Clemson Means
BUSINESS

Chapter 4
Structural Query Language (SQL)

MGT 4560 Business Information Management

Instructor: He Li

Spring 2020

Structured Query Language (SQL) Overview

- The standard for Relational Database Management Systems (RDBMS)
- History of SQL
  - **1970** – E. F. Codd develops relational database concept
  - **1974-79** – System R with Sequel (later SQL) created at IBM Research Lab
  - **1979** – Oracle markets first relational DB with SQL
  - **1981** – SQL/DS first available RDBMS system on DOS/VSE
    - Others followed: INGRES (1981), IDM (1982), DG/SGL (1984), Sybase (1986)
  - **1986** – ANSI SQL standard released
    - Major ANSI standard updates in 1989, 1992, 1999, 2003, 2006, 2008, 2011, 2016
  - **Today** – SQL is supported by most major database vendors

Purpose and Benefits of SQL Standard

- **Original Purpose of SQL Standard:**
  - Specify syntax/semantics for data definition and manipulation
  - Define data structures and basic operations
  - Enable portability of database definition and application modules
  - Specify minimal (level 1) and complete (level 2) standards
  - Allow for later growth/enhancement to standard (referential integrity, transaction management, user-defined functions, extended join operations, national character sets)
- **Benefits of SQL Standards:**
  - Reduced training costs
  - Productivity
  - Application portability
  - Application longevity
  - Reduced dependence on a single vendor
  - Cross-system communication

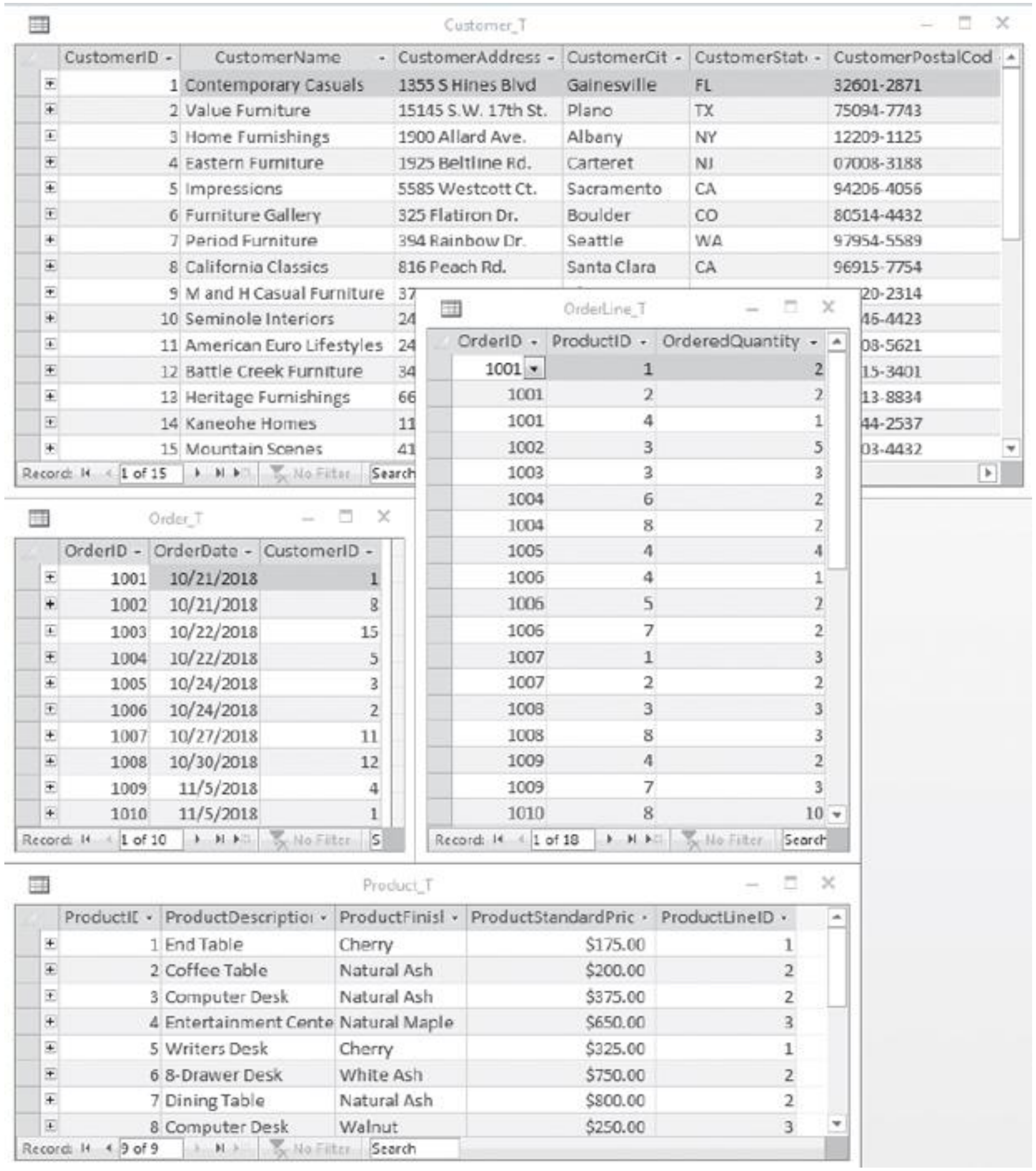
SQL Environment

- **Catalog:** A set of schemas that constitute the description of a database
- **Schema:** The structure that contains descriptions of objects created by a user (base tables, views, constraints)
- **Data Definition Language (DDL):** Commands that define a database, including creating, altering, and dropping tables and establishing constraints
- **Data Manipulation Language (DML):** Commands that maintain and query a database
- **Data Control Language (DCL):** Commands that control a database, including administering privileges and committing data

SQL Data Types

String	CHARACTER (CHAR)	Stores string values containing any characters in a character set. CHAR is defined to be a fixed length.
	CHARACTER VARYING (VARCHAR or VARCHAR2)	Stores string values containing any characters in a character set but of definable variable length.
	BINARY LARGE OBJECT (BLOB)	Stores binary string values in hexadecimal format. BLOB is defined to be a variable length. (Oracle also has CLOB and NCLOB, as well as BFILE for storing unstructured data outside the database.)
Number	NUMERIC	Stores exact numbers with a defined precision and scale.
	INTEGER (INT)	Stores exact numbers with a predefined precision and scale of zero.
Temporal	TIMESTAMP TIMESTAMP WITH LOCAL TIME ZONE	Stores a moment an event occurs, using a definable fraction-of-a-second precision.Value adjusted to the user's session time zone (available in Oracle and MySQL)
Boolean	BOOLEAN	Stores truth values: TRUE, FALSE, or UNKNOWN.

Illustrative Sample Data



Customer_T

CustomerID	CustomerName	CustomerAddress	CustomerCit	CustomerStat	CustomerPostalCod
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Period Furniture	394 Rainbow Dr.	Seattle	WA	97954-5589
8	California Classics	816 Peach Rd.	Santa Clara	CA	96915-7754
9	M and H Casual Furniture	37			20-2314
10	Seminole Interiors	24			45-4423
11	American Euro Lifestyles	24			08-5621
12	Battle Creek Furniture	34			15-3401
13	Heritage Furnishings	66			13-8834
14	Kaneohe Homes	11			44-2537
15	Mountain Scenes	41			03-4432

Record: 1 of 15 No Filter Search

Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2018	1
1002	10/21/2018	8
1003	10/22/2018	15
1004	10/22/2018	5
1005	10/24/2018	3
1006	10/24/2018	2
1007	10/27/2018	11
1008	10/30/2018	12
1009	11/5/2018	4
1010	11/5/2018	1

Record: 1 of 10 No Filter S

OrderLine_T

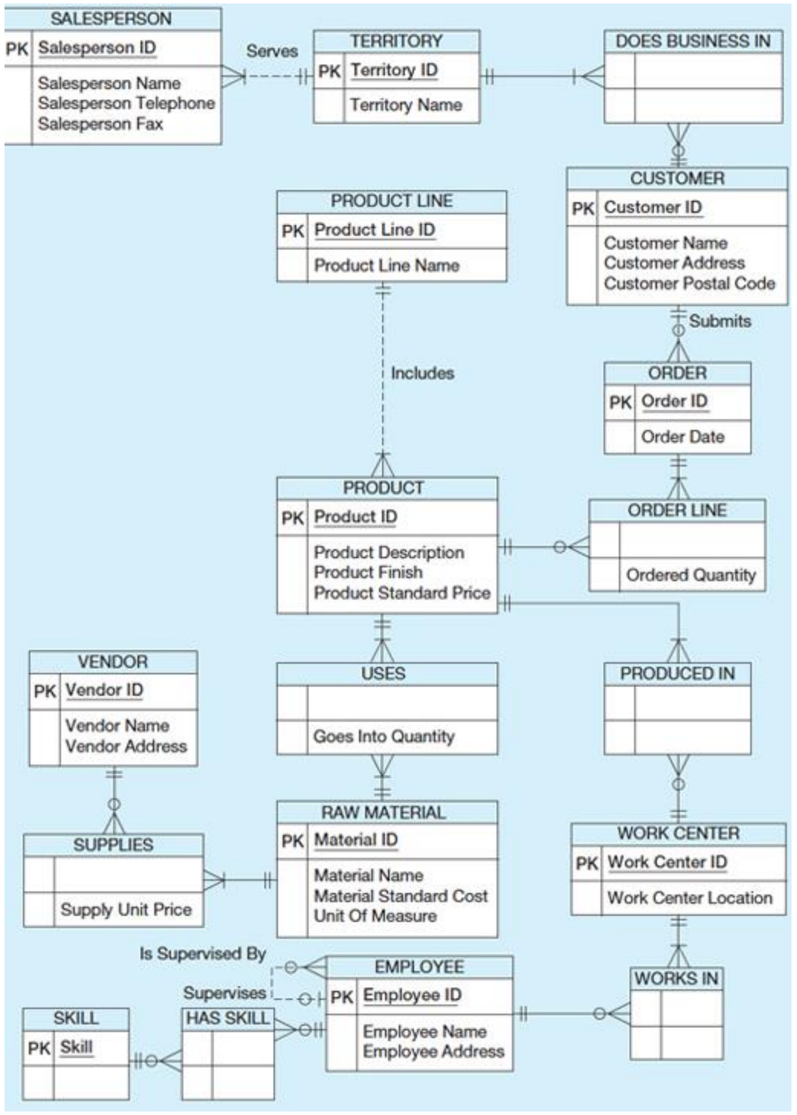
OrderID	ProductID	OrderedQuantity
1001	1	2
1001	2	2
1001	4	1
1002	3	5
1003	3	3
1004	6	2
1004	8	2
1005	4	4
1006	4	1
1006	5	2
1006	7	2
1007	1	3
1007	2	2
1008	3	3
1008	8	3
1009	4	2
1009	7	3
1010	8	10

Record: 1 of 18 No Filter Search

Product_T

ProductIE	ProductDescriptio	ProductFinisl	ProductStandardPric	ProductLineID
1	End Table	Cherry	$175.00	1
2	Coffee Table	Natural Ash	$200.00	2
3	Computer Desk	Natural Ash	$375.00	2
4	Entertainment Cente	Natural Maple	$650.00	3
5	Writers Desk	Cherry	$325.00	1
6	8-Drawer Desk	White Ash	$750.00	2
7	Dining Table	Natural Ash	$800.00	2
8	Computer Desk	Walnut	$250.00	3

Record: 9 of 9 No Filter Search

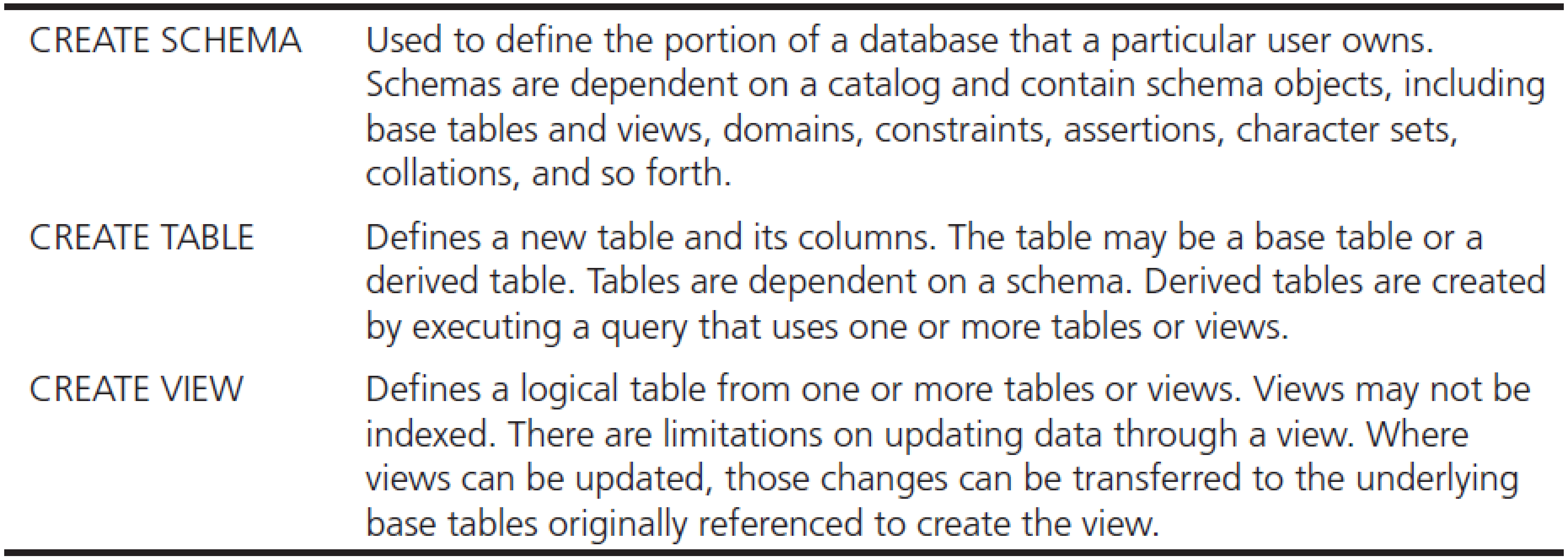


Creating a Database in SQL

- Syntax:

