

Introduction to SQL

Structured Query Language



Beginners to Advance

Genial Code

www.genial-code.com

Objectives

- Explore basic commands and functions of SQL
- How to use SQL for data administration (to create tables, indexes, and views)
- How to use SQL for data manipulation (to add, modify, delete, and retrieve data)
- How to use SQL to query a database to extract useful information

Introduction to SQL

- SQL functions fit into two broad categories:
 - Data definition language
 - SQL includes commands to:
 - Create database objects, such as tables, indexes, and views
 - Define access rights to those database objects
 - Data manipulation language
 - Includes commands to insert, update, delete, and retrieve data within database tables

Introduction to SQL (continued)

- SQL is relatively easy to learn
- Basic command set has vocabulary of less than 100 words
- Nonprocedural language
- American National Standards Institute (ANSI) prescribes a standard SQL
- Several SQL dialects exist

Introduction to SQL (continued)

**TABLE
7.1**

SQL Data Definition Commands

COMMAND OR OPTION	DESCRIPTION
CREATE SCHEMA AUTHORIZATION	Creates a database schema
CREATE TABLE	Creates a new table in the user's database schema
NOT NULL	Ensures that a column will not have null values
UNIQUE	Ensures that a column will not have duplicate values
PRIMARY KEY	Defines a primary key for a table
FOREIGN KEY	Defines a foreign key for a table
DEFAULT	Defines a default value for a column (when no value is given)
CHECK	Constraint used to validate data in an attribute
CREATE INDEX	Creates an index for a table
CREATE VIEW	Creates a dynamic subset of rows/columns from one or more tables
ALTER TABLE	Modifies a table's definition (adds, modifies, or deletes attributes or constraints)
CREATE TABLE AS	Creates a new table based on a query in the user's database schema
DROP TABLE	Permanently deletes a table (and thus its data)
DROP INDEX	Permanently deletes an index
DROP VIEW	Permanently deletes a view

Introduction to SQL (continued)

**TABLE
7.2**

SQL Data Manipulation Commands

COMMAND OR OPTION	DESCRIPTION
INSERT	Inserts row(s) into a table
SELECT	Selects attributes from rows in one or more tables or views
WHERE	Restricts the selection of rows based on a conditional expression
GROUP BY	Groups the selected rows based on one or more attributes
HAVING	Restricts the selection of grouped rows based on a condition
ORDER BY	Orders the selected rows based on one or more attributes
UPDATE	Modifies an attribute's values in one or more table's rows
DELETE	Deletes one or more rows from a table
COMMIT	Permanently saves data changes
ROLLBACK	Restores data to their original values

Introduction to SQL (continued)

**TABLE
7.2**

SQL Data Manipulation Commands (continued)

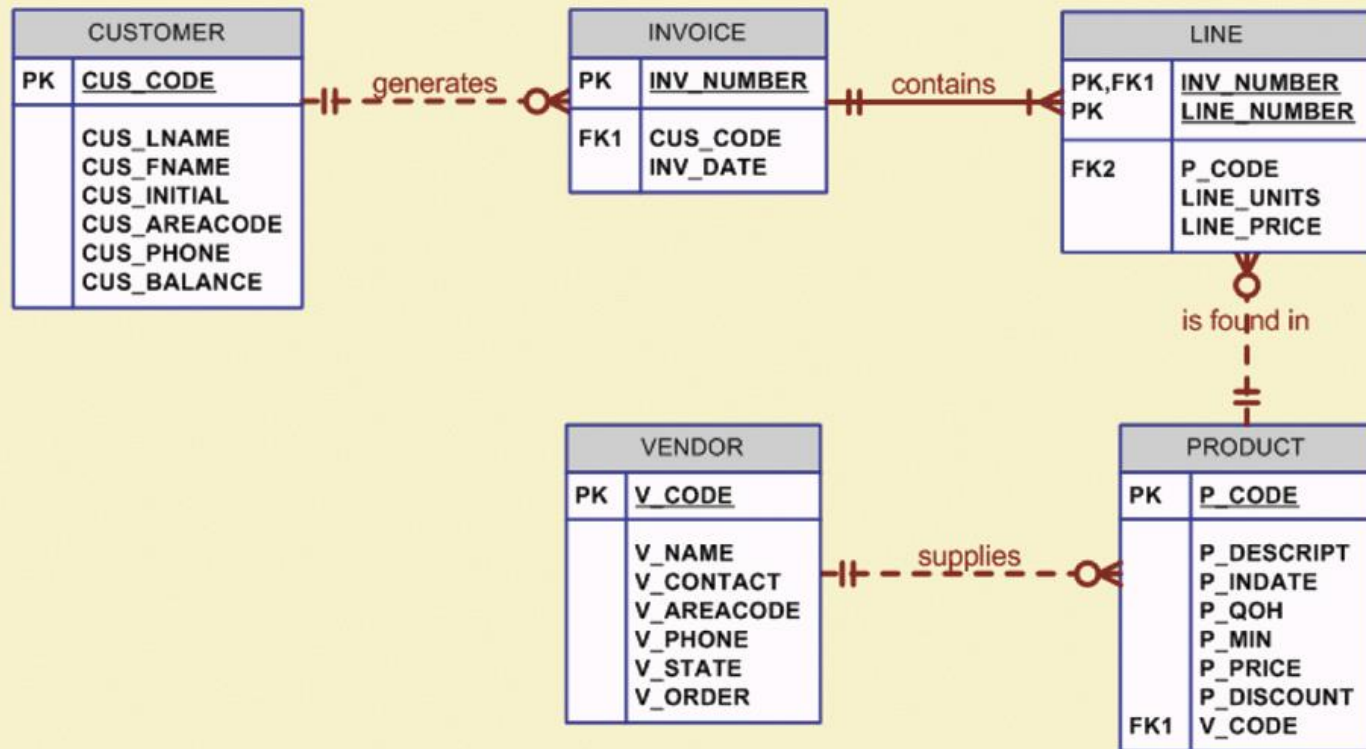
COMMAND OR OPTION	DESCRIPTION
COMPARISON OPERATORS	
=, <, >, <=, >=, <>	Used in conditional expressions
LOGICAL OPERATORS	
AND/OR/NOT	Used in conditional expressions
SPECIAL OPERATORS	Used in conditional expressions
BETWEEN	Checks whether an attribute value is within a range
IS NULL	Checks whether an attribute value is null
LIKE	Checks whether an attribute value matches a given string pattern
IN	Checks whether an attribute value matches any value within a value list
EXISTS	Checks whether a subquery returns any rows
DISTINCT	Limits values to unique values
AGGREGATE FUNCTIONS	Used with SELECT to return mathematical summaries on columns
COUNT	Returns the number of rows with non-null values for a given column
MIN	Returns the minimum attribute value found in a given column
MAX	Returns the maximum attribute value found in a given column
SUM	Returns the sum of all values for a given column
AVG	Returns the average of all values for a given column

