INTRODUCTION TO SQL

CHAPTER 6:
INTRODUCTION TO SQL

Modern Database Management

12th Edition

Jeff Hoffer, Ramesh Venkataraman, Heikki Topi

OBJECTIVES

- Define terms
- Interpret history and role of SQL
- Define a database using SQL data definition language
- Write single table queries using SQL

SQL OVERVIEW

- Structured Query Language – often pronounced “Sequel”

- The standard for relational database management systems (RDBMS)

- RDBMS: A database management system that manages data as a collection of tables in which all relationships are represented by common values in related tables

HISTORY OF SQL

- 1970–E. F. Codd develops relational database concept
- 1974-1979–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
- 1989, 1992, 1999, 2003, 2006, 2008, 2011–Major ANSI
- standard updates
- Current–SQL is supported by most major database vendors

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)

SQL ENVIRONMENT

Ò Catalog

É A set of schemas that constitute the description of a database

Ò Schema

É The structure that contains descriptions of objects created by a user (base tables, views, constraints)

Ò Data Definition Language (DDL)

É Commands that define a database, including creating, altering, and dropping tables and establishing constraints

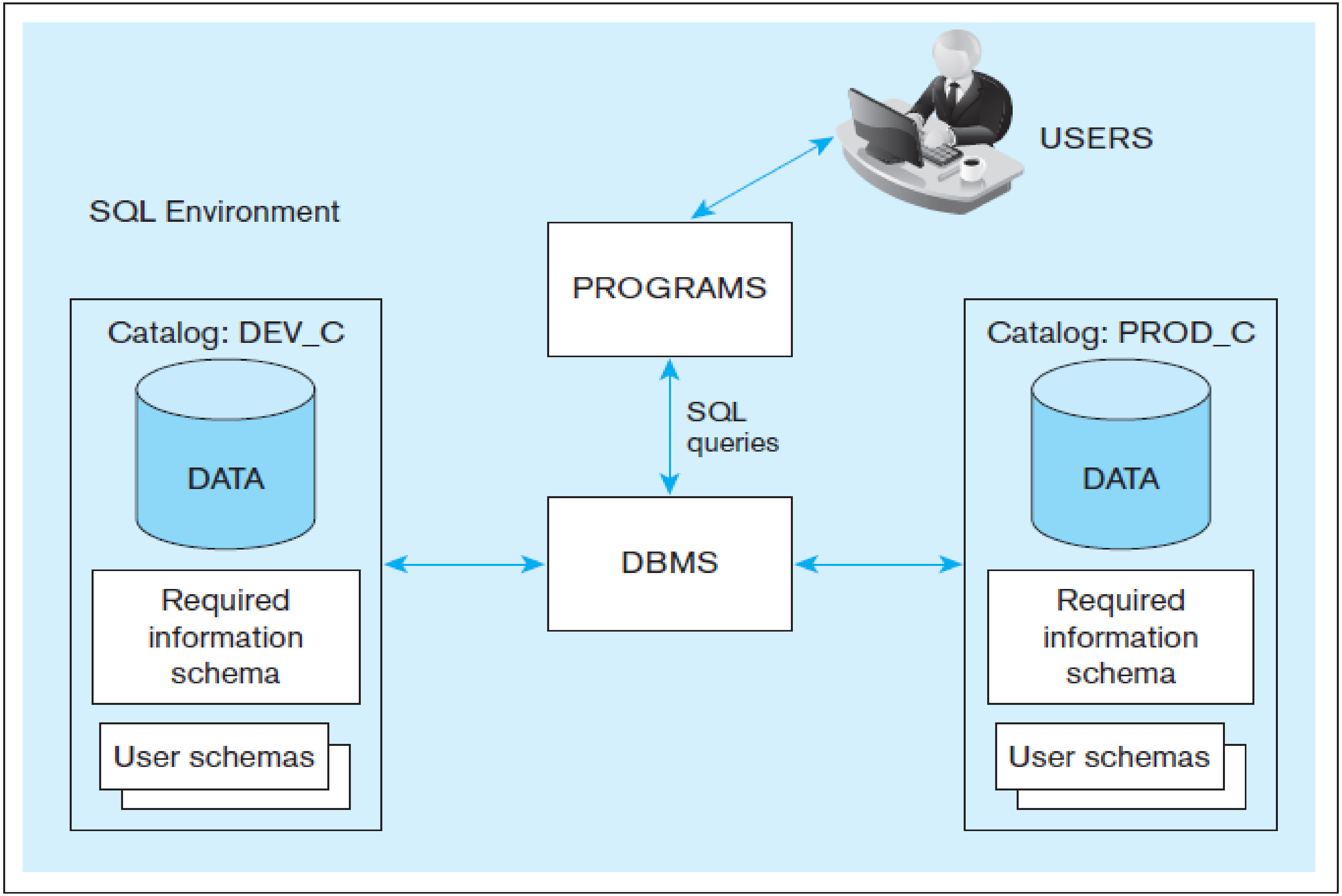
Ò Data Manipulation Language (DML)

É Commands that maintain and query a database

Ò Data Control Language (DCL)

É Commands that control a database, including administering privileges and committing data

A simplified schematic of a typical SQL environment, as described by the SQL: 2011 standard



Structured Query Language

- CRUD operations
  - **C**reate
  - **R**ead
  - **U**pdate
  - **D**elete
- Categories of SQL functions
  - Data definition language (DDL): create, drop , alter
  - Data manipulation language (DML): select, insert, update, delete, different operators & functions
  - Transaction control language (TCL): commit, rollback
  - Data control language (DCL): grant, revoke
- Other SQL commands: indexes, constraints, triggers, authorization, transactions, …

SQL DATABASE DEFINITION

- Data Definition Language (DDL)
- Major CREATE statements:
- CREATE SCHEMA–defines a portion of the database owned by a particular user
- CREATE TABLE–defines a new table and its columns
- CREATE VIEW–defines a logical table from one or more tables or views
- Other CREATE statements: CHARACTER SET, COLLATION, TRANSLATION, ASSERTION, DOMAIN

Data Definition (DDL)

Command or Option	Description
CREATE TABLE	Creates a new table in the user’s database schema
CREATE INDEX	Creates an index for a table
CREATE VIEW	Creates a dynamic subset of rows and columns from one or more tables
ALTER TABLE	Modifies a table’s definition (adds, modifies, or deletes attributes or constraints)
DROP TABLE	Permanently deletes a table (and its data)
DROP INDEX	Permanently deletes an index
DROP VIEW	Permanently deletes a view

Data Manipulation (DML)

Command or Option	Description
SELECT	Selects attributes from rows in one or more tables or views
INSERT	Inserts row(s) into a table
UPDATE	Modifies the value of one or more attributes in one or more table’s rows
DELETE	Deletes one or more rows from a table
Comparison Operators	=, <, >, <=, >=, <>, != used in conditional expressions
Logical Operators	AND/OR/NOT: used in conditional expressions
Special Operators	BETWEEN, IN, LIKE, IS NULL, EXISTS, DISTINCT: used in conditional expressions
Aggregate Functions	COUNT, MIN, MAX, SUM, AVG: usedwith SELECT to return mathematical summaries on columns

Transaction Control (TCL) & Data Control (DCL)

Command or Option	Description
Transaction Control Language	
COMMIT	Permanently saves data changes
ROLLBACK	Restores data to its original values
Data Control Language	
GRANT	Gives a user permission to take a system action or access a data object
REVOKE	Removes a previously granted permission from a user

SQL

- Data type: specification about the kinds of data that can be stored in an attribute
  - Influence queries that retrieve data
- Fundamental types of data
  - Character data: text, constant & variable character string
  - Numeric data: Integer (different sizes) , Floating (singe & double precision)
  - Boolean data:
  - Blob data
  - MySQL is not case sensitive, and neither is the SQL standard. It's just common practice to write the commands upper-case.

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 1:M relationships
- Restricting:
- Deletes of primary records
- Updates of primary records
- Inserts of dependent records

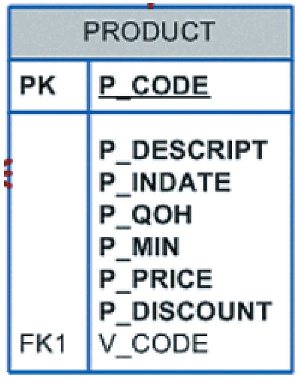
Creating Tables

```
CREATE TABLE tablename (
  column1        data type [constraint] [, [constraint] ] [,
  column2        data type [, column2]) ] [,
  PRIMARY KEY (column1   [, column2]) REFERENCES tablename
  FOREIGN KEY (column1   [,
  CONSTRAINT   constraint ] );
```

- Some SQL constraints
  - FOREIGN KEY, NOT NULL, UNIQUE, DEFAULT, CHECK
  - CHECK Constraints support on MySQL added in Version 8.0.16
  - Check out: https://dev.mysql.com/doc/refman/8.0/en/create-table-check-constraints.html

Table Creation Example

```
CREATE TABLE PRODUCT (
    P_CODE      VARCHAR(35) NOT NULL  UNIQUE,
    P_DESCRIPT  VARCHAR(35) NOT NULL,
    P_INDATE    DATE        NOT NULL,
    P_QOH       SMALLINT    NOT NULL,
    P_MIN       SMALLINT    NOT NULL,
    P_PRICE     NUMBER(8,2) NOT NULL,
    P_DISCOUNT  NUMBER(5,2) NOT NULL,
    V_CODE      INTEGER     NOT NULL, PRIMARY KEY (P_CODE),
    FOREIGN KEY (V_CODE) REFERENCES VENDOR ON UPDATE CASCADE);
```



PRODUCT	
PK	P_CODE
	P_DESCRIPT P_INDATE P_QOH P_MIN P_PRICE P_DISCOUNT
FK1	V_CODE

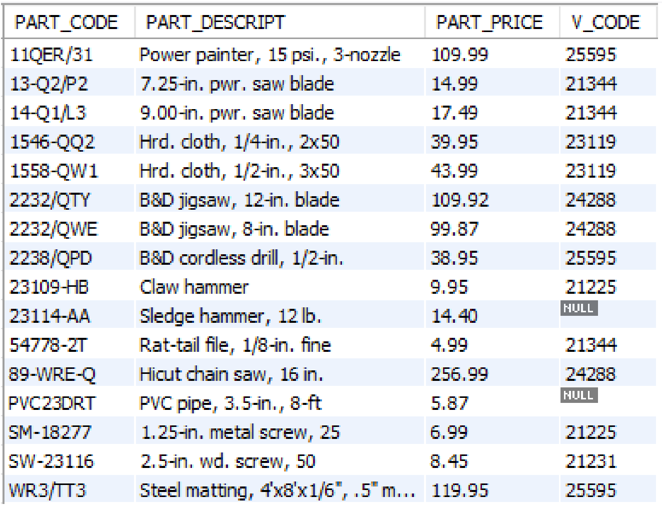
Table Creation Example

```
CREATE TABLE INVOICE (
 INV_NUMBER    INTEGER        NOT NULL    PRIMARY KEY,
 CUS_CODE      INTEGER        NOT NULL,
 INV_DATE      DATE           NOT NULL,

  CONSTRAINT INV_CHK1 CHECK (INV_DATE > '01-JAN-2018'));
```

Creating Tables From Existing Ones Using Queries

- Create a table with a SELECT statement
  - Rapidly creates a new table based on selected columns and rows of an existing table using a subquery
  - Automatically copies all the data rows returned

```
CREATE TABLE PART AS SELECT P_CODE AS PAR
  P_DESCRIPT AS PART_DESCRIPT,
P_PRICE AS PART_PRICE, V_CODE FROM
PRODUCT;
```



PART_CODE	PART_DESCRIPT	PART_PRICE	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	109.99	25595
13-Q2/P2	7.25-in. pwr. saw blade	14.99	21344
14-Q1/L3	9.00-in. pwr. saw blade	17.49	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23119
2232/QTY	B&D jigsaw, 12-in. blade	109.92	24288
2232/QWE	B&D jigsaw, 8-in. blade	99.87	24288
2238/QPD	B&D cordless drill, 1/2-in.	38.95	25595
23109-HB	Claw hammer	9.95	21225
23114-AA	Sledge hammer, 12 lb.	14.40	NULL
54778-2T	Rat-tail file, 1/8-in. fine	4.99	21344
89-WRE-Q	Hicut chain saw, 16 in.	256.99	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.87	NULL
SM-18277	1.25-in. metal screw, 25	6.99	21225
SW-23116	2.5-in. wd. screw, 50	8.45	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	119.95	25595

Creating SQL Indexes

- SQL indexes
  - CREATE INDEX improves the efficiency of searches and avoids duplicate column values
  - DROP Index deletes an index

Altering Table Structures

- ALTER TABLE - the command to make changes in the table structure followed by a keyword that produces the specific change you want to make
  - ADD, MODIFY, and DROP
- Changing a column's data characteristics
  - If the column to be changed already contains data, you can make changes in the column's characteristics if those changes do not alter the data type
- Adding a column
  - You can alter an existing table by adding one or more columns
  - Be careful not to include the NOT NULL clause for the new column

Altering Table Structures

- ALTER TABLE - the command to make changes in the table structure followed by a keyword that produces the specific change you want to make
- – ADD, MODIFY, and DROP

- **ALTER TABLE** tablename
- {**ADD** | **MODIFY**} (column_name *datatype*
- [{**ADD** | **MODIFY**} column_name *datatype*]);

- **ALTER TABLE** tablename
- **ADD CONSTRAINT** [ADD constraint];

- **ALTER TABLE** tablename
- **DROP** { **COLUMN** column_name | **CONSTRAINT** constraint_name |
- **PRIMARY KEY** column_name | **FOREIGN KEY** column_name };

Adding Primary, Foreign Keys & Checks to Existing Tables

- Add primary key:

```
ALTER TABLE PART
ADD PRIMARY KEY (PART_CODE);
```

- Add foreign key:

```
ALTER TABLE PART
ADD FOREIGN KEY (V_CODE) REFERENCES VENDOR;
```

- Add check constraint: (MySQL Version 8.0.16 or later)

```
ALTER TABLE PART
ADD CHECK (PART_PRICE >= 0);
```

Adding & Dropping Columns From Tables

- Adding a column:

```
ALTER TABLE PART
ADD(V_ORDER CHAR(1));
```

- Dropping a column:

```
ALTER TABLE PART
DROP COLUMN V_ORDER;
```

- Modifying a column:

```
ALTER TABLE PERSONS
MODIFY AGE INT NOT NULL;
```

Dropping Tables

- Deleting a table from the database

```
DROP TABLE PART;
```

Inserting Data into Tables

- Adding table rows: INSERT command
  - Inserting rows with null attributes: use NULL entry
  - Inserting rows with optional attributes: indicate attributes that have required values

```
INSERT INTO tablename VALUES (value1,
 value2, …, valueN;)
```

Data Manipulation Commands

- UPDATE: modify data in a table

```
UPDATE    tablename
SET columnname = expression [, columnname =
 expression] [WHERE conditionlist ];
```

- DELETE: deleting table rows

```
DELETE FROM tablename
[WHERE    conditionlist ];
```

- ROLLBACK: restore the database to its previous condition

```
ROLLBACK;
```

Sequences

- Sequences are an independent object (not a data type) in database
- Sequences are lists of integers
- Usually used as IDs of entities such as CUS_CODE, INV_NUMBER etc
- In MySQL, use AUTO_INCREMENT as constraints in CREATE TABLE

```
CREATE SEQUENCE
sequence_1
start with 1
increment by 1
minvalue 0
maxvalue 100
cycle;
```

```
_________________
| ID | NAME |
-----------------
| 1 | Ramesh |
| 2 | Suresh |
-----------------
```

```
CREATE TABLE students
( ID number(10), NAME char(20) );

INSERT into students
VALUES(sequence_1.nextval,'Ramesh’);
INSERT into students
VALUES(sequence_1.nextval,'Suresh');
```

The SELECT Statement Structure

```
SELECT      attribute(s)
FROM        table(s)
```

REQUIRED

```
WHERE       condition(s)
GROUP BY    attribute(s)
HAVING      condition(s)
ORDER BY
            attribute(s)  [ASC | DESC]
LIMIT
            quantity
```