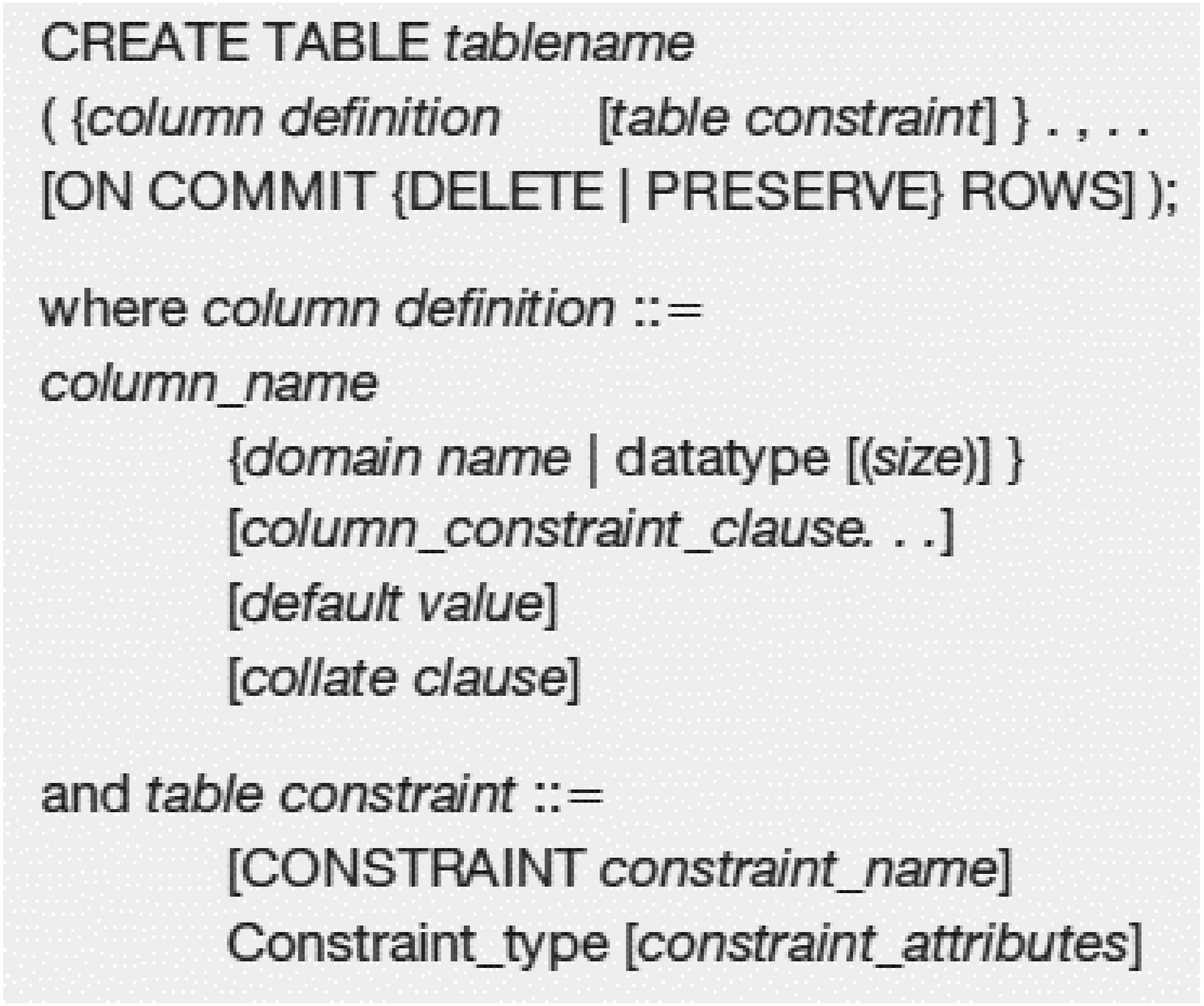
CREATE SCHEMA database_name

AUTHORIZATION owner_user id



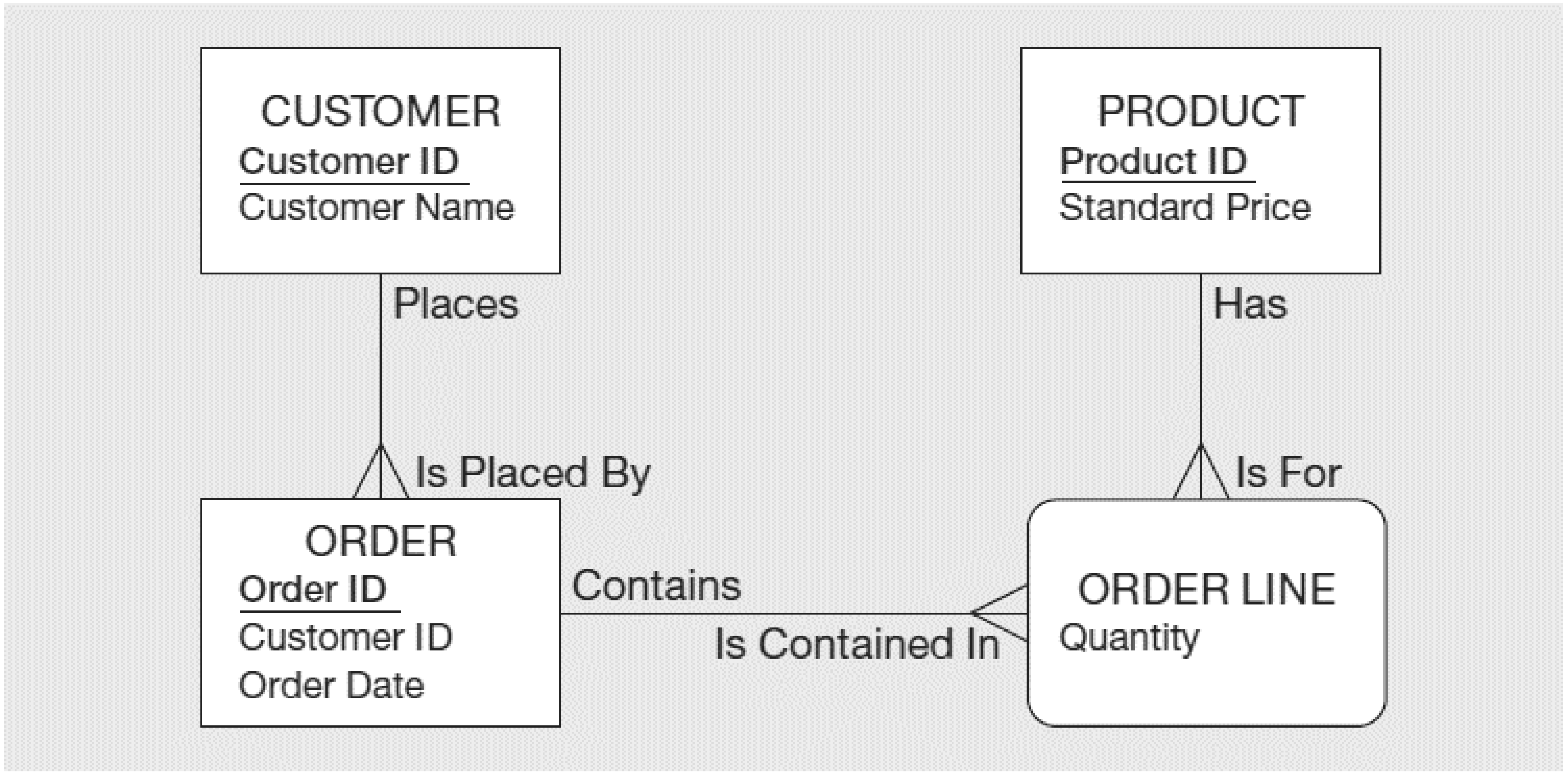
CREATE SCHEMA	Used to define the portion of a database that a particular user owns. Schemas are dependent on a catalog and contain schema objects, including base tables and views, domains, constraints, assertions, character sets, collations, and so forth.
CREATE TABLE	Defines a new table and its columns. The table may be a base table or a derived table. Tables are dependent on a schema. Derived tables are created by executing a query that uses one or more tables or views.
CREATE VIEW	Defines a logical table from one or more tables or views. Views may not be indexed. There are limitations on updating data through a view. Where views can be updated, those changes can be transferred to the underlying base tables originally referenced to create the view.

Creating Tables: General Syntax



```
CREATE TABLE tablename
( {column definition      [table constraint] } . , . .
[ON COMMIT {DELETE | PRESERVE} ROWS] );

where column definition ::=
column_name
          {domain name | datatype [(size)] }
          [column_constraint_clause. . .]
          [default value]
          [collate clause]

and table constraint ::=
          [CONSTRAINT constraint_name]
          Constraint_type [constraint_attributes]
```

The Following Slides Create Tables for This Enterprise Data Model



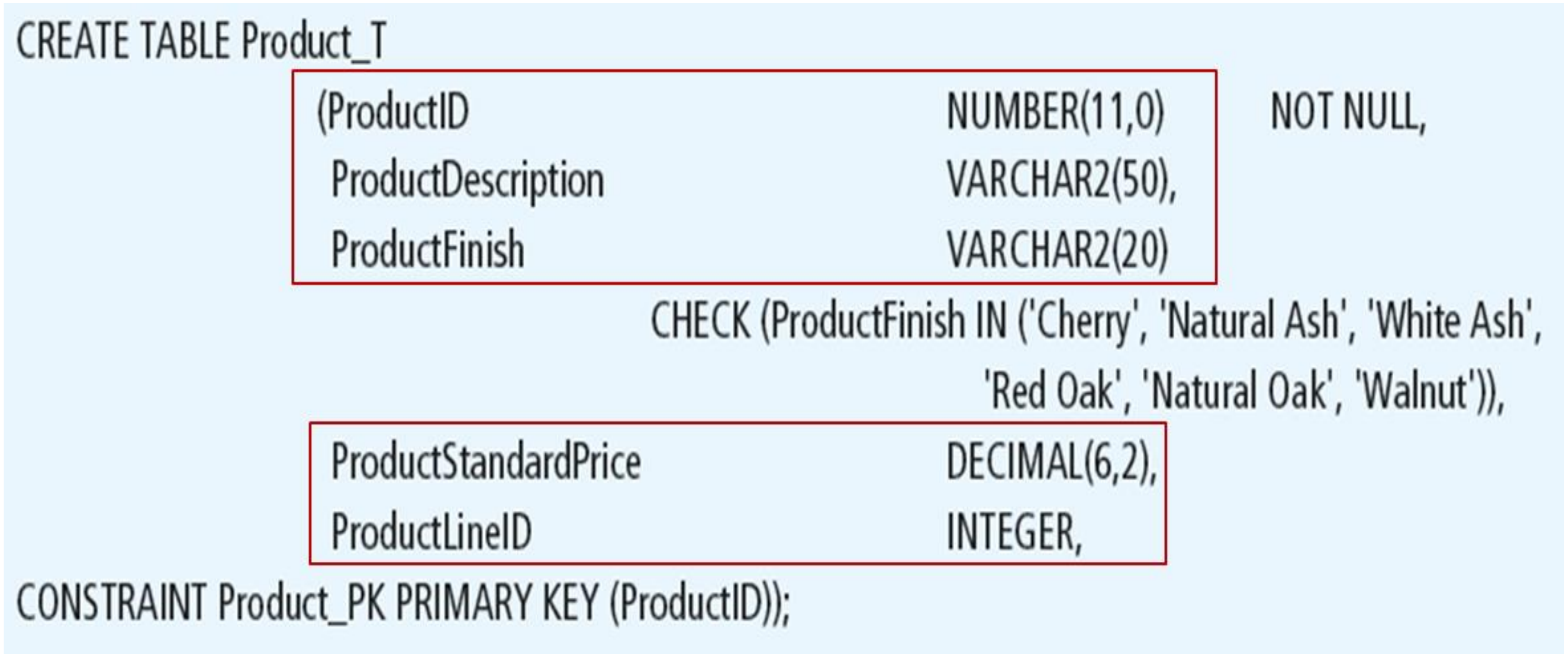
CLEMSON College of BUSINESS

SQL Database Definition Commands

```
CREATE TABLE Customer_T
                    (CustomerID                          NUMBER(11,0)        NOT NULL,
                     CustomerName                        VARCHAR2(25)        NOT NULL,
                     CustomerAddress                     VARCHAR2(30),
                     CustomerCity                        VARCHAR2(20),
                     CustomerState                       CHAR(2),
                     CustomerPostalCode                  VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
                   (OrderID                              NUMBER(11,0)        NOT NULL,
                    OrderDate                            DATE DEFAULT SYSDATE,
                    CustomerID                           NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T(CustomerID));

CREATE TABLE Product_T
                   (ProductID                            NUMBER(11,0)        NOT NULL,
                    ProductDescription                   VARCHAR2(50),
                    ProductFinish                        VARCHAR2(20)
                                     CHECK (ProductFinish IN ('Cherry', 'Natural Ash', 'White Ash',
                                                        'Red Oak', 'Natural Oak', 'Walnut')),
                    ProductStandardPrice                 DECIMAL(6,2),
                    ProductLineID                        INTEGER,
CONSTRAINT Product_PK PRIMARY KEY (ProductID));

CREATE TABLE OrderLine_T
                   (OrderID                              NUMBER(11,0)        NOT NULL,
                    ProductID                            INTEGER             NOT NULL,
                    OrderedQuantity                      NUMBER(11,0),
CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T(OrderID),
CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T(ProductID));
```