Data Definition Commands

- Examine simple database model and database tables that will form basis for many SQL examples
- Understand data environment

The Database Model

FIGURE 7.1
The database model



Creating the Database

- Following two tasks must be completed:
 - Create database structure
 - Create tables that will hold end-user data
- First task:
 - RDBMS creates physical files that will hold database
 - Tends to differ substantially from one RDBMS to another

The Database Schema

- **Authentication**
 - Process through which DBMS verifies that only registered users are able to access database
 - Log on to RDBMS using user ID and password created by database administrator
- **Schema**
 - Group of database objects—such as tables and indexes—that are related to each other

Data Types

- Data type selection is usually dictated by nature of data and by intended use
- Pay close attention to expected use of attributes for sorting and data retrieval purposes

Data Types (continued)

TABLE
7.4

Some Common SQL Data Types

DATA TYPE	FORMAT	COMMENTS
Numeric	NUMBER(L,D)	The declaration NUMBER(7,2) indicates numbers that will be stored with two decimal places and may be up to six digits long, including the sign and the decimal place. Examples: 12.32, -134.99.
	INTEGER	May be abbreviated as INT. Integers are (whole) counting numbers, so they cannot be used if you want to store numbers that require decimal places.
	SMALLINT	Like INTEGER, but limited to integer values up to six digits. If your integer values are relatively small, use SMALLINT instead of INT.
	DECIMAL(L,D)	Like the NUMBER specification, but the storage length is a <i>minimum</i> specification. That is, greater lengths are acceptable, but smaller ones are not. DECIMAL(9,2), DECIMAL(9), and DECIMAL are all acceptable.
Character	CHAR(L)	Fixed-length character data for up to 255 characters. If you store strings that are not as long as the CHAR parameter value, the remaining spaces are left unused. Therefore, if you specify CHAR(25), strings such as "Smith" and "Katzenjammer" are each stored as 25 characters. However, a U.S. area code is always three digits long, so CHAR(3) would be appropriate if you wanted to store such codes.
	VARCHAR(L) or VARCHAR2(L)	Variable-length character data. The designation VARCHAR2(25) will let you store characters up to 25 characters long. However, VARCHAR will not leave unused spaces. Oracle users may use VARCHAR2 as well as VARCHAR.
Date	DATE	Stores dates in the Julian date format.

Creating Table Structures

- Use one line per column (attribute) definition
- Use spaces to line up attribute characteristics and constraints
- Table and attribute names are capitalized
- NOT NULL specification
- UNIQUE specification

Creating Table Structures (continued)

- Primary key attributes contain both a NOT NULL and a UNIQUE specification
- RDBMS will automatically enforce referential integrity for foreign keys
- Command sequence ends with semicolon

SQL Constraints

- **NOT NULL constraint**
 - Ensures that column does not accept nulls
- **UNIQUE constraint**
 - Ensures that all values in column are unique
- **DEFAULT constraint**
 - Assigns value to attribute when a new row is added to table
- **CHECK constraint**
 - Validates data when attribute value is entered

SQL Indexes

- When primary key is declared, DBMS automatically creates unique index
- Often need additional indexes
- Using CREATE INDEX command, SQL indexes can be created on basis of any selected attribute
- Composite index
 - Index based on two or more attributes
 - Often used to prevent data duplication

Data Manipulation Commands

- Adding table rows
- Saving table changes
- Listing table rows
- Updating table rows
- Restoring table contents
- Deleting table rows
- Inserting table rows with a select subquery

Adding Table Rows

- **INSERT**

- Used to enter data into table
- Syntax:
 - **INSERT INTO** *columnname*
VALUES (*value1, value2, ... , valuen*);

Adding Table Rows (continued)

- When entering values, notice that:
 - Row contents are entered between parentheses
 - Character and date values are entered between apostrophes
 - Numerical entries are not enclosed in apostrophes
 - Attribute entries are separated by commas
 - A value is required for each column
- Use NULL for unknown values

Saving Table Changes

- Changes made to table contents are not physically saved on disk until, one of the following occurs:
 - Database is closed
 - Program is closed
 - COMMIT command is used
- Syntax:
 - COMMIT [WORK];
- Will permanently save any changes made to any table in the database

Listing Table Rows

- **SELECT**

- Used to list contents of table
- Syntax:
 - `SELECT columnlist`
`FROM tablename;`
- *Columnlist* represents one or more attributes, separated by commas
- Asterisk can be used as wildcard character to list all attributes

Updating Table Rows

- **UPDATE**
 - Modify data in a table
 - Syntax:
 - `UPDATE tablename`
`SET columnname = expression [,`
`columnname = expression]`
`[WHERE conditionlist];`
 - If more than one attribute is to be updated in row, separate corrections with commas

Restoring Table Contents

- **ROLLBACK**
 - Used to restore database to its previous condition
 - Only applicable if COMMIT command has not been used to permanently store changes in database
- Syntax:
 - ROLLBACK;
- COMMIT and ROLLBACK only work with data manipulation commands that are used to add, modify, or delete table rows

