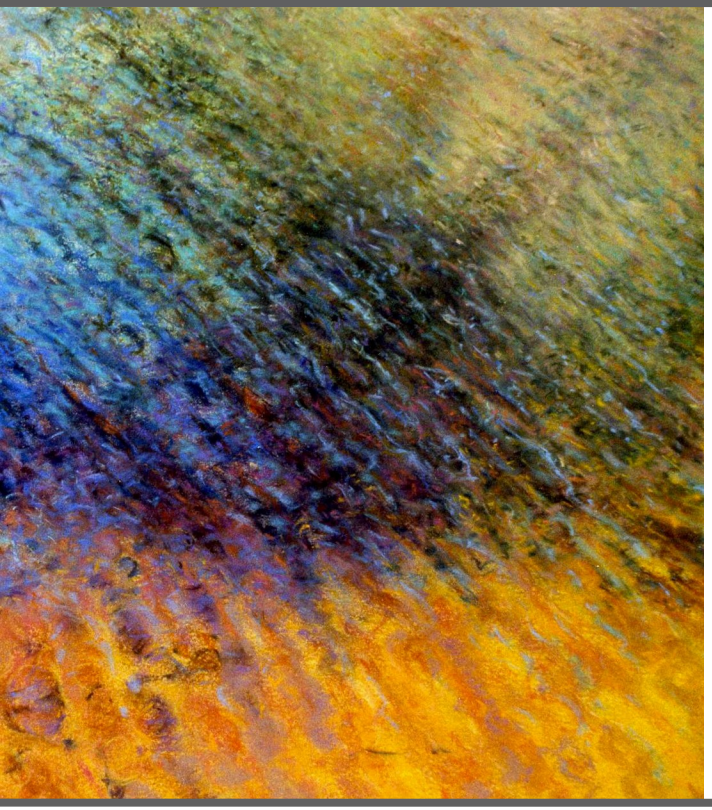
David M. Kroenke and David J. Auer

Database Processing:

Fundamentals, Design, and Implementation



Chapter Two:

Introduction to Structured Query Language

Part 1: Single Table Queries

Objectives

Understand SQL

SELECT

FROM

WHERE

as basis for database queries.

Create SQL queries to retrieve data from a single table.

Ad-Hoc Queries

- **Ad-hoc queries**:
  - Questions that can be answered using database data
  - Example: “How many customers in Atlanta bought blue GSU caps?”
  - Created by user as needed, instead of programmed into application

Oracle Database

- For Oracle Database 12c and Oracle Database XE:
  - See Online Chapter 10B [POSTED UNDER "ORACLE Information"]
- Online chapters 10A, **10B**, and 10C are available for download at:

  http://www.pearsonhighered.com/kroenke/

SQL As a Data Sublanguage

- SQL: not a full featured **programming language**.
  - C, C#, Java
- SQL: **data sublanguage** for creating and processing database data and metadata.
- SQL: ubiquitous in enterprise-class DBMS products.
- SQL: non-procedural
  - State the results you want; not how to get them.

SQL DDL

- **Data definition language (DDL)** statements
  - Used for creating tables, Chapter 7

SQL DML

Data manipulation language (DML) statements

- Used for:
  - Queries – SQL **SELECT** statement
  - Inserting data – SQL **INSERT** statement
  - Modifying data – SQL **UPDATE** statement
  - Deleting data – SQL **DELETE** statement
- See Chapter 2

The SQL SELECT Statement

Fundamental framework for SQL query:

- **SQL SELECT statement.**
  - **SELECT {ColumnName(s)}**
  - **FROM {TableName(s)}**
  - **WHERE {Condition(s)}**
- All SQL statements end with a **semi-colon (;)**

Specific Columns on One Table

```
/* *** SQL-Query-CH02-01 *** */
SELECT     SKU, SKU_Description, Department, Buyer
FROM       SKU_DATA;
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
5	201000	Half-dome Tent	Camping	Cindy Lo
6	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
7	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
8	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

Selecting All Columns:
The SQL Asterisk (*) Wildcard Character