Four table definitions

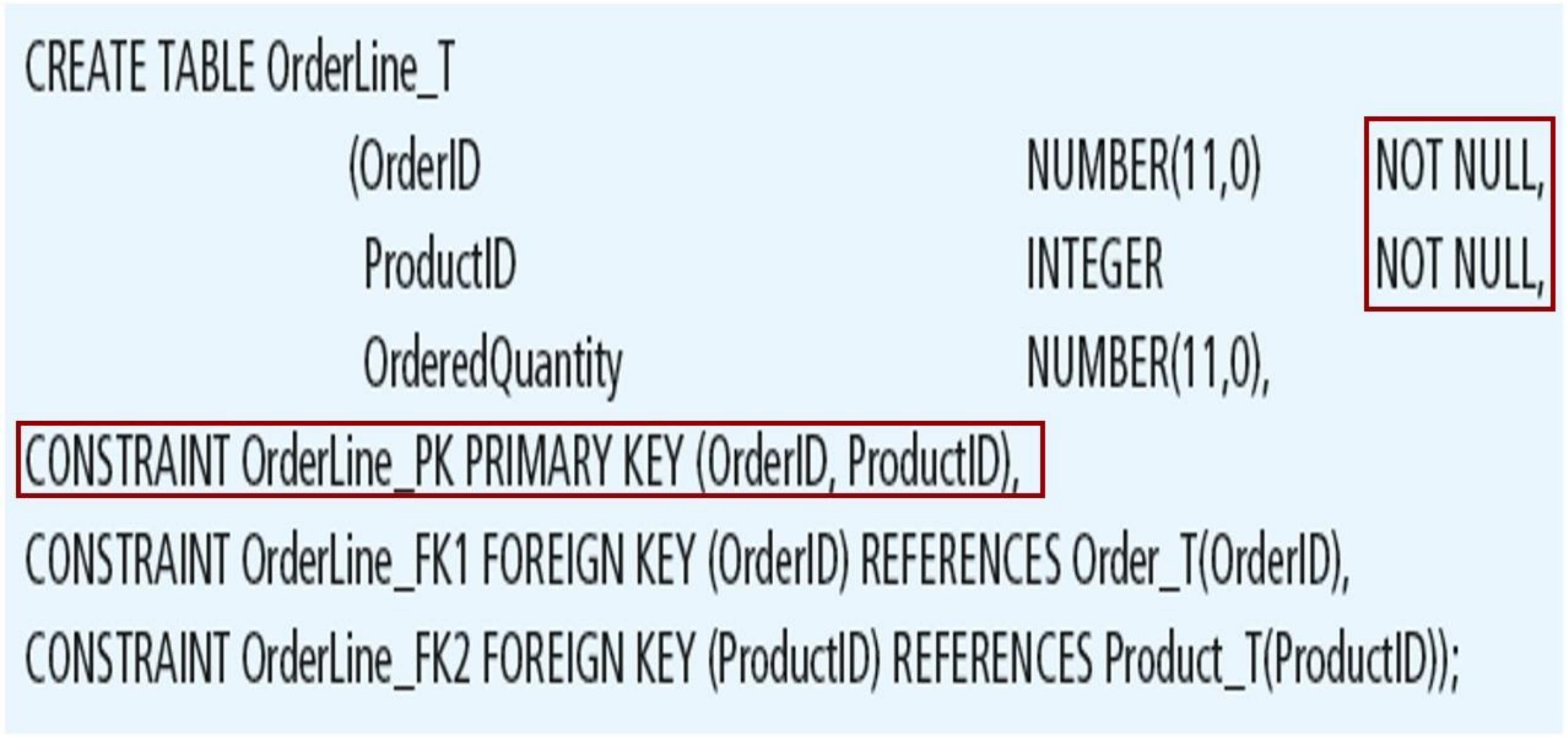
Defining Attributes and Their Data Types



```
CREATE TABLE Product_T
        (ProductID                  NUMBER(11,0)     NOT NULL,
         ProductDescription         VARCHAR2(50),
         ProductFinish              VARCHAR2(20)
                   CHECK (ProductFinish IN ('Cherry', 'Natural Ash', 'White Ash',
                              'Red Oak', 'Natural Oak', 'Walnut')),
         ProductStandardPrice       DECIMAL(6,2),
         ProductLineID              INTEGER,
CONSTRAINT Product_PK PRIMARY KEY (ProductID));
```

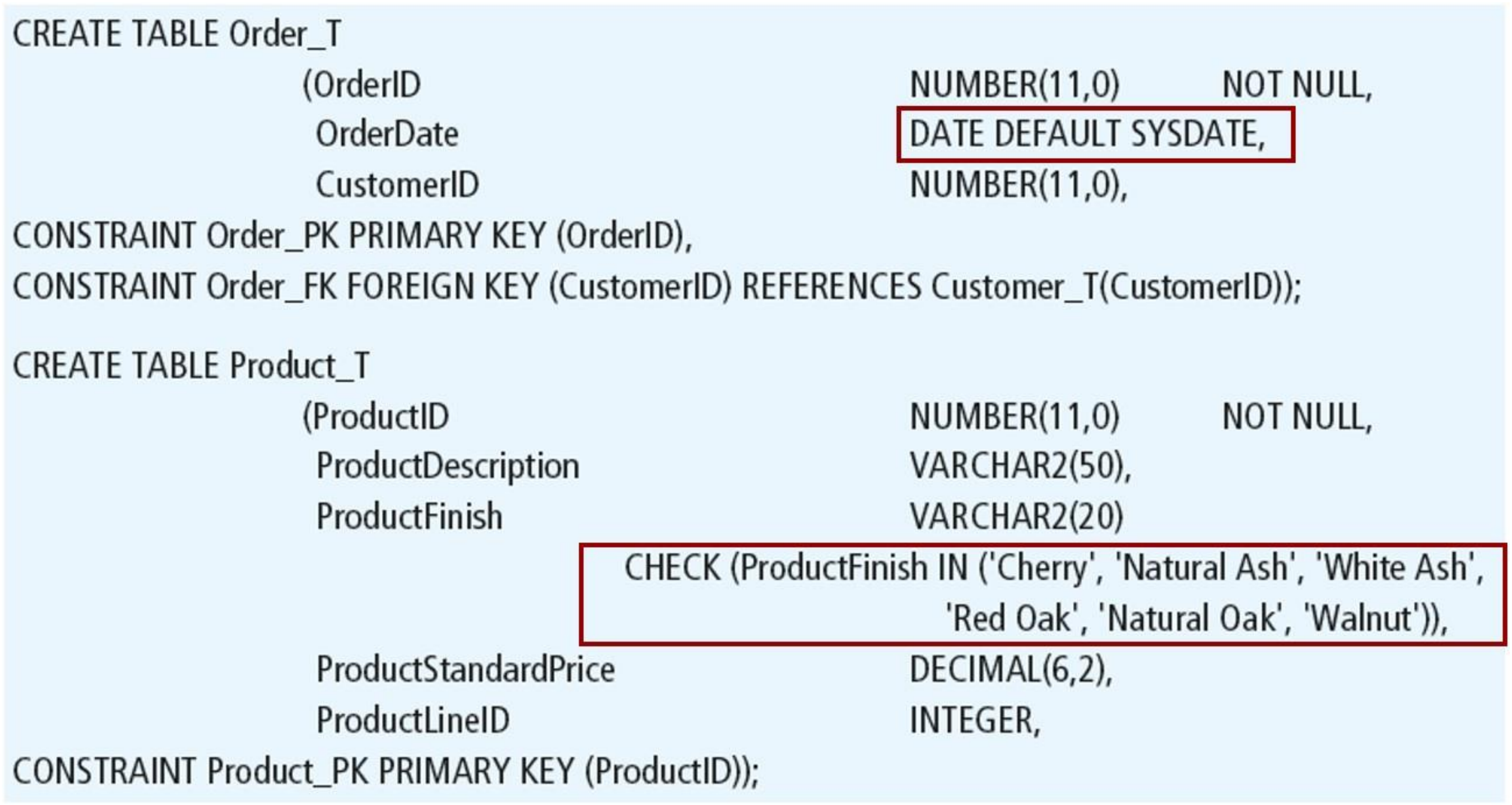
Non-Nullable Specifications

- Some primary keys are composite – composed of multiple attributes



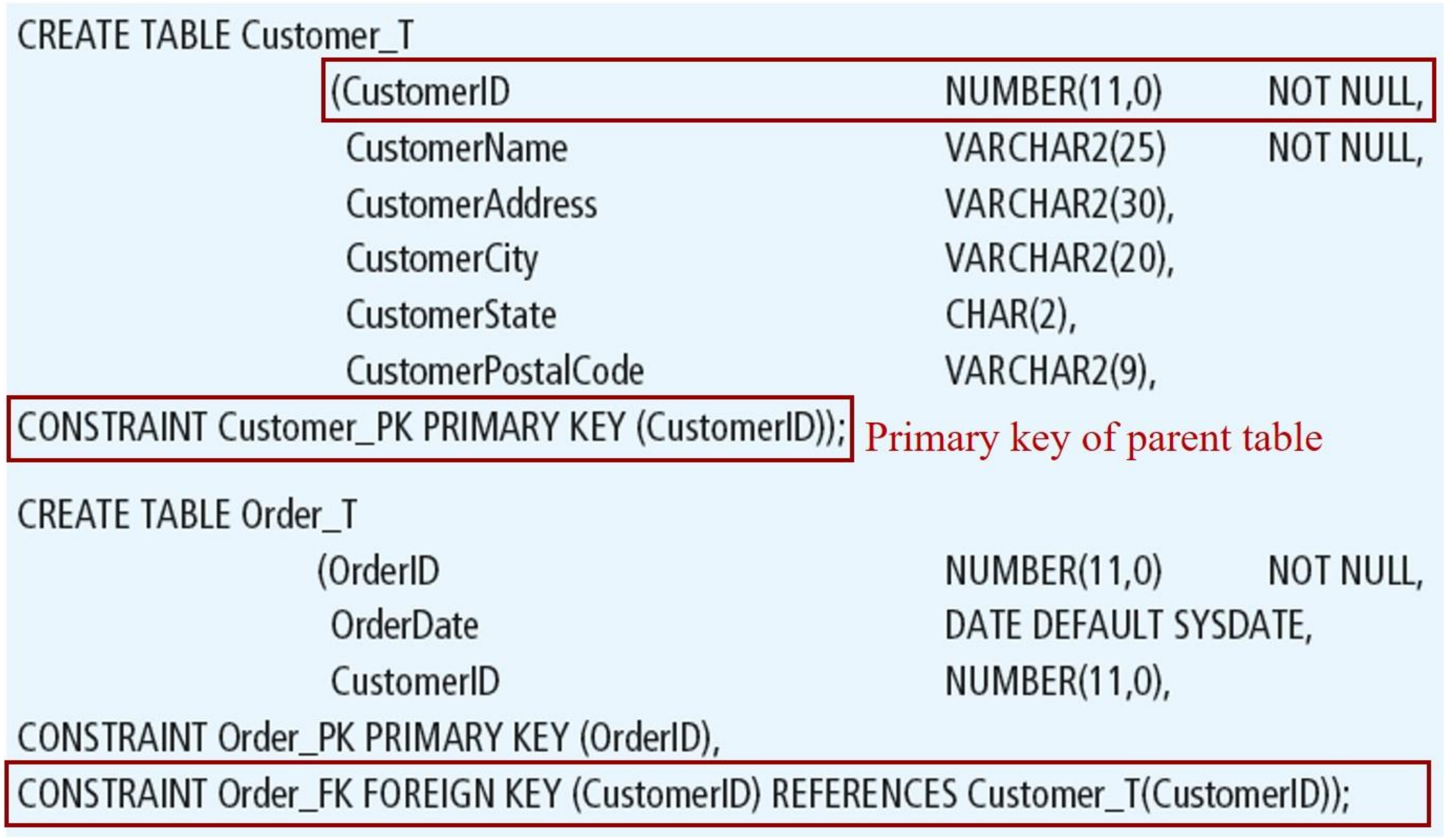
```
CREATE TABLE OrderLine_T
          (OrderID                    NUMBER(11,0)    NOT NULL,
           ProductID                  INTEGER         NOT NULL,
           OrderedQuantity            NUMBER(11,0),
CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T(OrderID),
CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T(ProductID));
```

Controlling the Values in Attributes



```
CREATE TABLE Order_T
            (OrderID                              NUMBER(11,0)         NOT NULL,
             OrderDate                            DATE DEFAULT SYSDATE,
             CustomerID                           NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T(CustomerID));

CREATE TABLE Product_T
            (ProductID                            NUMBER(11,0)         NOT NULL,
             ProductDescription                   VARCHAR2(50),
             ProductFinish                        VARCHAR2(20)
                          CHECK (ProductFinish IN ('Cherry', 'Natural Ash', 'White Ash',
                                        'Red Oak', 'Natural Oak', 'Walnut')),
             ProductStandardPrice                 DECIMAL(6,2),
             ProductLineID                        INTEGER,
CONSTRAINT Product_PK PRIMARY KEY (ProductID));
```

Identifying Foreign Keys and Establishing Relationships



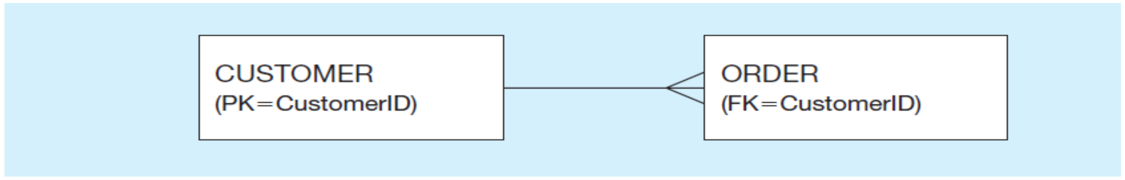
```
CREATE TABLE Customer_T
                    (CustomerID                          NUMBER(11,0)        NOT NULL,
                     CustomerName                        VARCHAR2(25)        NOT NULL,
                     CustomerAddress                     VARCHAR2(30),
                     CustomerCity                        VARCHAR2(20),
                     CustomerState                       CHAR(2),
                     CustomerPostalCode                  VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));
```

Primary key of parent table

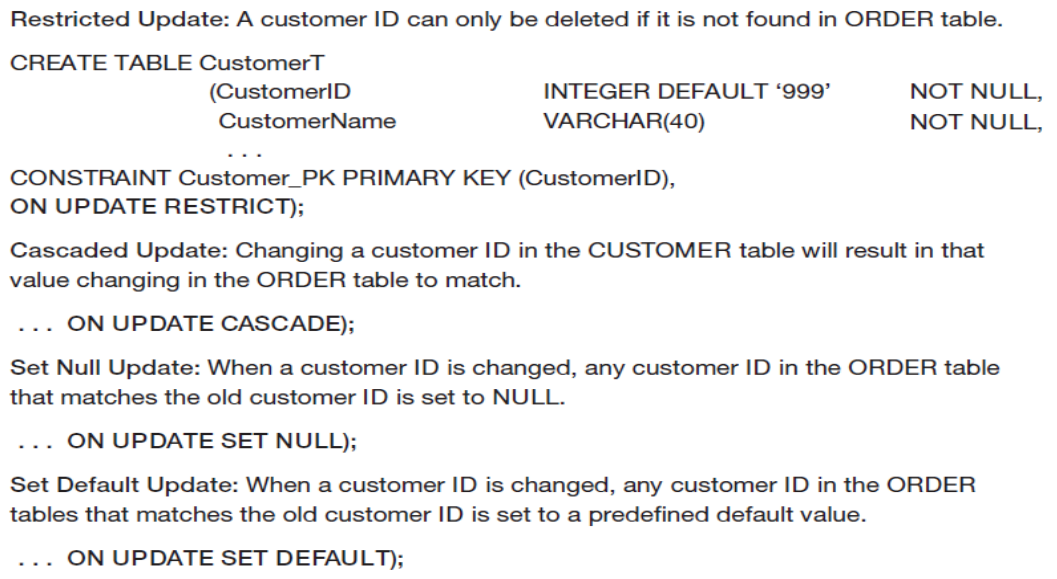
```
CREATE TABLE Order_T
                    (OrderID                             NUMBER(11,0)        NOT NULL,
                     OrderDate                           DATE DEFAULT SYSDATE,
                     CustomerID                          NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T(CustomerID));
```

Data Integrity Controls

- Referential integrity – constraint that ensures that foreign key values of a table must match primary key values of a related table in a 1:N relationship



- Restricting:
  - Deletes of primary records
  - Updates of primary records
  - Inserts of dependent records
- Relational integrity is enforced via the primary-key to foreign-key match