Deleting Table Rows

- **DELETE**
 - Deletes a table row
 - Syntax:
 - `DELETE FROM tablename`
`[WHERE conditionlist];`
- **WHERE** condition is optional
- If **WHERE** condition is not specified, all rows from specified table will be deleted

Inserting Table Rows with a Select Subquery

■ INSERT

- Inserts multiple rows from another table (source)
- Uses SELECT subquery
 - Query that is embedded (or nested) inside another query
 - Executed first
- Syntax:
 - INSERT INTO *tablename* SELECT *columnlist* FROM *tablename*;

Selecting Rows with Conditional Restrictions

- Select partial table contents by placing restrictions on rows to be included in output
 - Add conditional restrictions to SELECT statement, using WHERE clause
- Syntax:
 - `SELECT columnlist`
`FROM tablelist`
`[WHERE conditionlist] ;`

Selecting Rows with Conditional Restrictions (continued)

FIGURE 7.5 The Microsoft Access QBE and its SQL

The screenshot shows the Microsoft Access QBE (Query By Examples) grid for a query named 'qryFig7-05 : Select Query'. The grid is set to 'Design View'. The 'Field' row contains 'P_DESCRPT', 'P_INDATE', 'P_PRICE', and 'V_CODE'. The 'Table' row contains 'PRODUCT', 'PRODUCT', 'PRODUCT', and 'PRODUCT'. The 'Criteria' row contains '21344' under the 'V_CODE' column. The 'Show' row has checkboxes for 'P_DESCRPT', 'P_INDATE', 'P_PRICE', and 'V_CODE', all of which are checked. The 'Sort' row is empty. The 'Criteria' row has a value of '21344' under the 'V_CODE' column.

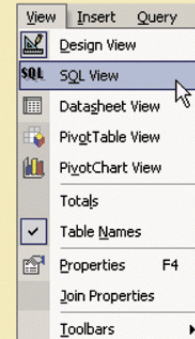
Field:	P_DESCRPT	P_INDATE	P_PRICE	V_CODE
Table:	PRODUCT	PRODUCT	PRODUCT	PRODUCT
Sort:				
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:				21344
or:				

Microsoft Access-generated SQL

The screenshot shows the Microsoft Access SQL View for the query 'qryFig7-05 : Select Query'. The SQL statement is: `SELECT PRODUCT.P_DESCRPT, PRODUCT.P_INDATE, PRODUCT.P_PRICE, PRODUCT.V_CODE FROM PRODUCT WHERE (((PRODUCT.V_CODE)=21344));`

User-entered SQL

The screenshot shows the Microsoft Access SQL View for the query 'qryFig7-05 : Select Query' with user-entered SQL. The SQL statement is: `SELECT P_DESCRPT, P_INDATE, P_PRICE, V_CODE FROM PRODUCT WHERE V_CODE=21344;`



Selecting Rows with Conditional Restrictions (continued)

**TABLE
7.6**

Comparison Operators

SYMBOL	MEANING
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<> or !=	Not equal to

Arithmetic Operators: The Rule of Precedence

- Perform operations within parentheses
- Perform power operations
- Perform multiplications and divisions
- Perform additions and subtractions

Arithmetic Operators: The Rule of Precedence (continued)

TABLE
7.7

The Arithmetic Operators

ARITHMETIC OPERATOR	DESCRIPTION
+	Add
-	Subtract
*	Multiply
/	Divide
^	Raise to the power of (Some applications use ** instead of ^.)

Special Operators

- **BETWEEN**
 - Used to check whether attribute value is within a range
- **IS NULL**
 - Used to check whether attribute value is null
- **LIKE**
 - Used to check whether attribute value matches given string pattern

Special Operators (continued)

- **IN**
 - Used to check whether attribute value matches any value within a value list
- **EXISTS**
 - Used to check if subquery returns any rows

Advanced Data Definition Commands

- All changes in table structure are made by using **ALTER** command
 - Followed by keyword that produces specific change
 - Following three options are available:
 - ADD
 - MODIFY
 - DROP

Changing a Column's Data Type

- ALTER can be used to change data type
- Some RDBMSs (such as Oracle) do not permit changes to data types unless column to be changed is empty

Changing a Column's Data Characteristics

- Use `ALTER` to change data characteristics
- If column to be changed already contains data, changes in column's characteristics are permitted if those changes do not alter the data type

Adding a Column

- Use **ALTER** to add column
 - Do not include the **NOT NULL** clause for new column