OPTIONAL

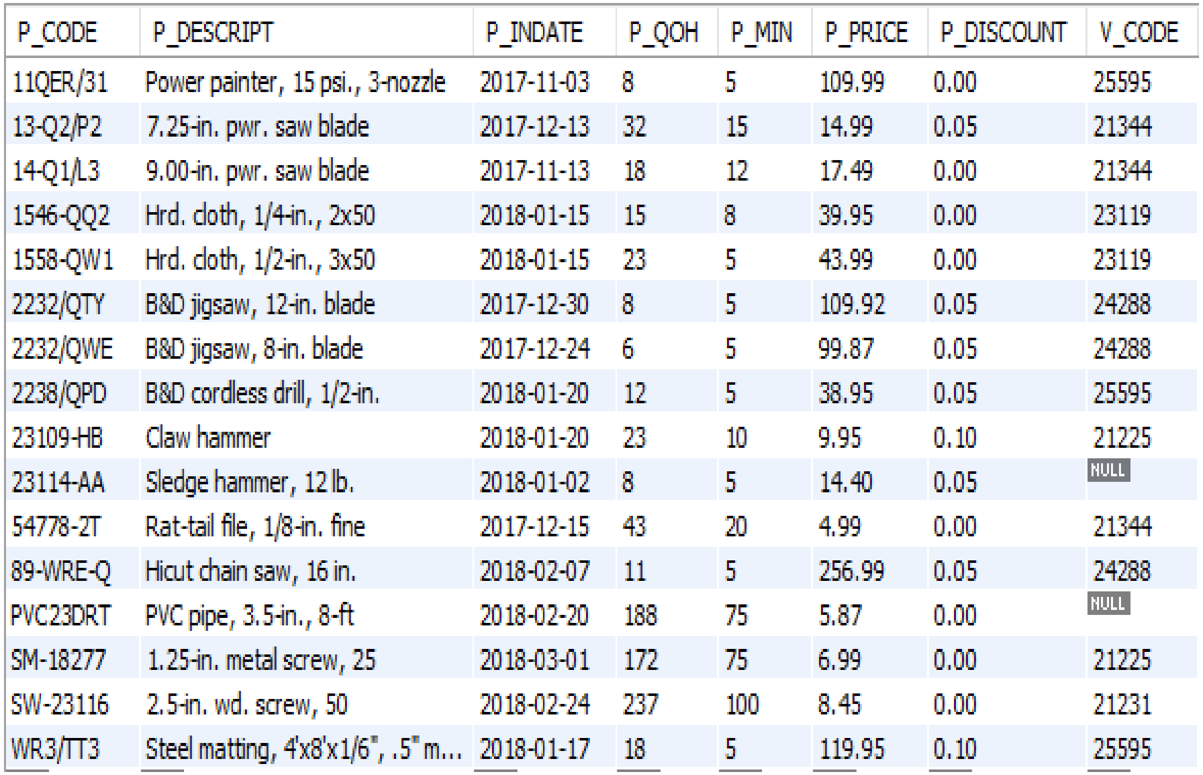
A Simple SELECT Statement

```
SELECT
attribute list
FROM
table list;
```

- A wildcard character * is a symbol that can be used as a general substitute for other characters or commands
- Column aliases - alternative name for a column or table in a SQL statement
- Computed/computed columns - represents a derived attribute
- Arithmetic operators - **+, - , *, /, ^** ; follows the rule of precedence

The SELECT Statement – Include Everything

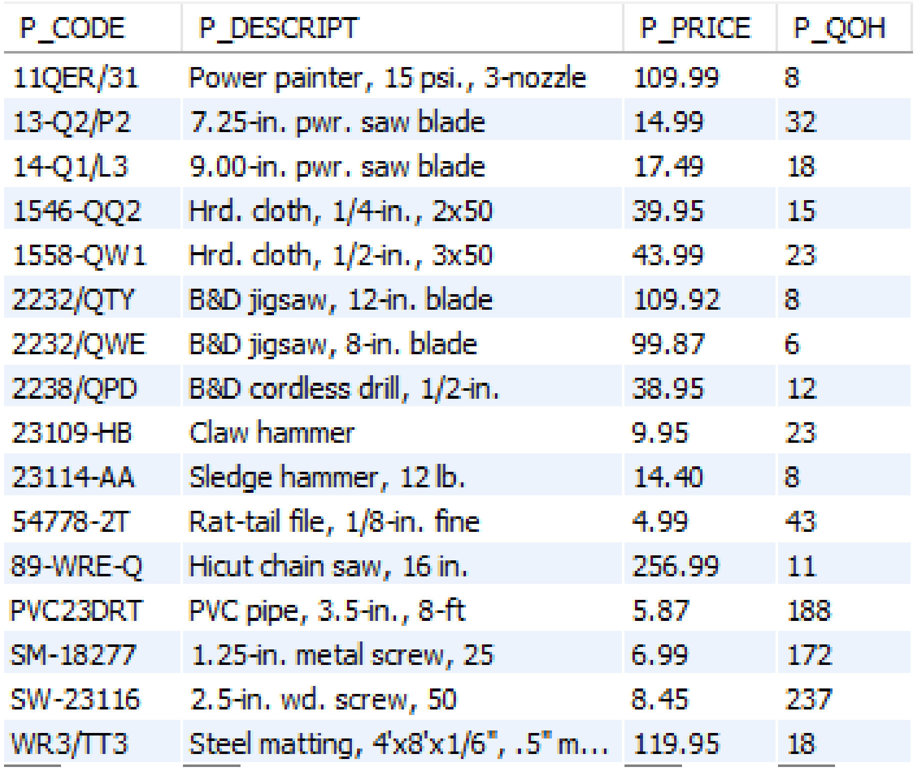
```
SELECT * FROM  PRODUCT;
```



P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	2017-11-03	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	2017-12-13	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	2017-11-13	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	2018-01-15	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	2018-01-15	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	2017-12-30	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	2017-12-24	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	2018-01-20	12	5	38.95	0.05	25595
23109-HB	Claw hammer	2018-01-20	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	2018-01-02	8	5	14.40	0.05	NULL
54778-2T	Rat-tail file, 1/8-in. fine	2017-12-15	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	2018-02-07	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	2018-02-20	188	75	5.87	0.00	NULL
SM-18277	1.25-in. metal screw, 25	2018-03-01	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	2018-02-24	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	2018-01-17	18	5	119.95	0.10	25595

The SELECT Statement with a Column List

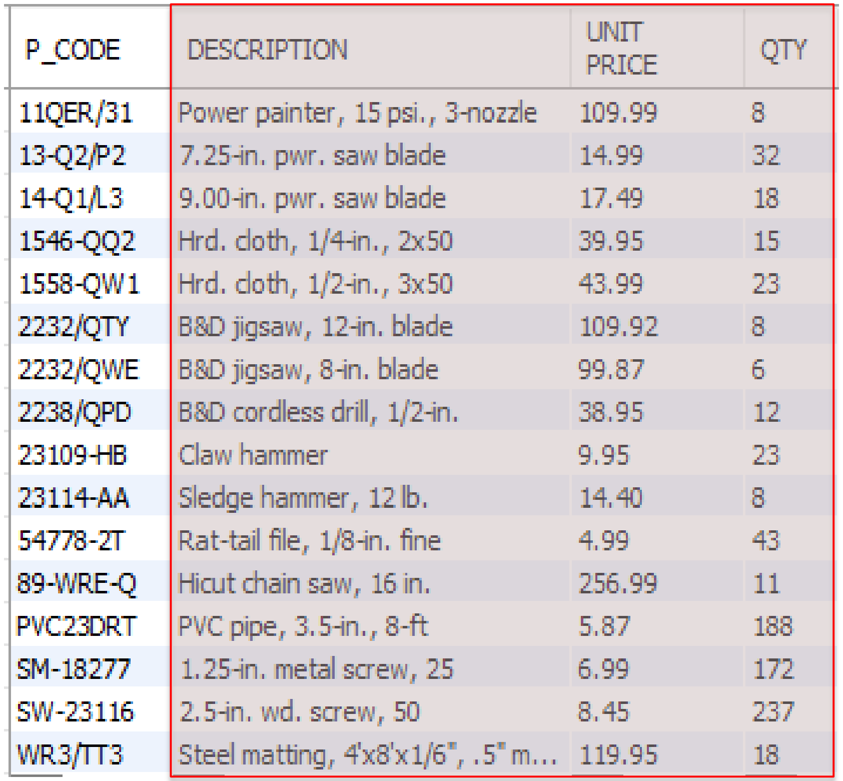
```
SELECT
      P_CODE,
      P_DESCRIPT,
      P_PRICE,
      P_QOH
FROM
      PRODUCT;
```



P_CODE	P_DESCRIPT	P_PRICE	P_QOH
11QER/31	Power painter, 15 psi., 3-nozzle	109.99	8
13-Q2/P2	7.25-in. pwr. saw blade	14.99	32
14-Q1/L3	9.00-in. pwr. saw blade	17.49	18
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	15
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23
2232/QTY	B&D jigsaw, 12-in. blade	109.92	8
2232/QWE	B&D jigsaw, 8-in. blade	99.87	6
2238/QPD	B&D cordless drill, 1/2-in.	38.95	12
23109-HB	Claw hammer	9.95	23
23114-AA	Sledge hammer, 12 lb.	14.40	8
54778-2T	Rat-tail file, 1/8-in. fine	4.99	43
89-WRE-Q	Hicut chain saw, 16 in.	256.99	11
PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.87	188
SM-18277	1.25-in. metal screw, 25	6.99	172
SW-23116	2.5-in. wd. screw, 50	8.45	237
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	119.95	18

The SELECT Statement with Column Aliases

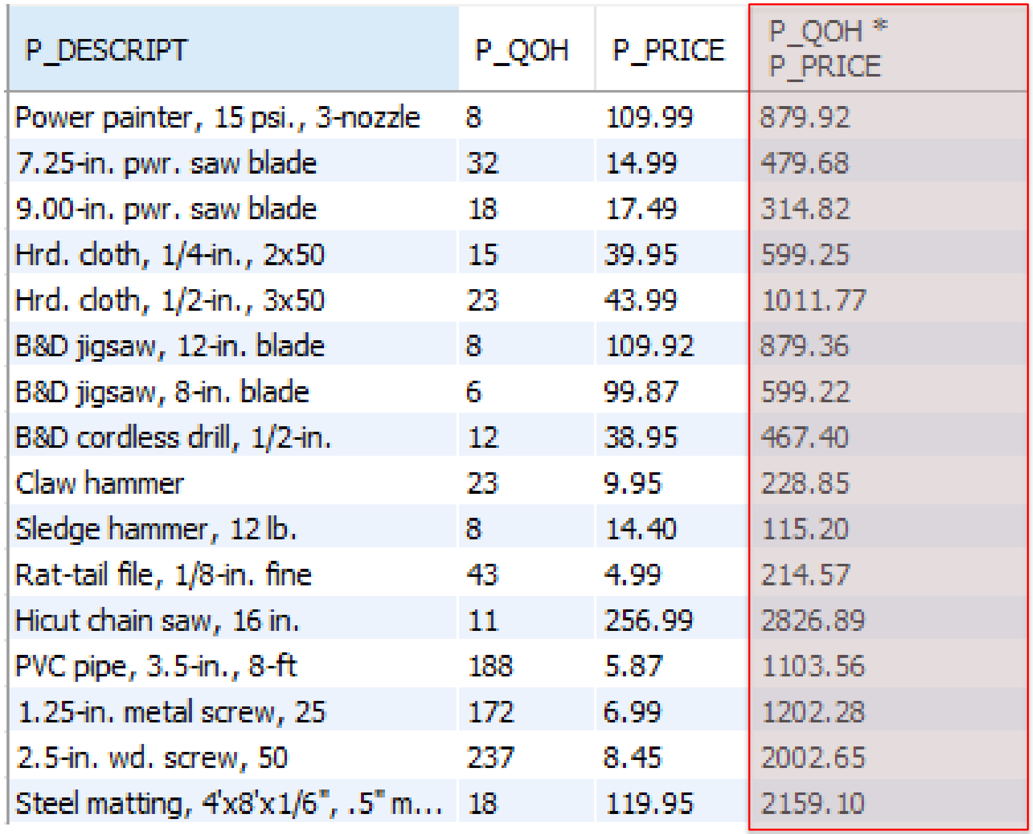
```
SELECT
    P_CODE,
    P_DESCRIPT AS DESCRIPTION,
    P_PRICE AS "UNIT PRICE",
    P_QOH AS QTY
FROM
    PRODUCT;
```



P_CODE	DESCRIPTION	UNIT PRICE	QTY
11QER/31	Power painter, 15 psi., 3-nozzle	109.99	8
13-Q2/P2	7.25-in. pwr. saw blade	14.99	32
14-Q1/L3	9.00-in. pwr. saw blade	17.49	18
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	15
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23
2232/QTY	B&D jigsaw, 12-in. blade	109.92	8
2232/QWE	B&D jigsaw, 8-in. blade	99.87	6
2238/QPD	B&D cordless drill, 1/2-in.	38.95	12
23109-HB	Claw hammer	9.95	23
23114-AA	Sledge hammer, 12 lb.	14.40	8
54778-2T	Rat-tail file, 1/8-in. fine	4.99	43
89-WRE-Q	Hicut chain saw, 16 in.	256.99	11
PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.87	188
SM-18277	1.25-in. metal screw, 25	6.99	172
SW-23116	2.5-in. wd. screw, 50	8.45	237
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	119.95	18

The SELECT Statement with Computed Column

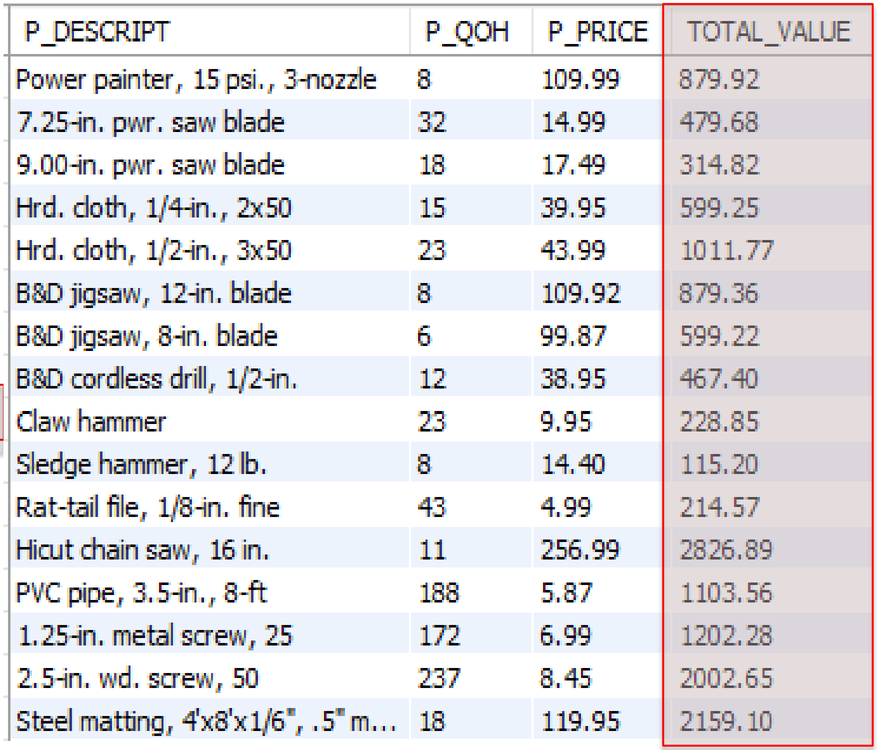
```
SELECT
P_DESCRIPT,
P_QOH, P_PRICE,

        P_QOH*P_PRICE
FROM
        PRODUCT;
```



P_DESCRIPT	P_QOH	P_PRICE	P_QOH * P_PRICE
Power painter, 15 psi., 3-nozzle	8	109.99	879.92
7.25-in. pwr. saw blade	32	14.99	479.68
9.00-in. pwr. saw blade	18	17.49	314.82
Hrd. cloth, 1/4-in., 2x50	15	39.95	599.25
Hrd. cloth, 1/2-in., 3x50	23	43.99	1011.77
B&D jigsaw, 12-in. blade	8	109.92	879.36
B&D jigsaw, 8-in. blade	6	99.87	599.22
B&D cordless drill, 1/2-in.	12	38.95	467.40
Claw hammer	23	9.95	228.85
Sledge hammer, 12 lb.	8	14.40	115.20
Rat-tail file, 1/8-in. fine	43	4.99	214.57
Hicut chain saw, 16 in.	11	256.99	2826.89
PVC pipe, 3.5-in., 8-ft	188	5.87	1103.56
1.25-in. metal screw, 25	172	6.99	1202.28
2.5-in. wd. screw, 50	237	8.45	2002.65
Steel matting, 4'x8'x1/6", .5" m...	18	119.95	2159.10