Restricted Update: A customer ID can only be deleted if it is not found in ORDER table.

```
CREATE TABLE CustomerT
                (CustomerID            INTEGER DEFAULT '999'       NOT NULL,
                 CustomerName          VARCHAR(40)                 NOT NULL,
                  . . .
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID),
ON UPDATE RESTRICT);
```

Cascaded Update: Changing a customer ID in the CUSTOMER table will result in that value changing in the ORDER table to match.

```
 . . . ON UPDATE CASCADE);
```

Set Null Update: When a customer ID is changed, any customer ID in the ORDER table that matches the old customer ID is set to NULL.

```
 . . . ON UPDATE SET NULL);
```

Set Default Update: When a customer ID is changed, any customer ID in the ORDER tables that matches the old customer ID is set to a predefined default value.

```
 . . . ON UPDATE SET DEFAULT);
```

Changing Table Definitions

- ALTER TABLE statement allows you to change column specifications:

  ALTER TABLE table_name alter_table_action;

- Table Actions:

  ADD [COLUMN] column_definition
  ALTER [COLUMN] column_name **SET DEFAULT** default-value
  ALTER [COLUMN] column_name **DROP DEFAULT**
  DROP [COLUMN] column_name **[RESTRICT] [CASCADE]**
  ADD table_constraint

- Example (adding a new column with a default value):

```
ALTER TABLE CUSTOMER_T ADD COLUMN
CustomerType VARCHAR2 (10) DEFAULT
“Commercial”;
```

Removing Tables

- DROP TABLE statement allows you to remove tables from your schema:

```
DROP TABLE CUSTOMER_T
```

Inserting Data

- Adds one or more rows to a table
- Inserting into a table:

```
INSERT INTO Customer_T VALUES (001, ‘Contemporary
Casuals’, ‘1355 S. Himes Blvd.’, ‘Gainesville’, ‘FL’, 32601);
```

- Inserting a record that has some null attributes requires identifying the fields that actually get data:

```
INSERT INTO Product_T (ProductID, ProductDescription,
ProductFinish, ProductStandardPrice) VALUES (1, ‘End
Table’, ‘Cherry’, 175, 8);
```

- Inserting from another table:

```
INSERT INTO CaCustomer_T SELECT * FROM
Customer_T WHERE CustomerState = ‘CA’;
```

Deleting Data

- Removes rows from a table
- Delete certain rows

```
DELETE FROM CUSTOMER_T WHERE
CUSTOMERSTATE = 'HI';
```

- Delete all rows

```
DELETE FROM CUSTOMER_T;
```

Updating and Merging Data

- **Updating Data:** Modifies data in existing rows

```
UPDATE Product_T SET
ProductStandardPrice = 775 WHERE
ProductID = 7;
```

- **Merging Data:** Makes it easier to update a table. It allows combination of Insert and Update in one statement. It is useful for updating master tables with new data.

```
MERGE INTO Product_T AS PROD
USING
(SELECT ProductID, ProductDescription, ProductFinish,
ProductStandardPrice, ProductLineID FROM Purchases_T) AS PURCH
      ON (PROD.ProductID = PURCH.ProductID)
WHEN MATCHED THEN UPDATE
      PROD.ProductStandardPrice = PURCH.ProductStandardPrice
WHEN NOT MATCHED THEN INSERT
      (ProductID, ProductDescription, ProductFinish, ProductStandardPrice,
      ProductLineID)
      VALUES(PURCH.ProductID, PURCH.ProductDescription,
      PURCH.ProductFinish, PURCH.ProductStandardPrice,
         PURCH.ProductLineID);
```

PART I: Queries for Single Tables

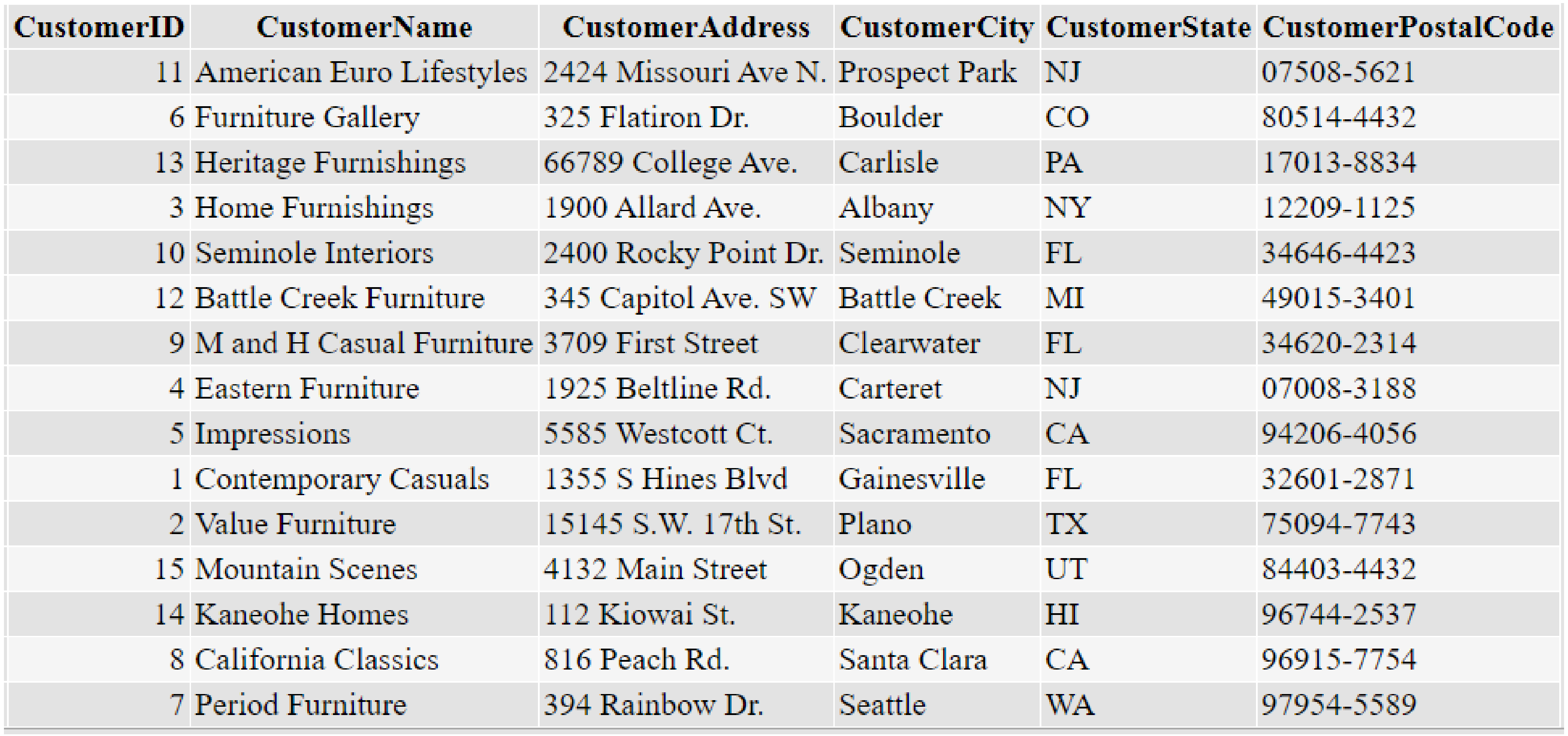
- Clauses of the **SELECT** statement:
  - SELECT: List the columns (and expressions) to be returned from the query
  - FROM: Indicate the table(s) or view(s) from which data will be obtained
  - WHERE: Indicate the conditions under which a row will be included in the result
  - GROUP BY: Indicate categorization of results
  - HAVING: Indicate the conditions under which a category (group) will be included
  - ORDER BY: Sorts the result according to specified criteria

```
SELECT [ALL | DISTINCT]
  {* | TABLE.* | expression [AS alias] [, expression [AS Alias] ...}
    FROM tablename [alias] [, tablename [alias] ...
      [WHERE condition]
      [GROUP BY expression [, expression] ... ] [HAVING condition]
[ { UNION [ALL] | INTERSECT | MINUS } SELECT ... ]
      [ORDER BY {expression | position} [ASC | DESC] [, expression | position
[ASC | DESC ] ...
```

Basic SELECT Statement

- **Q1:** List all the data about products

```
SELECT *
FROM CUSTOMER_T;
```

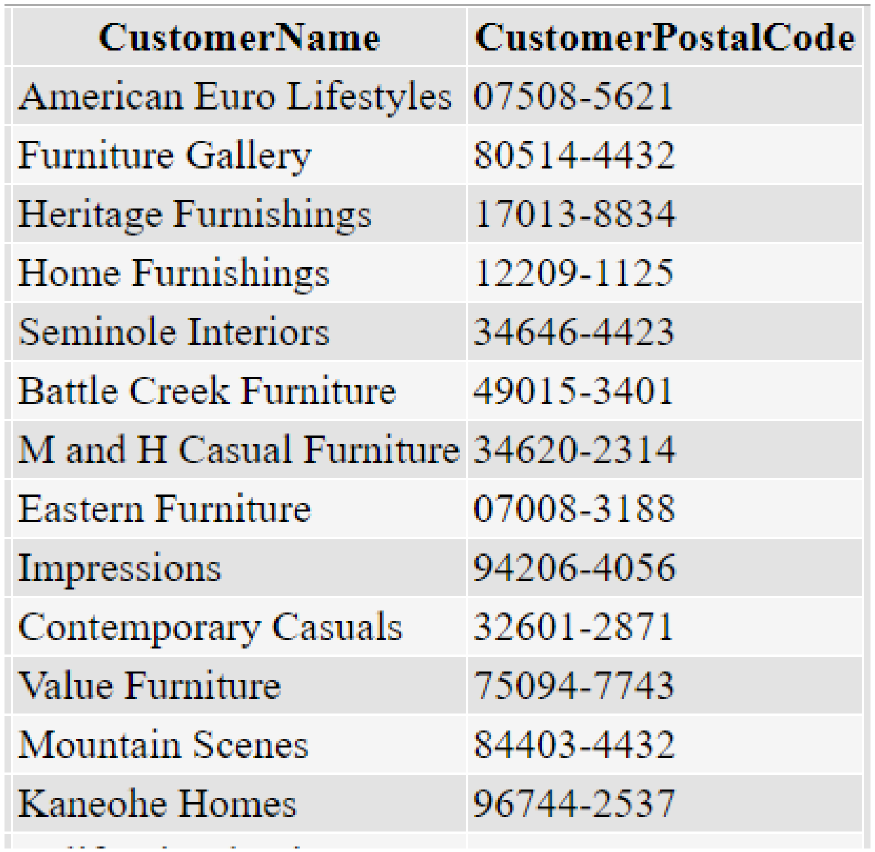


CustomerID	CustomerName	CustomerAddress	CustomerCity	CustomerState	CustomerPostalCode
11	American Euro Lifestyles	2424 Missouri Ave N.	Prospect Park	NJ	07508-5621
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
13	Heritage Furnishings	66789 College Ave.	Carlisle	PA	17013-8834
3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125
10	Seminole Interiors	2400 Rocky Point Dr.	Seminole	FL	34646-4423
12	Battle Creek Furniture	345 Capitol Ave. SW	Battle Creek	MI	49015-3401
9	M and H Casual Furniture	3709 First Street	Clearwater	FL	34620-2314
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
15	Mountain Scenes	4132 Main Street	Ogden	UT	84403-4432
14	Kaneohe Homes	112 Kiowai St.	Kaneohe	HI	96744-2537
8	California Classics	816 Peach Rd.	Santa Clara	CA	96915-7754
7	Period Furniture	394 Rainbow Dr.	Seattle	WA	97954-5589

Basic SELECT Statement

- **Q2:** List name and post code for all customers

```
SELECT CustomerName, CustomerPostalCode
FROM CUSTOMER_T;
```

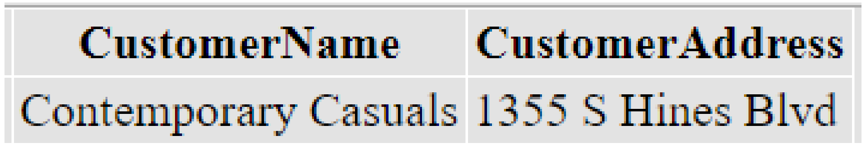


CustomerName	CustomerPostalCode
American Euro Lifestyles	07508-5621
Furniture Gallery	80514-4432
Heritage Furnishings	17013-8834
Home Furnishings	12209-1125
Seminole Interiors	34646-4423
Battle Creek Furniture	49015-3401
M and H Casual Furniture	34620-2314
Eastern Furniture	07008-3188
Impressions	94206-4056
Contemporary Casuals	32601-2871
Value Furniture	75094-7743
Mountain Scenes	84403-4432
Kaneohe Homes	96744-2537

Using Comparison Operators

- **Q3:** List the name and address of all customers in postal code 32601-2871

```
SELECT CustomerName, CustomerAddress
FROM CUSTOMER_T
WHERE CustomerPostalCode = '32601-2871';
```



CustomerName	CustomerAddress
Contemporary Casuals	1355 S Hines Blvd

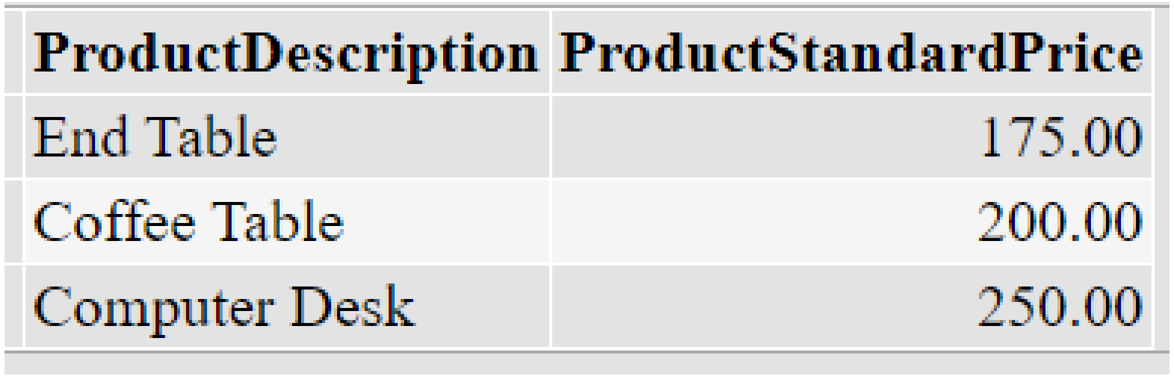
Comparison operators include:

= Equal to
> Greater than
>= Greater than or equal to
< Less than
<= Less than or equal to
<> Not equal to

Using Comparison Operators

- **Q4:** Which products have a standard price of less than $275? Show the product description and standard price.

```
SELECT ProductDescription, ProductStandardPrice
FROM Product_T
WHERE ProductStandardPrice < 275;
```