```
/* *** SQL-Query-CH02-02 *** */
SELECT     *
FROM       SKU_DATA;
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
5	201000	Half-dome Tent	Camping	Cindy Lo
6	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
7	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
8	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

Specifying Column Order I

```
/* *** SQL-Query-CH02-03 *** */
SELECT     Department, Buyer
FROM       SKU_DATA;
```

	Department	Buyer
1	Water Sports	Pete Hansen
2	Water Sports	Pete Hansen
3	Water Sports	Nancy Meyers
4	Water Sports	Nancy Meyers
5	Camping	Cindy Lo
6	Camping	Cindy Lo
7	Climbing	Jerry Martin
8	Climbing	Jerry Martin

Specifying Column Order II

```
/* *** SQL-Query-CH02-04 *** */
SELECT      Buyer, Department
FROM        SKU_DATA;
```

	Buyer	Department
1	Pete Hansen	Water Sports
2	Pete Hansen	Water Sports
3	Nancy Meyers	Water Sports
4	Nancy Meyers	Water Sports
5	Cindy Lo	Camping
6	Cindy Lo	Camping
7	Jerry Martin	Climbing
8	Jerry Martin	Climbing

Using Oracle Database I
Oracle SQL Developer

The **SQL Worksheet**

Connections object browser shows connected databases

The **New Connection** button

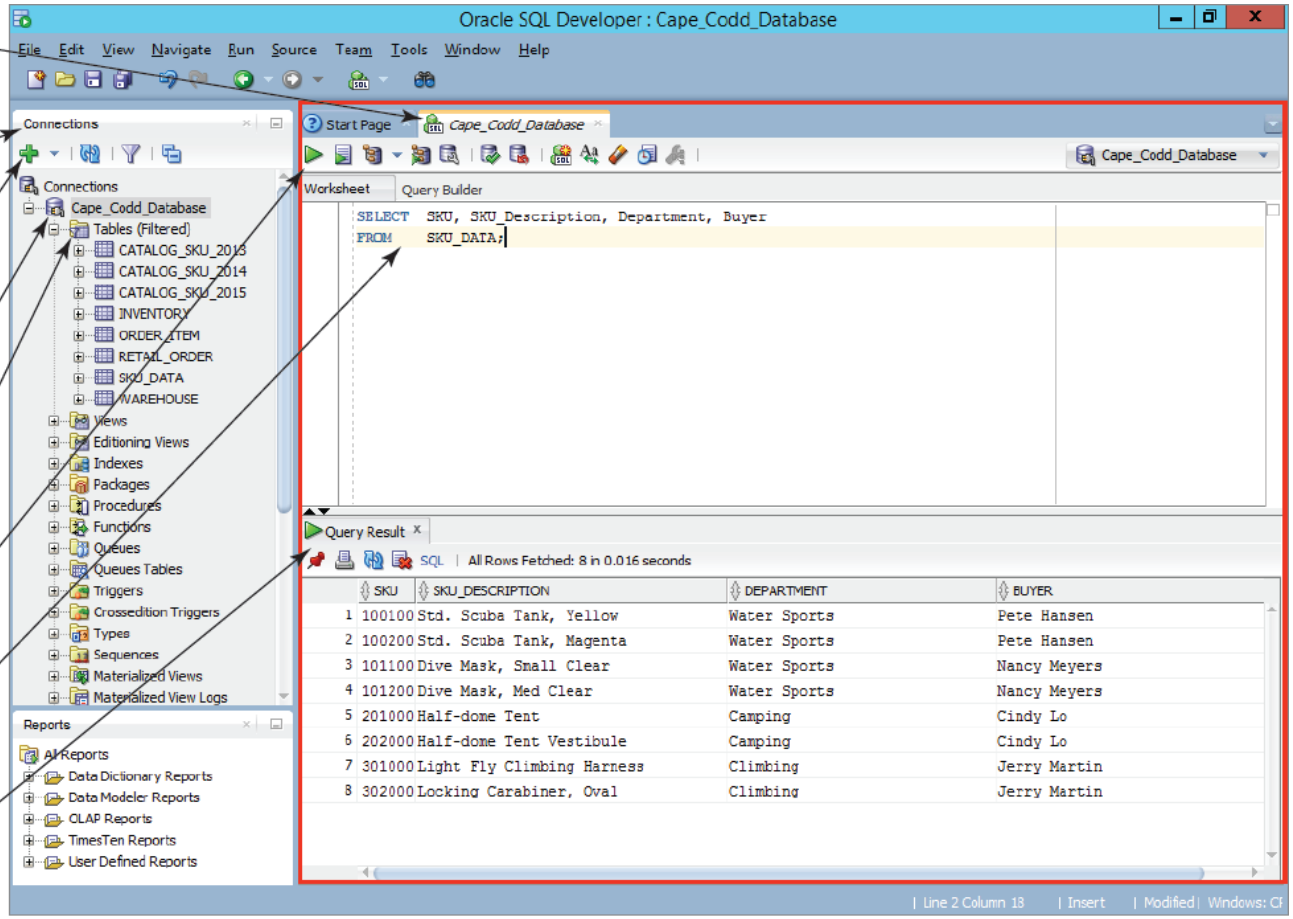
The Cape Codd database

The Cape Codd database tables

The **Execute** button

The SQL query in the SQL Worksheet

The Query Result tabbed window



Using Oracle Database II
Saving an Oracle Database SQL Query

The **Save** button

The **Save** dialog box

The **DBP-e14-Cape-Codd-Database** folder

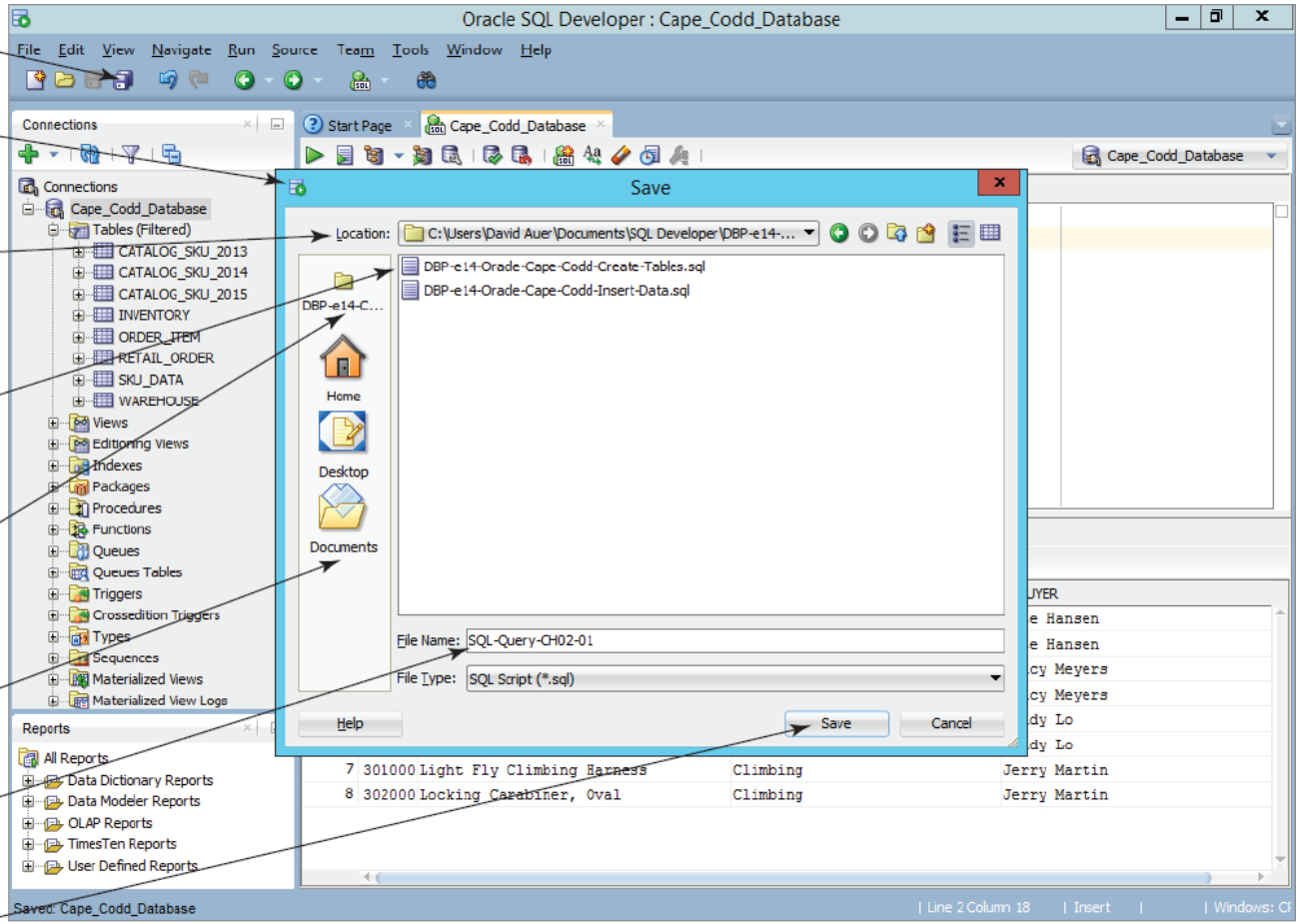
Existing SQL scripts—these were used to create and populate the Cape-Codd database

