Unit 8: Advanced SQL

Learning Objectives (1 of 2)

- In this chapter, you will learn:
  - How to use the advanced SQL JOIN operator syntax
  - About the different types of subqueries and correlated queries
  - How to use SQL functions to manipulate dates, strings, and other data
  - About the relational set operators UNION, UNION ALL, INTERSECT, and MINUS

Learning Objectives (2 of 2)

- In this chapter, you will learn:
  - How to create and use views and updatable views
  - How to create and use triggers and stored procedures
  - How to create embedded SQL

Case Study - Tables

- V
- VENDOR
- P
- PRODUCT
- CUSTOMER
- INVOICE
- LINE
- EMPLOYEE
- EMP

```
CREATE TABLE V (
V_CODE       INTEGER PRIMARY KEY,
V_NAME       VARCHAR(35) NOT NULL,
V_CONTACT    VARCHAR(15) NOT NULL,
V_AREACODE   CHAR(3) NOT NULL,
V_PHONE      CHAR(8) NOT NULL,
V_STATE      CHAR(2) NOT NULL,
V_ORDER      CHAR(1) NOT NULL);
```

```
CREATE TABLE P (
P_CODE       VARCHAR(10) PRIMARY KEY,
P_DESCRIPT   VARCHAR(35) NOT NULL,
P_INDATE     DATETIME NOT NULL,
P_QOH            INT NOT NULL,
P_MIN        INT NOT NULL,
P_PRICE      NUMERIC(8,2) NOT NULL,
P_DISCOUNT   NUMERIC(4,2) NOT NULL,
V_CODE       INT);
```

```
CREATE TABLE VENDOR (
V_CODE       INTEGER,
V_NAME       VARCHAR(35) NOT NULL,
V_CONTACT    VARCHAR(15) NOT NULL,
V_AREACODE   CHAR(3) NOT NULL,
V_PHONE      CHAR(8) NOT NULL,
V_STATE      CHAR(2) NOT NULL,
V_ORDER      CHAR(1) NOT NULL,
PRIMARY KEY (V_CODE));
```

```
CREATE TABLE PRODUCT (
P_CODE  VARCHAR(10) PRIMARY KEY,
P_DESCRIPT  VARCHAR(35) NOT NULL,
P_INDATE    DATETIME NOT NULL,
P_QOH           INTEGER NOT NULL,
P_MIN       INTEGER NOT NULL,
P_PRICE     NUMERIC(8,2) NOT NULL,
P_DISCOUNT  NUMERIC(4,2) NOT NULL,
V_CODE      INTEGER,
CONSTRAINT PRODUCT_V_CODE_FK FOREIGN KEY (V_CODE)
REFERENCES VENDOR (V_CODE));
```

•

```
CREATE TABLE CUSTOMER (
CUS_CODE     INTEGER PRIMARY KEY,
CUS_LNAME    VARCHAR(15) NOT NULL,
CUS_FNAME    VARCHAR(15) NOT NULL,
CUS_INITIAL CHAR(1),
CUS_AREACODE     CHAR(3) DEFAULT '615' NOT NULL
CHECK(CUS_AREACODE IN ('615','713','931')),
CUS_PHONE    CHAR(8) NOT NULL,
CUS_BALANCE NUMERIC(9,2) DEFAULT 0.00,
CONSTRAINT CUS_UI1 UNIQUE(CUS_LNAME,CUS_FNAME));
```

```
CREATE TABLE INVOICE (
INV_NUMBER       INTEGER PRIMARY KEY,
CUS_CODE     INTEGER NOT NULL,
INV_DATE     DATETIME NOT NULL,
CONSTRAINT INVOICE_CUS_CODE_FK FOREIGN KEY (CUS_CODE)
REFERENCES CUSTOMER(CUS_CODE));
```

```
CREATE TABLE LINE (
INV_NUMBER  INTEGER NOT NULL,
LINE_NUMBER NUMERIC(2,0) NOT NULL,
P_CODE      VARCHAR(10) NOT NULL,
LINE_UNITS  NUMERIC(9,2) DEFAULT 0.00 NOT NULL,
LINE_PRICE  NUMERIC(9,2) DEFAULT 0.00 NOT NULL,
PRIMARY KEY (INV_NUMBER,LINE_NUMBER),
FOREIGN KEY (INV_NUMBER) REFERENCES INVOICE
(INV_NUMBER) ON DELETE CASCADE,
FOREIGN KEY (P_CODE) REFERENCES PRODUCT(P_CODE),
CONSTRAINT LINE_UI1 UNIQUE(INV_NUMBER, P_CODE));
```

```
CREATE TABLE EMPLOYEE (
EMP_NUM      INTEGER PRIMARY KEY,
EMP_TITLE    CHAR(10),
EMP_LNAME    VARCHAR(15) NOT NULL,
EMP_FNAME    VARCHAR(15) NOT NULL,
EMP_INITIAL  CHAR(1),
EMP_DOB      DATETIME,
EMP_HIRE_DATE    DATETIME,
EMP_YEARS    INTEGER,
EMP_AREACODE     CHAR(3),
EMP_PHONE    CHAR(8));
```

```
CREATE TABLE EMP (
EMP_NUM      INTEGER PRIMARY KEY,
EMP_TITLE    CHAR(10),
EMP_LNAME    VARCHAR(15) NOT NULL,
EMP_FNAME    VARCHAR(15) NOT NULL,
EMP_INITIAL  CHAR(1),
EMP_DOB      DATETIME,
EMP_HIRE_DATE    DATETIME,
EMP_AREACODE     CHAR(3),
EMP_PHONE    CHAR(8),
EMP_MGR      INTEGER);
```

```
• INSERT INTO V VALUES(21225,'Bryson, Inc.'    ,'Smithson','615','223-3234','TN','Y');
• INSERT INTO V VALUES(21226,'SuperLoo, Inc.'  ,'Flushing','904','215-8995','FL','N');
• INSERT INTO V VALUES(21231,'D&E Supply'      ,'Singh'   ,'615','228-3245','TN','Y');
• INSERT INTO V VALUES(21344,'Gomez Bros.'     ,'Ortega'  ,'615','889-2546','KY','N');
• INSERT INTO V VALUES(22567,'Dome Supply'     ,'Smith'   ,'901','678-1419','GA','N');
• INSERT INTO V VALUES(23119,'Randsets Ltd.'   ,'Anderson','901','678-3998','GA','Y');
• INSERT INTO V VALUES(24004,'Brackman Bros.'  ,'Browning','615','228-1410','TN','N');
• INSERT INTO V VALUES(24288,'ORDVA, Inc.'     ,'Hakford' ,'615','898-1234','TN','Y');
• INSERT INTO V VALUES(25443,'B&K, Inc.'       ,'Smith'   ,'904','227-0093','FL','N');
• INSERT INTO V VALUES(25501,'Damal Supplies'  ,'Smythe'  ,'615','890-3529','TN','N');
• INSERT INTO V VALUES(25595,'Rubicon Systems' ,'Orton'   ,'904','456-0092','FL','Y');
```

```
INSERT INTO P VALUES('11QER/31','Power painter, 15 psi., 3-nozzle'     ,'2015-11-03',  8,  5,109.99,0.00,25595);
INSERT INTO P VALUES('13-Q2/P2','7.25-in. pwr. saw blade'             ,'2015-12-13', 32, 15, 14.99,0.05,21344);
INSERT INTO P VALUES('14-Q1/L3','9.00-in. pwr. saw blade'             ,'2015-11-13', 18, 12, 17.49,0.00,21344);
INSERT INTO P VALUES('1546-QQ2','Hrd. cloth, 1/4-in., 2x50'           ,'2016-01-15', 15,  8, 39.95,0.00,23119);
INSERT INTO P VALUES('1558-QW1','Hrd. cloth, 1/2-in., 3x50'           ,'2016-01-15', 23,  5, 43.99,0.00,23119);
INSERT INTO P VALUES('2232/QTY','B&D jigsaw, 12-in. blade'            ,'2015-12-30',  8,  5,109.92,0.05,24288);
INSERT INTO P VALUES('2232/QWE','B&D jigsaw, 8-in. blade'             ,'2015-12-24',  6,  5, 99.87,0.05,24288);
INSERT INTO P VALUES('2238/QPD','B&D cordless drill, 1/2-in.'         ,'2016-01-20', 12,  5, 38.95,0.05,25595);
INSERT INTO P VALUES('23109-HB','Claw hammer'                         ,'2016-01-20', 23, 10,  9.95,0.10,21225);
INSERT INTO P VALUES('23114-AA','Sledge hammer, 12 lb.'               ,'2016-01-02',  8,  5, 14.40,0.05,NULL);
INSERT INTO P VALUES('54778-2T','Rat-tail file, 1/8-in. fine'         ,'2015-12-15', 43, 20,  4.99,0.00,21344);
INSERT INTO P VALUES('89-WRE-Q','Hicut chain saw, 16 in.'             ,'2016-02-07', 11,  5,256.99,0.05,24288);
INSERT INTO P VALUES('PVC23DRT','PVC pipe, 3.5-in., 8-ft'             ,'2016-02-20',188, 75,  5.87,0.00,NULL);
INSERT INTO P VALUES('SM-18277','1.25-in. metal screw, 25'            ,'2016-03-01',172, 75,  6.99,0.00,21225);
INSERT INTO P VALUES('SW-23116','2.5-in. wd. screw, 50'               ,'2016-02-24',237,100,  8.45,0.00,21231);
INSERT INTO P VALUES('WR3/TT3' ,'Steel matting, 4''x8''x1/6", .5" mesh','2016-01-17', 18,  5,119.95,0.10,25595);
```

```
INSERT INTO CUSTOMER VALUES(10010,'Ramas'   ,'Alfred','A' ,'615','844-2573',0);
INSERT INTO CUSTOMER VALUES(10011,'Dunne'   ,'Leona' ,'K' ,'713','894-1238',0);
INSERT INTO CUSTOMER VALUES(10012,'Smith'   ,'Kathy' ,'W' ,'615','894-2285',345.86);
INSERT INTO CUSTOMER VALUES(10013,'Olowski' ,'Paul'  ,'F' ,'615','894-2180',536.75);
INSERT INTO CUSTOMER VALUES(10014,'Orlando' ,'Myron' ,NULL,'615','222-1672',0);
INSERT INTO CUSTOMER VALUES(10015,'O''Brian','Amy'   ,'B' ,'713','442-3381',0);
INSERT INTO CUSTOMER VALUES(10016,'Brown'   ,'James' ,'G' ,'615','297-1228',221.19);
INSERT INTO CUSTOMER VALUES(10017,'Williams','George',NULL,'615','290-2556',768.93);
INSERT INTO CUSTOMER VALUES(10018,'Farriss' ,'Anne'  ,'G' ,'713','382-7185',216.55);
INSERT INTO CUSTOMER VALUES(10019,'Smith'   ,'Olette','K' ,'615','297-3809',0);
```

