clause and store them in a result set

Example using AP database

```
SELECT VendorName, VendorCity
      FROM Vendors;
```

- The statement above would retrieve the VendorName and the VendorCity from the Vendors table and store them in a result set

Using Wildcard *

Syntax

```
SELECT *
	FROM table_name
```

- The SELECT statement above would be used to retrieve ALL columns in the table specified in the FROM clause and store them in a result set.

Example

```
SELECT *
	FROM Vendors;
```

Naming the columns in a result set

- By default, a column in a result set is named the same as the name of the column. For example:

```
SELECT VendorName, Vendor City
  FROM Vendors;
```

The result set for this SELECT statement is:

VendorName	VendorCity
US Postal Service	Madison
National Information Data Ctr	Washington
Register of Copyrights	Washington
Jobtrak	Los Angeles

Naming the columns in a result set…continued

- If you wanted the VendorName in the header in the result set to be Vendor Name (with a space) and VendorCity to be City, you could use the AS keyword like below:

```
SELECT VendorName AS [Vendor Name], Vendor City AS City
  FROM Vendors;
```

The result set for this SELECT statement is:

Vendor Name	City
US Postal Service	Madison
National Information Data Ctr	Washington
Register of Copyrights	Washington
Jobtrak	Los Angeles

Notice the square brackets. If the column name you want to use is more than one word, you need to use square brackets

Concatenating Data

- You can concatenate multiple columns and or literal strings in your SELECT statements as well. For example, I want to return the Vendor City and the Vendor State separated by a comma:

```
SELECT VendorCity, VendorState, VendorCity + ‘, ‘ + VendorState
    FROM Vendors;
```

ORDER BY Clause

- Often, we want to sort the result set that is returned to us. We do that by using the ORDER BY Clause. For Example, the query below would sort by VendorName

```
SELECT VendorName, VendorCity
       FROM Vendors
       ORDER BY VendorName;
```

- You can sort by Ascending or Descending order. Ascending is the default, so in the query above, it will sort by ascending since we didn’t explicitly state the order by direction. We could explicitly state the sort direction to be ascending by using ASC like below:

```
SELECT VendorName, VendorCity
       FROM Vendors
       ORDER BY VendorName ASC;
```

- To order by Descending, we use DESC like below:

```
SELECT VendorName, VendorCity
       FROM Vendors
       ORDER BY VendorName DESC
```

SELECT TOP

```
SELECT TOP 5 column1, column2
        FROM table_name
```

- We use the TOP keyword to limit the number or records we get returned to us in the result set.

Example

```
SELECT TOP 5 VendorName, VendorCity
        FROM Vendors
        ORDER BY VendorName;
```

Arithmetic Expressions

- It is possible to have arithmetic expressions in your select statement. For example:

```
SELECT
          InvoiceTotal,
            PaymentTotal,
            CreditTotal,
            InvoiceTotal – PaymentTotal – CreditTotal AS BalanceDue
FROM
            Invoices;
```

SELECT DISTINCT

```
SELECT DISTINCT column1, column2
    FROM table_name;
```

- The distinct keyword prevents duplicate rows from being included in the results set

Example

```
SELECT DISTINCT VendorCity, VendorState
    FROM Vendors;
```