The **DBP-e14-Cape-Codd-Database folder** button

The **Documents Folder** button

Type the SQL script file name here

The dialog box **Save** button



Reading Specified Rows from a Single Table
The SQL DISTINCT Keyword

```
/* *** SQL-Query-CH02-05 *** */
SELECT     DISTINCT Buyer, Department
FROM       SKU_DATA;
```

	Buyer	Department
1	Cindy Lo	Camping
2	Jerry Martin	Climbing
3	Nancy Meyers	Water Sports
4	Pete Hansen	Water Sports

Reading Specified Rows from a Single Table
The SQL WHERE Clause: Character Strings

```
/* *** SQL-Query-CH02-08 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Department = 'Water Sports';
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

NOTE: SQL wants a plain ASCII single quote: ' NOT ` !

SQL Comparison Operators

SQL Comparison Operators	
Operator	**Meaning**
=	Is equal to
<>	Is NOT Equal to
<	Is less than
>	Is greater than
<=	Is less than OR equal to
>=	Is greater than OR equal to
IN	Is equal to one of a set of values
NOT IN	Is NOT Equal to one of a set of values
BETWEEN	Is within a range of numbers (includes the end points)
NOT BETWEEN	Is NOT within a range of numbers (includes the end points)
LIKE	Matches a set of characters
NOT LIKE	Does NOT match a set of characters
IS NULL	Is equal to NULL
IS NOT NULL	Is NOT equal to NULL

Reading Specified Rows from a Single Table
The SQL WHERE Clause: Dates

```
/* *** SQL-Query-CH02-09 *** */

SELECT      *

FROM        CATALOG_SKU_2014

WHERE       DateOnWebSite = '01-JAN-2014';
```

	CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWebSite
1	20140001	100100	Std. Scuba Tank, Yellow	Water Sports	23	2014-01-01
2	20140002	100300	Std. Scuba Tank, Light Blue	Water Sports	23	2014-01-01
3	20140004	101100	Dive Mask, Small Clear	Water Sports	26	2014-01-01
4	20140005	101200	Dive Mask, Med Clear	Water Sports	26	2014-01-01
5	20140006	201000	Half-dome Tent	Camping	46	2014-01-01
6	20140007	202000	Half-dome Tent Vestibule	Camping	46	2014-01-01
7	20140008	301000	Light Fly Climbing Harness	Climbing	77	2014-01-01
8	20140009	302000	Locking Carabiner, Oval	Climbing	79	2014-01-01

NOTE: See Chapters 10B and 10C for a discussion of dates in Oracle Database and MySQL respectively.

NOTE: Microsoft Access 2013 dates must be enclosed within # symbols: **#01/01/14#**

Reading Specified Rows from a Single Table
The SQL WHERE Clause: Numbers