- `INSERT INTO INVOICE VALUES(1001,10014,'2016-01-16');`
- `INSERT INTO INVOICE VALUES(1002,10011,'2016-01-16');`
- `INSERT INTO INVOICE VALUES(1003,10012,'2016-01-16');`
- `INSERT INTO INVOICE VALUES(1004,10011,'2016-01-17');`
- `INSERT INTO INVOICE VALUES(1005,10018,'2016-01-17');`
- `INSERT INTO INVOICE VALUES(1006,10014,'2016-01-17');`
- `INSERT INTO INVOICE VALUES(1007,10015,'2016-01-17');`
- `INSERT INTO INVOICE VALUES(1008,10011,'2016-01-17');`

```
INSERT INTO LINE VALUES(1001,1,'13-Q2/P2',1,14.99);
INSERT INTO LINE VALUES(1001,2,'23109-HB',1,9.95);
INSERT INTO LINE VALUES(1002,1,'54778-2T',2,4.99);
INSERT INTO LINE VALUES(1003,1,'2238/QPD',1,38.95);
INSERT INTO LINE VALUES(1003,2,'1546-QQ2',1,39.95);
INSERT INTO LINE VALUES(1003,3,'13-Q2/P2',5,14.99);
INSERT INTO LINE VALUES(1004,1,'54778-2T',3,4.99);
INSERT INTO LINE VALUES(1004,2,'23109-HB',2,9.95);
INSERT INTO LINE VALUES(1005,1,'PVC23DRT',12,5.87);
INSERT INTO LINE VALUES(1006,1,'SM-18277',3,6.99);
INSERT INTO LINE VALUES(1006,2,'2232/QTY',1,109.92);
INSERT INTO LINE VALUES(1006,3,'23109-HB',1,9.95);
INSERT INTO LINE VALUES(1006,4,'89-WRE-Q',1,256.99);
INSERT INTO LINE VALUES(1007,1,'13-Q2/P2',2,14.99);
INSERT INTO LINE VALUES(1007,2,'54778-2T',1,4.99);
INSERT INTO LINE VALUES(1008,1,'PVC23DRT',5,5.87);
INSERT INTO LINE VALUES(1008,2,'WR3/TT3',3,119.95);
INSERT INTO LINE VALUES(1008,3,'23109-HB',1,9.95);
```

```
INSERT INTO EMP VALUES(100,'Mr.' ,'Kolmycz'   ,'George' ,'D' ,'1942-06-15','1985-03-15','615','324-5456',NULL);
INSERT INTO EMP VALUES(101,'Ms.' ,'Lewis'     ,'Rhonda' ,'G' ,'1965-03-19','1986-04-25','615','324-4472',100);
INSERT INTO EMP VALUES(102,'Mr.' ,'Vandam'    ,'Rhett'  ,NULL,'1958-11-14','1990-12-20','901','675-8993',100);
INSERT INTO EMP VALUES(103,'Ms.' ,'Jones'     ,'Anne'   ,'M' ,'1974-10-16','1994-08-28','615','898-3456',100);
INSERT INTO EMP VALUES(104,'Mr.' ,'Lange'     ,'John'   ,'P' ,'1971-11-08','1994-10-20','901','504-4430',105);
INSERT INTO EMP VALUES(105,'Mr.' ,'Williams'  ,'Robert' ,'D' ,'1975-03-14','1998-11-08','615','890-3220',NULL);
INSERT INTO EMP VALUES(106,'Mrs.','Smith'     ,'Jeanine','K' ,'1968-02-12','1989-01-05','615','324-7883',105);
INSERT INTO EMP VALUES(107,'Mr.' ,'Diante'    ,'Jorge'  ,'D' ,'1974-08-21','1994-07-02','615','890-4567',105);
INSERT INTO EMP VALUES(108,'Mr.' ,'Wiesenbach','Paul'   ,'R' ,'1966-02-14','1992-11-18','615','897-4358',NULL);
INSERT INTO EMP VALUES(109,'Mr.' ,'Smith'     ,'George' ,'K' ,'1961-06-18','1989-04-14','901','504-3339',108);
INSERT INTO EMP VALUES(110,'Mrs.','Genkazi'   ,'Leighla','W' ,'1970-05-19','1990-12-01','901','569-0093',108);
INSERT INTO EMP VALUES(111,'Mr.' ,'Washington','Rupert' ,'E' ,'1966-01-03','1993-06-21','615','890-4925',105);
INSERT INTO EMP VALUES(112,'Mr.' ,'Johnson'   ,'Edward' ,'E' ,'1961-05-14','1983-12-01','615','898-4387',100);
INSERT INTO EMP VALUES(113,'Ms.' ,'Smythe'    ,'Melanie','P' ,'1970-09-15','1999-05-11','615','324-9006',105);
INSERT INTO EMP VALUES(114,'Ms.' ,'Brandon'   ,'Marie'  ,'G' ,'1956-11-02','1979-11-15','901','882-0845',108);
INSERT INTO EMP VALUES(115,'Mrs.','Saranda'   ,'Hermine','R' ,'1972-07-25','1993-04-23','615','324-5505',105);
INSERT INTO EMP VALUES(116,'Mr.' ,'Smith'     ,'George' ,'A' ,'1965-11-08','1988-12-10','615','890-2984',108);
```

```
INSERT INTO EMPLOYEE VALUES(100,'Mr.' ,'Kolmycz'   ,'George' ,'D' ,'1942-06-15','1985-03-15',18,'615','324-5456');
INSERT INTO EMPLOYEE VALUES(101,'Ms.' ,'Lewis'     ,'Rhonda' ,'G' ,'1965-03-19','1986-04-25',16,'615','324-4472');
INSERT INTO EMPLOYEE VALUES(102,'Mr.' ,'Vandam'    ,'Rhett'  ,NULL,'1958-11-14','1990-12-20',12,'901','675-8993');
INSERT INTO EMPLOYEE VALUES(103,'Ms.' ,'Jones'     ,'Anne'   ,'M' ,'1974-10-16','1994-08-28', 8,'615','898-3456');
INSERT INTO EMPLOYEE VALUES(104,'Mr.' ,'Lange'     ,'John'   ,'P' ,'1971-11-08','1994-10-20', 8,'901','504-4430');
INSERT INTO EMPLOYEE VALUES(105,'Mr.' ,'Williams'  ,'Robert' ,'D' ,'1975-03-14','1998-11-08', 4,'615','890-3220');
INSERT INTO EMPLOYEE VALUES(106,'Mrs.','Smith'     ,'Jeanine','K' ,'1968-02-12','1989-01-05',14,'615','324-7883');
INSERT INTO EMPLOYEE VALUES(107,'Mr.' ,'Diante'    ,'Jorge'  ,'D' ,'1974-08-21','1994-07-02', 8,'615','890-4567');
INSERT INTO EMPLOYEE VALUES(108,'Mr.' ,'Wiesenbach','Paul'   ,'R' ,'1966-02-14','1992-11-18',10,'615','897-4358');
INSERT INTO EMPLOYEE VALUES(109,'Mr.' ,'Smith'     ,'George' ,'K' ,'1961-06-18','1989-04-14',13,'901','504-3339');
INSERT INTO EMPLOYEE VALUES(110,'Mrs.','Genkazi'   ,'Leighla','W' ,'1970-05-19','1990-12-01',12,'901','569-0093');
INSERT INTO EMPLOYEE VALUES(111,'Mr.' ,'Washington','Rupert' ,'E' ,'1966-01-03','1993-06-21', 9,'615','890-4925');
INSERT INTO EMPLOYEE VALUES(112,'Mr.' ,'Johnson'   ,'Edward' ,'E' ,'1961-05-14','1983-12-01',19,'615','898-4387');
INSERT INTO EMPLOYEE VALUES(113,'Ms.' ,'Smythe'    ,'Melanie','P' ,'1970-09-15','1999-05-11', 3,'615','324-9006');
INSERT INTO EMPLOYEE VALUES(114,'Ms.' ,'Brandon'   ,'Marie'  ,'G' ,'1956-11-02','1979-11-15',23,'901','882-0845');
INSERT INTO EMPLOYEE VALUES(115,'Mrs.','Saranda'   ,'Hermine','R' ,'1972-07-25','1993-04-23', 9,'615','324-5505');
INSERT INTO EMPLOYEE VALUES(116,'Mr.' ,'Smith'     ,'George' ,'A' ,'1965-11-08','1988-12-10',14,'615','890-2984');
```