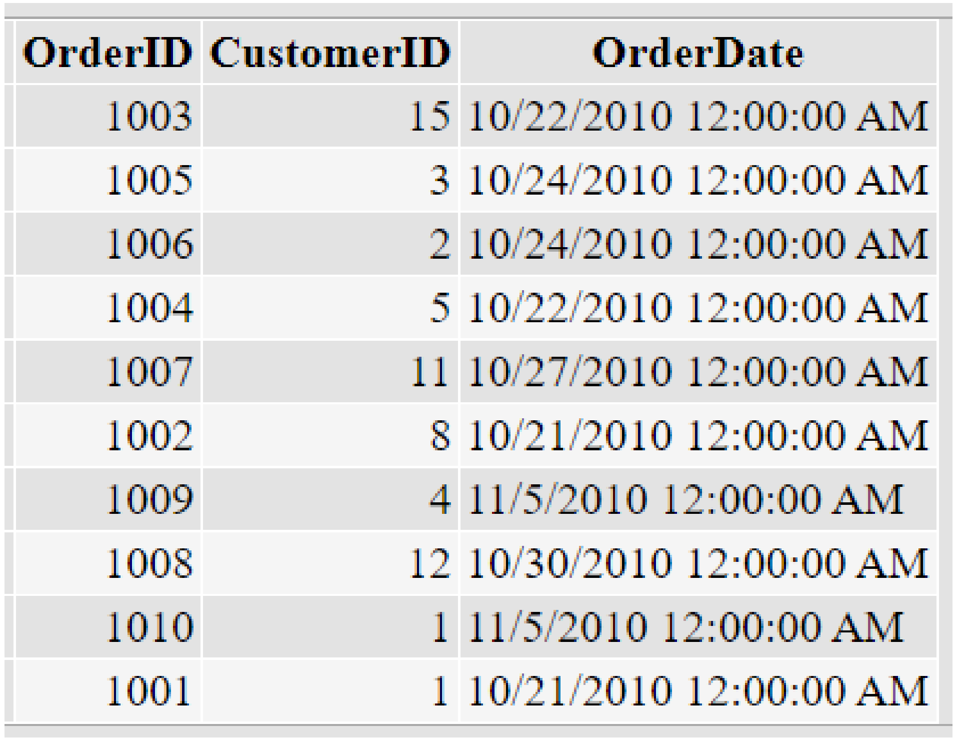


ProductDescription	ProductStandardPrice
End Table	175.00
Coffee Table	200.00
Computer Desk	250.00

Using Comparison Operators

- **Q5:** List OrderID, CustomerID, and OrderDate for all orders ordered since August 1, 2010.

```
SELECT OrderID, CustomerID, OrderDate
FROM Order_T
WHERE OrderDate > '2010-08-01';
```

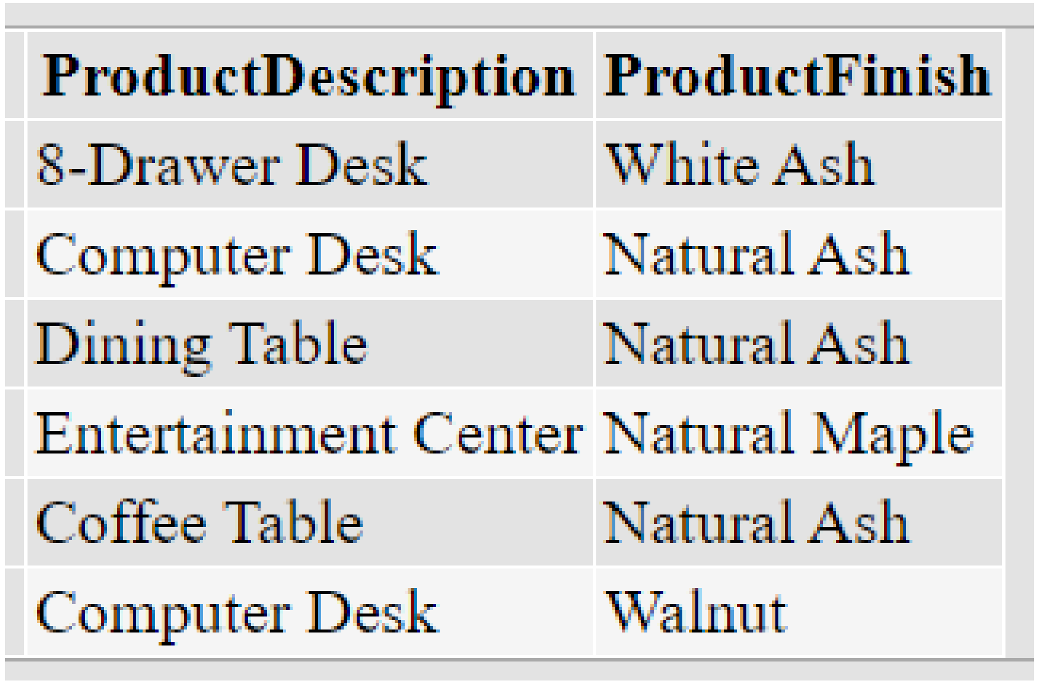


OrderID	CustomerID	OrderDate
1003	15	10/22/2010 12:00:00 AM
1005	3	10/24/2010 12:00:00 AM
1006	2	10/24/2010 12:00:00 AM
1004	5	10/22/2010 12:00:00 AM
1007	11	10/27/2010 12:00:00 AM
1002	8	10/21/2010 12:00:00 AM
1009	4	11/5/2010 12:00:00 AM
1008	12	10/30/2010 12:00:00 AM
1010	1	11/5/2010 12:00:00 AM
1001	1	10/21/2010 12:00:00 AM

Using Comparison Operators

- **Q6:** List ProductDescription and ProductFinish of products that isn’t made of cherry.

```
SELECT ProductDescription, ProductFinish
FROM Product_T
WHERE ProductFinish <> 'Cherry';
```



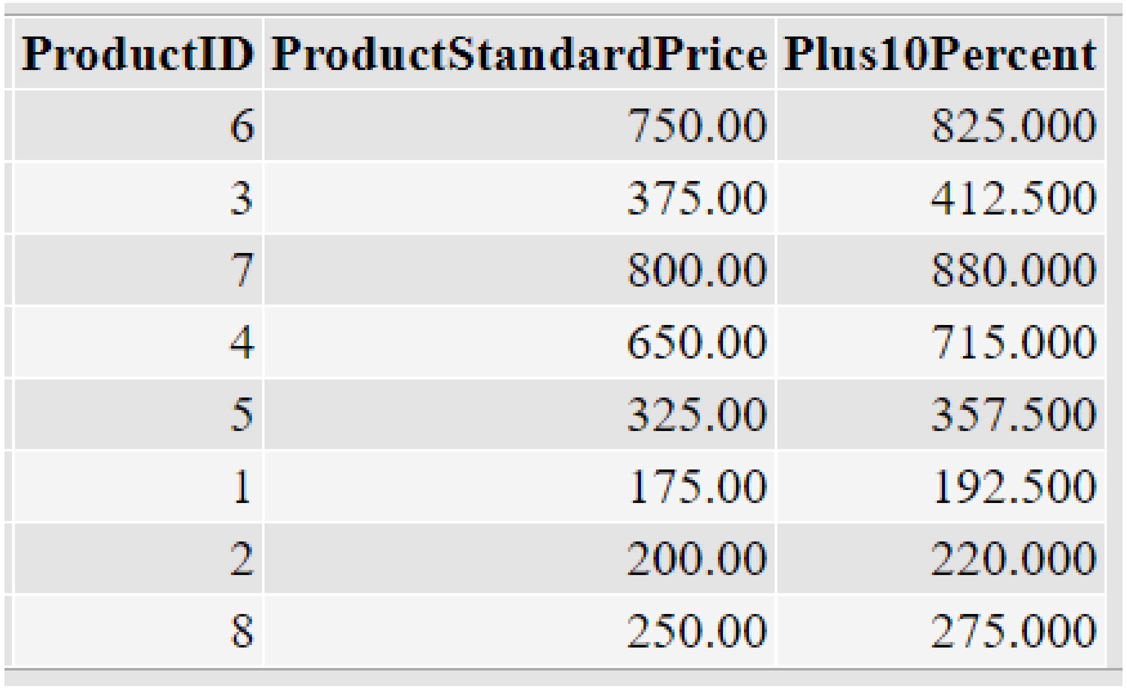
ProductDescription	ProductFinish
8-Drawer Desk	White Ash
Computer Desk	Natural Ash
Dining Table	Natural Ash
Entertainment Center	Natural Maple
Coffee Table	Natural Ash
Computer Desk	Walnut

Using Expressions

CLEMSON College of BUSINESS

- **Q7:** What are the standard price and standard price if increased by 10 percent for every product?

```
SELECT ProductID, ProductStandardPrice,
ProductStandardPrice*1.1 AS Plus10Percent
FROM Product_T;
```



ProductID	ProductStandardPrice	Plus10Percent
6	750.00	825.000
3	375.00	412.500
7	800.00	880.000
4	650.00	715.000
5	325.00	357.500
1	175.00	192.500
2	200.00	220.000
8	250.00	275.000

Using Functions

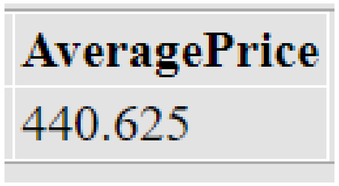
- **Common SQL functions:**

Mathematical	MIN, MAX, COUNT, SUM, ROUND (to round up a number to a specific number of decimal places), TRUNC (to truncate insignificant digits), and MOD (for modular arithmetic)
String	LOWER (to change to all lower case), UPPER (to change to all capital letters), INITCAP (to change to only an initial capital letter), CONCAT (to concatenate), SUBSTR (to isolate certain character positions), and COALESCE (finding the first not NULL values in a list of columns)
Date	NEXT_DAY (to compute the next date in sequence), ADD_MONTHS (to compute a date a given number of months before or after a given date), and MONTHS_BETWEEN (to compute the number of months between specified dates)
Analytical	TOP (find the top n values in a set, e.g., the top 5 customers by total annual sales)

Using Expressions

- **Q8:** What is the average standard price for all products in inventory?

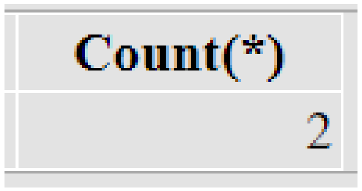
```
SELECT AVG(ProductStandardPrice) AS AveragePrice
FROM Product_T;
```



AveragePrice
440.625

- **Q9:** How many different items were ordered on order number 1004?

```
SELECT COUNT(*)
FROM OrderLine_T
WHERE OrderID = 1004;
```



Count(*)
2

Using Null Values

- **Q10:** Display all customers for whom we do not know their postal code.

```
SELECT *
FROM Customer_T
WHERE CustomerPostalCode IS NULL;
```

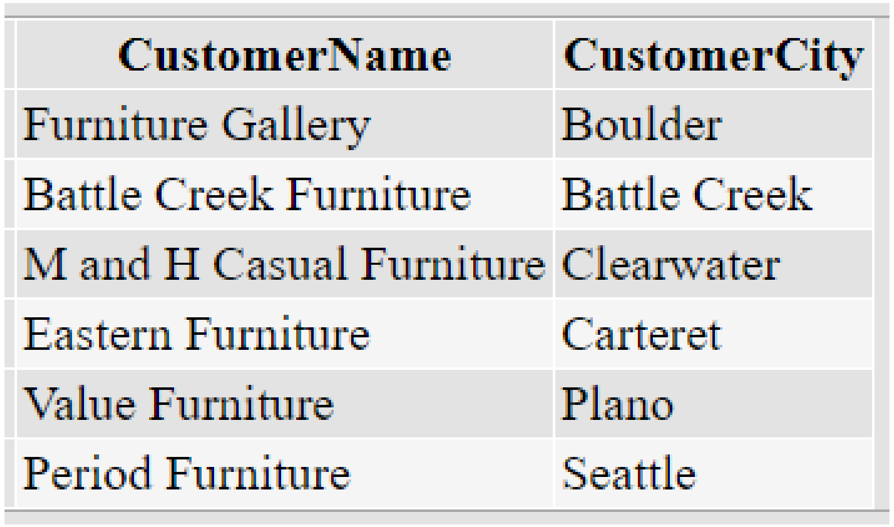
- **Q11:** Display all customers for whom we know their postal code.

```
SELECT *
FROM Customer_T
WHERE CustomerPostalCode IS NOT NULL;
```

Using Wildcards

- **Q12:** List CustomerName and CustomerCity for customers whose name includes 'Furniture'

```
SELECT CustomerName, CustomerCity
FROM Customer_T
WHERE CustomerName LIKE '%Furniture%';
```



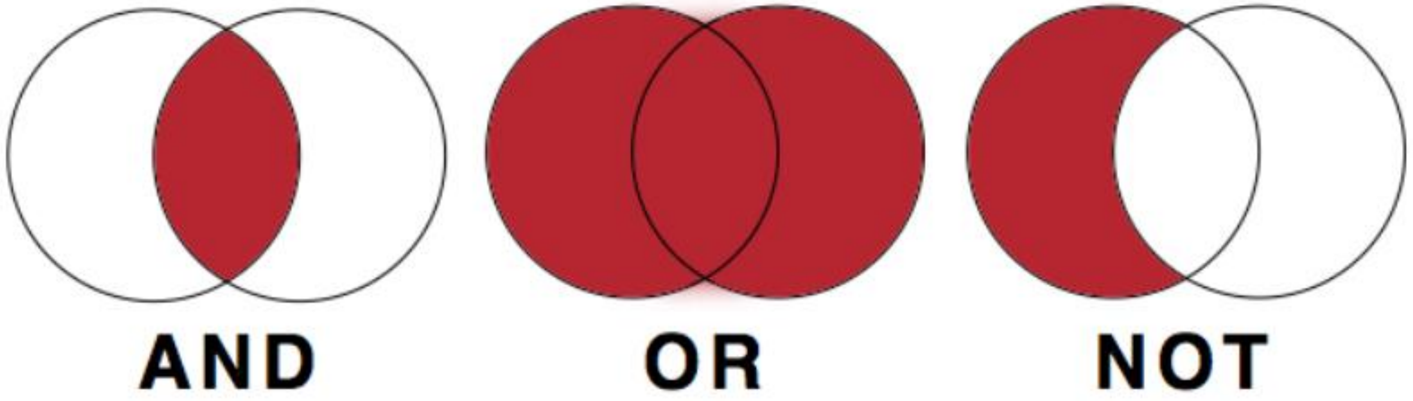
CustomerName	CustomerCity
Furniture Gallery	Boulder
Battle Creek Furniture	Battle Creek
M and H Casual Furniture	Clearwater
Eastern Furniture	Carteret
Value Furniture	Plano
Period Furniture	Seattle

% any collection of characters

_ exactly one character

Using Boolean Operators

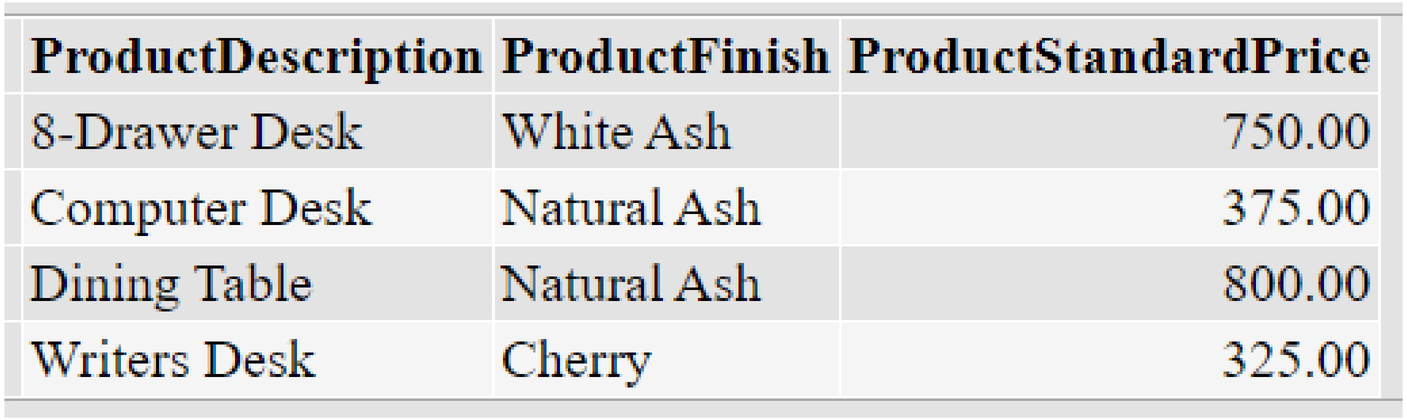
AND	Joins two or more conditions and returns results only when all conditions are true.
OR	Joins two or more conditions and returns results when any conditions are true.
NOT	Negates an expression.