The SELECT Statement with Computed Column with Alias

```
SELECT
P_DESCRIPT,
P_QOH, P_PRICE,

    P_QOH*P_PRICE AS TOTAL_VALUE
FROM
    PRODUCT;
```



P_DESCRIPT	P_QOH	P_PRICE	TOTAL_VALUE
Power painter, 15 psi., 3-nozzle	8	109.99	879.92
7.25-in. pwr. saw blade	32	14.99	479.68
9.00-in. pwr. saw blade	18	17.49	314.82
Hrd. cloth, 1/4-in., 2x50	15	39.95	599.25
Hrd. cloth, 1/2-in., 3x50	23	43.99	1011.77
B&D jigsaw, 12-in. blade	8	109.92	879.36
B&D jigsaw, 8-in. blade	6	99.87	599.22
B&D cordless drill, 1/2-in.	12	38.95	467.40
Claw hammer	23	9.95	228.85
Sledge hammer, 12 lb.	8	14.40	115.20
Rat-tail file, 1/8-in. fine	43	4.99	214.57
Hicut chain saw, 16 in.	11	256.99	2826.89
PVC pipe, 3.5-in., 8-ft	188	5.87	1103.56
1.25-in. metal screw, 25	172	6.99	1202.28
2.5-in. wd. screw, 50	237	8.45	2002.65
Steel matting, 4'x8'x1/6", .5" m...	18	119.95	2159.10

SELECT Statement: Date Arithmetic

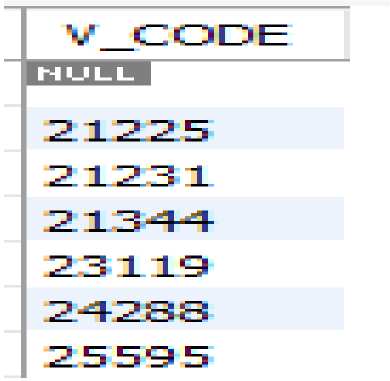
- Date arithmetic
  - Values are stored as a number of days
  - Can perform date arithmetic in a query

```
SELECT P_CODE, P_INDATE, P_INDATE + 90 AS
 EXPDATE
FROM PRODUCT;
```

```
SELECT P_CODE, P_INDATE, SYSDATE – 90 AS
 CUTOFF
FROM PRODUCT;
```

The SELECT Statement: Listing Unique Values

```
SELECT DISTINCT
V_CODE
FROM
PRODUCT;
```



```
V_CODE
NULL
21225
21231
21344
23119
24288
25595
```

- Generates a list of only those values that are different from one another
- sometimes called UNIQUE

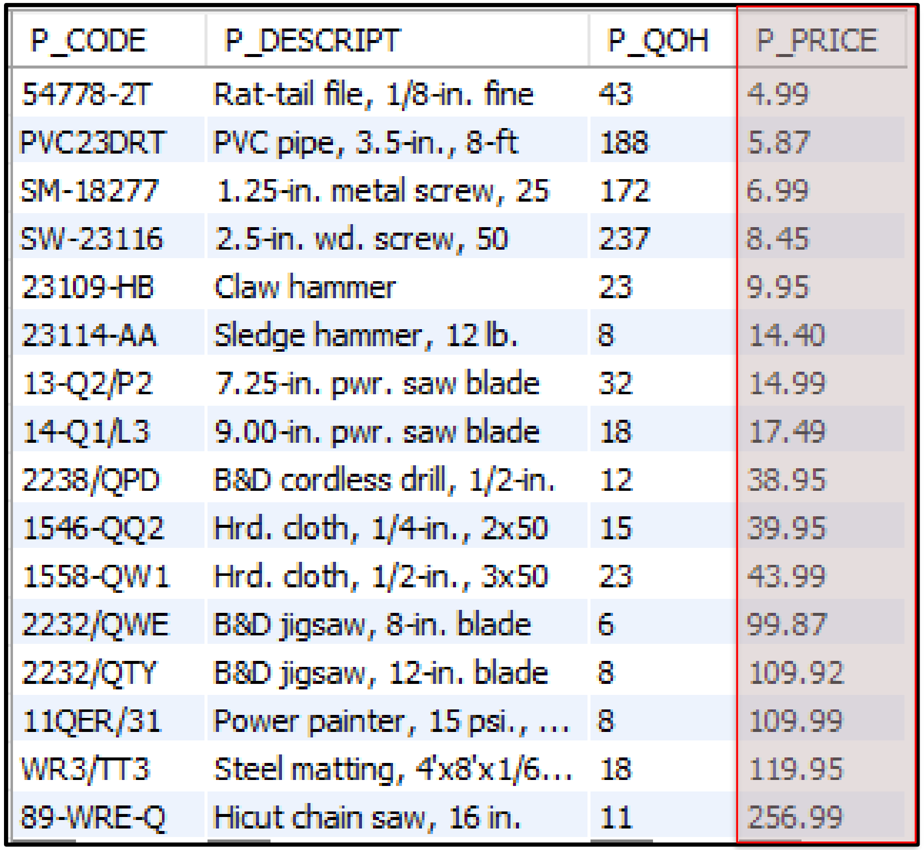
ORDER BY Clause

- ORDER BY clause is especially useful when the listing order is important:

```
SELECT      attribute(s) table(s)

FROM        attribute(s) [ASC | DESC] ;

ORDER BY
```

ORDER BY Example

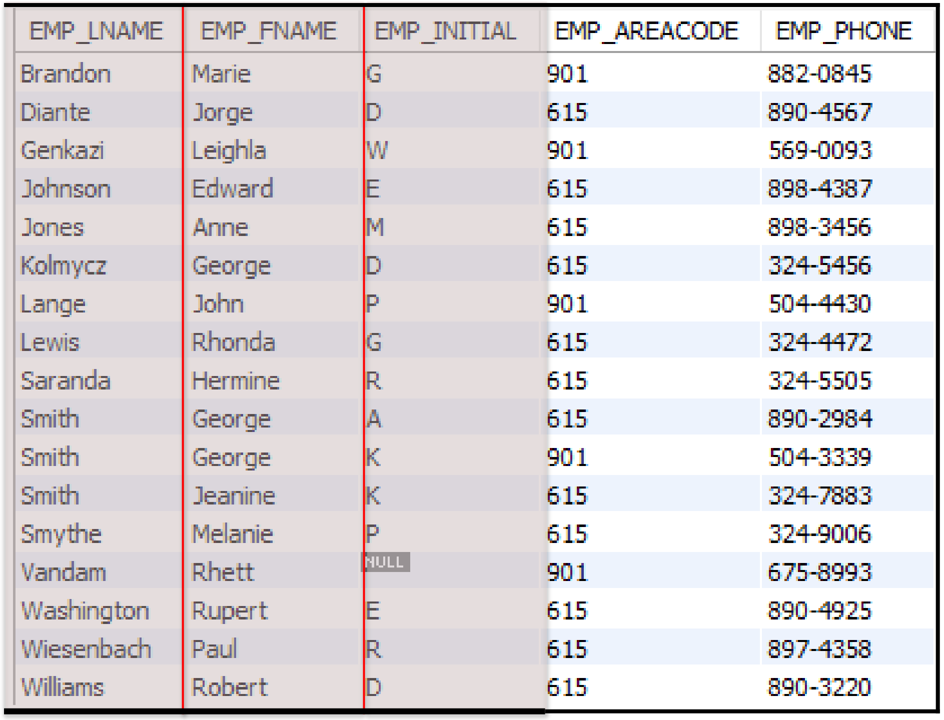
```
SELECT
P_CODE,
P_DESCRIPT,
P_QOH, P_PRICE,
FROM
PRODUCT

ORDER BY P_PRICE;
```



P_CODE	P_DESCRIPT	P_QOH	P_PRICE
54778-2T	Rat-tail file, 1/8-in. fine	43	4.99
PVC23DRT	PVC pipe, 3.5-in., 8-ft	188	5.87
SM-18277	1.25-in. metal screw, 25	172	6.99
SW-23116	2.5-in. wd. screw, 50	237	8.45
23109-HB	Claw hammer	23	9.95
23114-AA	Sledge hammer, 12 lb.	8	14.40
13-Q2/P2	7.25-in. pwr. saw blade	32	14.99
14-Q1/L3	9.00-in. pwr. saw blade	18	17.49
2238/QPD	B&D cordless drill, 1/2-in.	12	38.95
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15	39.95
1558-QW1	Hrd. cloth, 1/2-in., 3x50	23	43.99
2232/QWE	B&D jigsaw, 8-in. blade	6	99.87
2232/QTY	B&D jigsaw, 12-in. blade	8	109.92
11QER/31	Power painter, 15 psi., ...	8	109.99
WR3/TT3	Steel matting, 4'x8'x1/6...	18	119.95
89-WRE-Q	Hicut chain saw, 16 in.	11	256.99

ORDER BY Example – Cascading Order

```
SELECT
EMP_LNAME,
EMP_FNAME,
EMP_INITIAL,
EMP_AREACODE,
EMP_PHONE
FROM
EMPLOYEE
ORDER BY EMP_LNAME,
     EMP_FNAME,
     EMP_INITIAL;
```



EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_AREACODE	EMP_PHONE
Brandon	Marie	G	901	882-0845
Diante	Jorge	D	615	890-4567
Genkazi	Leighla	W	901	569-0093
Johnson	Edward	E	615	898-4387
Jones	Anne	M	615	898-3456
Kolmycz	George	D	615	324-5456
Lange	John	P	901	504-4430
Lewis	Rhonda	G	615	324-4472
Saranda	Hermine	R	615	324-5505
Smith	George	A	615	890-2984
Smith	George	K	901	504-3339
Smith	Jeanine	K	615	324-7883
Smythe	Melanie	P	615	324-9006
Vandam	Rhett	NULL	901	675-8993
Washington	Rupert	E	615	890-4925
Wiesenbach	Paul	R	615	897-4358
Williams	Robert	D	615	890-3220

WHERE Clause

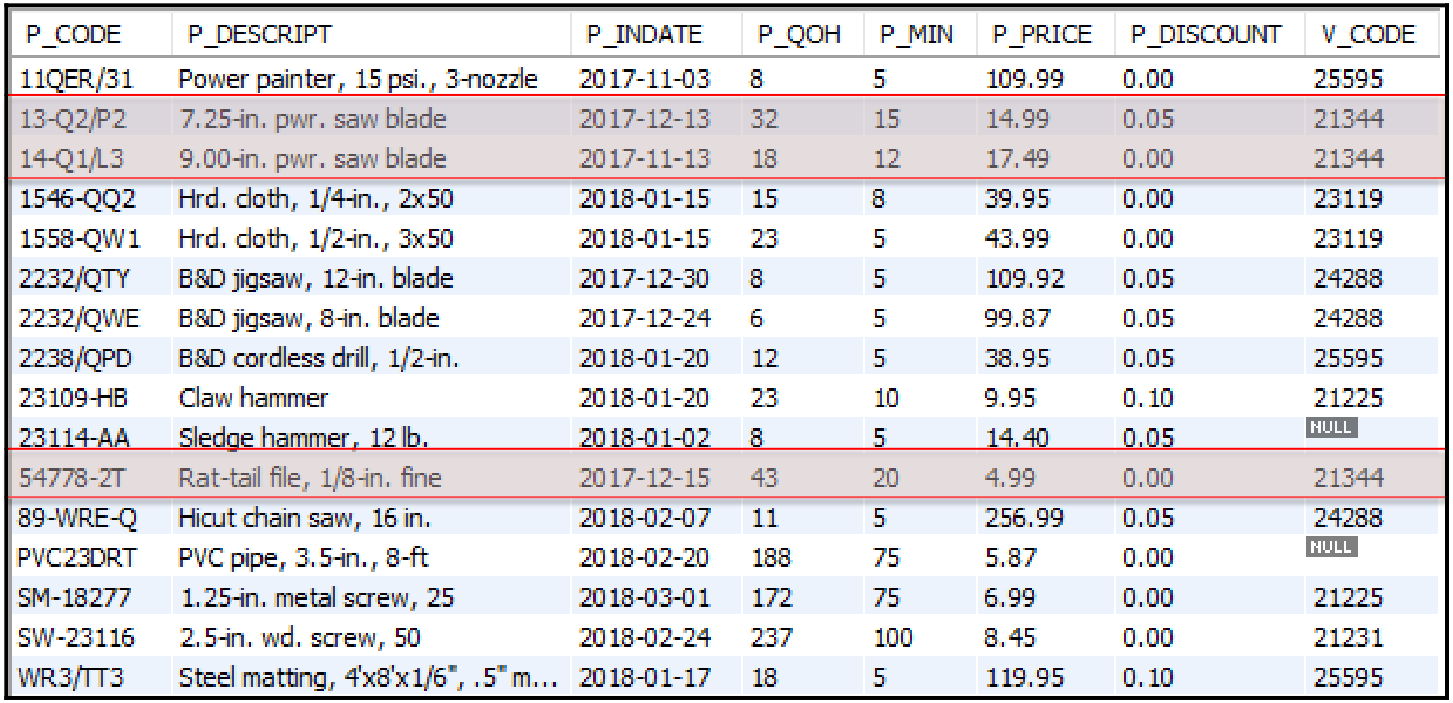
- Select rows with conditional restrictions - add conditional restrictions to the SELECT statement that limit the rows returned by the query

```
SELECT        attribute(s)
FROM          table(s)
WHERE         condition(s)
```

WHERE Clause Options

- Using comparison operators: **=**, **<**, **<=**, **>**, **>=**, **<>** or **!=**
  - May be used to place restrictions on character-based attributes
  - Using comparison operators on dates: date procedures are often more software-specific than other SQL procedures
- Logical operators: **AND**, **OR**, **NOT**
  - SQL allows you to include multiple conditions in a query through the use of these logical operators
- Special operators: **BETWEEN**, **IN**, **LIKE**, **IS NULL**, **NOT**

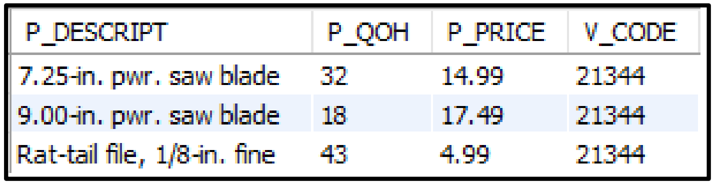
WHERE Example



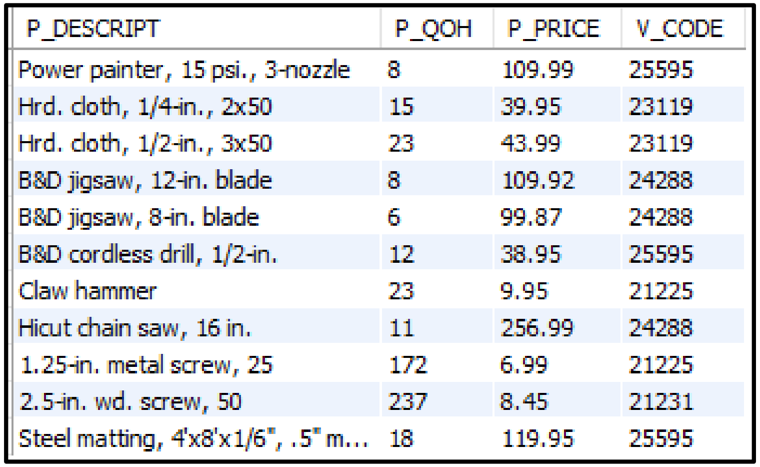
P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	2017-11-03	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	2017-12-13	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	2017-11-13	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	2018-01-15	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	2018-01-15	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	2017-12-30	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	2017-12-24	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	2018-01-20	12	5	38.95	0.05	25595
23109-HB	Claw hammer	2018-01-20	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	2018-01-02	8	5	14.40	0.05	NULL
54778-2T	Rat-tail file, 1/8-in. fine	2017-12-15	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	2018-02-07	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	2018-02-20	188	75	5.87	0.00	NULL
SM-18277	1.25-in. metal screw, 25	2018-03-01	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	2018-02-24	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	2018-01-17	18	5	119.95	0.10	25595

WHERE Example

```
SELECT
P_DESCRIPT, P_QOH,
P_PRICE, V_CODE
FROM
PRODUCT
WHERE
V_CODE = 21344;
WHERE
V_CODE != 21344;
```