SQL Join Operators

- Relational join operation merges rows from two tables and returns rows with one of the following:
  - Natural join - common values in common columns
  - Equality or inequality - meet a given join condition
  - Outer join – common values in common columns or no matching values
- **Inner join**: Rows that meet a given criterion are selected
  - Equality condition (natural join or equijoin) or inequality condition (theta join)
- **Outer join:** Returns matching rows and rows with unmatched attribute values for one or both joined tables

Table 8.1 – SQL Join Expression Styles (1 of 2)

JOIN CLASSIFICATION	JOIN TYPE	SQL SYNTAX EXAMPLE	DESCRIPTION
CROSS	CROSS JOIN	SELECT * FROM T1, T2	Returns the Cartesian product of T1 and T2 (old style)
		SELECT * FROM T1 CROSS JOIN T2	Returns the Cartesian product of T1 and T2
INNER	Old-style JOIN	SELECT * FROM T1, T2 WHERE T1.C1=T2.C1	Returns only the rows that meet the join condition in the WHERE clause (old style); only rows with matching values are selected
	NATURAL JOIN	SELECT * FROM T1 NATURAL JOIN T2	Returns only the rows with matching values in the matching columns; the matching columns must have the same names and similar data types

Table 8.1 – SQL Join Expression Styles (2 of 2)

	JOIN USING	SELECT * FROM T1 JOIN T2 USING (C1)	Returns only the rows with matching values in the columns indicated in the USING clause
	JOIN ON	SELECT * FROM T1 JOIN T2 ON T1.C1=T2.C1	Returns only the rows that meet the join Condition indicated in the ON clause
OUTER	LEFT JOIN	SELECT * FROM T1 LEFT OUTER JOIN T2 ON T1.C1=T2.C1	Returns rows with matching values and includes all rows from the left table (T1) with unmatched values
	RIGHT JOIN	SELECT * FROM T1 RIGHT OUTER JOIN T2 ON T1.C1=T2.C1	Returns rows with matching values and includes all rows from the right table (T2) with unmatched Values
	FULL JOIN	SELECT * FROM T1 FULL OUTER JOIN T2 ON T1.C1=T2.C1	Returns rows with matching values and includes all rows from both tables (T1 and T2) with unmatched Values

MySQL does not support the FULL OUTER JOIN. See Lab 5 for details.

Subqueries

- A subquery is a query (SELECT statement) inside another query.
- A subquery is normally expressed inside parentheses.
- The first query in the SQL statement is known as the outer query.
- The query inside the SQL statement is known as the inner query.
- The inner query is executed first.
- The output of an inner query is used as the input for the outer query.
- The entire SQL statement is sometimes referred to as a nested query.

Subqueries

- Subquery is a query inside another query
- Subquery can return:
  - One single value - One column and one row
  - A list of values - One column and multiple rows
  - A virtual table - Multicolumn, multirow set of values
  - No value - Output of the outer query might result in an error or a null empty set

TABLE 8.2

SELECT SUBQUERY EXAMPLES

SELECT SUBQUERY EXAMPLES	EXPLANATION
INSERT INTO PRODUCT SELECT * FROM P;	Inserts all rows from Table P into the PRODUCT table. Both tables must have the same attributes. The subquery returns all rows from Table P.
UPDATE PRODUCT SET P_PRICE = (SELECT AVG(P_PRICE) FROM PRODUCT) WHERE V_CODE IN (SELECT V_CODE FROM VENDOR WHERE V_AREACODE = '615')	Updates the product price to the average product price, but only for products provided by vendors who have an area code equal to 615. The first subquery returns the average price; the second subquery returns the list of vendors with an area code equal to 615.
DELETE FROM PRODUCT WHERE V_CODE IN (SELECT V_CODE FROM VENDOR WHERE V_AREACODE = '615')	Deletes the PRODUCT table rows provided by vendors with an area code equal to 615. The subquery returns the list of vendor codes with an area code equal to 615.

Subqueries

- /* VENDOR rows */
- INSERT INTO VENDOR SELECT * FROM V;

/* PRODUCT rows */
- INSERT INTO PRODUCT SELECT * FROM P;

```
mysql> SELECT * FROM VENDOR;
Empty set (0.00 sec)

mysql> SELECT * FROM V;
+--------+----------------+-----------+------------+----------+---------+---------+
| V_CODE | V_NAME         | V_CONTACT | V_AREACODE | V_PHONE  | V_STATE | V_ORDER |
+--------+----------------+-----------+------------+----------+---------+---------+
|  21225 | Bryson, Inc.   | Smithson  | 615        | 223-3234 | TN      | Y       |
|  21226 | SuperLoo, Inc. | Flushing  | 904        | 215-8995 | FL      | N       |
|  21231 | D&E Supply     | Singh     | 615        | 228-3245 | TN      | Y       |
|  21344 | Gomez Bros.    | Ortega    | 615        | 889-2546 | KY      | N       |
|  22567 | Dome Supply    | Smith     | 901        | 678-1419 | GA      | N       |
|  23119 | Randsets Ltd.  | Anderson  | 901        | 678-3998 | GA      | Y       |
|  24004 | Brackman Bros. | Browning  | 615        | 228-1410 | TN      | N       |
|  24288 | ORDVA, Inc.    | Hakford   | 615        | 898-1234 | TN      | Y       |
|  25443 | B&K, Inc.      | Smith     | 904        | 227-0093 | FL      | N       |
|  25501 | Damal Supplies | Smythe    | 615        | 890-3529 | TN      | N       |
|  25595 | Rubicon Systems | Orton    | 904        | 456-0092 | FL      | Y       |
+--------+----------------+-----------+------------+----------+---------+---------+
11 rows in set (0.00 sec)

mysql> INSERT INTO VENDOR SELECT * FROM V;
Query OK, 11 rows affected (0.00 sec)
Records: 11  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM VENDOR;
+--------+----------------+-----------+------------+----------+---------+---------+
| V_CODE | V_NAME         | V_CONTACT | V_AREACODE | V_PHONE  | V_STATE | V_ORDER |
+--------+----------------+-----------+------------+----------+---------+---------+
|  21225 | Bryson, Inc.   | Smithson  | 615        | 223-3234 | TN      | Y       |
|  21226 | SuperLoo, Inc. | Flushing  | 904        | 215-8995 | FL      | N       |
|  21231 | D&E Supply     | Singh     | 615        | 228-3245 | TN      | Y       |
|  21344 | Gomez Bros.    | Ortega    | 615        | 889-2546 | KY      | N       |
|  22567 | Dome Supply    | Smith     | 901        | 678-1419 | GA      | N       |
|  23119 | Randsets Ltd.  | Anderson  | 901        | 678-3998 | GA      | Y       |
|  24004 | Brackman Bros. | Browning  | 615        | 228-1410 | TN      | N       |
|  24288 | ORDVA, Inc.    | Hakford   | 615        | 898-1234 | TN      | Y       |
|  25443 | B&K, Inc.      | Smith     | 904        | 227-0093 | FL      | N       |
|  25501 | Damal Supplies | Smythe    | 615        | 890-3529 | TN      | N       |
|  25595 | Rubicon Systems | Orton    | 904        | 456-0092 | FL      | Y       |
+--------+----------------+-----------+------------+----------+---------+---------+
11 rows in set (0.00 sec)
```