Using Boolean Operators

- **Q13:** List product name, finish, and standard price for all desks and all tables that costs more than $300 in the Product table.

```
SELECT ProductDescription, ProductFinish, ProductStandardPrice
FROM Product_T
WHERE (ProductDescription LIKE '%Desk'
  OR ProductDescription LIKE '%Table')
  AND ProductStandardPrice > 300;
```

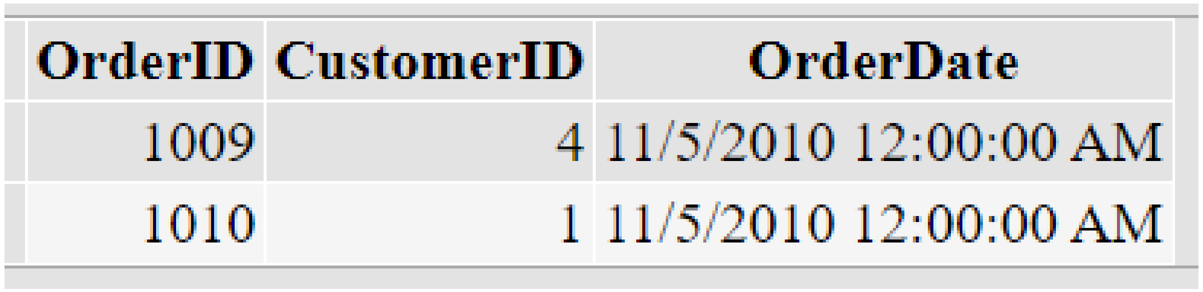


ProductDescription	ProductFinish	ProductStandardPrice
8-Drawer Desk	White Ash	750.00
Computer Desk	Natural Ash	375.00
Dining Table	Natural Ash	800.00
Writers Desk	Cherry	325.00

Using Ranges for Qualification

- **Q14:** List OrderID, CustomerID, and OrderDate for all orders ordered during the first 10 days of November, 2010.

```
SELECT OrderID, CustomerID, OrderDate
FROM Order_T
WHERE OrderDate BETWEEN '2010-11-01' AND '2010-11-10';
```

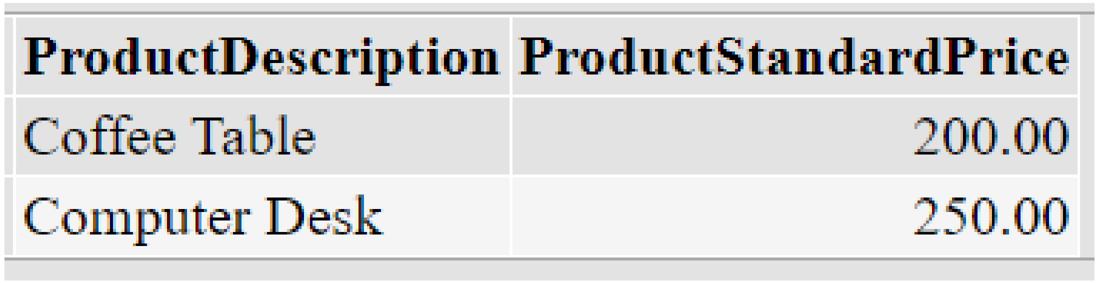


OrderID	CustomerID	OrderDate
1009	4	11/5/2010 12:00:00 AM
1010	1	11/5/2010 12:00:00 AM

Using Ranges for Qualification

- **Q15:** Which products in the Product table have a standard price between $200 and $300.

```
SELECT ProductDescription, ProductStandardPrice
FROM Product_T
WHERE ProductStandardPrice BETWEEN 200 AND 300;
```

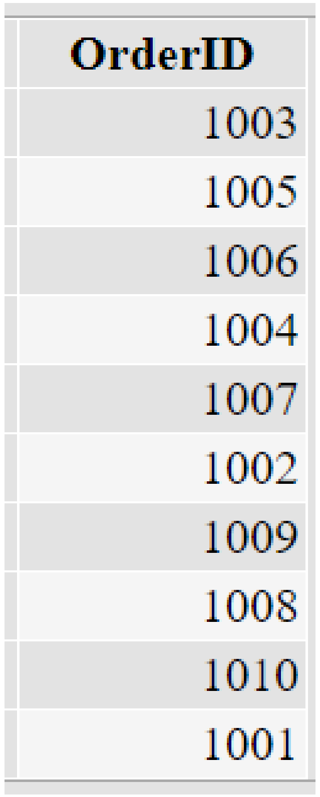


ProductDescription	ProductStandardPrice
Coffee Table	200.00
Computer Desk	250.00

Using Distinct Values

- **Q16:** What unique order numbers are included in the OrderLine table?

```
SELECT DISTINCT OrderID
FROM OrderLine_T;
```

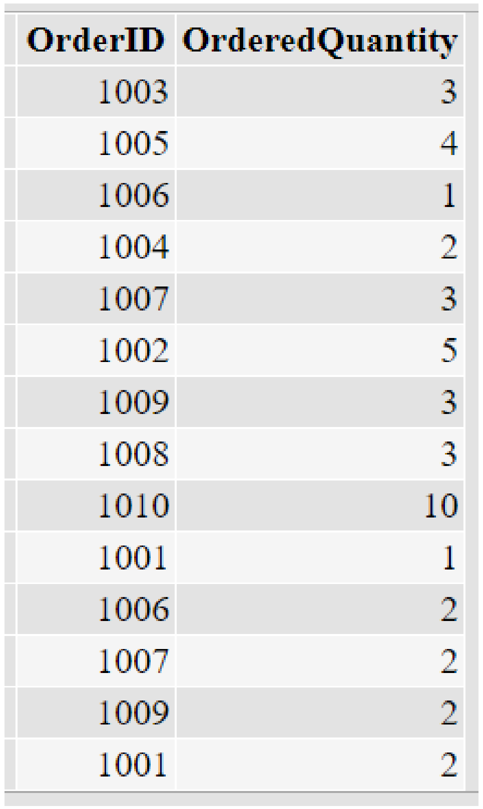


OrderID
1003
1005
1006
1004
1007
1002
1009
1008
1010
1001

Using Distinct Values

- **Q17:** What are the unique combinations of order number and order quantity included in the OrderLine table?

```
SELECT DISTINCT OrderID, OrderedQuantity
FROM OrderLine_T;
```

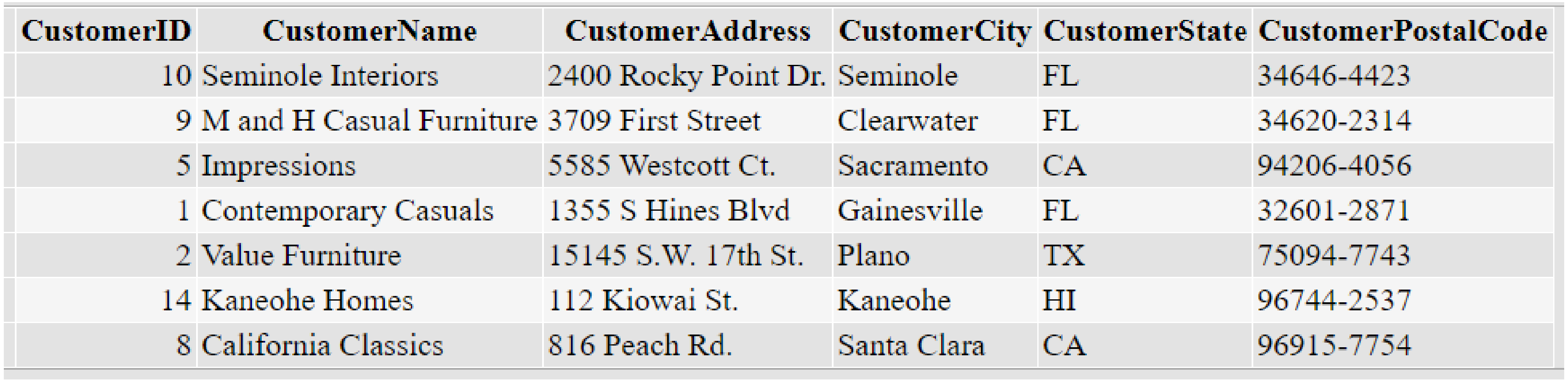


OrderID	OrderedQuantity
1003	3
1005	4
1006	1
1004	2
1007	3
1002	5
1009	3
1008	3
1010	10
1001	1
1006	2
1007	2
1009	2
1001	2

Using IN and NOT IN with Lists

- **Q18:** List all customers who live in warmer states (i.e., FL, TX, CA, HI).

```
SELECT *
FROM Customer_T
WHERE CustomerState IN ('FL', 'TX', 'CA', 'HI');
```

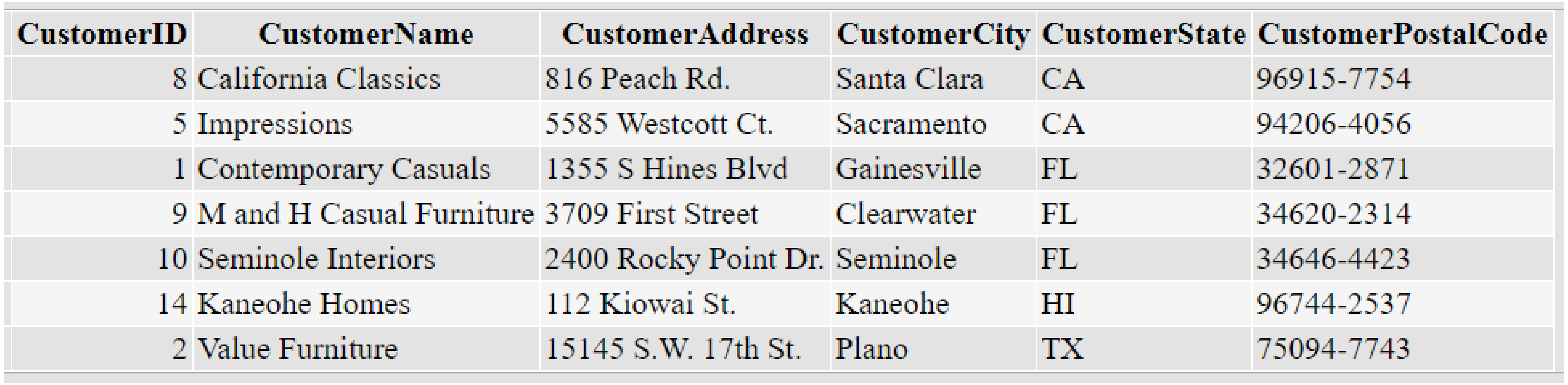


CustomerID	CustomerName	CustomerAddress	CustomerCity	CustomerState	CustomerPostalCode
10	Seminole Interiors	2400 Rocky Point Dr.	Seminole	FL	34646-4423
9	M and H Casual Furniture	3709 First Street	Clearwater	FL	34620-2314
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
14	Kaneohe Homes	112 Kiowai St.	Kaneohe	HI	96744-2537
8	California Classics	816 Peach Rd.	Santa Clara	CA	96915-7754

Sorting Results Using ORDER BY

- **Q19:** List all customers who live in warmer states (i.e., FL, TX, CA, HI). List customers alphabetically by state and alphabetically by customer within each state.

```
SELECT *
FROM Customer_T
WHERE CustomerState IN ('FL', 'TX', 'CA', 'HI')
ORDER BY CustomerState, CustomerName;
```



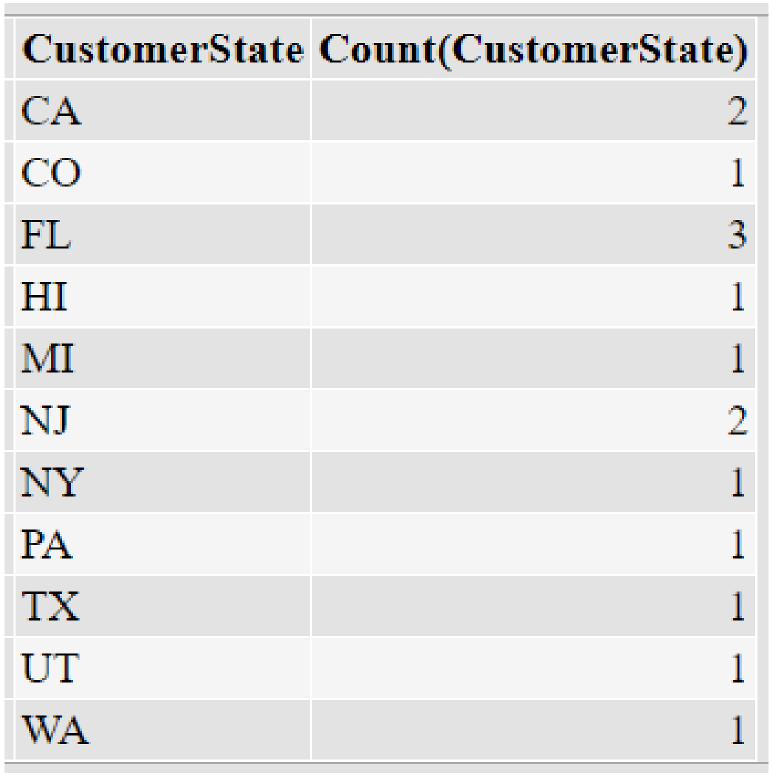
CustomerID	CustomerName	CustomerAddress	CustomerCity	CustomerState	CustomerPostalCode
8	California Classics	816 Peach Rd.	Santa Clara	CA	96915-7754
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
9	M and H Casual Furniture	3709 First Street	Clearwater	FL	34620-2314
10	Seminole Interiors	2400 Rocky Point Dr.	Seminole	FL	34646-4423
14	Kaneohe Homes	112 Kiowai St.	Kaneohe	HI	96744-2537
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743