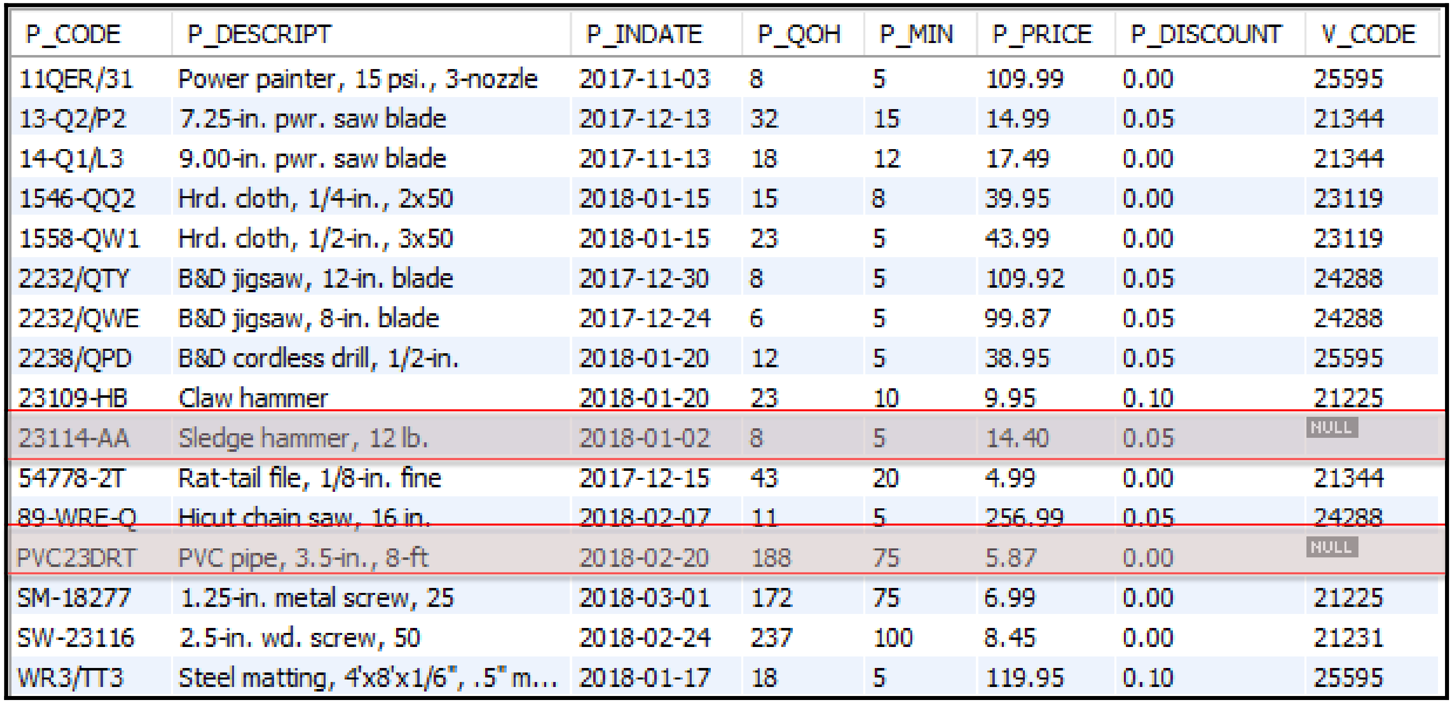


P_DESCRIPT	P_QOH	P_PRICE	V_CODE
7.25-in. pwr. saw blade	32	14.99	21344
9.00-in. pwr. saw blade	18	17.49	21344
Rat-tail file, 1/8-in. fine	43	4.99	21344



P_DESCRIPT	P_QOH	P_PRICE	V_CODE
Power painter, 15 psi., 3-nozzle	8	109.99	25595
Hrd. cloth, 1/4-in., 2x50	15	39.95	23119
Hrd. cloth, 1/2-in., 3x50	23	43.99	23119
B&D jigsaw, 12-in. blade	8	109.92	24288
B&D jigsaw, 8-in. blade	6	99.87	24288
B&D cordless drill, 1/2-in.	12	38.95	25595
Claw hammer	23	9.95	21225
Hicut chain saw, 16 in.	11	256.99	24288
1.25-in. metal screw, 25	172	6.99	21225
2.5-in. wd. screw, 50	237	8.45	21231
Steel matting, 4'x8'x1/6", .5" m...	18	119.95	25595

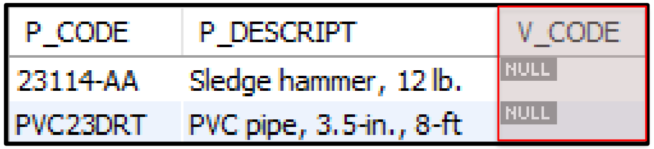
WHERE Example



P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	2017-11-03	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	2017-12-13	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	2017-11-13	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	2018-01-15	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	2018-01-15	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	2017-12-30	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	2017-12-24	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	2018-01-20	12	5	38.95	0.05	25595
23109-HB	Claw hammer	2018-01-20	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	2018-01-02	8	5	14.40	0.05	NULL
54778-2T	Rat-tail file, 1/8-in. fine	2017-12-15	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	2018-02-07	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	2018-02-20	188	75	5.87	0.00	NULL
SM-18277	1.25-in. metal screw, 25	2018-03-01	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	2018-02-24	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	2018-01-17	18	5	119.95	0.10	25595

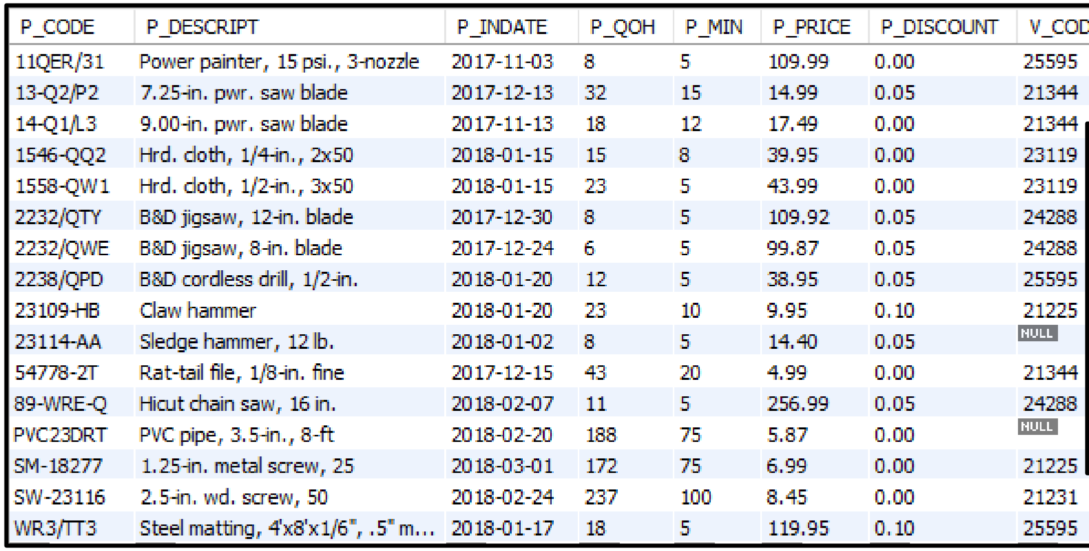
WHERE Example with NULLs

```
SELECT
P_CODE,
P_DESCRIPT,
V_CODE
FROM
PRODUCT
WHERE
     V_CODE IS NULL;
```

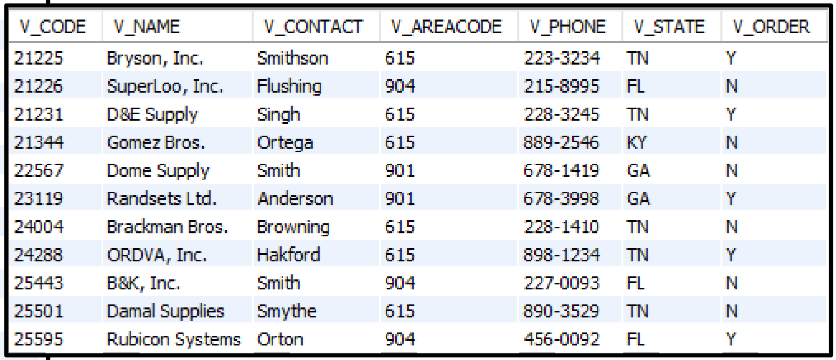


P_CODE	P_DESCRIPT	V_CODE
23114-AA	Sledge hammer, 12 lb.	NULL
PVC23DRT	PVC pipe, 3.5-in., 8-ft	NULL

PRODUCT & VENDOR Tables



P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	2017-11-03	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	2017-12-13	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	2017-11-13	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	2018-01-15	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	2018-01-15	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	2017-12-30	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	2017-12-24	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	2018-01-20	12	5	38.95	0.05	25595
23109-HB	Claw hammer	2018-01-20	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	2018-01-02	8	5	14.40	0.05	NULL
54778-2T	Rat-tail file, 1/8-in. fine	2017-12-15	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	2018-02-07	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	2018-02-20	188	75	5.87	0.00	NULL
SM-18277	1.25-in. metal screw, 25	2018-03-01	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	2018-02-24	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	2018-01-17	18	5	119.95	0.10	25595

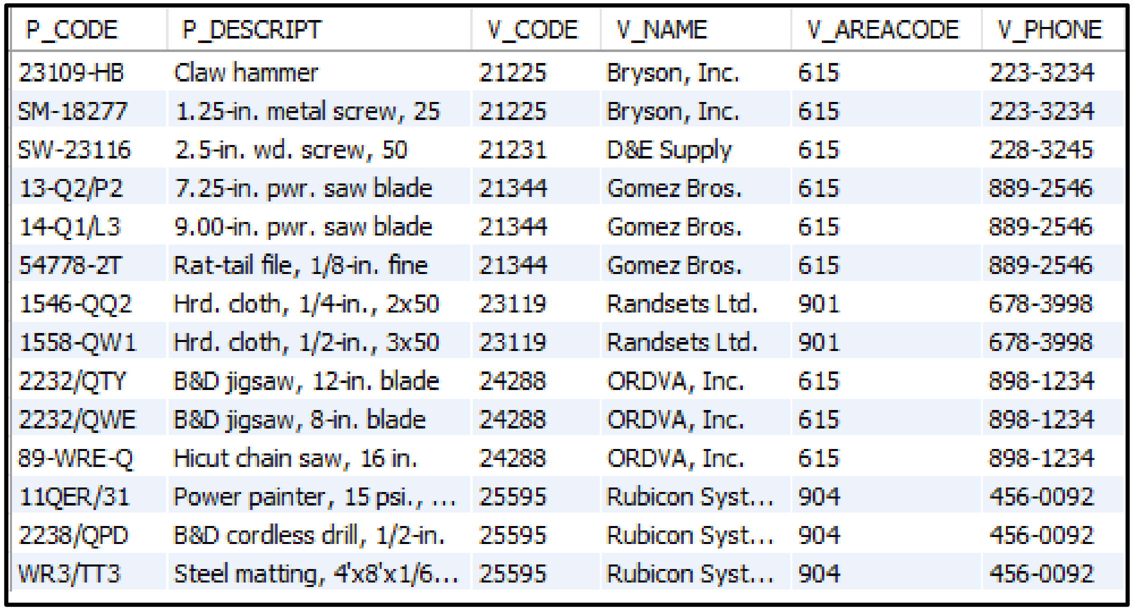


V_CODE	V_NAME	V_CONTACT	V_AREACODE	V_PHONE	V_STATE	V_ORDER
21225	Bryson, Inc.	Smithson	615	223-3234	TN	Y
21226	SuperLoo, Inc.	Flushing	904	215-8995	FL	N
21231	D&E Supply	Singh	615	228-3245	TN	Y
21344	Gomez Bros.	Ortega	615	889-2546	KY	N
22567	Dome Supply	Smith	901	678-1419	GA	N
23119	Randsets Ltd.	Anderson	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

JOIN Using WHERE Example (with aliases)

```
SELECT
P_CODE,
P_DESCRIPT,
V_CODE,
V_NAME,
V_AREACODE,
V_PHONE
```

```
FROM
PRODUCT p, VENDOR v,
WHERE
v.V_CODE = p.V_CODE;
```



P_CODE	P_DESCRIPT	V_CODE	V_NAME	V_AREACODE	V_PHONE
23109-HB	Claw hammer	21225	Bryson, Inc.	615	223-3234
SM-18277	1.25-in. metal screw, 25	21225	Bryson, Inc.	615	223-3234
SW-23116	2.5-in. wd. screw, 50	21231	D&E Supply	615	228-3245
13-Q2/P2	7.25-in. pwr. saw blade	21344	Gomez Bros.	615	889-2546
14-Q1/L3	9.00-in. pwr. saw blade	21344	Gomez Bros.	615	889-2546
54778-2T	Rat-tail file, 1/8-in. fine	21344	Gomez Bros.	615	889-2546
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	23119	Randsets Ltd.	901	678-3998
1558-QW1	Hrd. cloth, 1/2-in., 3x50	23119	Randsets Ltd.	901	678-3998
2232/QTY	B&D jigsaw, 12-in. blade	24288	ORDVA, Inc.	615	898-1234
2232/QWE	B&D jigsaw, 8-in. blade	24288	ORDVA, Inc.	615	898-1234
89-WRE-Q	Hicut chain saw, 16 in.	24288	ORDVA, Inc.	615	898-1234
11QER/31	Power painter, 15 psi., ...	25595	Rubicon Syst...	904	456-0092
2238/QPD	B&D cordless drill, 1/2-in.	25595	Rubicon Syst...	904	456-0092
WR3/TT3	Steel matting, 4'x8'x1/6...	25595	Rubicon Syst...	904	456-0092

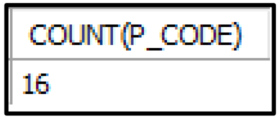
SQL Aggregates

- 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

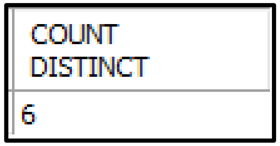
Basic SQL Aggregate Function Examples

```
SELECT
COUNT(P_CODE)
FROM
PRODUCT;
```



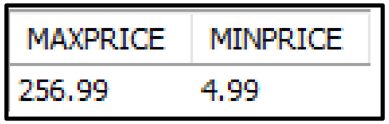
COUNT(P_CODE)
16

```
SELECT
COUNT(DISTINCT V_CODE) AS "COUNT
DISTINCT"
FROM
PRODUCT;
```



COUNT DISTINCT
6

```
SELECT
MAX(P_PRICE) AS MAXPRICE, MIN(P_PRICE)
AS MINPRICE
FROM
PRODUCT;
```



MAXPRICE	MINPRICE
256.99	4.99

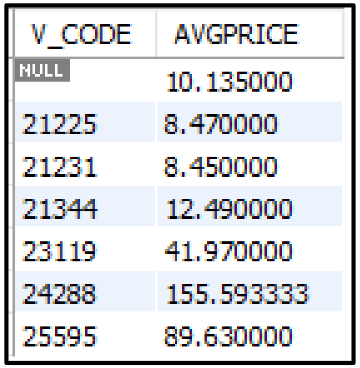
GROUP BY - Grouping Data

- Reduces a collection of rows to a single row
- Can apply aggregate functions that count, find minimum and maximum values, calculate averages: COUNT, MIN, MAX, SUM, AVG

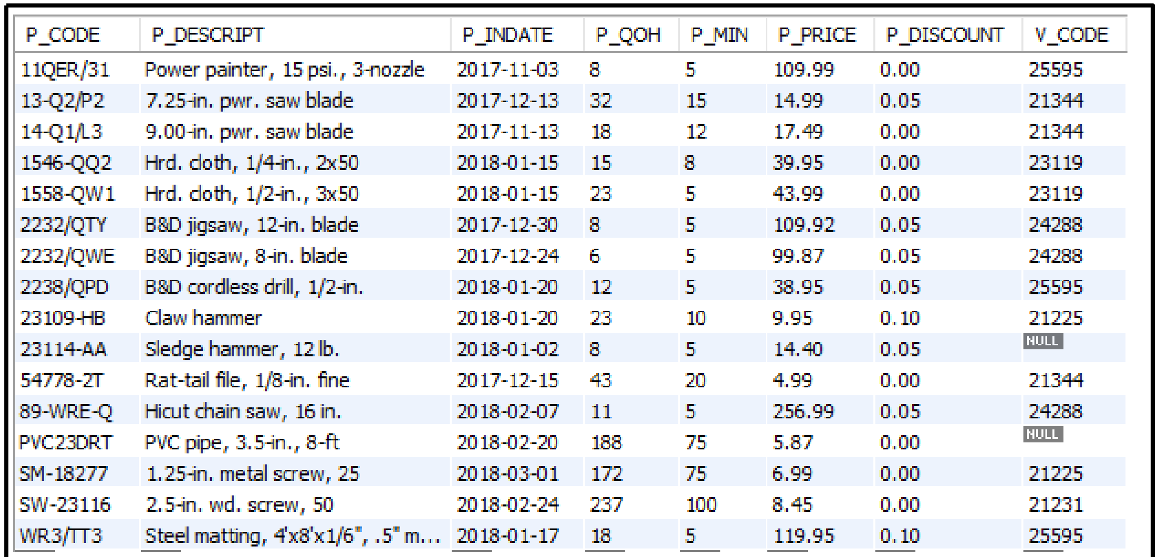
```
SELECT      attribute(s)
FROM        table(s)
[WHERE       condition(s) ]
GROUP BY    attribute(s)
[ORDER BY    attribute(s) [ASC | DESC] ] ;
```