Subqueries

- Suppose that you want to generate a list of vendors who do not provide products.

```
mysql> SELECT V_CODE, V_NAME FROM VENDOR
    -> WHERE V_CODE NOT IN (SELECT V_CODE FROM PRODUCT WHERE V_CODE IS NOT NULL);
+--------+----------------+
| V_CODE | V_NAME         |
+--------+----------------+
|  21226 | SuperLoo, Inc. |
|  22567 | Dome Supply    |
|  24004 | Brackman Bros. |
|  25443 | B&K, Inc.      |
|  25501 | Damal Supplies |
+--------+----------------+
5 rows in set (0.00 sec)

mysql> SELECT V_CODE, V_NAME FROM VENDOR
    -> WHERE V_CODE NOT IN (SELECT V_CODE FROM PRODUCT);
Empty set (0.00 sec)
```

Watch out!
Remember that having NULLs in your table can affect how your queries perform

WHERE Subqueries

- Uses inner SELECT subquery on the right side of a WHERE comparison expression
- Value generated by the subquery must be of a comparable data type
- If the query returns more than a single value, the DBMS will generate an error
- Can be used in combination with joins

WHERE Subqueries

- To generate a list of all products with a price greater than or equal to the average product price:

```
mysql> SELECT P_CODE, P_PRICE FROM PRODUCT
    -> WHERE P_PRICE >= (SELECT AVG(P_PRICE) FROM PRODUCT);
+----------+---------+
| P_CODE   | P_PRICE |
+----------+---------+
| 11QER/31 |  109.99 |
| 2232/QTY |  109.92 |
| 2232/QWE |   99.87 |
| 89-WRE-Q |  256.99 |
| WR3/TT3  |  119.95 |
+----------+---------+
5 rows in set (0.01 sec)
```

The nested query computes the average price, which is then used in the outer query

Subqueries in JOINS

- Subqueries can also be used in combination with joins.

```
mysql> SELECT DISTINCT CUS_CODE, CUS_LNAME, CUS_FNAME
    -> FROM CUSTOMER JOIN INVOICE USING (CUS_CODE)
    -> JOIN LINE USING (INV_NUMBER)
    -> JOIN PRODUCT USING (P_CODE)
    -> WHERE P_CODE = (SELECT P_CODE FROM PRODUCT WHERE
    -> P_DESCRIPT = 'Claw hammer');
+----------+-----------+-----------+
| CUS_CODE | CUS_LNAME | CUS_FNAME |
+----------+-----------+-----------+
|    10014 | Orlando   | Myron     |
|    10011 | Dunne     | Leona     |
+----------+-----------+-----------+
2 rows in set (0.00 sec)
```

IN and HAVING Subqueries

- IN subqueries
  - Used to compare a single attribute to a list of values
- HAVING subqueries
  - HAVING clause restricts the output of a GROUP BY query by applying conditional criteria to the grouped rows

IN and HAVING Subqueries

```
mysql> SELECT DISTINCT CUS_CODE, CUS_LNAME, CUS_FNAME
    -> FROM CUSTOMER JOIN INVOICE USING (CUS_CODE)
    -> JOIN LINE USING (INV_NUMBER)
    -> JOIN PRODUCT USING (P_CODE)
    -> WHERE P_CODE IN (SELECT P_CODE FROM PRODUCT
    -> WHERE P_DESCRIPT LIKE '%hammer%'
    -> OR P_DESCRIPT LIKE '%saw%');
+----------+-----------+-----------+
| CUS_CODE | CUS_LNAME | CUS_FNAME |
+----------+-----------+-----------+
|    10014 | Orlando   | Myron     |
|    10012 | Smith     | Kathy     |
|    10015 | O'Brian   | Amy       |
|    10011 | Dunne     | Leona     |
+----------+-----------+-----------+
4 rows in set (0.00 sec)
```

IN and HAVING Subqueries

- To list all products with a total quantity sold greater than the average quantity sold:

```
mysql> SELECT P_CODE, SUM(LINE_UNITS)
    -> FROM LINE
    -> GROUP BY P_CODE
    -> HAVING SUM(LINE_UNITS) > (SELECT AVG(LINE_UNITS) FROM LINE);
+----------+-----------------+
| P_CODE   | SUM(LINE_UNITS) |
+----------+-----------------+
| 13-Q2/P2 |            8.00 |
| 23109-HB |            5.00 |
| 54778-2T |            6.00 |
| PVC23DRT |           17.00 |
| SM-18277 |            3.00 |
| WR3/TT3  |            3.00 |
+----------+-----------------+
6 rows in set (0.00 sec)
```

Multirow Subquery Operators: ANY and ALL

- ALL operator
  - Allows comparison of a single value with a list of values returned by the first subquery
    - Uses a comparison operator other than equals
- ANY operator
  - Allows comparison of a single value to a list of values and selects only the rows for which the value is greater than or less than any value in the list

Multirow Subquery Operators: ANY and ALL

- Suppose you want to know which products cost more than all individual products provided by vendors from Florida:

```
mysql> SELECT P_CODE, P_QOH * P_PRICE
    -> FROM PRODUCT
    -> WHERE P_QOH * P_PRICE > ALL (SELECT P_QOH * P_PRICE
    -> FROM PRODUCT
    -> WHERE V_CODE IN (SELECT V_CODE
    -> FROM VENDOR
    -> WHERE V_STATE = 'FL'));
+----------+-----------------+
| P_CODE   | P_QOH * P_PRICE |
+----------+-----------------+
| 89-WRE-Q |         2826.89 |
+----------+-----------------+
1 row in set (0.00 sec)
```

SQL Functions (1 of 2)

- Functions always use a numerical, date, or string value
- Value may be part of a command or may be an attribute located in a table
- Function may appear anywhere in an SQL statement where a value or an attribute can be used

SQL Functions (2 of 2)

- Aggregate Functions
- Date and time functions
- Numeric functions
- String functions
- Conversion functions

Aggregate Functions

- Min()
- Max()
- Avg()
- Count()

- The AS command
  - used to rename a column or table with an alias.
  - only exists for the duration of the query.

COUNT()

```
mysql> SELECT COUNT(*) AS COUNT FROM CUSTOMER;
+-------+
| COUNT |
+-------+
|    10 |
+-------+
1 row in set (0.02 sec)
```

MAX()

```
mysql> SELECT MAX(P_PRICE)
    -> FROM PRODUCT;
+--------------+
| MAX(P_PRICE) |
+--------------+
|       256.99 |
+--------------+
1 row in set (0.01 sec)
```

AVG()

```
mysql> SELECT AVG(P_PRICE) AS AVERAGE
    -> FROM PRODUCT;
+-----------+
| AVERAGE   |
+-----------+
| 56.421250 |
+-----------+
1 row in set (0.00 sec)
```

GROUP BY

- The GROUP BY statement is often used with aggregate functions

(MAX, MIN, SUM, AVG) to group the result set by one or more columns

GROUP BY

```
mysql> SELECT CUS_AREACODE, COUNT(*) AS COUNT FROM CUSTOMER
    -> GROUP BY CUS_AREACODE;
+--------------+-------+
| CUS_AREACODE | COUNT |
+--------------+-------+
| 615          |     7 |
| 713          |     3 |
+--------------+-------+
2 rows in set (0.00 sec)
```

GROUP BY

```
mysql> SELECT V_CODE, MAX(P_PRICE)
    -> FROM PRODUCT
    -> GROUP BY V_CODE;
+--------+---------------+
| V_CODE | MAX(P_PRICE) |
+--------+---------------+
|   NULL |         14.40 |
|  21225 |          9.95 |
|  21231 |          8.45 |
|  21344 |         17.49 |
|  23119 |         43.99 |
|  24288 |        256.99 |
|  25595 |        119.95 |
+--------+---------------+
7 rows in set (0.00 sec)
```

Date and time functions

TABLE 8.5

SELECTED MYSQL DATE/TIME FUNCTIONS

FUNCTION	EXAMPLE(S)
Date_Format Returns a character string or a formatted string from a date value Syntax: DATE_FORMAT(date_value, fmt) fmt = format used; can be: %M: name of month %m: two-digit month number %b: abbreviated month name %d: number of day of month %W: weekday name %a: abbreviated weekday name %Y: four-digit year %y: two-digit year	Displays the product code and date the product was last received into stock for all products: SELECT P_CODE, DATE_FORMAT(P_INDATE, '%m/%d/%y') FROM PRODUCT; SELECT P_CODE, DATE_FORMAT(P_INDATE, '%M %d, %Y') FROM PRODUCT;
YEAR Returns a four-digit year Syntax: YEAR(date_value)	Lists all employees born in 1982: SELECT EMP_LNAME, EMP_FNAME, EMP_DOB, YEAR(EMP_DOB) AS YEAR FROM EMPLOYEE WHERE YEAR(EMP_DOB) = 1982;
MONTH Returns a two-digit month code Syntax: MONTH(date_value)	Lists all employees born in November: SELECT EMP_LNAME, EMP_FNAME, EMP_DOB, MONTH(EMP_DOB) AS MONTH FROM EMPLOYEE WHERE MONTH(EMP_DOB) = 11;
DAY Returns the number of the day Syntax: DAY(date_value)	Lists all employees born on the 14th day of the month: SELECT EMP_LNAME, EMP_FNAME, EMP_DOB, DAY(EMP_DOB) AS DAY FROM EMPLOYEE WHERE DAY(EMP_DOB) = 14;