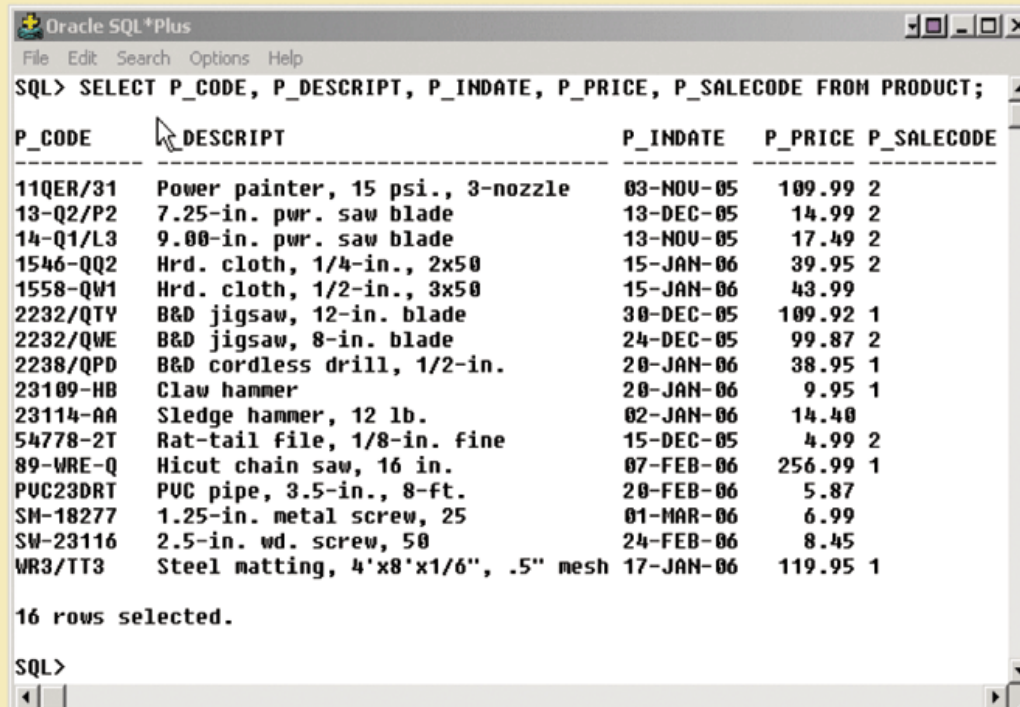
Dropping a Column

- Use **ALTER** to drop column
 - Some RDBMSs impose restrictions on the deletion of an attribute

Advanced Data Updates

FIGURE
7.15

The cumulative effect of the multiple updates in the PRODUCT table (Oracle)



The screenshot shows the Oracle SQL*Plus interface. The command prompt displays the SQL query: `SQL> SELECT P_CODE, P_DESCRIPT, P_INDATE, P_PRICE, P_SALECODE FROM PRODUCT;`. The results are displayed in a table with 5 columns: P_CODE, P_DESCRIPT, P_INDATE, P_PRICE, and P_SALECODE. There are 16 rows of data. Below the table, it says "16 rows selected." and the prompt "SQL>" is visible at the bottom.

P_CODE	P_DESCRIPT	P_INDATE	P_PRICE	P_SALECODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-NOV-05	109.99	2
13-Q2/P2	7.25-in. pwr. saw blade	13-DEC-05	14.99	2
14-Q1/L3	9.00-in. pwr. saw blade	13-NOV-05	17.49	2
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-JAN-06	39.95	2
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-JAN-06	43.99	
2232/QTV	B&D jigsaw, 12-in. blade	30-DEC-05	109.92	1
2232/QWE	B&D jigsaw, 8-in. blade	24-DEC-05	99.87	2
2238/QPD	B&D cordless drill, 1/2-in.	20-JAN-06	38.95	1
23109-HB	Claw hammer	20-JAN-06	9.95	1
23114-AA	Sledge hammer, 12 lb.	02-JAN-06	14.40	
54778-2T	Rat-tail file, 1/8-in. fine	15-DEC-05	4.99	2
89-WRE-Q	Hicut chain saw, 16 in.	07-FEB-06	256.99	1
PVC23DRT	PVC pipe, 3.5-in., 8-ft.	20-FEB-06	5.87	
SM-18277	1.25-in. metal screw, 25	01-MAR-06	6.99	
SW-23116	2.5-in. wd. screw, 50	24-FEB-06	8.45	
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-JAN-06	119.95	1

16 rows selected.

SQL>

Copying Parts of Tables

- SQL permits copying contents of selected table columns so that the data need not be reentered manually into newly created table(s)
- First create the PART table structure
- Next add rows to new PART table using PRODUCT table rows

Adding Primary and Foreign Key Designations

- When table is copied, integrity rules do not copy, so primary and foreign keys need to be manually defined on new table
- User ALTER TABLE command
 - Syntax:
 - `ALTER TABLE tablename ADD PRIMARY KEY(fieldname);`
 - For foreign key, use FOREIGN KEY in place of PRIMARY KEY

Deleting a Table from the Database

- **DROP**
 - Deletes table from database
 - Syntax:
 - `DROP TABLE tablename;`

Advanced Select Queries

- SQL provides useful functions that can:
 - Count
 - Find minimum and maximum values
 - Calculate averages
- SQL allows user to limit queries to only those entries having no duplicates or entries whose duplicates may be grouped

Aggregate Functions

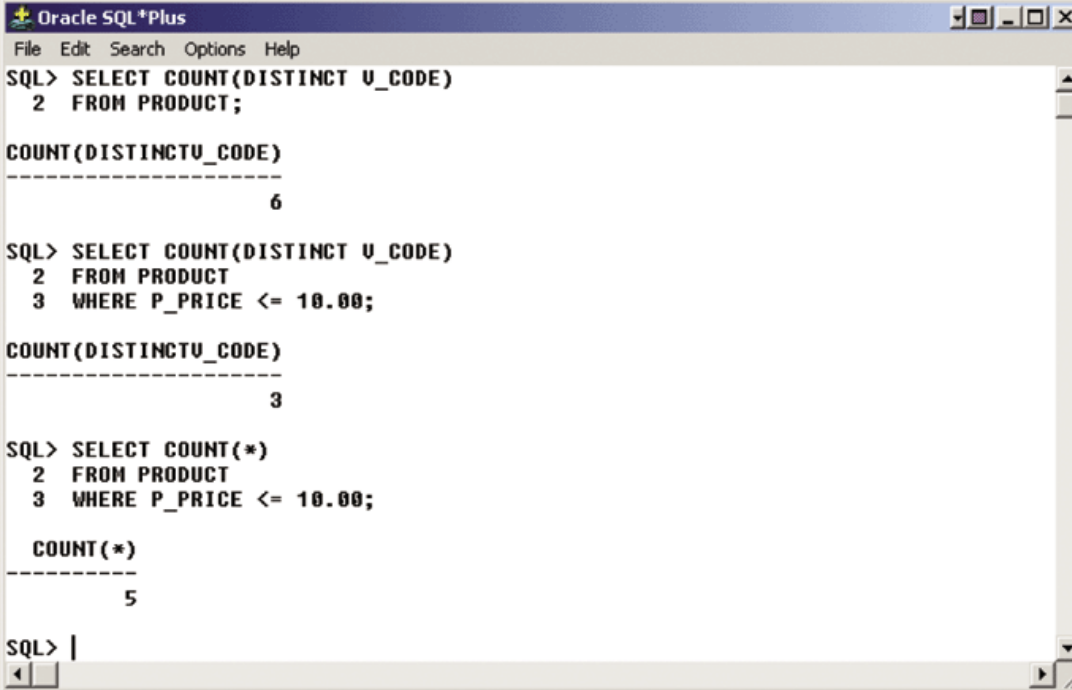
**TABLE
7.8**

**Some Basic SQL Aggregate
Functions**

FUNCTION	OUTPUT
COUNT	The number of rows containing non-null values
MIN	The minimum attribute value encountered in a given column
MAX	The maximum attribute value encountered in a given column
SUM	The sum of all values for a given column
AVG	The arithmetic mean (average) for a specified column