```
/* *** SQL-Query-CH02-10 *** */
SELECT      *
FROM        SKU_DATA
WHERE       SKU > 200000;
```

	SKU	SKU_Description	Department	Buyer
1	201000	Half-dome Tent	Camping	Cindy Lo
2	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
3	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
4	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

Reading Specified Columns and Rows from a Single Table
Column Names and the SQL WHERE Clause

```
/* *** SQL-Query-CH02-11 *** */
SELECT      SKU_Description, Department
FROM        SKU_DATA
WHERE       Department = 'Climbing';
```

	SKU_Description	Department
1	Light Fly Climbing Harness	Climbing
2	Locking Carabiner, Oval	Climbing

NOTE: A column used in the WHERE clause does ***not*** have to be one of the columns listed in the SELECT clause.

Sorting the SQL Query Results
The SQL ORDER BY Clause

```
/* *** SQL-Query-CH02-13 *** */
SELECT      *
FROM        ORDER_ITEM
ORDER BY    OrderNumber;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	201000	1	300.00	300.00
2	1000	202000	1	130.00	130.00
3	2000	101100	4	50.00	200.00
4	2000	101200	2	50.00	100.00
5	3000	101200	1	50.00	50.00
6	3000	101100	2	50.00	100.00

Sorting the SQL Query Results
Ascending and Descending Sort Order

```
/* *** SQL-Query-CH02-16 *** */
SELECT     *
FROM       ORDER_ITEM
ORDER BY   Price DESC, OrderNumber ASC;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	201000	1	300.00	300.00
2	1000	202000	1	130.00	130.00
3	2000	101100	4	50.00	200.00
4	2000	101200	2	50.00	100.00
5	3000	101200	1	50.00	50.00
6	3000	101100	2	50.00	100.00

NOTE: The default sort order is ASC—it does not have to be specified.

SQL WHERE Clause Options
SQL Logical Operators

SQL Logical Operators	
Operator	**Meaning**
AND	Both arguments are TRUE
OR	One or the other or both of the arguments are TRUE
NOT	Negates the associated operator

SQL WHERE Clause Options
SQL AND Operator

```
/* *** SQL-Query-CH02-18 *** */
SELECT     *
FROM       SKU_DATA
WHERE      Department='Water Sports'
   AND     Buyer='Nancy Meyers';
```

	SKU	SKU_Description	Department	Buyer
1	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

SQL WHERE Clause Options
SQL OR Operator

```
/* *** SQL-Query-CH02-19 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Department='Camping'
   OR       Department='Climbing';
```

	SKU	SKU_Description	Department	Buyer
1	201000	Half-dome Tent	Camping	Cindy Lo
2	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
3	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
4	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

SQL WHERE Clause Options
SQL NOT Operator

```
/* *** SQL-Query-CH02-20 *** */
SELECT       *
FROM         SKU_DATA
WHERE        Department='Water Sports'
   AND NOT   Buyer='Nancy Meyers';
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen

SQL WHERE Clause Options
Sets of Values: SQL IN Operator

```
/* *** SQL-Query-CH02-21 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Buyer IN ('Nancy Meyers', 'Cindy Lo', 'Jerry Martin');
```

	SKU	SKU_Description	Department	Buyer
1	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
3	201000	Half-dome Tent	Camping	Cindy Lo
4	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
5	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
6	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

SQL WHERE Clause Options
Sets of Values: SQL NOT IN Operator

```
/* *** SQL-Query-CH02-18 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Buyer NOT IN ('Nancy Meyers', 'Cindy Lo', 'Jerry Martin');
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen

- A row qualifies for an **IN** condition if the column is *equal* to *any* of the values in the parentheses.
- A row qualifies for a **NOT IN** condition if it is *not equal* to *all* of the items in the parentheses.

SQL WHERE Clause Options

Ranges of Values: Using Math Symbols