Date and time functions

ADDDATE Adds a number of days to a date Syntax: ADDDATE(date_value, n) n = number of days **DATE_ADD** Adds a number of days, months, or years to a date. This is similar to ADDDATE except it is more robust. It allows the user to specify the date unit to add. Syntax: DATE_ADD(date, INTERVAL n unit) n = number to add unit = date unit, can be: DAY: add n days WEEK: add n weeks MONTH: add n months YEAR: add n years	List all products with the date they will have been on the shelf for 30 days. SELECT P_CODE, P_INDATE, ADDDATE(P_INDATE, 30) FROM PRODUCT ORDER BY ADDDATE(P_INDATE, 30); Lists all products with their expiration date (two years from the purchase date): SELECT P_CODE, P_INDATE, DATE_ADD(P_INDATE, INTERVAL 2 YEAR) FROM PRODUCT ORDER BY DATE_ADD(P_INDATE, INTERVAL 2 YEAR);
LAST_DAY Returns the date of the last day of the month given in a date Syntax: LAST_DAY(date_value)	Lists all employees who were hired within the last seven days of a month: SELECT EMP_LNAME, EMP_FNAME, EMP_HIRE_DATE FROM EMPLOYEE WHERE EMP_HIRE_DATE >= DATE_ADD(LAST_DAY (EMP_HIRE_DATE), INTERVAL -7 DAY);

Date and time functions

```
mysql> SELECT P_CODE, DATE_FORMAT(P_INDATE, '%m/%d/
    -> FROM PRODUCT;
+----------+-------------------------------------+
| P_CODE   | DATE_FORMAT(P_INDATE, '%m/%d/%y') |
+----------+-------------------------------------+
| 11QER/31 | 11/03/15                            |
| 13-Q2/P2 | 12/13/15                            |
| 14-Q1/L3 | 11/13/15                            |
| 1546-QQ2 | 01/15/16                            |
| 1558-QW1 | 01/15/16                            |
| 2232/QTY | 12/30/15                            |
| 2232/QWE | 12/24/15                            |
| 2238/QPD | 01/20/16                            |
| 23109-HB | 01/20/16                            |
| 23114-AA | 01/02/16                            |
| 54778-2T | 12/15/15                            |
| 89-WRE-Q | 02/07/16                            |
| PVC23DRT | 02/20/16                            |
| SM-18277 | 03/01/16                            |
| SW-23116 | 02/24/16                            |
| WR3/TT3  | 01/17/16                            |
+----------+-------------------------------------+
16 rows in set (0.01 sec)
```

```
mysql> SELECT P_CODE, DATE_FORMAT(P_INDATE, '%M %d, %Y')
    -> FROM PRODUCT;
+----------+-----------------------------------+
| P_CODE   | DATE_FORMAT(P_INDATE, '%M %d, %Y') |
+----------+-----------------------------------+
| 11QER/31 | November 03, 2015                 |
| 13-Q2/P2 | December 13, 2015                 |
| 14-Q1/L3 | November 13, 2015                 |
| 1546-QQ2 | January 15, 2016                  |
| 1558-QW1 | January 15, 2016                  |
| 2232/QTY | December 30, 2015                 |
| 2232/QWE | December 24, 2015                 |
| 2238/QPD | January 20, 2016                  |
| 23109-HB | January 20, 2016                  |
| 23114-AA | January 02, 2016                  |
| 54778-2T | December 15, 2015                 |
| 89-WRE-Q | February 07, 2016                 |
| PVC23DRT | February 20, 2016                 |
| SM-18277 | March 01, 2016                    |
| SW-23116 | February 24, 2016                 |
| WR3/TT3  | January 17, 2016                  |
+----------+-----------------------------------+
16 rows in set (0.00 sec)
```

Numeric Functions

TABLE 8.6

SELECTED NUMERIC FUNCTIONS

FUNCTION	EXAMPLE(S)
ABS Returns the absolute value of a number Syntax: ABS(numeric_value)	In Oracle, use the following: SELECT 1.95, –1.93, ABS(1.95), ABS(–1.93) FROM DUAL; In MS Access, MySQL, and MS SQL Server, use the following: SELECT 1.95, –1.93, ABS(1.95), ABS(–1.93);
ROUND Rounds a value to a specified precision (number of digits) Syntax: ROUND(numeric_value, p) p = precision	Lists the product prices rounded to one and zero decimal places: SELECT P_CODE, P_PRICE, ROUND(P_PRICE,1) AS PRICE1, ROUND(P_PRICE,0) AS PRICE0 FROM PRODUCT;
CEIL/CEILING/FLOOR Returns the smallest integer greater than or equal to a number or returns the largest integer equal to or less than a number, respectively Syntax: CEIL(numeric_value) Oracle or MySQL CEILING(numeric_value) MS SQL Server or MySQL FLOOR(numeric_value)	Lists the product price, the smallest integer greater than or equal to the product price, and the largest integer equal to or less than the product price. In Oracle or MySQL, use the following: SELECT P_PRICE, CEIL(P_PRICE), FLOOR(P_PRICE) FROM PRODUCT; In MS SQL Server or MySQL, use the following: SELECT P_PRICE, CEILING(P_PRICE), FLOOR(P_PRICE) FROM PRODUCT; MS Access does not support these functions. Note that MySQL supports both CEIL and CEILING.

Numeric functions

```
mysql> SELECT P_CODE, P_PRICE,
    -> ROUND(P_PRICE,1) AS PRICE1,
    -> ROUND(P_PRICE,0) AS PRICE0
    -> FROM PRODUCT;
+----------+---------+--------+--------+
| P_CODE   | P_PRICE | PRICE1 | PRICE0 |
+----------+---------+--------+--------+
| 11QER/31 |  109.99 |  110.0 |    110 |
| 13-Q2/P2 |   14.99 |   15.0 |     15 |
| 14-Q1/L3 |   17.49 |   17.5 |     17 |
| 1546-QQ2 |   39.95 |   40.0 |     40 |
| 1558-QW1 |   43.99 |   44.0 |     44 |
| 2232/QTY |  109.92 |  109.9 |    110 |
| 2232/QWE |   99.87 |   99.9 |    100 |
| 2238/QPD |   38.95 |   39.0 |     39 |
| 23109-HB |    9.95 |   10.0 |     10 |
| 23114-AA |   14.40 |   14.4 |     14 |
| 54778-2T |    4.99 |    5.0 |      5 |
| 89-WRE-Q |  256.99 |  257.0 |    257 |
| PVC23DRT |    5.87 |    5.9 |      6 |
| SM-18277 |    6.99 |    7.0 |      7 |
| SW-23116 |    8.45 |    8.5 |      8 |
| WR3/TT3  |  119.95 |  120.0 |    120 |
+----------+---------+--------+--------+
16 rows in set (0.00 sec)
```

String Functions

- CONCAT() MySQL
- Concatenates data from two different character columns and returns a single column.
- CONCAT(strg_value, strg_value)
- The CONCAT function can only accept two string values so nested CONCAT functions are required when more than two values are to be concatenated.

String Functions

- Lists all employee names (concatenated).

```
mysql> SELECT CONCAT(CONCAT(EMP_LNAME, ', '),
    -> EMP_FNAME) AS NAME
    -> FROM EMPLOYEE;
+--------------------+
| NAME               |
+--------------------+
| Kolmycz, George    |
| Lewis, Rhonda      |
| Vandam, Rhett      |
| Jones, Anne        |
| Lange, John        |
| Williams, Robert   |
| Smith, Jeanine     |
| Diante, Jorge      |
| Wiesenbach, Paul   |
| Smith, George      |
| Genkazi, Leighla   |
| Washington, Rupert |
| Johnson, Edward    |
| Smythe, Melanie    |
| Brandon, Marie     |
| Saranda, Hermine   |
| Smith, George      |
+--------------------+
17 rows in set (0.02 sec)
```

String Functions

- Lists all employee names in all capital letters (concatenated).

```
mysql> SELECT UPPER(CONCAT(CONCAT(EMP_LNAME, ', '),
    -> EMP_FNAME)) AS NAME
    -> FROM EMPLOYEE;
+--------------------+
| NAME               |
+--------------------+
| KOLMYCZ, GEORGE    |
| LEWIS, RHONDA      |
| VANDAM, RHETT      |
| JONES, ANNE        |
| LANGE, JOHN        |
| WILLIAMS, ROBERT   |
| SMITH, JEANINE     |
| DIANTE, JORGE      |
| WIESENBACH, PAUL   |
| SMITH, GEORGE      |
| GENKAZI, LEIGHLA   |
| WASHINGTON, RUPERT |
| JOHNSON, EDWARD    |
| SMYTHE, MELANIE    |
| BRANDON, MARIE     |
| SARANDA, HERMINE   |
| SMITH, GEORGE      |
+--------------------+
17 rows in set (0.00 sec)
```

String Functions

- What function returns a string in all lowercase letters
- Syntax:
- LOWER(strg_value)

String Functions

- SUBSTRING
- Returns a substring or part of a given string parameter
- Syntax:
- SUBSTR(strg_value, p, l) or
- SUBSTRING(strg_value,p,l)
- p = start position
- l = length of characters
- If the length of characters is omitted, the functions will return the remainder of the string value.