Aggregate Functions (continued)

FIGURE 7.21 COUNT function output examples



```
Oracle SQL*Plus
File Edit Search Options Help

SQL> SELECT COUNT(DISTINCT U_CODE)
2 FROM PRODUCT;

COUNT(DISTINCT U_CODE)
-----
6

SQL> SELECT COUNT(DISTINCT U_CODE)
2 FROM PRODUCT
3 WHERE P_PRICE <= 10.00;

COUNT(DISTINCT U_CODE)
-----
3

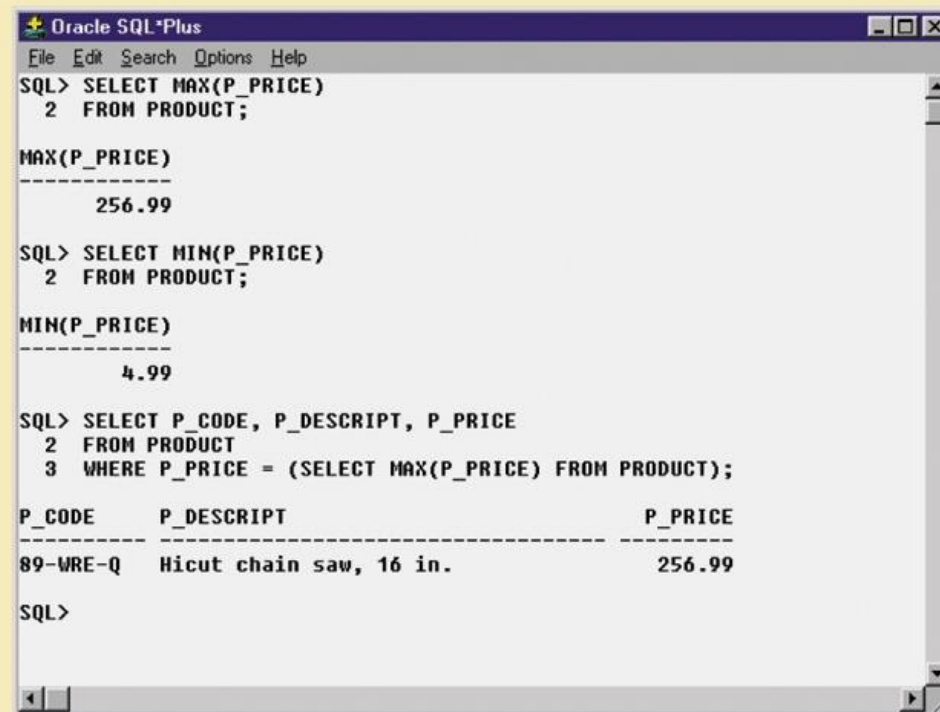
SQL> SELECT COUNT(*)
2 FROM PRODUCT
3 WHERE P_PRICE <= 10.00;

COUNT(*)
-----
5

SQL> |
```

Aggregate Functions (continued)

FIGURE 7.22 MAX and MIN function output examples



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT MAX(P_PRICE)
2 FROM PRODUCT;

MAX(P_PRICE)
-----
      256.99

SQL> SELECT MIN(P_PRICE)
2 FROM PRODUCT;

MIN(P_PRICE)
-----
        4.99

SQL> SELECT P_CODE, P_DESCRIPT, P_PRICE
2 FROM PRODUCT
3 WHERE P_PRICE = (SELECT MAX(P_PRICE) FROM PRODUCT);

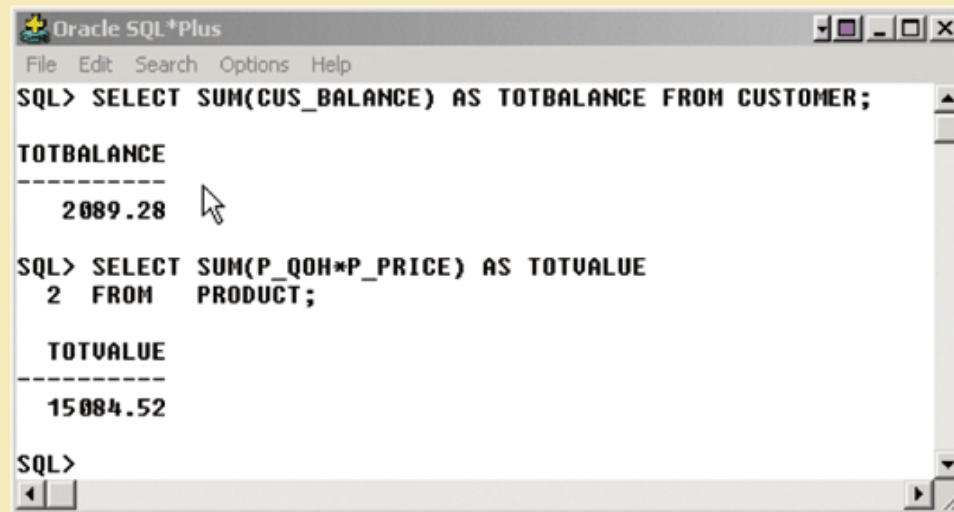
P_CODE      P_DESCRIPT      P_PRICE
-----
89-WRE-Q    Hicut chain saw, 16 in.      256.99