GROUP BY Example

```
SELECT
     V_CODE,
     AVG(P_PRICE) AS AVGPRICE,
FROM
     PRODUCT

 GROUP BY  V_CODE;
```



V_CODE	AVGPRICE
NULL	10.135000
21225	8.470000
21231	8.450000
21344	12.490000
23119	41.970000
24288	155.593333
25595	89.630000



P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	2017-11-03	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	2017-12-13	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	2017-11-13	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	2018-01-15	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	2018-01-15	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	2017-12-30	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	2017-12-24	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	2018-01-20	12	5	38.95	0.05	25595
23109-HB	Claw hammer	2018-01-20	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	2018-01-02	8	5	14.40	0.05	NULL
54778-2T	Rat-tail file, 1/8-in. fine	2017-12-15	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	2018-02-07	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	2018-02-20	188	75	5.87	0.00	NULL
SM-18277	1.25-in. metal screw, 25	2018-03-01	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	2018-02-24	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	2018-01-17	18	5	119.95	0.10	25595

HAVING Clause

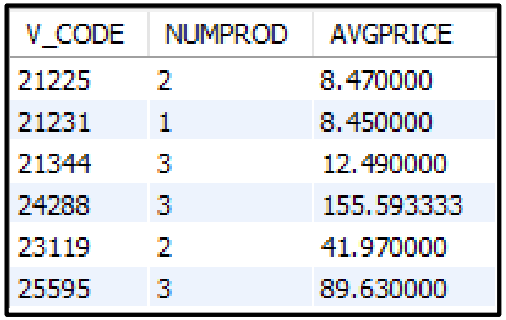
- Operates much like the WHERE clause in the SELECT statement
- HAVING clause is applied to the output of a GROUP BY operation

```
SELECT           attribute(s)

FROM             table(s)

[WHERE           condition(s)  ]

GROUP BY         attribute(s)


HAVING  condition(s)

[ORDER BY   attribute(s)[ASC | DESC] ] ;
```

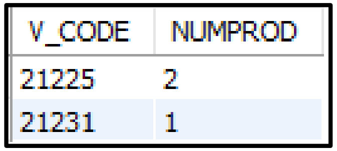
HAVING Example

- Find number of products & average price for each vendor with vendor names

```
SELECT
V_CODE,
COUNT(P_CODE) AS NUMPROD
FROM
PRODUCT

GROUP BY V_CODE
HAVING AVG(P_PRICE) < 10
ORDER BY V_CODE;
```



V_CODE	NUMPROD	AVGPRICE
21225	2	8.470000
21231	1	8.450000
21344	3	12.490000
24288	3	155.593333
23119	2	41.970000
25595	3	89.630000



V_CODE	NUMPROD
21225	2
21231	1

SQL Functions

- SQL functions are very useful tools and there are several types.
- Date and time functions
  - All date/time functions take one parameter of a date or character data type and return a value
- Numeric functions
  - Can be grouped in different ways, such as algebraic, trigonometric, and logarithmic
- String functions
  - Among the most-used functions
- Conversion functions
  - Allow you to take a value of a given data type and convert it to the equivalent value in another data type

SQL Date & Time Functions

- Many functions to help with different operations associated with Date & Time
  - CURRENT_DATE(), CURRENT_TIME(), NOW(), YEAR(), MONTH(), DAY()
  - DATE_FORMAT()
- Check out: https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html

SQL Numeric Functions

- Many functions to help with different numerical operations:
- +, -, *, /, %
  - ABS(), CEIL(), FLOOR(), TRUNCATE(), ROUND(), SIGN(), CONV()
  - DEGREES(), RADIANS()
  - POW(), DIV(), SQRT()
  - COS(), SIN(), TAN(), ACOS(), ASIN(), ATAN2(), LOG(), EXP()
- Check out: https://dev.mysql.com/doc/refman/8.0/en/numeric-functions.html

SQL String Functions

- Many functions to help with different string operations:
  - LIKE, NOT LIKE, STRCMP()
  - FORMAT(), LOWER(), UPPER(), LEFT(), RIGHT(), TRIM(), LTRIM(), RTRIM(), LENGTH()
  - CONCAT(), SUBSTRING()
- Check out: https://dev.mysql.com/doc/refman/8.0/en/string-functions.html

- https://www.w3schools.com/sql/func_sqlserver_left.asp

Relational Set Operators

- UNION
  - Combines rows from two or more queries **excluding** duplicate rows
  - Syntax:

query **UNION** *query*

- UNION ALL
  - Used to produce a relation that **retains** the duplicate rows
  - Used to unite more than just two queries

Relational Set Operators

- INTERSECT
  - Can be used to combine rows from two queries, returning only the rows that appear in both sets
  - Syntax:

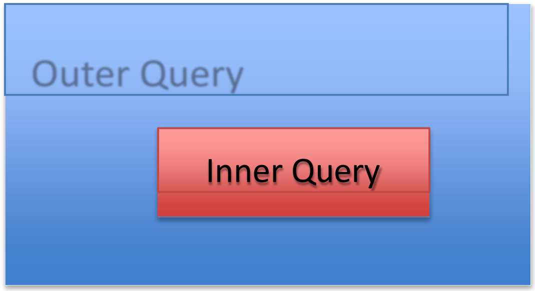
```
query         INTERSECT  query
```

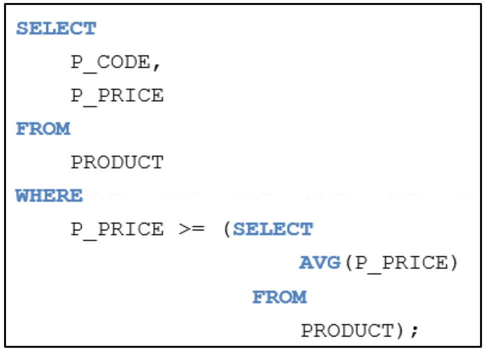
- EXCEPT or MINUS
  - Combines rows from two queries and returns only the rows that appear in the first set but not in the second
  - Syntax:

```
query   EXCEPT     query
        or
query   MINUS      query
```

Subqueries

- A subquery is a query (SELECT statement) inside another query
  - A subquery is normally expressed inside parentheses
  - 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





```
SELECT
    P_CODE,
    P_PRICE
FROM
    PRODUCT
WHERE
    P_PRICE >= (SELECT
                     AVG(P_PRICE)
                  FROM
                     PRODUCT);
```

- Subquery can return one or more values: single value, list of values etc

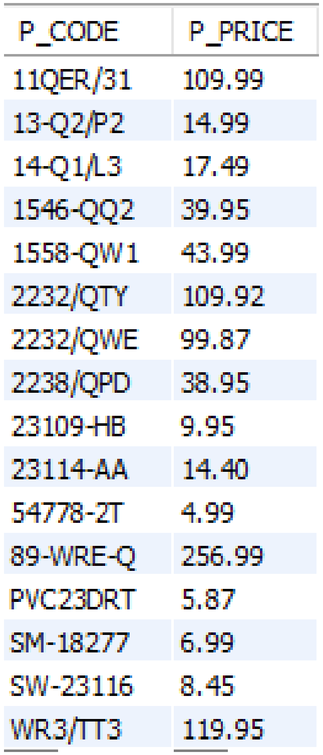
Subqueries

- WHERE subqueries
- IN subqueries
- HAVING subqueries
- FROM subqueries
- EXIST subqueries
- Attribute list subqueries
- Subqueries involving ALL and ANY operators

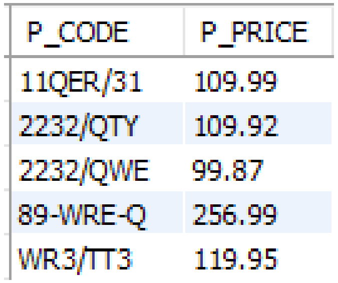
WHERE Subqueries

- Most common subquery: use an inner SELECT subquery on the right side of a WHERE comparison expression

```
SELECT
P_CODE, P_PRICE
FROM
PRODUCT

WHERE
P_PRICE >= (SELECT
AVG(P_PRICE) FROM
PRODUCT);
```



P_CODE	P_PRICE
11QER/31	109.99
13-Q2/P2	14.99
14-Q1/L3	17.49
1546-QQ2	39.95
1558-QW1	43.99
2232/QTY	109.92
2232/QWE	99.87
2238/QPD	38.95
23109-HB	9.95
23114-AA	14.40
54778-2T	4.99
89-WRE-Q	256.99
PVC23DRT	5.87
SM-18277	6.99
SW-23116	8.45
WR3/TT3	119.95

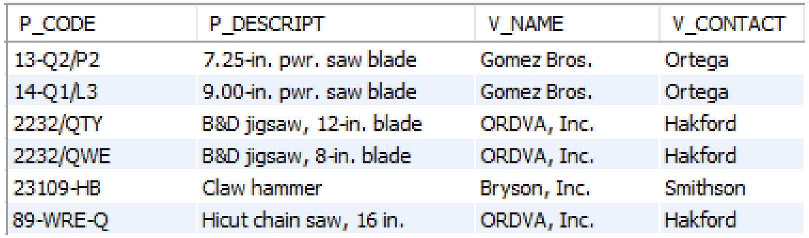


P_CODE	P_PRICE
11QER/31	109.99
2232/QTY	109.92
2232/QWE	99.87
89-WRE-Q	256.99
WR3/TT3	119.95

IN Subqueries

- Compare a single attribute to a list of values that’s not known beforehand, but can be derived using a (sub)query

```
SELECT
P_CODE,
FROM
VENDOR
JOIN
PRODUCT ON PRODUCT.V_CODE = VENDOR.V_CODE

WHERE
P_CODE IN (SELECT P_CODE FROM PRODUCT
WHERE P_DESCRIPT LIKE '%hammer%’
OR P_DESCRIPT LIKE '%saw%');
```



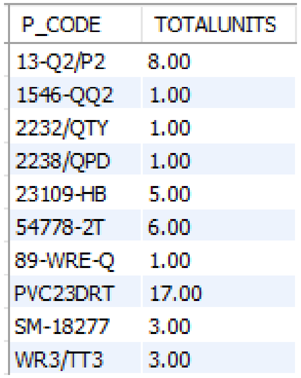
P_CODE	P_DESCRIPT	V_NAME	V_CONTACT
13-Q2/P2	7.25-in. pwr. saw blade	Gomez Bros.	Ortega
14-Q1/L3	9.00-in. pwr. saw blade	Gomez Bros.	Ortega
2232/QTY	B&D jigsaw, 12-in. blade	ORDVA, Inc.	Hakford
2232/QWE	B&D jigsaw, 8-in. blade	ORDVA, Inc.	Hakford
23109-HB	Claw hammer	Bryson, Inc.	Smithson
89-WRE-Q	Hicut chain saw, 16 in.	ORDVA, Inc.	Hakford

HAVING Subqueries

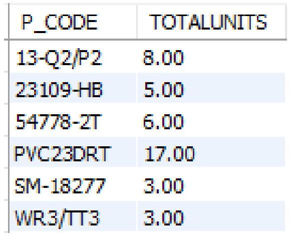
- HAVING clause: used to restrict the output of a GROUP BY query by applying conditional criteria to the grouped rows

Find all products with total units above average across all products

```
SELECT
   P_CODE,
   SUM(LINE_UNITS) AS TOTALUNITS
FROM
   LINE
GROUP BY P_CODE
HAVING SUM(LINE_UNITS) >
(SELECT
      AVG(LINE_UNITS)
   FROM
      LINE);
```



P_CODE	TOTALUNITS
13-Q2/P2	8.00
1546-QQ2	1.00
2232/QTY	1.00
2238/QPD	1.00
23109-HB	5.00
54778-2T	6.00
89-WRE-Q	1.00
PVC23DRT	17.00
SM-18277	3.00
WR3/TT3	3.00



P_CODE	TOTALUNITS
13-Q2/P2	8.00
23109-HB	5.00
54778-2T	6.00
PVC23DRT	17.00
SM-18277	3.00
WR3/TT3	3.00

FROM Subqueries

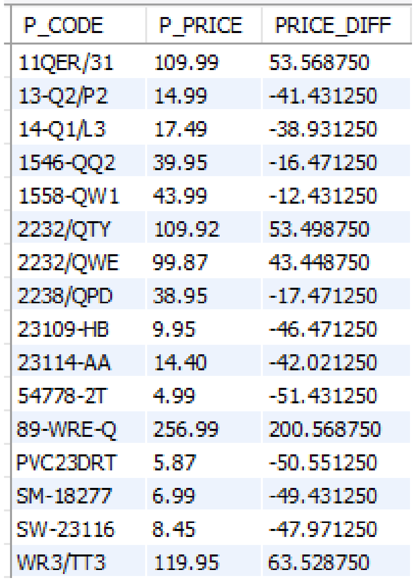
- FROM clause specifies the table(s) from which the data will be drawn. Can use a subquery in lieu of a table.

```
SELECT

 CP_INV.CUS_CODE, CP.CUS_LNAME, CP.CUS_FNAME,
 FROM
  (SELECT
      INVOICE.CUS_CODE
   FROM
   INVOICE
   JOIN LINE ON INVOICE.INV_NUMBER =
   LINE.INV_NUMBER
   WHERE P_CODE = '13-Q2/P2') CP_INV
   JOIN
   CUSTOMER CP ON CP_INV.CUS_CODE =
   CP.CUS_CODE;
```

Attribute list subqueries

- Inline subquery in attribute list: subquery expression
  - Example: list the difference between each product’s price and the average product price

```
SELECT
P_CODE, P_PRICE,

    P_PRICE - (SELECT
    AVG(P_PRICE) FROM
    PRODUCT) AS PRICE_DIFF

FROM
    PRODUCT;
```



P_CODE	P_PRICE	PRICE_DIFF
11QER/31	109.99	53.568750
13-Q2/P2	14.99	-41.431250
14-Q1/L3	17.49	-38.931250
1546-QQ2	39.95	-16.471250
1558-QW1	43.99	-12.431250
2232/QTY	109.92	53.498750
2232/QWE	99.87	43.448750
2238/QPD	38.95	-17.471250
23109-HB	9.95	-46.471250
23114-AA	14.40	-42.021250
54778-2T	4.99	-51.431250
89-WRE-Q	256.99	200.568750
PVC23DRT	5.87	-50.551250
SM-18277	6.99	-49.431250
SW-23116	8.45	-47.971250
WR3/TT3	119.95	63.528750

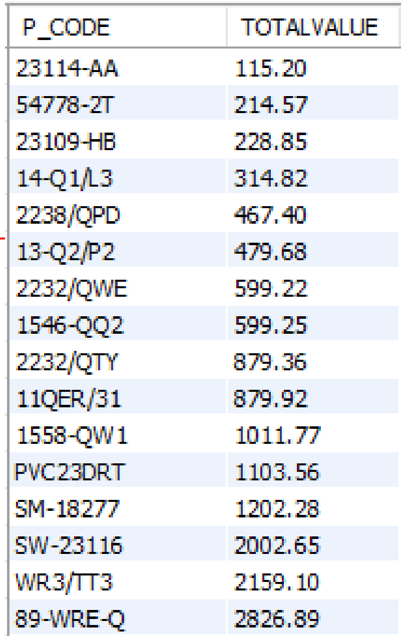
Subqueries with ANY or ALL Operators

- Multirow subquery operators: ALL and ANY
  - ALL operator compares a single value with a list of values returned by the first subquery using a comparison operator other than equals
  - ANY operator compares a single value to a list of values and select only the rows greater than or less than any value in the list. Using greater than (>) with ANY means greater than at least one value.