Lists the first three characters of all employee phone numbers.

```
mysql> SELECT EMP_PHONE, SUBSTRING(EMP_PHONE,1,3) AS PREFIX
    -> FROM EMPLOYEE;
+-----------+--------+
| EMP_PHONE | PREFIX |
+-----------+--------+
| 324-5456  | 324    |
| 324-4472  | 324    |
| 675-8993  | 675    |
| 898-3456  | 898    |
| 504-4430  | 504    |
| 890-3220  | 890    |
| 324-7883  | 324    |
| 890-4567  | 890    |
| 897-4358  | 897    |
| 504-3339  | 504    |
| 569-0093  | 569    |
| 890-4925  | 890    |
| 898-4387  | 898    |
| 324-9006  | 324    |
| 882-0845  | 882    |
| 324-5505  | 324    |
| 890-2984  | 890    |
+-----------+--------+
17 rows in set (0.00 sec)
```

String Functions

- LENGTH
- Returns the number of characters in a string value
- Syntax:
- LENGTH(strg_value)

- Lists all employee last names and the length of their names in descending order by last name length.

```
mysql> SELECT EMP_LNAME, LENGTH(EMP_LNAME) AS NAMESIZE
    -> FROM EMPLOYEE;
+-----------+----------+
| EMP_LNAME | NAMESIZE |
+-----------+----------+
| Kolmycz   |        7 |
| Lewis     |        5 |
| Vandam    |        6 |
| Jones     |        5 |
| Lange     |        5 |
| Williams  |        8 |
| Smith     |        5 |
| Diante    |        6 |
| Wiesenbach|       10 |
| Smith     |        5 |
| Genkazi   |        7 |
| Washington|       10 |
| Johnson   |        7 |
| Smythe    |        6 |
| Brandon   |        7 |
| Saranda   |        7 |
| Smith     |        5 |
+-----------+----------+
17 rows in set (0.01 sec)
```

Conversion functions

- Numeric or Date to Character:
  - CAST
  - CAST (value-to-convert AS char(length))
  - CONVERT(value-to-convert, decimal(l,d))

- Lists all product prices, product received date, and percent discount using formatted values.

```
mysql> SELECT P_CODE, CAST(P_PRICE AS CHAR(8)) AS PRICE,
    -> CAST(P_INDATE AS CHAR(20)) AS INDATE,
    -> CAST(P_DISCOUNT AS CHAR(4)) AS DISC
    -> FROM PRODUCT;
+----------+--------+---------------------+------+
| P_CODE   | PRICE  | INDATE              | DISC |
+----------+--------+---------------------+------+
| 11QER/31 | 109.99 | 2015-11-03 00:00:00 | 0.00 |
| 13-Q2/P2 | 14.99  | 2015-12-13 00:00:00 | 0.05 |
| 14-Q1/L3 | 17.49  | 2015-11-13 00:00:00 | 0.00 |
| 1546-QQ2 | 39.95  | 2016-01-15 00:00:00 | 0.00 |
| 1558-QW1 | 43.99  | 2016-01-15 00:00:00 | 0.00 |
| 2232/QTY | 109.92 | 2015-12-30 00:00:00 | 0.05 |
| 2232/QWE | 99.87  | 2015-12-24 00:00:00 | 0.05 |
| 2238/QPD | 38.95  | 2016-01-20 00:00:00 | 0.05 |
| 23109-HB | 9.95   | 2016-01-20 00:00:00 | 0.10 |
| 23114-AA | 14.40  | 2016-01-02 00:00:00 | 0.05 |
| 54778-2T | 4.99   | 2015-12-15 00:00:00 | 0.00 |
| 89-WRE-Q | 256.99 | 2016-02-07 00:00:00 | 0.05 |
| PVC23DRT | 5.87   | 2016-02-20 00:00:00 | 0.00 |
| SM-18277 | 6.99   | 2016-03-01 00:00:00 | 0.00 |
| SW-23116 | 8.45   | 2016-02-24 00:00:00 | 0.00 |
| WR3/TT3  | 119.95 | 2016-01-17 00:00:00 | 0.10 |
+----------+--------+---------------------+------+
16 rows in set (0.00 sec)
```

Relational Set Operators (1 of 3)

- SQL data manipulation commands are set-oriented
  - **Set-oriented**: Operate over entire sets of rows and columns at once
- UNION, INTERSECT, and Except (MINUS) work properly when relations are union-compatible
  - **Union-compatible**: Number of attributes are the same and their corresponding data types are alike
- UNION
  - Combines rows from two or more queries without including duplicate rows

Relational Set Operators (2 of 3)

- Syntax - query UNION query
- UNION ALL
  - Produces a relation that retains duplicate rows
  - Can be used to unite more than two queries
- INTERSECT
  - Combines rows from two queries, returning only the rows that appear in both sets
  - Syntax - query INTERSECT query

Relational Set Operators (3 of 3)

- EXCEPT (MINUS)
  - Combines rows from two queries and returns only the rows that appear in the first set
  - Syntax
    - query EXCEPT query
    - query MINUS query
- Syntax alternatives
  - IN and NOT IN subqueries can be used in place of INTERSECT

- To show the combined CUSTOMER and CUSTOMER_2 records without duplicates, the UNION query is written as follows:
- The UNION statement can be used to unite more than just two queries.

```
mysql> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE,
    -> CUS_PHONE
    -> FROM CUSTOMER
    -> UNION
    -> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE,
    -> CUS_PHONE
    -> FROM CUSTOMER_2;
+-----------+-----------+-------------+--------------+-----------+
| CUS_LNAME | CUS_FNAME | CUS_INITIAL | CUS_AREACODE | CUS_PHONE |
+-----------+-----------+-------------+--------------+-----------+
| Ramas     | Alfred    | A           | 615          | 844-2573  |
| Dunne     | Leona     | K           | 713          | 894-1238  |
| Smith     | Kathy     | W           | 615          | 894-2285  |
| Olowski   | Paul      | F           | 615          | 894-2180  |
| Orlando   | Myron     | NULL        | 615          | 222-1672  |
| O'Brian   | Amy       | B           | 713          | 442-3381  |
| Brown     | James     | G           | 615          | 297-1228  |
| Williams  | George    | NULL        | 615          | 290-2556  |
| Farriss   | Anne      | G           | 713          | 382-7185  |
| Smith     | Olette    | K           | 615          | 297-3809  |
| Terrell   | Justine   | H           | 615          | 322-9870  |
| Hernandez | Carlos    | J           | 723          | 123-7654  |
| McDowell  | George    | NULL        | 723          | 123-7768  |
| Tirpin    | Khaleed   | G           | 723          | 123-9876  |
| Lewis     | Marie     | J           | 734          | 332-1789  |
+-----------+-----------+-------------+--------------+-----------+
15 rows in set (0.02 sec)
```

FIGURE 8.16 UNION QUERY RESULTS

Database name: Ch08_SaleCo

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_BALANCE
10010	Ramas	Alfred	A	615	844-2573	0.00
10011	Dunne	Leona	K	713	894-1238	0.00
10012	Smith	Kathy	W	615	894-2285	345.86
10013	Olowski	Paul	F	615	894-2180	536.75
10014	Orlando	Myron		615	222-1672	0.00
10015	O'Brian	Amy	B	713	442-3381	0.00
10016	Brown	James	G	615	297-1228	221.19
10017	Williams	George		615	290-2556	768.93
10018	Farriss	Anne	G	713	382-7185	216.55
10019	Smith	Olette	K	615	297-3809	0.00

Table name: CUSTOMER_2

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
345	Terrell	Justine	H	615	322-9870
347	Olowski	Paul	F	615	894-2180
351	Hernandez	Carlos	J	723	123-7654
352	McDowell	George		723	123-7768
365	Tirpin	Khaleed	G	723	123-9876
368	Lewis	Marie	J	734	332-1789
369	Dunne	Leona	K	713	894-1238