SQL>
```

Aggregate Functions (continued)

**FIGURE
7.23**

The total value of all items in the PRODUCT table



The screenshot shows the Oracle SQL*Plus interface. The first query calculates the sum of customer balances, and the second query calculates the total value of products based on quantity and price.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT SUM(CUS_BALANCE) AS TOTBALANCE FROM CUSTOMER;

TOTBALANCE
-----
  2089.28

SQL> SELECT SUM(P_QOH*P_PRICE) AS TOTVALUE
2 FROM PRODUCT;

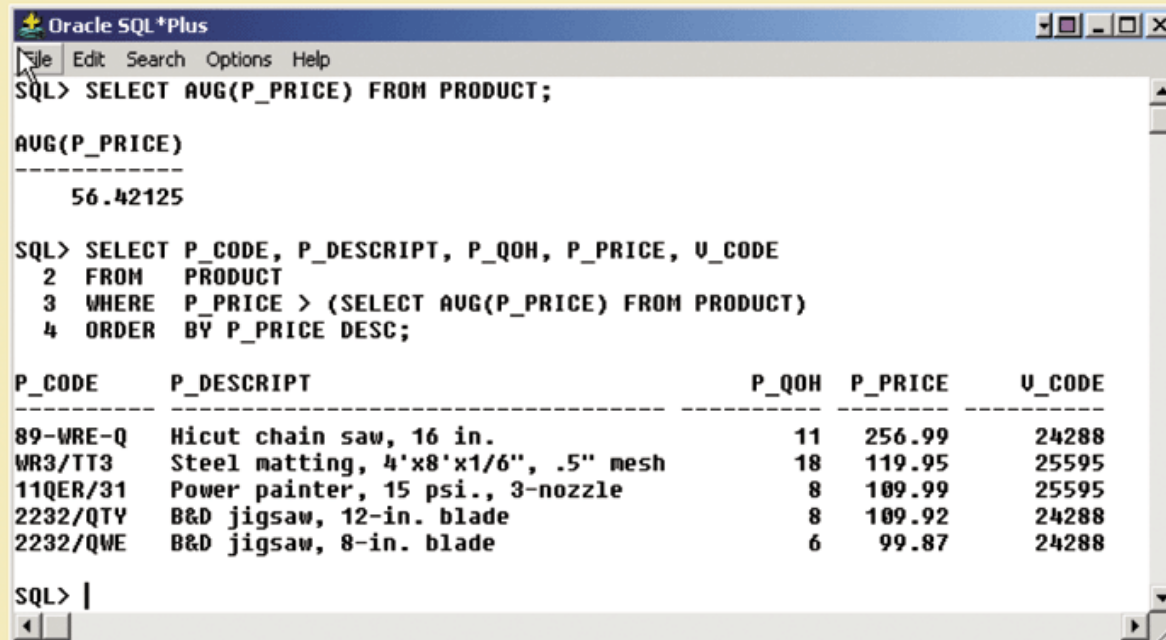
TOTVALUE
-----
15084.52

SQL>
```

Aggregate Functions (continued)

FIGURE
7.24

AVG function output examples



The screenshot shows the Oracle SQL*Plus interface. The first query calculates the average price of products. The second query filters products whose price is greater than the average price and orders them by price in descending order.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT AVG(P_PRICE) FROM PRODUCT;

AVG(P_PRICE)
-----
56.42125

SQL> SELECT P_CODE, P_DESCRIPT, P_QOH, P_PRICE, V_CODE
2 FROM PRODUCT
3 WHERE P_PRICE > (SELECT AVG(P_PRICE) FROM PRODUCT)
4 ORDER BY P_PRICE DESC;

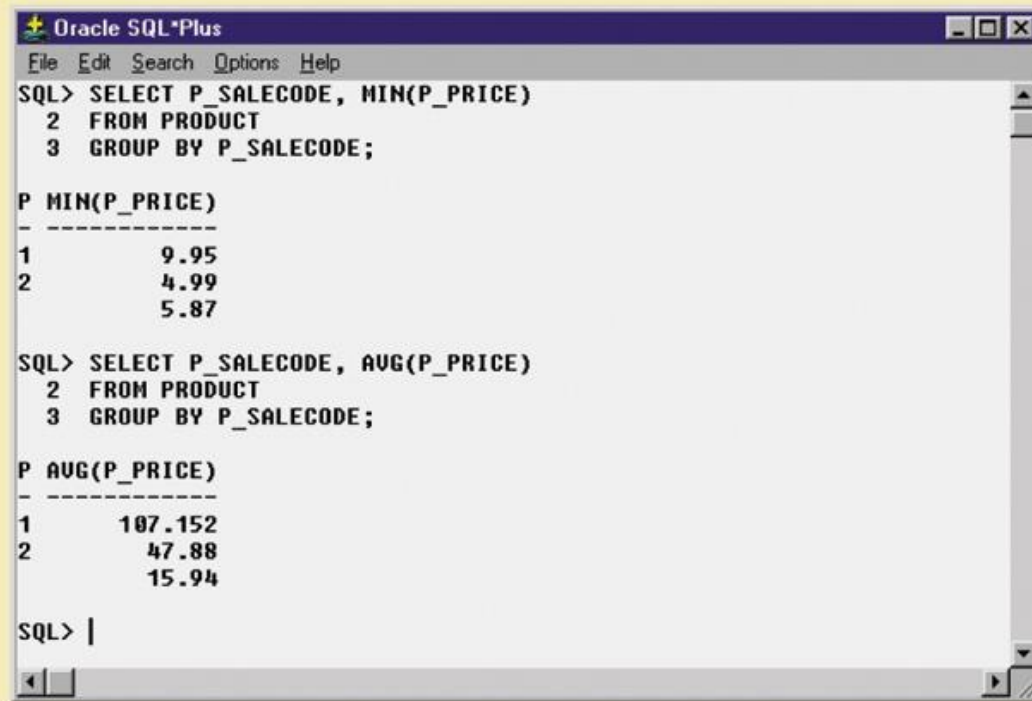
P_CODE      P_DESCRIPT                                P_QOH  P_PRICE  V_CODE
-----
89-WRE-Q    Hicut chain saw, 16 in.                   11    256.99   24288
WR3/TT3     Steel matting, 4'x8'x1/6", .5" mesh       18    119.95   25595
11QER/31    Power painter, 15 psi., 3-nozzle          8    109.99   25595
2232/QTY    B&D jigsaw, 12-in. blade                  8    109.92   24288
2232/QWE    B&D jigsaw, 8-in. blade                   6     99.87   24288