```
/* *** SQL-Query-CH02-23 *** */
SELECT      *
FROM        ORDER_ITEM
WHERE       ExtendedPrice >= 100
    AND     ExtendedPrice <= 200
ORDER BY    ExtendedPrice;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	3000	101100	2	50.00	100.00
2	2000	101200	2	50.00	100.00
3	1000	202000	1	130.00	130.00
4	2000	101100	4	50.00	200.00

SQL WHERE Clause Options
Ranges of Values: SQL BETWEEN Operator

```
/* *** SQL-Query-CH02-24 *** */
SELECT      *
FROM        ORDER_ITEM
WHERE       ExtendedPrice BETWEEN 100 AND 200
ORDER BY    ExtendedPrice;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	3000	101100	2	50.00	100.00
2	2000	101200	2	50.00	100.00
3	1000	202000	1	130.00	130.00
4	2000	101100	4	50.00	200.00

SQL WHERE Clause Options
Ranges of Values: SQL NOT BETWEEN Operator

```
/* *** SQL-Query-CH02-25 *** */
SELECT      *
FROM        ORDER_ITEM
WHERE       ExtendedPrice NOT BETWEEN 100 AND 200
ORDER BY    ExtendedPrice;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	3000	101200	1	50.00	50.00
2	1000	201000	1	300.00	300.00
3	3000	100200	1	300.00	300.00

SQL WHERE Clause Options
Character String Patterns: SQL LIKE Operator I

```
/* *** SQL-Query-CH02-26 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Buyer LIKE 'Pete%';
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen

SQL WHERE Clause Options
Character String Patterns: SQL LIKE Operator II

```
/* *** SQL-Query-CH02-27 *** */
SELECT      *
FROM        SKU_DATA
WHERE       SKU_Description LIKE '%Tent%';
```

	SKU	SKU_Description	Department	Buyer
1	201000	Half-dome Tent	Camping	Cindy Lo
2	202000	Half-dome Tent Vestibule	Camping	Cindy Lo

SQL WHERE Clause Options
Character String Patterns: SQL LIKE Operator III

```
/* *** SQL-Query-CH02-29 *** */
SELECT      *
FROM        SKU_DATA
WHERE       SKU LIKE '%2__';
```

	SKU	SKU_Description	Department	Buyer
1	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

SQL WHERE Clause Options
Using NULL Values: SQL IS NULL Operator

```
/* *** SQL-Query-CH02-30 *** */
SELECT      *
FROM        CATALOG_SKU_2015
WHERE       CatalogPage IS NULL;
```

	CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWebSite
1	20150007	203000	Half-dome Tent Vestibule - Wide	Camping	NULL	2015-04-01

SQL WHERE Clause Options
Using NULL Values: SQL IS NOT NULL Operator

```
/* *** SQL-Query-CH02-31 *** */
SELECT      *
FROM        CATALOG_SKU_2015
WHERE       CatalogPage IS NOT NULL;
```

	CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWebSite
1	20150001	100100	Std. Scuba Tank, Yellow	Water Sports	23	2015-01-01
2	20150002	100200	Std. Scuba Tank, Magenta	Water Sports	23	2015-01-01
3	20150003	101100	Dive Mask, Small Clear	Water Sports	27	2015-01-01
4	20150004	101200	Dive Mask, Med Clear	Water Sports	27	2015-01-01
5	20150005	201000	Half-dome Tent	Camping	45	2015-01-01
6	20150006	202000	Half-dome Tent Vestibule	Camping	45	2015-01-01
7	20150008	301000	Light Fly Climbing Harness	Climbing	76	2015-01-01
8	20150009	302000	Locking Carabiner, Oval	Climbing	78	2015-01-01

Performing Calculations in SQL Queries
SQL Built-in Aggregate Functions

SQL Built-in Aggregate Functions	
Function	**Meaning**
COUNT(*)	Count the number of rows in the table
COUNT ({Name})	Count the number of rows in the table where column {Name} IS NOT NULL
SUM	Calculate the sum of all values (numeric columns only)
AVG	Calculate the average of all values (numeric columns only)
MIN	Calculate the minimum value of all values
MAX	Calculate the maximum value of all values

Performing Calculations in SQL Queries
Using SQL Built-in Aggregate Functions: SUM

We can assign meaningful column names using the **SQL AS keyword**.