Query name: qryUNION-of-CUSTOMER-and-CUSTOMER_2

CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
Brown	James	G	615	297-1228
Dunne	Leona	K	713	894-1238
Farriss	Anne	G	713	382-7185
Hernandez	Carlos	J	723	123-7654
Lewis	Marie	J	734	332-1789
McDowell	George		723	123-7768
O'Brian	Amy	B	713	442-3381
Olowski	Paul	F	615	894-2180
Orlando	Myron		615	222-1672
Ramas	Alfred	A	615	844-2573
Smith	Kathy	W	615	894-2285
Smith	Olette	K	615	297-3809
Terrell	Justine	H	615	322-9870
Tirpin	Khaleed	G	723	123-9876
Williams	George		615	290-2556

UNION ALL

```
mysql> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE,
    -> CUS_PHONE
    -> FROM CUSTOMER
    -> UNION ALL
    -> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE,
    -> CUS_PHONE
    -> FROM CUSTOMER_2;
+-----------+-----------+-------------+--------------+-----------+
| CUS_LNAME | CUS_FNAME | CUS_INITIAL | CUS_AREACODE | CUS_PHONE |
+-----------+-----------+-------------+--------------+-----------+
| Ramas     | Alfred    | A           | 615          | 844-2573  |
| Dunne     | Leona     | K           | 713          | 894-1238  |
| Smith     | Kathy     | W           | 615          | 894-2285  |
| Olowski   | Paul      | F           | 615          | 894-2180  |
| Orlando   | Myron     | NULL        | 615          | 222-1672  |
| O'Brian   | Amy       | B           | 713          | 442-3381  |
| Brown     | James     | G           | 615          | 297-1228  |
| Williams  | George    | NULL        | 615          | 290-2556  |
| Farriss   | Anne      | G           | 713          | 382-7185  |
| Smith     | Olette    | K           | 615          | 297-3809  |
| Terrell   | Justine   | H           | 615          | 322-9870  |
| Olowski   | Paul      | F           | 615          | 894-2180  |
| Hernandez | Carlos    | J           | 723          | 123-7654  |
| McDowell  | George    | NULL        | 723          | 123-7768  |
| Tirpin    | Khaleed   | G           | 723          | 123-9876  |
| Lewis     | Marie     | J           | 734          | 332-1789  |
| Dunne     | Leona     | K           | 713          | 894-1238  |
+-----------+-----------+-------------+--------------+-----------+
17 rows in set (0.00 sec)
```

INTERSECT

- To know which customer records are duplicated in the CUSTOMER and CUSTOMER_2 tables, the INTERSECT statement can be used to combine rows from two queries, returning only the rows that appear in both sets.

```
mysql> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE,
    -> CUS_PHONE
    -> FROM CUSTOMER
    -> INTERSECT
    -> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE,
    -> CUS_PHONE
    -> FROM CUSTOMER_2;
+-----------+-----------+-------------+--------------+-----------+
| CUS_LNAME | CUS_FNAME | CUS_INITIAL | CUS_AREACODE | CUS_PHONE |
+-----------+-----------+-------------+--------------+-----------+
| Dunne     | Leona     | K           | 713          | 894-1238  |
| Olowski   | Paul      | F           | 615          | 894-2180  |
+-----------+-----------+-------------+--------------+-----------+
2 rows in set (0.00 sec)
```

INTERSECT

- The following query returns the customer codes for all customers who are in area code 615 and who have made purchases. (If a customer has made a purchase, there must be an invoice record for that customer.)

```
mysql> SELECT CUS_CODE FROM CUSTOMER WHERE CUS_AREACODE = '615'
    -> INTERSECT
    -> SELECT DISTINCT CUS_CODE FROM INVOICE;
+----------+
| CUS_CODE |
+----------+
|    10012 |
|    10014 |
+----------+
2 rows in set (0.01 sec)
```

EXCEPT

- If the managers want to know which customers in the CUSTOMER_2 table are not found in the CUSTOMER table,

```
mysql> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE,
    -> CUS_PHONE
    -> FROM CUSTOMER_2
    -> EXCEPT
    -> SELECT CUS_LNAME, CUS_FNAME, CUS_INITIAL, CUS_AREACODE,
    -> CUS_PHONE
    -> FROM CUSTOMER;
+-----------+-----------+-------------+--------------+-----------+
| CUS_LNAME | CUS_FNAME | CUS_INITIAL | CUS_AREACODE | CUS_PHONE |
+-----------+-----------+-------------+--------------+-----------+
| Terrell   | Justine   | H           | 615          | 322-9870  |
| Hernandez | Carlos    | J           | 723          | 123-7654  |
| McDowell  | George    | NULL        | 723          | 123-7768  |
| Tirpin    | Khaleed   | G           | 723          | 123-9876  |
| Lewis     | Marie     | J           | 734          | 332-1789  |
+-----------+-----------+-------------+--------------+-----------+
5 rows in set (0.00 sec)
```

INTERSECT

- The following query returns the customer codes for all customers in area code 615 minus the ones who have made purchases, leaving the customers in area code 615 who have not made purchases.

```
mysql> SELECT CUS_CODE FROM CUSTOMER WHERE CUS_AREACODE = '615'
    -> EXCEPT
    -> SELECT DISTINCT CUS_CODE FROM INVOICE;
+----------+
| CUS_CODE |
+----------+
|    10010 |
|    10013 |
|    10016 |
|    10017 |
|    10019 |
+----------+
5 rows in set (0.00 sec)
```

Syntax Alternatives

- If your DBMS does not support the INTERSECT or EXCEPT (MINUS) statements, you can use IN and NOT IN subqueries to obtain similar results.

```
mysql> SELECT CUS_CODE FROM CUSTOMER
    -> WHERE CUS_AREACODE = '615' AND
    -> CUS_CODE IN (SELECT DISTINCT CUS_CODE FROM INVOICE);
+----------+
| CUS_CODE |
+----------+
|    10012 |
|    10014 |
+----------+
2 rows in set (0.01 sec)
```

Virtual Tables: Creating a View

- **View**: Virtual table based on a SELECT query
- **Base tables**: Tables on which the view is based
- **CREATE VIEW** statement: Data definition command that stores the subquery specification in the data dictionary
  - CREATE VIEW command
    - CREATE VIEW viewname AS SELECT query

Virtual Tables: Creating a View

- A relational view has several special characteristics:
- You can use the name of a view anywhere a table name is expected.
- Views are dynamically updated. That is, the view is re-created on demand each time it is invoked.
- Views provide a level of security in the database because they can restrict users to seeing only specified columns and rows in a table.

Virtual Tables: Creating a View

- Create a view to list all products with price greater than 50?

```
mysql> CREATE VIEW PRICEGT50 AS
    -> SELECT P_DESCRIPT, P_QOH, P_PRICE
    -> FROM PRODUCT
    -> WHERE P_PRICE > 50.00;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SELECT * FROM PRICEGT50;
+-------------------------------------+-------+---------+
| P_DESCRIPT                          | P_QOH | P_PRICE |
+-------------------------------------+-------+---------+
| Power painter, 15 psi., 3-nozzle    |     8 |  109.99 |
| B&D jigsaw, 12-in. blade            |     8 |  109.92 |
| B&D jigsaw, 8-in. blade             |     6 |   99.87 |
| Hicut chain saw, 16 in.             |    11 |  256.99 |
| Steel matting, 4'x8'x1/6", .5" mesh |    18 |  119.95 |
+-------------------------------------+-------+---------+
5 rows in set (0.01 sec)
```

Procedural SQL (1 of 2)

- Performs a conditional or looping operation by isolating critical code and making all application programs call the shared code
  - Yields better maintenance and logic control
- **Persistent stored module (PSM)**: Block of code containing:
  - Standard SQL statements
  - Procedural extensions that is stored and executed at the DBMS server

Procedural SQL (2 of 2)

- **Procedural Language SQL (PL/SQL)**
  - Use and storage of procedural code and SQL statements within the database
  - Merging of SQL and traditional programming constructs
- Procedural code is executed as a unit by DBMS when invoked by end user
- End users can use PL/SQL to create:
  - Anonymous PL/SQL blocks and triggers
  - Stored procedures and PL/SQL functions

Procedural SQL

- Do not confuse PL/SQL functions with SQL's built-in aggregate functions such as MIN and MAX.
- SQL built-in functions can be used only within SQL statements, while PL/SQL functions are mainly invoked within PL/SQL programs such as triggers and stored procedures.
- Functions can also be called within SQL statements, provided that they conform to very specific rules that are dependent on your DBMS environment.