Subquery with ALL Operator Example

Which products cost more than all the products provided by vendors from Florida?

```
  SELECT
     P_CODE,
    P_QOH*P_PRICE AS TOTALVALUE
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	TOTALVALUE
23114-AA	115.20
54778-2T	214.57
23109-HB	228.85
14-Q1/L3	314.82
2238/QPD	467.40
13-Q2/P2	479.68
2232/QWE	599.22
1546-QQ2	599.25
2232/QTY	879.36
11QER/31	879.92
1558-QW1	1011.77
PVC23DRT	1103.56
SM-18277	1202.28
SW-23116	2002.65
WR3/TT3	2159.10
89-WRE-Q	2826.89

Summary

- SQL commands can be divided into several categories: data definition language (DDL) commands, data manipulation language (DML), transaction control language (TCL) and data control language (DCL) commands
- The SELECT statement is the main data retrieval command in SQL
- Operations that join tables can be classified as inner joins and outer joins
- The ORDER BY clause is used to sort the output of a SELECT statement
- The WHERE clause can be used with the SELECT, UPDATE, and DELETE statements to restrict the rows affected by the DDL command
- Aggregate functions (COUNT, MIN, MAX, and AVG) are special functions that perform arithmetic computations over a set of rows

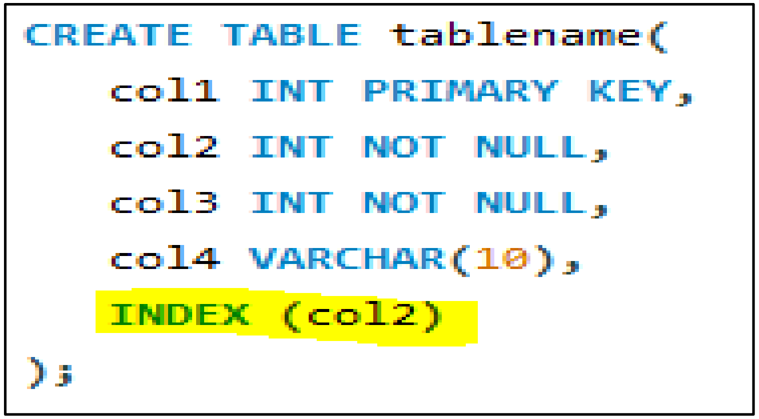
Indexes in Tables

- Indexes of a table are used to improve data retrieval performance
- DBMSs have query optimizers that use indexes to locate data quickly without having to scan every row in a table
- Indexes can be defined on a single column, or a list of columns
- Indexes can be simple or unique (2 rows can’t have the same index value)

- In MySQL, a special index **PRIMARY** is automatically created for tables with a primary key or unique key.

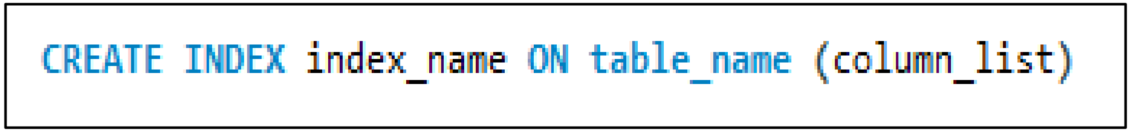
Creating Indexes

- Typically, indexes are created when the table is defined/created:



```
CREATE TABLE tablename(
    col1 INT PRIMARY KEY,
    col2 INT NOT NULL,
    col3 INT NOT NULL,
    col4 VARCHAR(10),
    INDEX (col2)
);
```

- Indexes can also be added using the **`CREATE [UNIQUE] INDEX`** command:

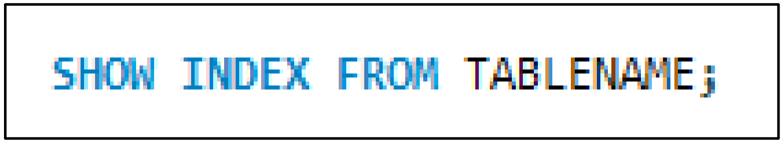


```
CREATE INDEX index_name ON table_name (column_list)
```

Altering, Listing & Dropping Indexes

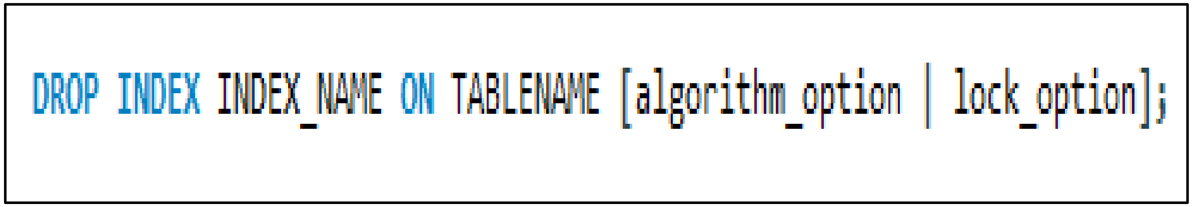
- Altering index on table: can also use the **ALTER TABLE** command to add or delete indexes on a table

- Listing index on table:



```
SHOW INDEX FROM TABLENAME;
```

- Dropping Index on table:



```
DROP INDEX INDEX_NAME ON TABLENAME [algorithm_option | lock_option];
```

Virtual Tables: Creating a View

- View:a virtual table based on a **SELECT** query
  - Base tables: tables on which the view is based

- **CREATE VIEW**: DDL command that stores the definition of a view based on a query statement in schema

```
CREATE VIEW viewname AS SELECT query;
```

Modifying & Deleting Views

- **ALTER VIEW**: update the definition of a view in schema

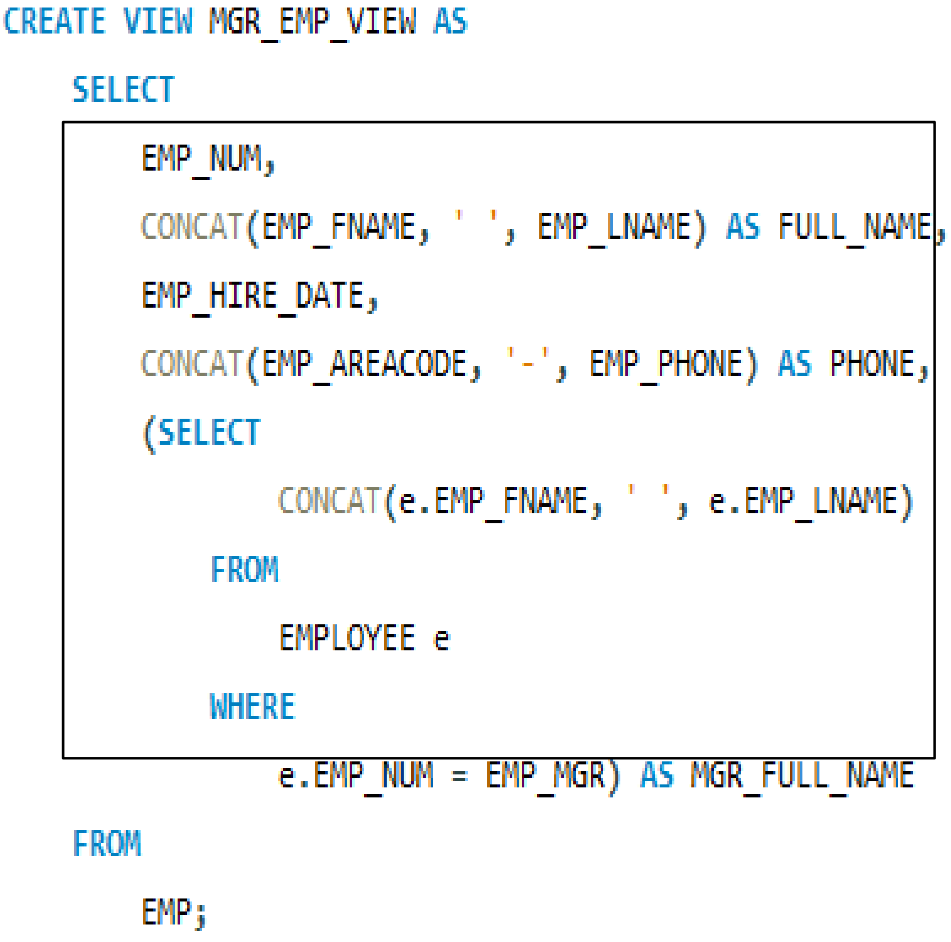
```
ALTER VIEW viewname AS SELECT query;
```

- **DROP VIEW**: delete view from schema

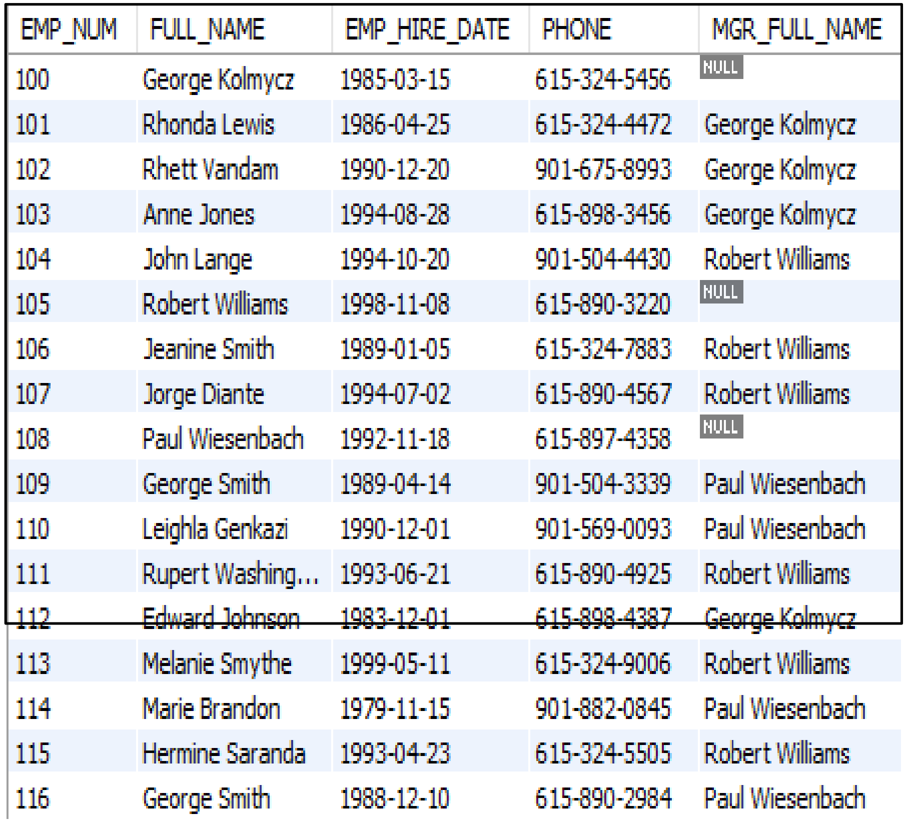
```
DROP VIEW viewname;
```

View Example #1

- Creating a new view based on a SELECT query:



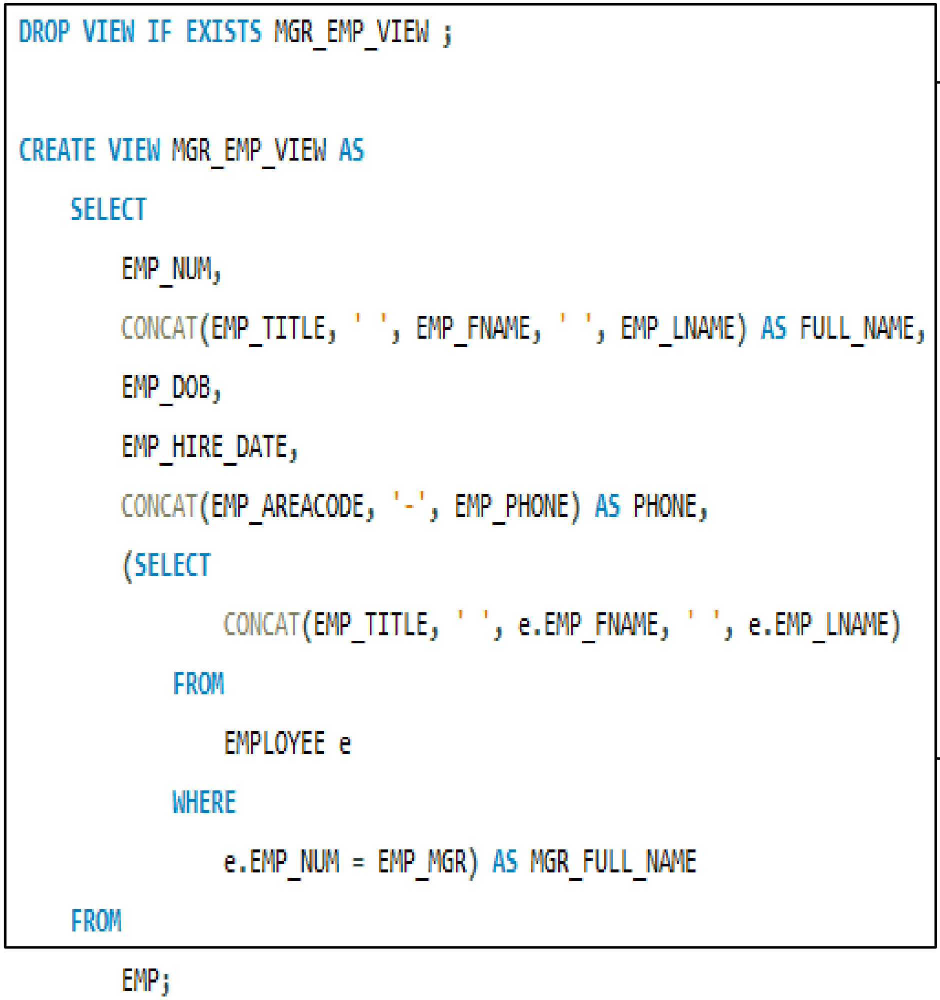
```
CREATE VIEW MGR_EMP_VIEW AS
    SELECT
        EMP_NUM,
        CONCAT(EMP_FNAME, ' ', EMP_LNAME) AS FULL_NAME,
        EMP_HIRE_DATE,
        CONCAT(EMP_AREACODE, '-', EMP_PHONE) AS PHONE,
        (SELECT
                CONCAT(e.EMP_FNAME, ' ', e.EMP_LNAME)
            FROM
                EMPLOYEE e
            WHERE
                e.EMP_NUM = EMP_MGR) AS MGR_FULL_NAME
    FROM
        EMP;
```



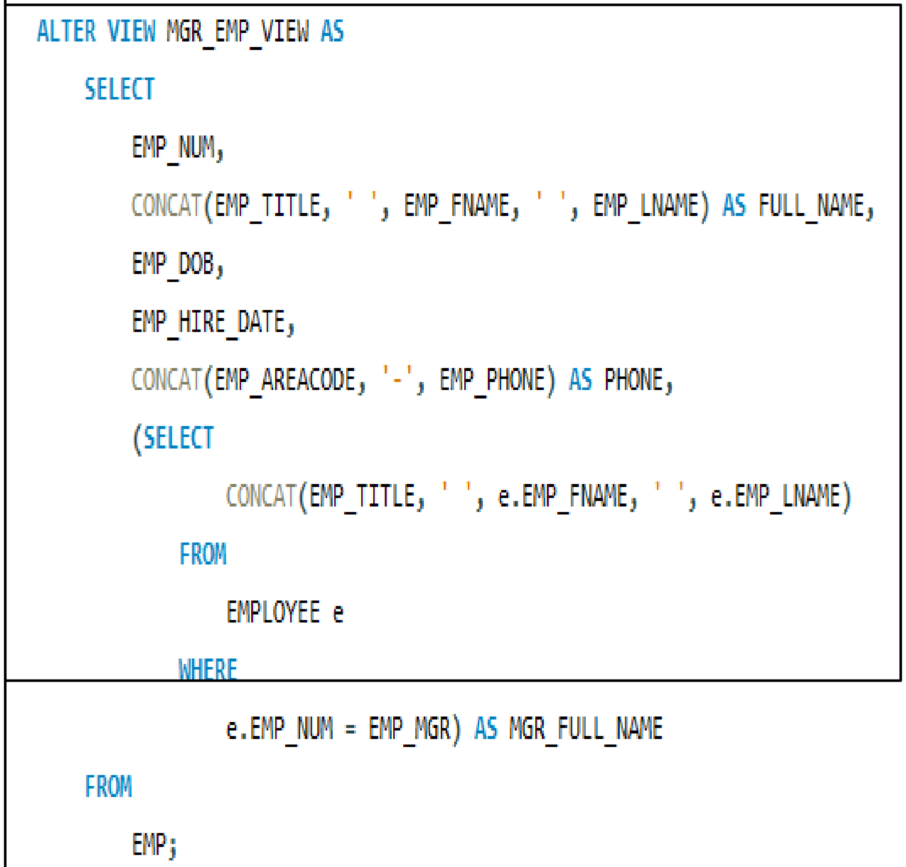
EMP_NUM	FULL_NAME	EMP_HIRE_DATE	PHONE	MGR_FULL_NAME
100	George Kolmycz	1985-03-15	615-324-5456	NULL
101	Rhonda Lewis	1986-04-25	615-324-4472	George Kolmycz
102	Rhett Vandam	1990-12-20	901-675-8993	George Kolmycz
103	Anne Jones	1994-08-28	615-898-3456	George Kolmycz
104	John Lange	1994-10-20	901-504-4430	Robert Williams
105	Robert Williams	1998-11-08	615-890-3220	NULL
106	Jeanine Smith	1989-01-05	615-324-7883	Robert Williams
107	Jorge Diante	1994-07-02	615-890-4567	Robert Williams
108	Paul Wiesenbach	1992-11-18	615-897-4358	NULL
109	George Smith	1989-04-14	901-504-3339	Paul Wiesenbach
110	Leighla Genkazi	1990-12-01	901-569-0093	Paul Wiesenbach
111	Rupert Washing...	1993-06-21	615-890-4925	Robert Williams
112	Edward Johnson	1983-12-01	615-898-4387	George Kolmycz
113	Melanie Smythe	1999-05-11	615-324-9006	Robert Williams
114	Marie Brandon	1979-11-15	901-882-0845	Paul Wiesenbach
115	Hermine Saranda	1993-04-23	615-324-5505	Robert Williams
116	George Smith	1988-12-10	615-890-2984	Paul Wiesenbach

View Example #2

- Changing a view definition:



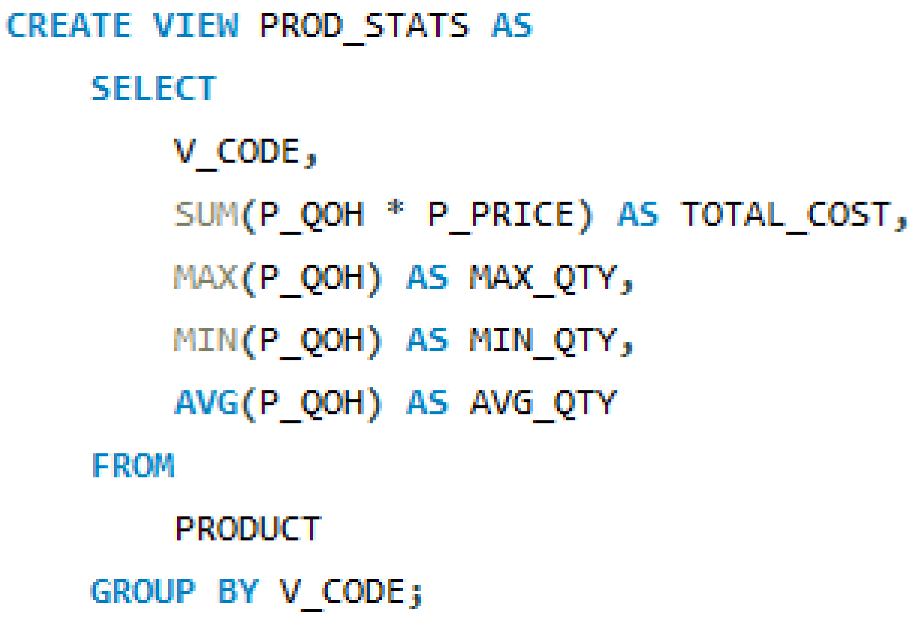
```sql
DROP VIEW IF EXISTS MGR_EMP_VIEW ;