```
/* *** SQL-Query-CH02-33 *** */
SELECT     SUM(OrderTotal) AS OrderSum
FROM       RETAIL_ORDER;
```

	OrderSum
1	1235.00

Performing Calculations in SQL Queries

Using SQL Built-in Aggregate Functions: SUM, AVG, MIN and MAX

We can assign meaningful column names using the **SQL AS keyword**.

```
/* *** SQL-Query-CH02-35 *** */
SELECT      SUM(ExtendedPrice) AS OrderItemSum,
            AVG(ExtendedPrice) AS OrderItemAvg,
            MIN(ExtendedPrice) AS OrderItemMin,
            MAX(ExtendedPrice) AS OrderItemMax
FROM        ORDER_ITEM;
```

	OrderItemSum	OrderItemAvg	OrderItemMin	OrderItemMax
1	1180.00	168.5714	50.00	300.00

Performing Calculations in SQL Queries

Using SQL Built-in Aggregate Functions: COUNT(*)

We can assign meaningful column names using the **SQL AS keyword**.

```
/* *** SQL-Query-CH02-36 *** */
SELECT      COUNT(*) AS NumberOfRows
FROM        ORDER_ITEM;
```

	NumberOfRows
1	7

Performing Calculations in SQL Queries

Using SQL Built-in Aggregate Functions: COUNT({Name})

We can assign meaningful column names using the **SQL AS keyword**.
We use the **SQL DISTINCT keyword** to count each value only once.

```
/* *** SQL-Query-CH02-38 *** */
SELECT     COUNT(DISTINCT Department) AS DeptCount
FROM       SKU_DATA;
```

	DeptCount
1	3

Performing Calculations in SQL Queries
Using SQL Expressions

We can assign meaningful column names using the **SQL AS keyword**.

```
/* *** SQL-Query-CH02-42 *** */
SELECT     OrderNumber, SKU, (Quantity * Price) AS EP
FROM       ORDER_ITEM
ORDER BY   OrderNumber, SKU;
```

	OrderNumber	SKU	EP
1	1000	201000	300.00
2	1000	202000	130.00
3	2000	101100	200.00
4	2000	101200	100.00
5	3000	100200	300.00
6	3000	101100	100.00
7	3000	101200	50.00

Grouping Rows in SQL Queries
The SQL GROUP BY Clause