Categorizing Results Using GROUP BY

CLEMSON College of BUSINESS

- **Q20:** Count the number of customers with addresses in each state to which we ship.

```
SELECT CustomerState, COUNT(CustomerState)
FROM Customer_T
GROUP BY CustomerState
ORDER BY CustomerState;
```

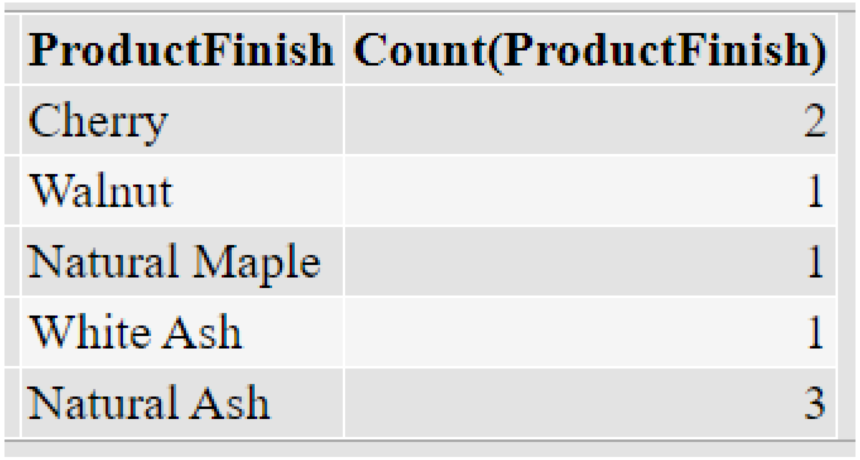


CustomerState	Count(CustomerState)
CA	2
CO	1
FL	3
HI	1
MI	1
NJ	2
NY	1
PA	1
TX	1
UT	1
WA	1

Categorizing Results Using GROUP BY

- **Q21:** How many products are there of each finish?

```
SELECT ProductFinish, COUNT(ProductFinish)
FROM PRODUCT_T
GROUP BY ProductFinish;
```



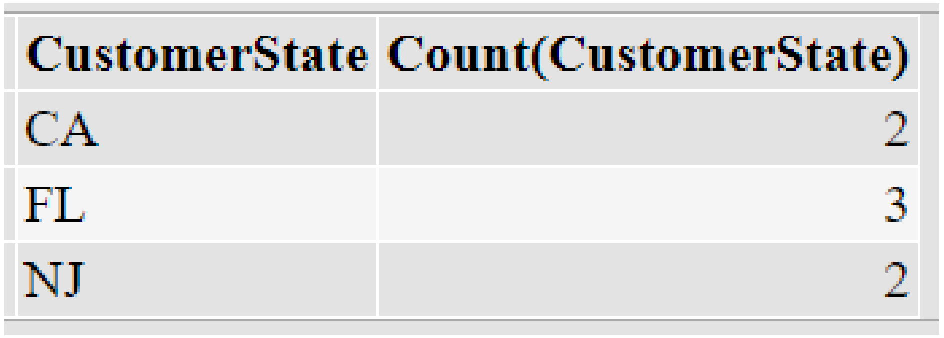
ProductFinish	Count(ProductFinish)
Cherry	2
Walnut	1
Natural Maple	1
White Ash	1
Natural Ash	3

Qualifying Results Using HAVING

- The HAVING clause acts like a WHERE clause, but it identifies groups, rather than rows, that meet a criterion.

- **Q22:** Find only states with more than one customer.

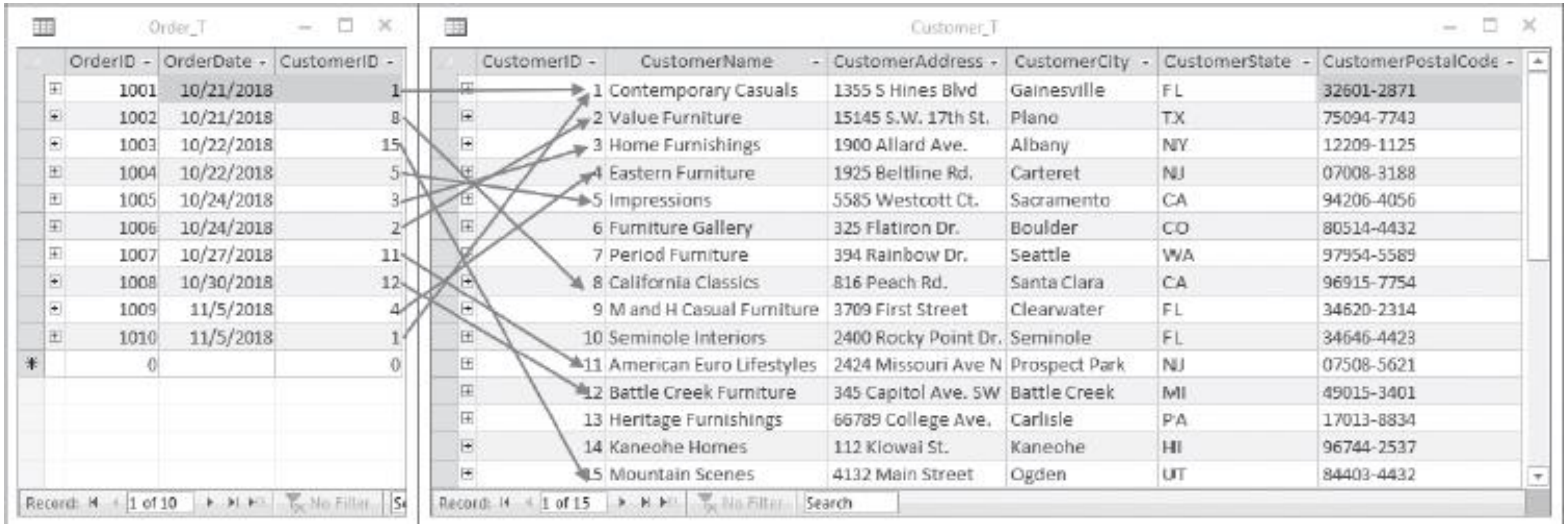
```
SELECT CustomerState, COUNT(CustomerState)
FROM Customer_T
GROUP BY CustomerState
HAVING COUNT(CustomerState) > 1;
```



CustomerState	Count(CustomerState)
CA	2
FL	3
NJ	2

PART II: Queries for Multiple Tables

- **Join:** A relational operation that causes two or more tables with a common domain to be combined into a single table or view



Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2018	1
1002	10/21/2018	8
1003	10/22/2018	15
1004	10/22/2018	5
1005	10/24/2018	3
1006	10/24/2018	2
1007	10/27/2018	11
1008	10/30/2018	12
1009	11/5/2018	4
1010	11/5/2018	1
0		0

Record: 1 of 10 No Filter Se

Customer_T

CustomerID	CustomerName	CustomerAddress	CustomerCity	CustomerState	CustomerPostalCode
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Period Furniture	394 Rainbow Dr.	Seattle	WA	97954-5589
8	California Classics	816 Peach Rd.	Santa Clara	CA	96915-7754
9	M and H Casual Furniture	3709 First Street	Clearwater	FL	34620-2314
10	Seminole Interiors	2400 Rocky Point Dr.	Seminole	FL	34646-4423
11	American Euro Lifestyles	2424 Missouri Ave N	Prospect Park	NJ	07508-5621
12	Battle Creek Furniture	345 Capitol Ave. SW	Battle Creek	MI	49015-3401
13	Heritage Furnishings	66789 College Ave.	Carlisle	PA	17013-8834
14	Kaneohe Homes	112 Kiowai St.	Kaneohe	HI	96744-2537
15	Mountain Scenes	4132 Main Street	Ogden	UT	84403-4432

Record: 1 of 15 No Filter Search

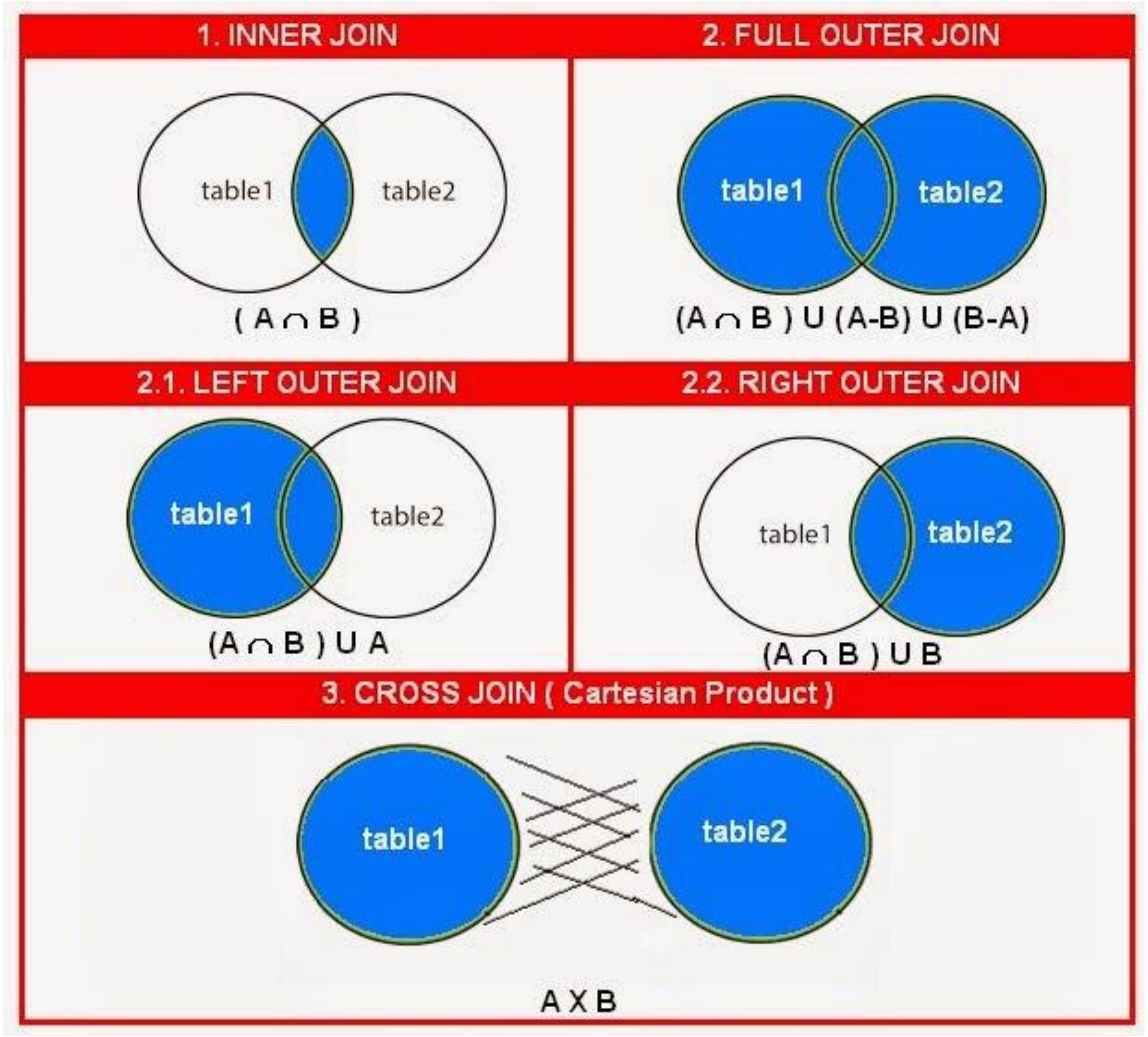
Types of Joins

Syntax: specify as **FROM** clause

table 1 explicit **JOIN** *table 2* **ON** joining condition

explicit: could be INNER, LEFT OUTER, RIGHT OUTER, FULL OUTER, or CROSS

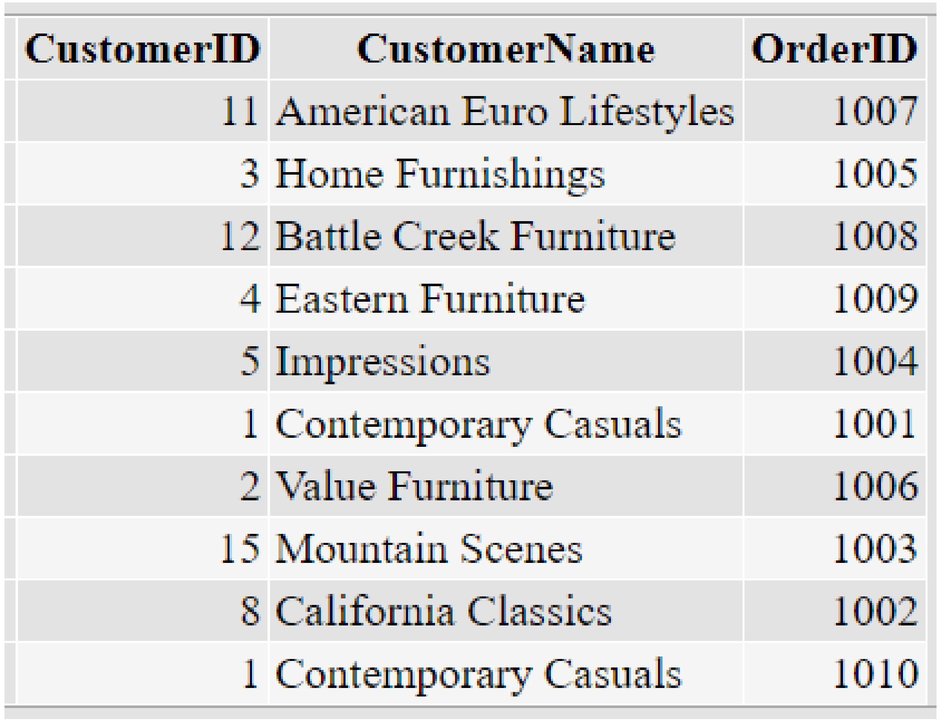
With cross join, no need to specify condition