CREATE VIEW MGR_EMP_VIEW AS
    SELECT
        EMP_NUM,
        CONCAT(EMP_TITLE, ' ', EMP_FNAME, ' ', EMP_LNAME) AS FULL_NAME,
        EMP_DOB,
        EMP_HIRE_DATE,
        CONCAT(EMP_AREACODE, '-', EMP_PHONE) AS PHONE,
        (SELECT
                CONCAT(EMP_TITLE, ' ', e.EMP_FNAME, ' ', e.EMP_LNAME)
            FROM
                EMPLOYEE e
            WHERE
                e.EMP_NUM = EMP_MGR) AS MGR_FULL_NAME
    FROM
        EMP;
```



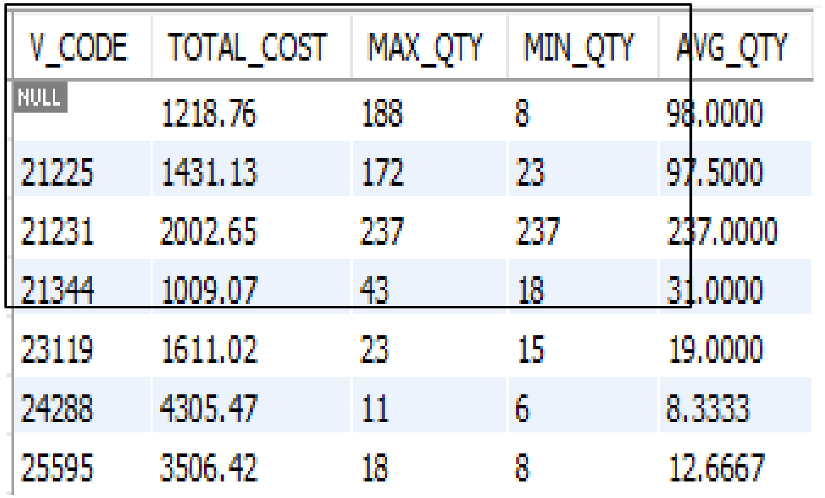
```sql
ALTER VIEW MGR_EMP_VIEW AS
    SELECT
        EMP_NUM,
        CONCAT(EMP_TITLE, ' ', EMP_FNAME, ' ', EMP_LNAME) AS FULL_NAME,
        EMP_DOB,
        EMP_HIRE_DATE,
        CONCAT(EMP_AREACODE, '-', EMP_PHONE) AS PHONE,
        (SELECT
                CONCAT(EMP_TITLE, ' ', e.EMP_FNAME, ' ', e.EMP_LNAME)
            FROM
                EMPLOYEE e
            WHERE
                e.EMP_NUM = EMP_MGR) AS MGR_FULL_NAME
    FROM
        EMP;
```

View Example #2

- Creating a new view based on a SELECT query:



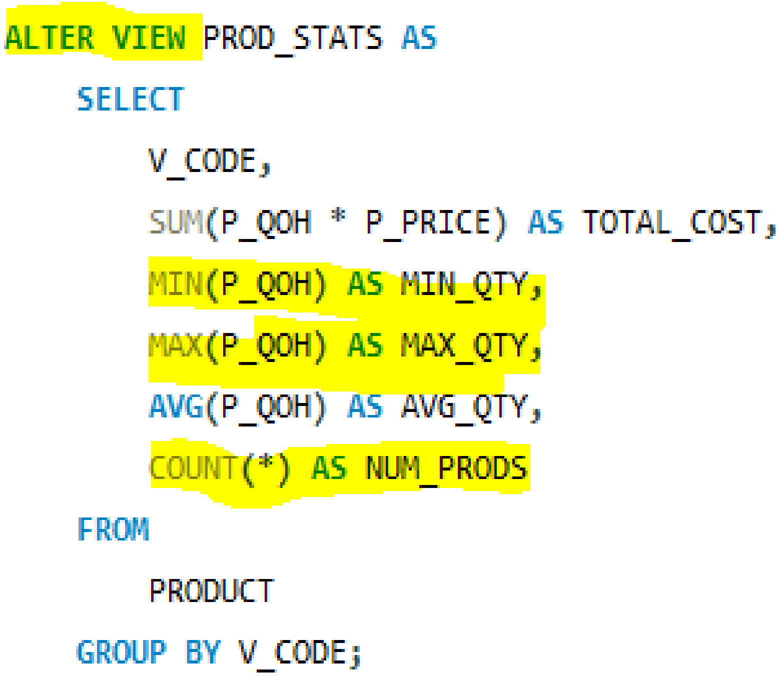
```
CREATE VIEW PROD_STATS AS
    SELECT
        V_CODE,
        SUM(P_QOH * P_PRICE) AS TOTAL_COST,
        MAX(P_QOH) AS MAX_QTY,
        MIN(P_QOH) AS MIN_QTY,
        AVG(P_QOH) AS AVG_QTY
    FROM
        PRODUCT
    GROUP BY V_CODE;
```



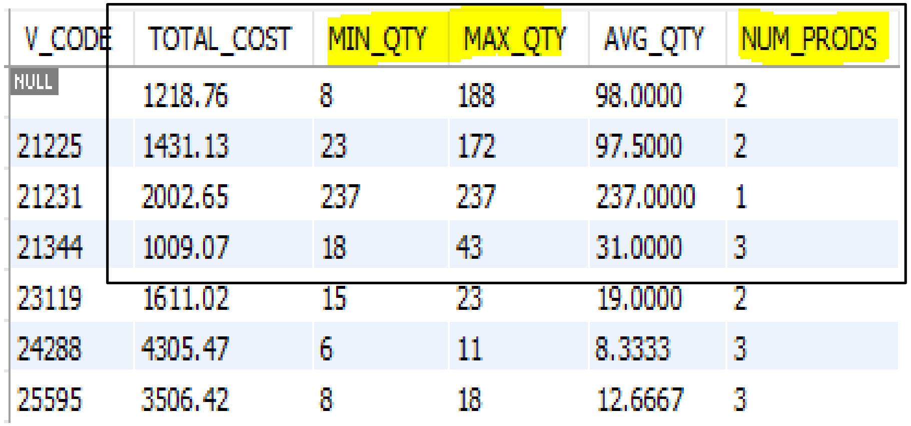
V_CODE	TOTAL_COST	MAX_QTY	MIN_QTY	AVG_QTY
NULL	1218.76	188	8	98.0000
21225	1431.13	172	23	97.5000
21231	2002.65	237	237	237.0000
21344	1009.07	43	18	31.0000
23119	1611.02	23	15	19.0000
24288	4305.47	11	6	8.3333
25595	3506.42	18	8	12.6667

View Example #2

- Changing a view definition:

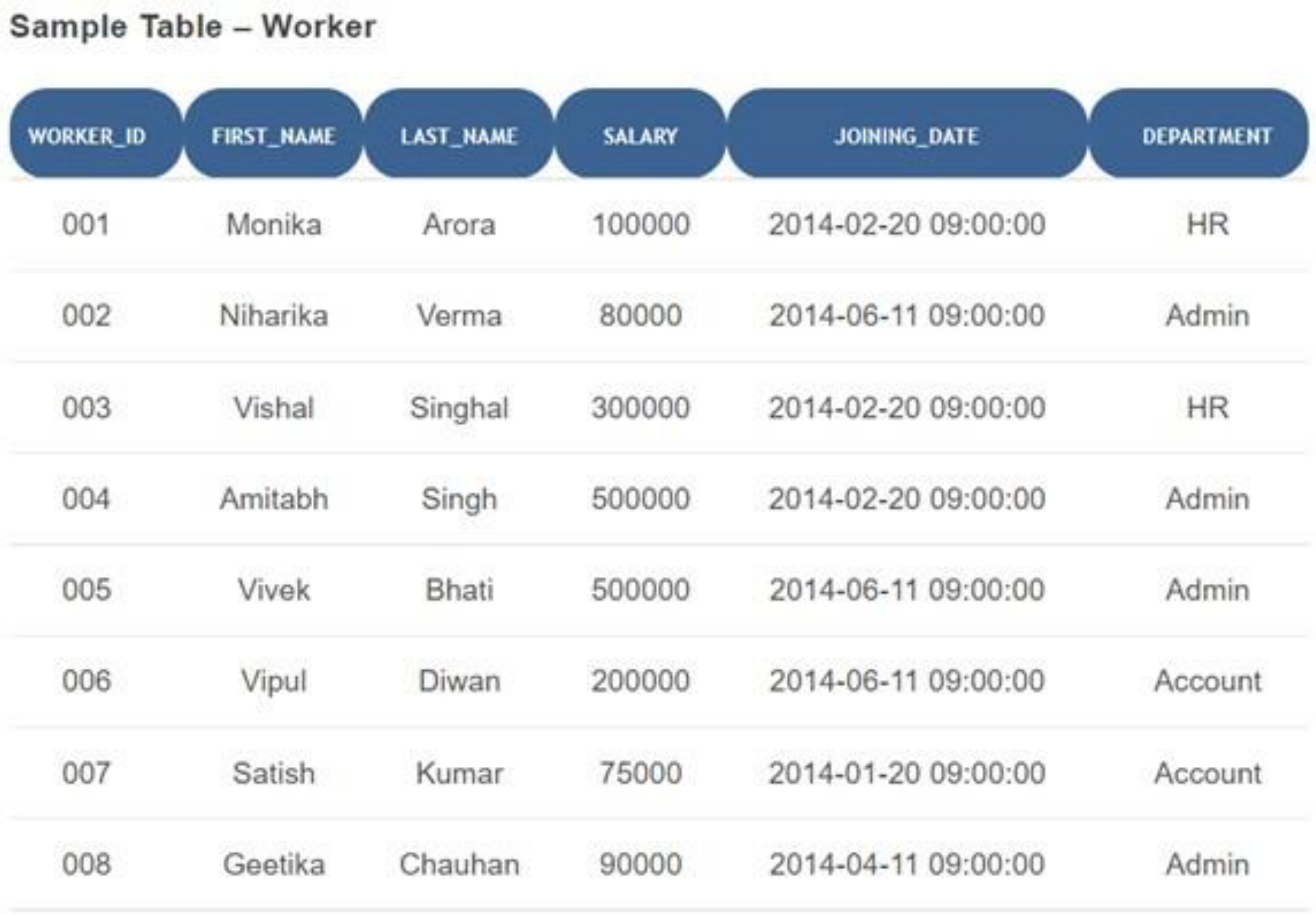