SQL> |
```


Grouping Data

FIGURE
7.25

GROUP BY clause output examples



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT P_SALECODE, MIN(P_PRICE)
2 FROM PRODUCT
3 GROUP BY P_SALECODE;

P MIN(P_PRICE)
-----
1          9.95
2          4.99
           5.87

SQL> SELECT P_SALECODE, AVG(P_PRICE)
2 FROM PRODUCT
3 GROUP BY P_SALECODE;

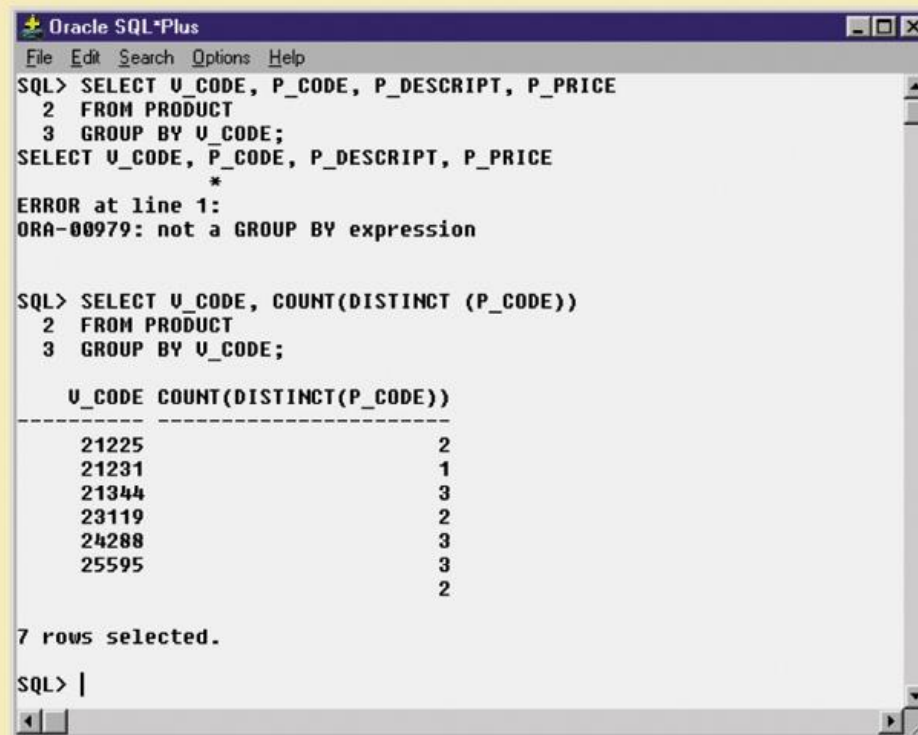
P AVG(P_PRICE)
-----
1        107.152
2         47.88
           15.94

SQL> |
```

Grouping Data (continued)

FIGURE
7.26

Incorrect and correct use of the GROUP BY clause



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT U_CODE, P_CODE, P_DESCRIPT, P_PRICE
      2 FROM PRODUCT
      3 GROUP BY U_CODE;
SELECT U_CODE, P_CODE, P_DESCRIPT, P_PRICE
      *
```

ERROR at line 1:
ORA-00979: not a GROUP BY expression

```
SQL> SELECT U_CODE, COUNT(DISTINCT (P_CODE))
      2 FROM PRODUCT
      3 GROUP BY U_CODE;
```

U_CODE	COUNT(DISTINCT(P_CODE))
21225	2
21231	1
21344	3
23119	2
24288	3
25595	3
	2

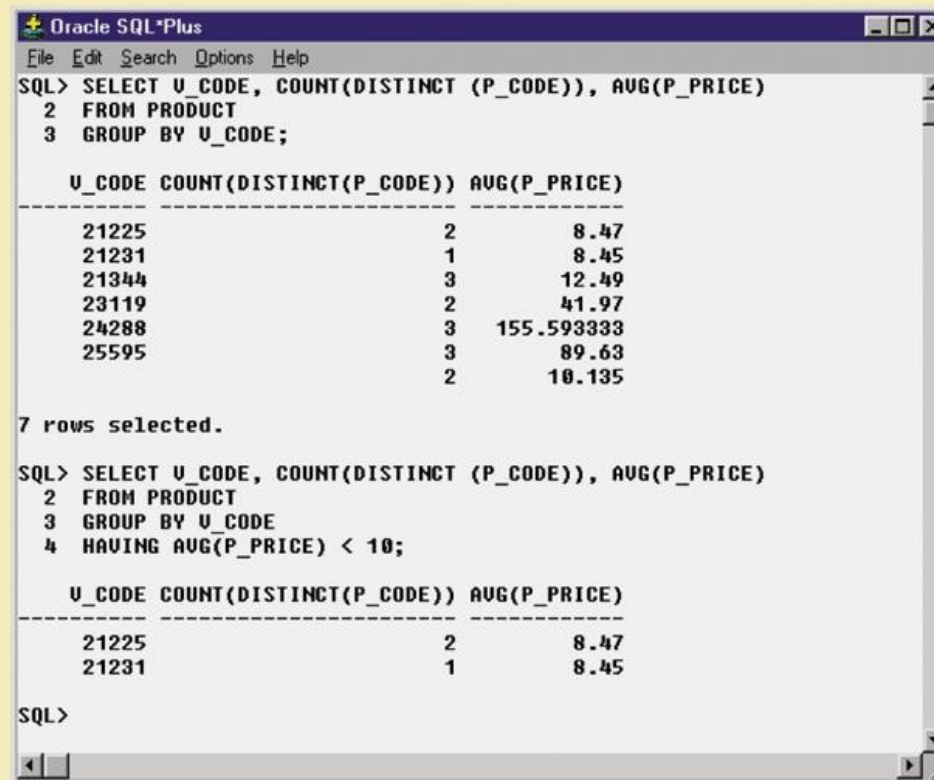
7 rows selected.