Inner Join

- **Q23:** What are the customer IDs and names of all customers, along with the order IDs for all the orders they have placed?

```
SELECT Customer_T.CustomerID, CustomerName, OrderID
FROM Customer_T INNER JOIN Order_T on
Customer_T.CustomerID = Order_T.CustomerID;
```

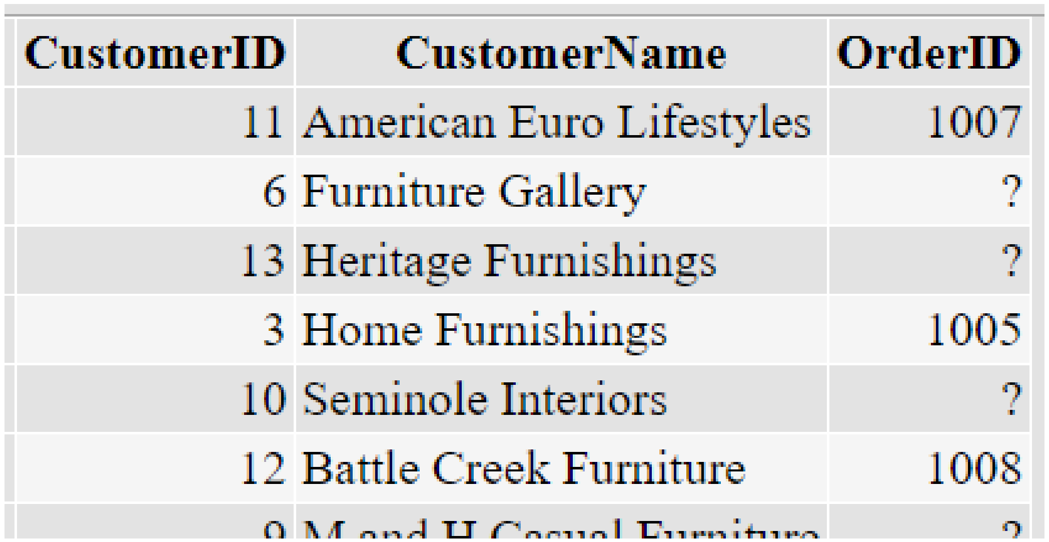


CustomerID	CustomerName	OrderID
11	American Euro Lifestyles	1007
3	Home Furnishings	1005
12	Battle Creek Furniture	1008
4	Eastern Furniture	1009
5	Impressions	1004
1	Contemporary Casuals	1001
2	Value Furniture	1006
15	Mountain Scenes	1003
8	California Classics	1002
1	Contemporary Casuals	1010

Left Outer Join

- **Q24:** List customer name, identification number, and order number for all customers listed in the customer table. Include the customer identification number and name even if there is no order available for that customer.

```
SELECT Customer_T.CustomerID, CustomerName, OrderID
FROM Customer_T LEFT OUTER JOIN Order_T on
Customer_T.CustomerID = Order_T.CustomerID;
```

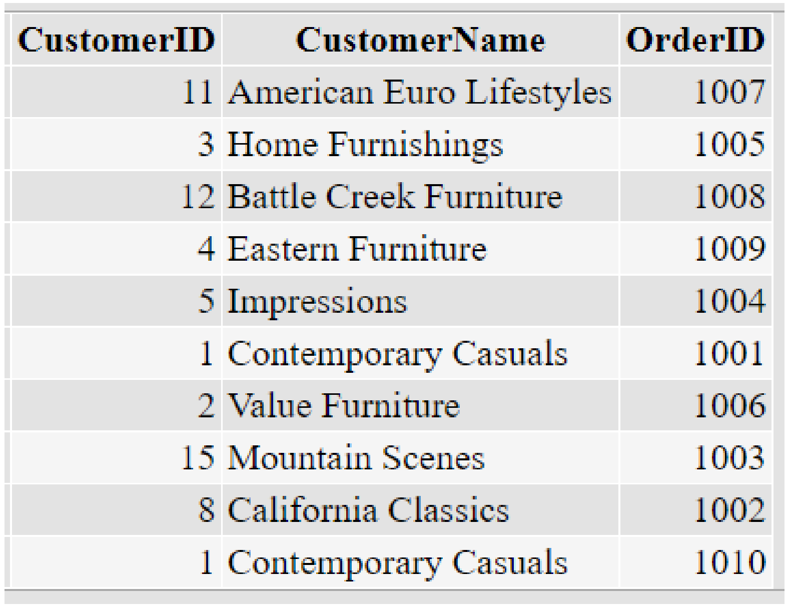


CustomerID	CustomerName	OrderID
11	American Euro Lifestyles	1007
6	Furniture Gallery	?
13	Heritage Furnishings	?
3	Home Furnishings	1005
10	Seminole Interiors	?
12	Battle Creek Furniture	1008
9	M and H Casual Furniture	?

Right Outer Join

- **Q25:** List customer name, identification number, and order number for all customers listed in the customer table. Include the order number, even if there is no customer name and identification number available.

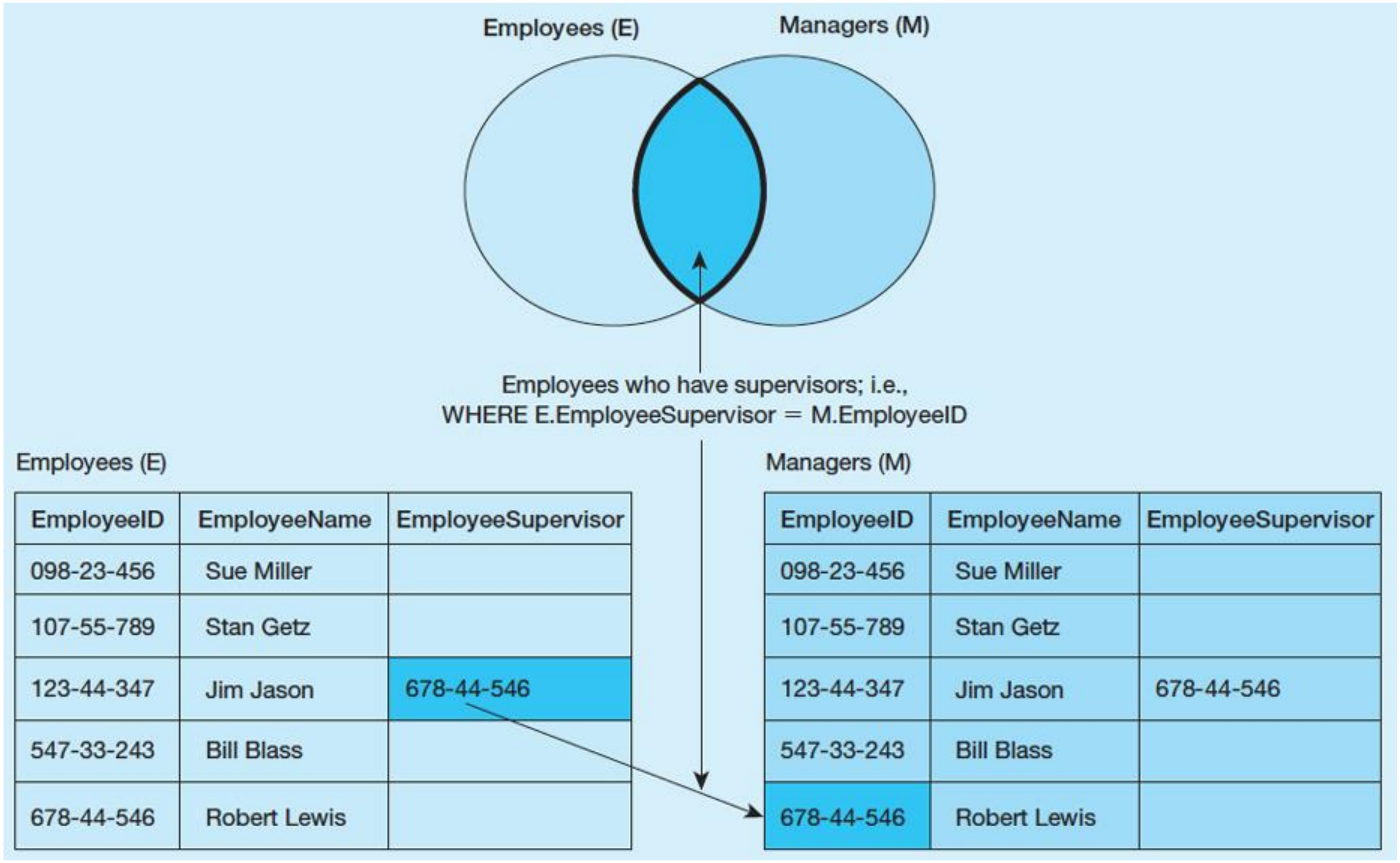
```
SELECT Customer_T.CustomerID, CustomerName, OrderID
FROM Customer_T RIGHT OUTER JOIN Order_T on
Customer_T.CustomerID = Order_T.CustomerID;
```



CustomerID	CustomerName	OrderID
11	American Euro Lifestyles	1007
3	Home Furnishings	1005
12	Battle Creek Furniture	1008
4	Eastern Furniture	1009
5	Impressions	1004
1	Contemporary Casuals	1001
2	Value Furniture	1006
15	Mountain Scenes	1003
8	California Classics	1002
1	Contemporary Casuals	1010

Self-Join

- A join requires matching rows in a table with other rows in that same table, e.g., unary relationship.



Employees (E)
Managers (M)
Employees who have supervisors; i.e.,
WHERE E.EmployeeSupervisor = M.EmployeeID

Employees (E)

EmployeeID	EmployeeName	EmployeeSupervisor
098-23-456	Sue Miller	
107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass	
678-44-546	Robert Lewis	

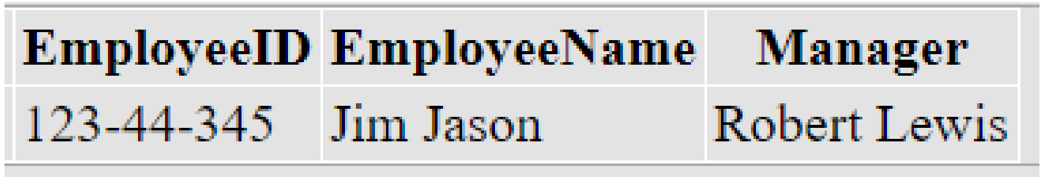
Managers (M)

EmployeeID	EmployeeName	EmployeeSupervisor
098-23-456	Sue Miller	
107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass	
678-44-546	Robert Lewis	

Self-Join

- **Q26:** What are the employee ID and name of each employee and the name of his or her supervisor?

```
SELECT E.EmployeeID, E.EmployeeName, M.EmployeeName AS Manager
FROM Employee_T AS E, Employee_T AS M
WHERE E.EmployeeSupervisor = M.EmployeeID;
```



EmployeeID	EmployeeName	Manager
123-44-345	Jim Jason	Robert Lewis

Subqueries

- Placing an inner query (SELECT … FROM … WHERE) within a WHERE or HAVING clause of another query
- **Joining Technique:** when data from several relations are to be retrieved and displayed and the relationships are not necessarily nested;
- **Subquery Technique:** display data from only the tables mentioned in the outer query

Subqueries

- **Q27:** What are the name and address of the customer who placed order number 1008?

Joining Technique:

```
SELECT CustomerName, CustomerAddress, CustomerCity, CustomerState,
CustomerPostalCode
FROM Customer_T, Order_T
WHERE Customer_T.CustomerID = Order_T.CustomerID
    AND OrderID = 1008;
```

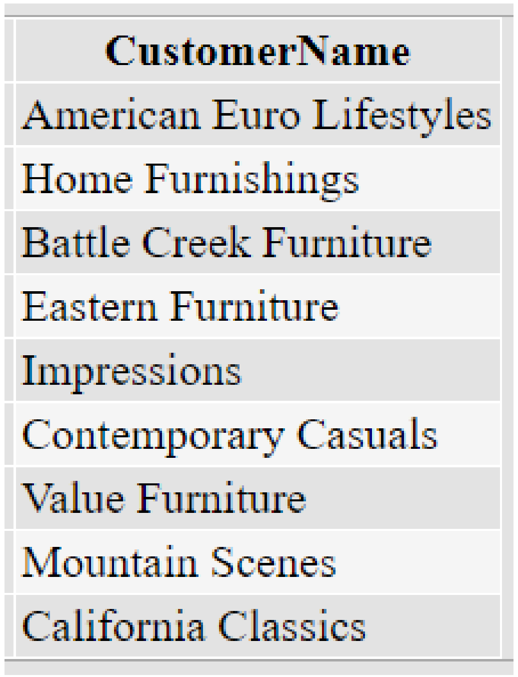
Subquery Technique:

```
SELECT CustomerName, CustomerAddress, CustomerCity, CustomerState,
CustomerPostalCode
FROM Customer_T
WHERE CustomerID =
   (SELECT Order_T.CustomerID
    FROM Order_T
    WHERE OrderID = 1008);
```

Subqueries

- **Q28:** What are the names of customers who have placed orders?

```
SELECT CustomerName
FROM Customer_T
WHERE CustomerID IN
   (SELECT DISTINCT CustomerID
    FROM Order_T);
```



CustomerName
American Euro Lifestyles
Home Furnishings
Battle Creek Furniture
Eastern Furniture
Impressions
Contemporary Casuals
Value Furniture
Mountain Scenes
California Classics

Subqueries

- The qualifiers NOT, ANY, and ALL may be used in front of IN or with logical operators such as =, >, and <.
- Also can use < ANY or >= ALL

- **Q29:** Which customers have not placed any orders for computer desk?

```
SELECT CustomerName
FROM Customer_T
WHERE CustomerID NOT IN
  (SELECT DISTINCT CustomerID
   FROM Order_T, OrderLine_T, Product_T
   WHERE Order_T.OrderID = OrderLine_T.OrderID
      AND OrderLine_T.ProductID = Product_T.ProductID
      AND ProductDescription = 'Computer Desk');
```

Subqueries

- Also can use < ANY or >= ALL

- **Q30:** List the details about the product with the highest standard price.

```
SELECT ProductDescription, ProductFinish, ProductStandardPrice
FROM Product_T
WHERE ProductStandardPrice >= ALL
   (SELECT ProductStandardPrice
    FROM Product_T);
```

EXISTS, NOT EXISTS

- EXISTS: True if the subquery returns an intermediate results table that is not empty; False if the subquery returns an empty table.
- NOT EXISTS: reverse of EXISTS

- **Q31:** What are the order IDs for all orders that have included furniture finished in natural ash?

```
SELECT DISTINCT OrderID
FROM OrderLine_T
WHERE EXISTS
   (SELECT *
    FROM Product_T
    WHERE ProductID = OrderLine_T.ProductID
       AND ProductFinish = 'Natural Ash');
```

Practicing SQL Queries

- In-class exercises
- Assignments
- DataCamp