```
/* *** SQL-Query-CH02-48 *** */
SELECT     Department, COUNT(SKU) AS NumberOfCatalogItems
FROM       CATALOG_SKU_2014
WHERE      CatalogPage IS NOT NULL
GROUP BY   Department;
```

	Department	NumberOfCatalogItems
1	Camping	2
2	Climbing	2
3	Water Sports	4

Grouping Rows in SQL Queries

Department Groups in the CATALOG_SKU_2014 Table

This group of rows is for the **Water Sports** department

This SKU did not appear in the catalog

This group of rows is for the **Camping** department

This group of rows is for the **Climbing** department

	CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWebSite
1	20140001	100100	Std. Scuba Tank, Yellow	Water Sports	23	2014-01-01
2	20140002	100300	Std. Scuba Tank, Light Blue	Water Sports	23	2014-01-01
3	20140003	100400	Std. Scuba Tank, Dark Blue	Water Sports	NULL	2014-08-01
4	20140004	101100	Dive Mask, Small Clear	Water Sports	26	2014-01-01
5	20140005	101200	Dive Mask, Med Clear	Water Sports	26	2014-01-01
6	20140006	201000	Half-dome Tent	Camping	46	2014-01-01
7	20140007	202000	Half-dome Tent Vestibule	Camping	46	2014-01-01
8	20140008	301000	Light Fly Climbing Harness	Climbing	77	2014-01-01
9	20140009	302000	Locking Carabiner, Oval	Climbing	79	2014-01-01

Grouping Rows in SQL Queries

The Department Groups Summarized in Microsoft Excel 2013

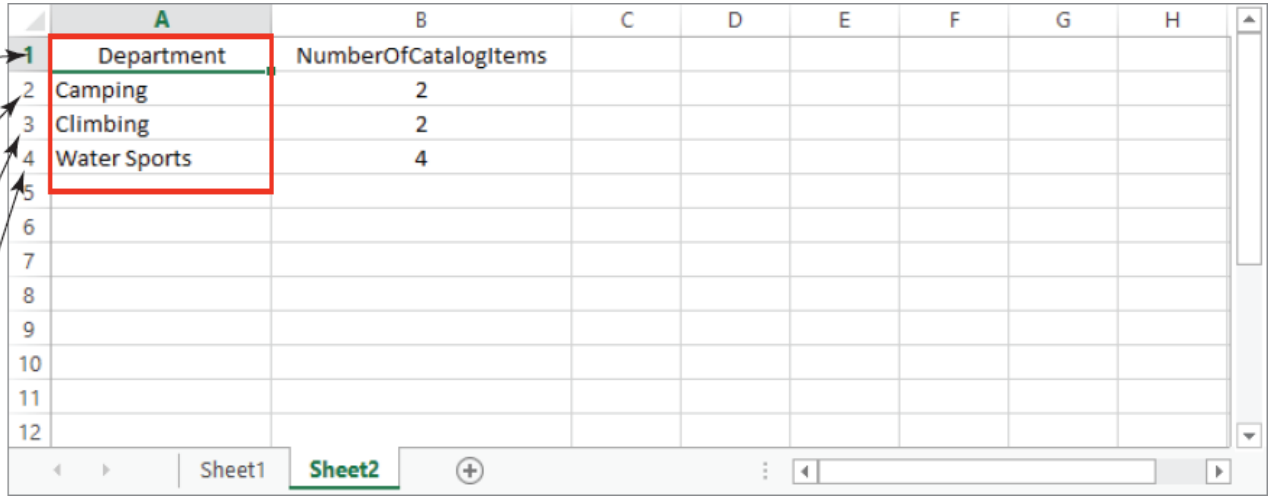
Note that only 8 of the 9 items in the CATALOG_SKU_2014 actually appeared in the catalog itself — the other item was only on the Web site.

Data is now grouped by **Department**

This is the **Camping** group

This is the **Climbing** group

This is the **Water Spots** group



	A	B	C	D	E	F	G	H
1	Department	NumberOfCatalogItems						
2	Camping	2						
3	Climbing	2						
4	Water Sports	4						
5								
6								
7								
8								
9								
10								
11								
12								

Sheet1 Sheet2

Grouping Rows in SQL Queries
The SQL HAVING Clause

The **SQL HAVING** clause controls which groups are in the query result:

```
/* *** SQL-Query-CH02-49 *** */
SELECT      Department, COUNT(SKU) AS NumberOfCatalogItems
FROM        CATALOG_SKU_2014
WHERE       CatalogPage IS NOT NULL
GROUP BY    Department
HAVING      COUNT(SKU) > 2;
```

	Department	NumberOfCatalogItems
1	Water Sports	4

In general, place **WHERE** before **GROUP BY**.

Grouping Rows in SQL Queries
Using the SQL ORDER BY Clause with Groupings

```
/* *** SQL-Query-CH02-52 *** */
SELECT      Department, COUNT(SKU) AS Dept_SKU_Count
FROM        SKU_DATA
WHERE       SKU <> 302000
GROUP BY    Department
HAVING      COUNT(SKU) > 1
ORDER BY    Dept_SKU_Count;
```

	Department	Dept_SKU_Count
1	Camping	2
2	Water Sports	4

End of Presentation:
Chapter Two Part One