```
SQL> |
```

Grouping Data (continued)

FIGURE
7.27

An application of the HAVING clause



The screenshot shows the Oracle SQL*Plus interface. The first query groups products by U_CODE and shows the count of distinct P_CODES and the average P_PRICE. The second query filters the results using the HAVING clause to only show groups where the average price is less than 10.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> SELECT U_CODE, COUNT(DISTINCT (P_CODE)), AVG(P_PRICE)
2  FROM PRODUCT
3  GROUP BY U_CODE;

  U_CODE COUNT(DISTINCT(P_CODE))  AVG(P_PRICE)
-----
21225          2             8.47
21231          1             8.45
21344          3            12.49
23119          2            41.97
24288          3       155.593333
25595          3             89.63
          2            10.135

7 rows selected.

SQL> SELECT U_CODE, COUNT(DISTINCT (P_CODE)), AVG(P_PRICE)
2  FROM PRODUCT
3  GROUP BY U_CODE
4  HAVING AVG(P_PRICE) < 10;

  U_CODE COUNT(DISTINCT(P_CODE))  AVG(P_PRICE)
-----
21225          2             8.47
21231          1             8.45

SQL>
```

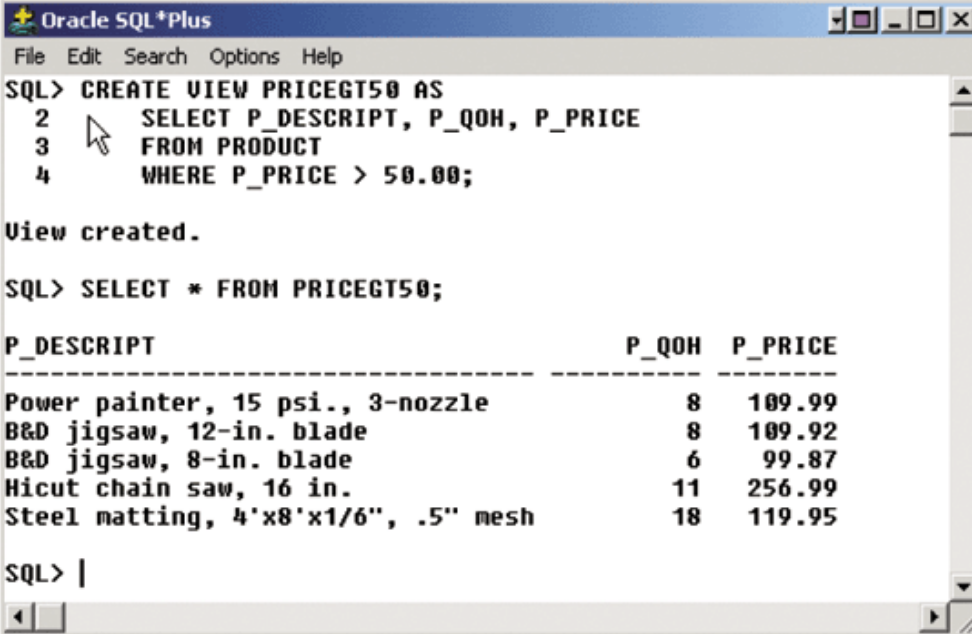
Virtual Tables: Creating a View

- View is virtual table based on SELECT query
 - Can contain columns, computed columns, aliases, and aggregate functions from one or more tables
- Base tables are tables on which view is based
- Create view by using CREATE VIEW command

Virtual Tables: Creating a View (continued)

FIGURE
7.28

Creating a virtual table with the CREATE VIEW command



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> CREATE VIEW PRICEGT50 AS
  2   SELECT P_DESCRIPTION, P_QOH, P_PRICE
  3   FROM PRODUCT
  4   WHERE P_PRICE > 50.00;

View created.

SQL> SELECT * FROM PRICEGT50;

P_DESCRIPTION                                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

SQL> |
```

Joining Database Tables

- Ability to combine (join) tables on common attributes is most important distinction between relational database and other databases
- Join is performed when data are retrieved from more than one table at a time
- Join is generally composed of an equality comparison between foreign key and primary key of related tables

Joining Tables with an Alias

- Alias can be used to identify source table
- Any legal table name can be used as alias
- Add alias after table name in FROM clause
 - FROM *tablename alias*

Summary

- SQL commands can be divided into two overall categories:
 - Data definition language commands
 - Data manipulation language commands
- The ANSI standard data types are supported by all RDBMS vendors in different ways
- Basic data definition commands allow you to create tables, indexes, and views

Summary (continued)

- DML commands allow you to add, modify, and delete rows from tables
- The basic DML commands are SELECT, INSERT, UPDATE, DELETE, COMMIT, and ROLLBACK
- INSERT command is used to add new rows to tables
- SELECT statement is main data retrieval command in SQL

Summary (continued)

- Many SQL constraints can be used with columns
- The column list represents one or more column names separated by commas
- WHERE clause can be used with SELECT, UPDATE, and DELETE statements to restrict rows affected by the DDL command

Summary (continued)

- **Aggregate functions**
 - Special functions that perform arithmetic computations over a set of rows
- **ORDER BY clause**
 - Used to sort output of SELECT statement
 - Can sort by one or more columns and use either an ascending or descending order
- **Join output of multiple tables with SELECT statement**

Summary (continued)

- Natural join uses join condition to match only rows with equal values in specified columns
- Right outer join and left outer join used to select rows that have no matching values in other related table