Table 8.9 – PL/SQL Basic Data Types

DATA TYPE	DESCRIPTION
CHAR	Character values of a fixed length; for example: W_ZIP CHAR(5)
VARCHAR2	Variable-length character values; for example: W_FNAME VARCHAR2(15)
NUMBER	Numeric values; for example: W_PRICE NUMBER(6,2)
DATE	Date values; for example: W_EMP_DOB DATE
%TYPE	Inherits the data type from a variable that you declared previously or from an attribute of a database table; for example: W_PRICE PRODUCT.P_PRICE%TYPE Assigns W_PRICE the same data type as the P_PRICE column in the PRODUCT table

Procedural SQL

- PL/SQL blocks can contain only standard SQL data manipulation language (DML) commands such as SELECT, INSERT, UPDATE, and DELETE.
- The use of data definition language (DDL) commands is not directly supported in a PL/SQL block.
- MySQL does not support Anonymous PL/SQL blocks

Stored Procedures

- Named collection of procedural and SQL statements
- Advantages
  - Reduce network traffic and increase performance
  - Reduce code duplication by means of code isolation and code sharing

Stored Procedures - MySQL

- In MySQL, *procedure* is a stored program that can take in parameters. It does not return a value like function does.
- CREATE/ DROP a procedure
- Always select a database before creating procedures
- As ; acts as a delimiter(end of statement/query) in MySQL, you will need to first change the delimiter to special character
  - `DELIMITER //`
  - Create a procedure ending with //
  - Change the delimiter back to ; `DELIMITER ;`
  - Call the procedure

Stored Procedures - MySQL

- Syntax to create a procedure in MySQL

```
CREATE PROCEDURE <procedure_name> (parameterdata_type)
BEGIN
<declaration_section> #for declaring local variables
<executable_section> #procedure code
END;
```

Stored Procedures - MySQL

```
mysql> CREATE PROCEDURE all_vendors()
    -> BEGIN
    -> SELECT * FROM vendor;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> CALL all_vendors();
    -> //
+--------+----------------+----------+------------+----------+---------+---------+
| V_CODE | V_NAME         | V_CONTACT | V_AREACODE | V_PHONE  | V_STATE | V_ORDER |
+--------+----------------+----------+------------+----------+---------+---------+
|  21225 | Bryson, Inc.   | Smithson | 615        | 223-3234 | TN      | Y       |
|  21226 | SuperLoo, Inc. | Flushing | 904        | 215-8995 | FL      | N       |
|  21231 | D&E Supply     | Singh    | 615        | 228-3245 | TN      | Y       |
|  21344 | Gomez Bros.    | Ortega   | 615        | 889-2546 | KY      | N       |
|  22567 | Dome Supply    | Smith    | 901        | 678-1419 | GA      | N       |
|  23119 | Randsets Ltd.  | Anderson | 901        | 678-3998 | GA      | Y       |
|  24004 | Brackman Bros. | Browning | 615        | 228-1410 | TN      | N       |
|  24288 | ORDVA, Inc.    | Hakford  | 615        | 898-1234 | TN      | Y       |
|  25443 | B&K, Inc.      | Smith    | 904        | 227-0093 | FL      | N       |
|  25501 | Damal Supplies | Smythe   | 615        | 890-3529 | TN      | N       |
|  25595 | Rubicon Systems | Orton   | 904        | 456-0092 | FL      | Y       |
+--------+----------------+----------+------------+----------+---------+---------+
11 rows in set (0.01 sec)

Query OK, 0 rows affected (0.04 sec)
```

- The delimiter was not changed back to a semicolon, therefore the semicolon did not terminate the SQL prompt, but the ‘//’ did.

Stored Procedures - MySQL

```
mysql> DELIMITER ;
mysql> CALL all_vendors();
+--------+-----------------+-----------+------------+----------+---------+---------+
| V_CODE | V_NAME          | V_CONTACT | V_AREACODE | V_PHONE  | V_STATE | V_ORDER |
+--------+-----------------+-----------+------------+----------+---------+---------+
|  21225 | Bryson, Inc.    | Smithson  | 615        | 223-3234 | TN      | Y       |
|  21226 | SuperLoo, Inc.  | Flushing  | 904        | 215-8995 | FL      | N       |
|  21231 | D&E Supply      | Singh     | 615        | 228-3245 | TN      | Y       |
|  21344 | Gomez Bros.     | Ortega    | 615        | 889-2546 | KY      | N       |
|  22567 | Dome Supply     | Smith     | 901        | 678-1419 | GA      | N       |
|  23119 | Randsets Ltd.   | Anderson  | 901        | 678-3998 | GA      | Y       |
|  24004 | Brackman Bros.  | Browning  | 615        | 228-1410 | TN      | N       |
|  24288 | ORDVA, Inc.     | Hakford   | 615        | 898-1234 | TN      | Y       |
|  25443 | B&K, Inc.       | Smith     | 904        | 227-0093 | FL      | N       |
|  25501 | Damal Supplies  | Smythe    | 615        | 890-3529 | TN      | N       |
|  25595 | Rubicon Systems | Orton     | 904        | 456-0092 | FL      | Y       |
+--------+-----------------+-----------+------------+----------+---------+---------+
11 rows in set (0.00 sec)

Query OK, 0 rows affected (0.04 sec)
```

Triggers (1 of 2)

- Procedural SQL code automatically invoked by RDBMS when given data manipulation event occurs
- Parts of a trigger definition
  - Triggering timing - Indicates when trigger's PL/SQL code executes
  - Triggering event - Statement that causes the trigger to execute
  - Triggering level - **Statement-** and **row-level**
  - Triggering action - PL/SQL code enclosed between the BEGIN and END keywords

Triggers (2 of 2)

- DROP TRIGGER trigger_name command
  - Deletes a trigger without deleting the table
- Trigger action based on DML predicates
  - Actions depend on the type of DML statement that fires the trigger

MySQL - Trigger

- The MySQL trigger is a database object that is associated with a table.
- It will be activated when a defined action is executed for the table.
- The trigger can be executed when you run one of the following MySQL statements on the table INSERT, UPDATE and DELETE
- It can be invoked before or after the event

MySQL - Trigger

```
CREATE TRIGGER trigger_name trigger_time trigger_event ON table_name
        FOR EACH ROW BEGIN
                {tasks to be performed}
        END;
```

- trigger_name: eg: BEFORE_EMPLOYEE_UPDATE, AFTER_CUSTOMER_INSERT
- trigger_time: BEFORE or AFTER (to process action before/after the *change*)
- trigger_event: INSERT / UPDATE/ DELETE
  - only one event per trigger

MySQL - Trigger

```
mysql> DELIMITER //
mysql> CREATE TRIGGER TRG_CUST
    -> BEFORE INSERT ON TEST_CUSTOMER
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.CUS_ZIP = 333 THEN
    -> SET NEW.CUS_ZIP = 555;
    -> END IF;
    -> END;
    -> //
```

```
mysql> INSERT INTO TEST_CUSTOMER
    -> VALUES ("CUS110", "333");
    -> //
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM TEST_CUSTOMER;
    -> //
+----------+---------+
| CUS_CODE | CUS_ZIP |
+----------+---------+
| CUS110   | 555     |
+----------+---------+
1 row in set (0.00 sec)
```

PL/SQL Stored Functions

- **Stored function**: Named group of procedural and SQL statements that returns a value
  - As indicated by a RETURN statement in its program code
- Can be invoked only from within stored procedures or triggers
- It can take input parameters

Stored Functions

- CREATE /DROP a function
- Always select a database before creating functions
- As “;” acts as a delimiter(end of statement/query) in MySQL, you
- will need to first change the delimiter to special character eg:
- DELIMITER //
  - create a function ending with //
  - change the delimiter back to “;” eg., DELIMITER ;
  - call the function

Stored Functions

- Syntax to create a function in MySQL

```
CREATE FUNCTION <function_name> (parameter data_type)
RETURNS return_data_type
BEGIN
    <declaration_section> #for declaring local variables
    <executable_section> #function code
END
```

Stored Functions

- A function to calculate Body Mass Index(BMI)
- Input parameters: Takes in weight (in kgs) and height (in meters)
- Output parameters: returns BMI

```
CREATE FUNCTION BMI(weight DOUBLE(4,2), height DOUBLE(4,2))
RETURNS DOUBLE
READS SQL DATA
BEGIN
DECLARE bmi_value DOUBLE(4,2);
SET bmi_value=0;
SET bmi_value=weight/(height*height);
RETURN bmi_value;
END; //
```

```
mysql> DELIMITER //
mysql> CREATE FUNCTION BMI(weight DOUBLE(4,2), height DOUBLE(4,2))
    -> RETURNS DOUBLE
    -> READS SQL DATA
    -> BEGIN
    -> DECLARE bmi_value DOUBLE(4,2);
    -> SET bmi_value=0;
    -> SET bmi_value=weight/(height*height);
    -> RETURN bmi_value;
    -> END; //
Query OK, 0 rows affected, 3 warnings (0.02 sec)
```

```
mysql> SELECT BMI(50,1.45);
+--------------+
| BMI(50,1.45) |
+--------------+
|        23.78 |
+--------------+
1 row in set (0.00 sec)
```