```
ALTER VIEW PROD_STATS AS
    SELECT
        V_CODE,
        SUM(P_QOH * P_PRICE) AS TOTAL_COST,
        MIN(P_QOH) AS MIN_QTY,
        MAX(P_QOH) AS MAX_QTY,
        AVG(P_QOH) AS AVG_QTY,
        COUNT(*) AS NUM_PRODS
    FROM
        PRODUCT
    GROUP BY V_CODE;
```



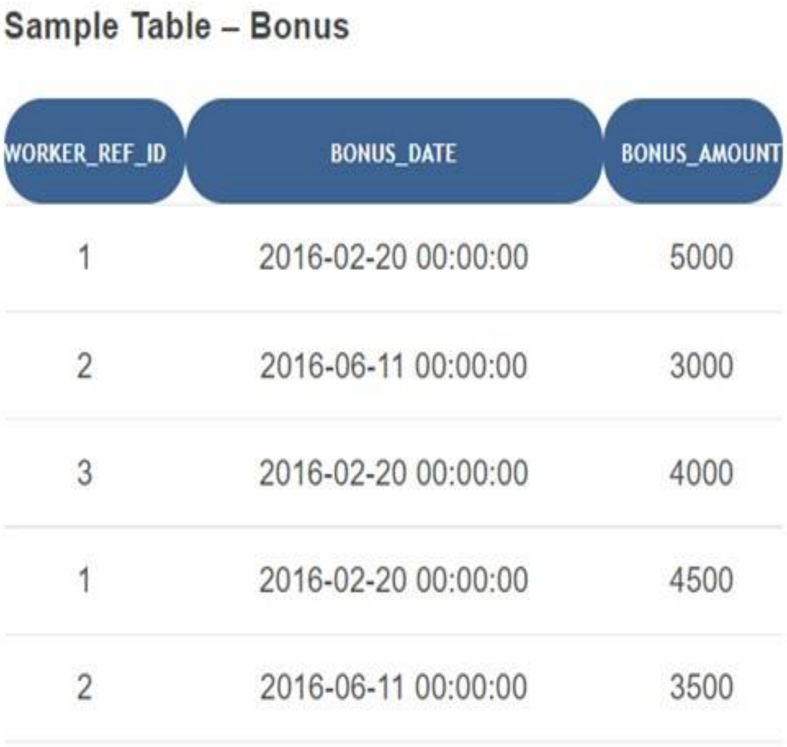
V_CODE	TOTAL_COST	MIN_QTY	MAX_QTY	AVG_QTY	NUM_PRODS
NULL	1218.76	8	188	98.0000	2
21225	1431.13	23	172	97.5000	2
21231	2002.65	237	237	237.0000	1
21344	1009.07	18	43	31.0000	3
23119	1611.02	15	23	19.0000	2
24288	4305.47	6	11	8.3333	3
25595	3506.42	8	18	12.6667	3

Example



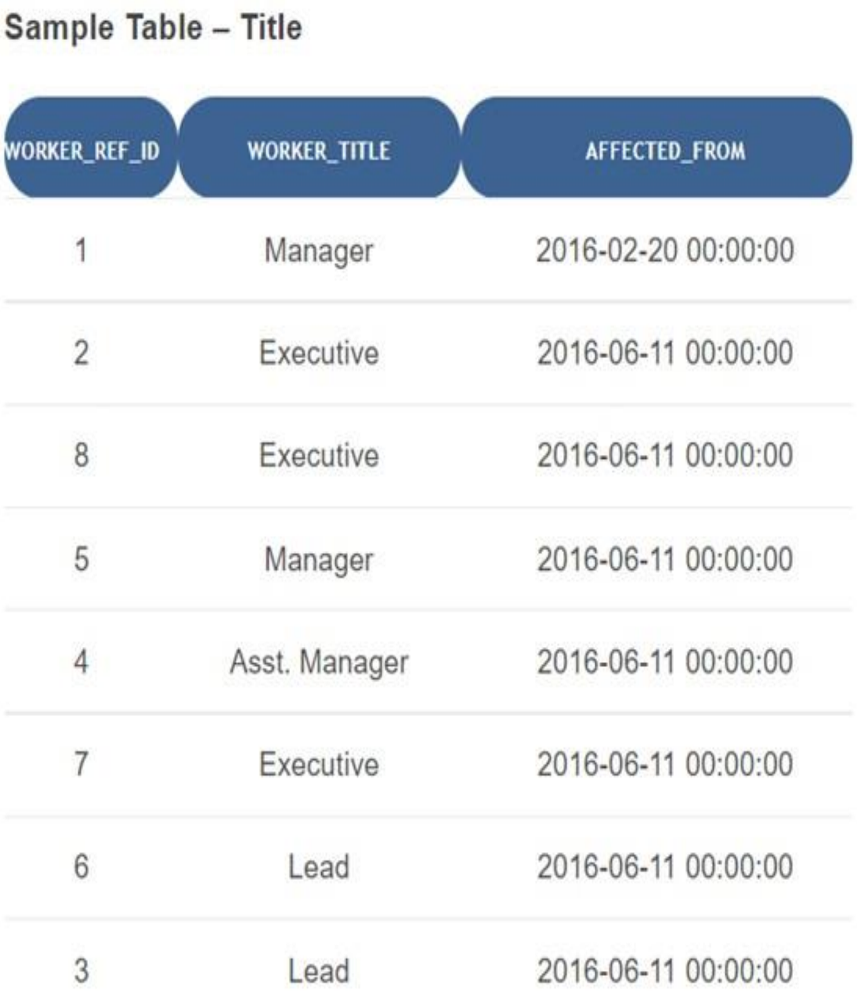
Sample Table – Worker

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
001	Monika	Arora	100000	2014-02-20 09:00:00	HR
002	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
003	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
004	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
005	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
006	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
007	Satish	Kumar	75000	2014-01-20 09:00:00	Account
008	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin



Sample Table – Bonus

WORKER_REF_ID	BONUS_DATE	BONUS_AMOUNT
1	2016-02-20 00:00:00	5000
2	2016-06-11 00:00:00	3000
3	2016-02-20 00:00:00	4000
1	2016-02-20 00:00:00	4500
2	2016-06-11 00:00:00	3500



Sample Table – Title

WORKER_REF_ID	WORKER_TITLE	AFFECTED_FROM
1	Manager	2016-02-20 00:00:00
2	Executive	2016-06-11 00:00:00
8	Executive	2016-06-11 00:00:00
5	Manager	2016-06-11 00:00:00
4	Asst. Manager	2016-06-11 00:00:00
7	Executive	2016-06-11 00:00:00
6	Lead	2016-06-11 00:00:00
3	Lead	2016-06-11 00:00:00

1. Write an SQL query to fetch “FIRST_NAME” from Worker table in upper case.

2. Write an SQL query to fetch unique values of DEPARTMENT from Worker table.

3. Write an SQL query to print the first three characters of FIRST_NAME from Worker table.

4. Write an SQL query to print details of the Workers whose FIRST_NAME contains ‘a’.

5. Write an SQL query to print details of the Workers whose FIRST_NAME ends with ‘h’ and contains six alphabets.

6. Write an SQL query to fetch the no. of workers for each department in the descending order.

7. Write an SQL query to print details of the Workers who are also Managers.

8. Write an SQL query to show only odd rows from a table.

9. Write an SQL query to show the top n (say 10) salary in records of a table.

10. Write an SQL query to determine the nth (say n=5) highest salary from a table.

11. Write an SQL query to fetch the list of employees with the same salary.

12. Write an SQL query to show the second highest salary from a table.

13. Write an SQL query to fetch the first 50% records from worker table. (assume id starts from 1 incremented by 1)

14. Write an SQL query to fetch the departments that have less than five people in it.

15. Write an SQL query to print highest salary in each department and the name of employees having the highest salary in each department.

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DN
	John	B	Smith	123456789	09-JAN-55	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	08-DEC-45	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	19-JUL-58	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	20-JUN-31	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	15-SEP-52	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	31-JUL-62	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	29-MAR-59	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	10-NOV-27	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	22-MAY-78
	Administration	4	987654321	01-JAN-85
	Headquarters	1	888665555	19-JUN-71

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	05-APR-76	DAUGHTER
	333445555	Theodore	M	25-OCT-73	SON
	333445555	Joy	F	03-MAY-48	SPOUSE
	987654321	Abner	M	29-FEB-32	SPOUSE
	123456789	Michael	M	01-JAN-78	SON
	123456789	Alice	F	31-DEC-78	DAUGHTER
	123456789	Elizabeth	F	05-MAY-57	SPOUSE

Example

- Retrieve the name and address of all employees who work for the ‘Research’ department

-
  ```
  SELECT FNAME, LANME, ADDRESS
  FROM      EMPLOYEE, DEPARTMENT
  WHERE     DNAME=‘Research’ AND DNUMBER=DNO;
  ```
- For every project located in ‘Stafford’, list project number, the controlling department number, and the department manager’s last name, address, and birthdate.

-
  ```
  SELECT PNUMBER, DNUM, LNAME, ADDRESS, BDATE
  FROM      PROJECT, DEPARTMENT, EMPLOYEE
  WHERE     DNUM=DNUMBER AND MGRSSN=SSN AND
  PLOCATION=‘Stafford’;
  ```

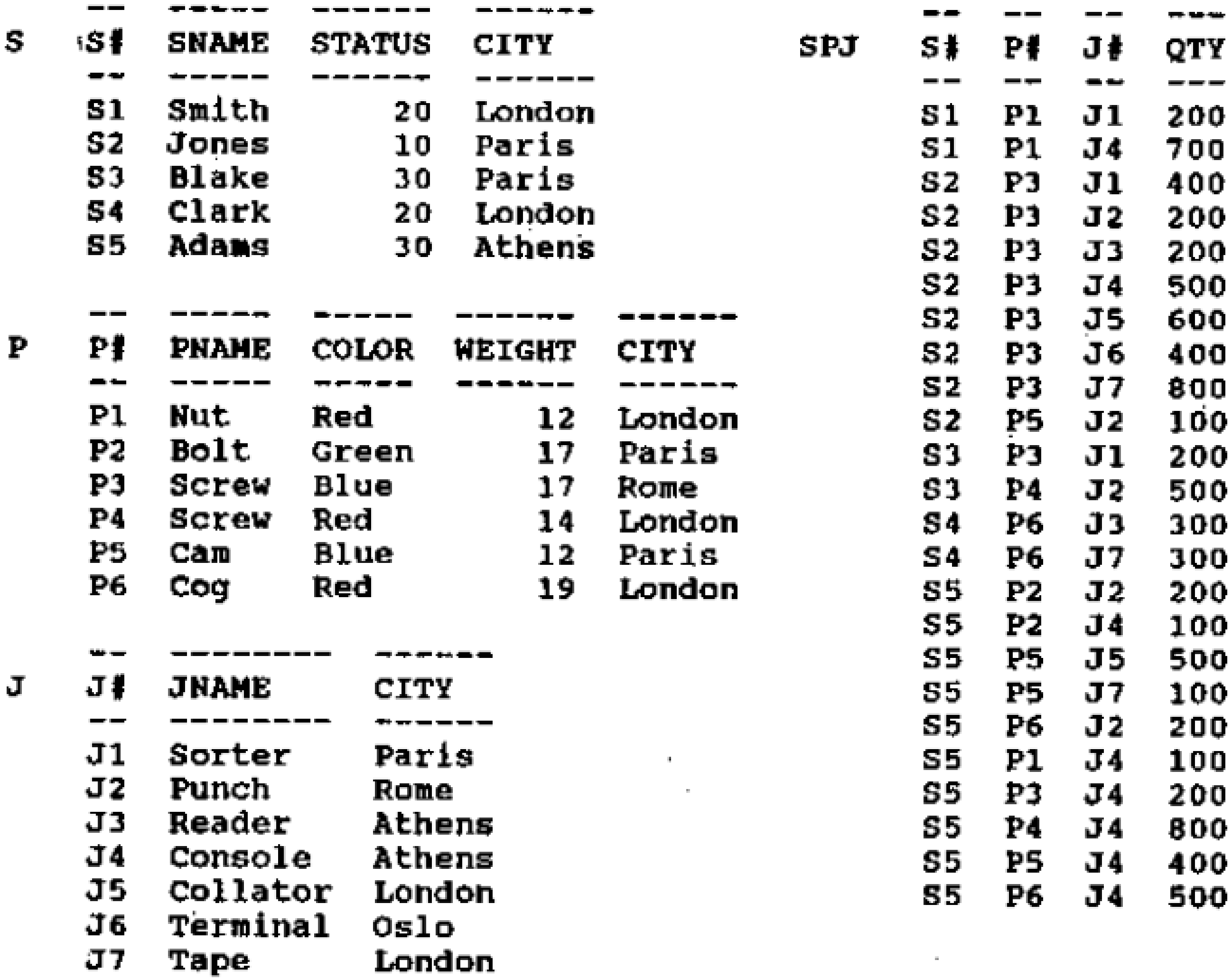
- For each employee, retrieve the employee’s first and last name and the first and last name of his or her immediate supervisor.
- SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
  FROM EMPLOYEE AS E, EMPLOYEE AS S
  WHERE E.SUPERSSN = S.SSN
- Find all employees who were born during the 1950s
- SELECT FNAME, LNAME
  FROM EMPLOYEE
  WHERE BDAT LIKE ‘__5______’;

- For each project, retrieve the project number, the project name, and the number of employees who work on that project.

-
  ```
  SELECT      PNUMBER, PNAME, COUNT(*)
  FROM        PROJECT, WORKS_ON
  WHERE       PNUMBER=PNO
  GROUP BY PNUMBER, PNAME;
  ```
- For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project

-
  ```
  SELECT      PNUMBER, PNAME, COUNT(*)
  FROM        PROJECT, WORKS_ON
  WHERE       PNUMBER=PNO
  GROUP BY PNUMBER,PNAME
  HAVING      COUNT (*) > 2;
  ```

Exercise 1



S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

J

J#	JNAME	CITY
J1	Sorter	Paris
J2	Punch	Rome
J3	Reader	Athens
J4	Console	Athens
J5	Collator	London
J6	Terminal	Oslo
J7	Tape	London

SPJ

S#	P#	J#	QTY
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

Exercise 1

1. Get names of suppliers located in Paris with status value greater than 20.
2. Get names of suppliers who supply red parts
3. Get the quantity of part P1 supplied by supplier Smith to job J1
4. Get the job name located in Paris where blue screws are supplied by suppliers located in Paris

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	09-JAN-55	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	08-DEC-45	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	19-JUL-58	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	20-JUN-31	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	15-SEP-52	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	31-JUL-62	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	29-MAR-59	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	10-NOV-27	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	22-MAY-78
	Administration	4	987654321	01-JAN-85
	Headquarters	1	888665555	19-JUN-71

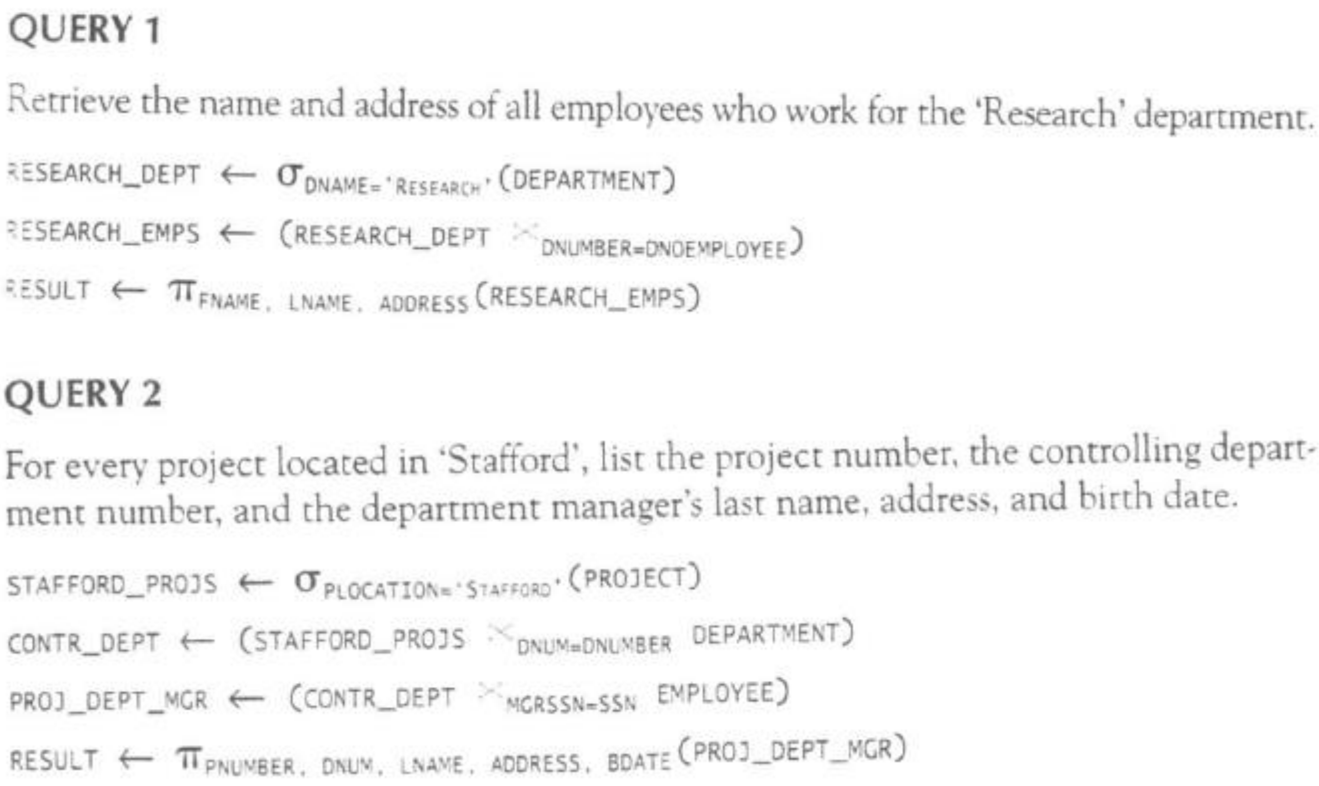
WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	05-APR-76	DAUGHTER
	333445555	Theodore	M	25-OCT-73	SON
	333445555	Joy	F	03-MAY-48	SPOUSE
	987654321	Abner	M	29-FEB-32	SPOUSE
	123456789	Michael	M	01-JAN-78	SON
	123456789	Alice	F	31-DEC-78	DAUGHTER
	123456789	Elizabeth	F	05-MAY-57	SPOUSE

Example

1. Retrieve the name and address of all employees who work for the ‘Research’ department
2. For every project located in ‘Stafford’, list the project number, the controlling department number, and department manager’s last name, address and birth date.
3. Find the names of employees who work on all the projects controlled by department number 5
4. Retrieve the names of employees who have no dependents



QUERY 1

Retrieve the name and address of all employees who work for the 'Research' department.

$\text{RESEARCH\_DEPT} \leftarrow \sigma_{\text{DNAME='Research'}}(\text{DEPARTMENT})$

$\text{RESEARCH\_EMPS} \leftarrow (\text{RESEARCH\_DEPT} \bowtie_{\text{DNUMBER=DNOEMPLOYEE}})$

$\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, ADDRESS}}(\text{RESEARCH\_EMPS})$

QUERY 2

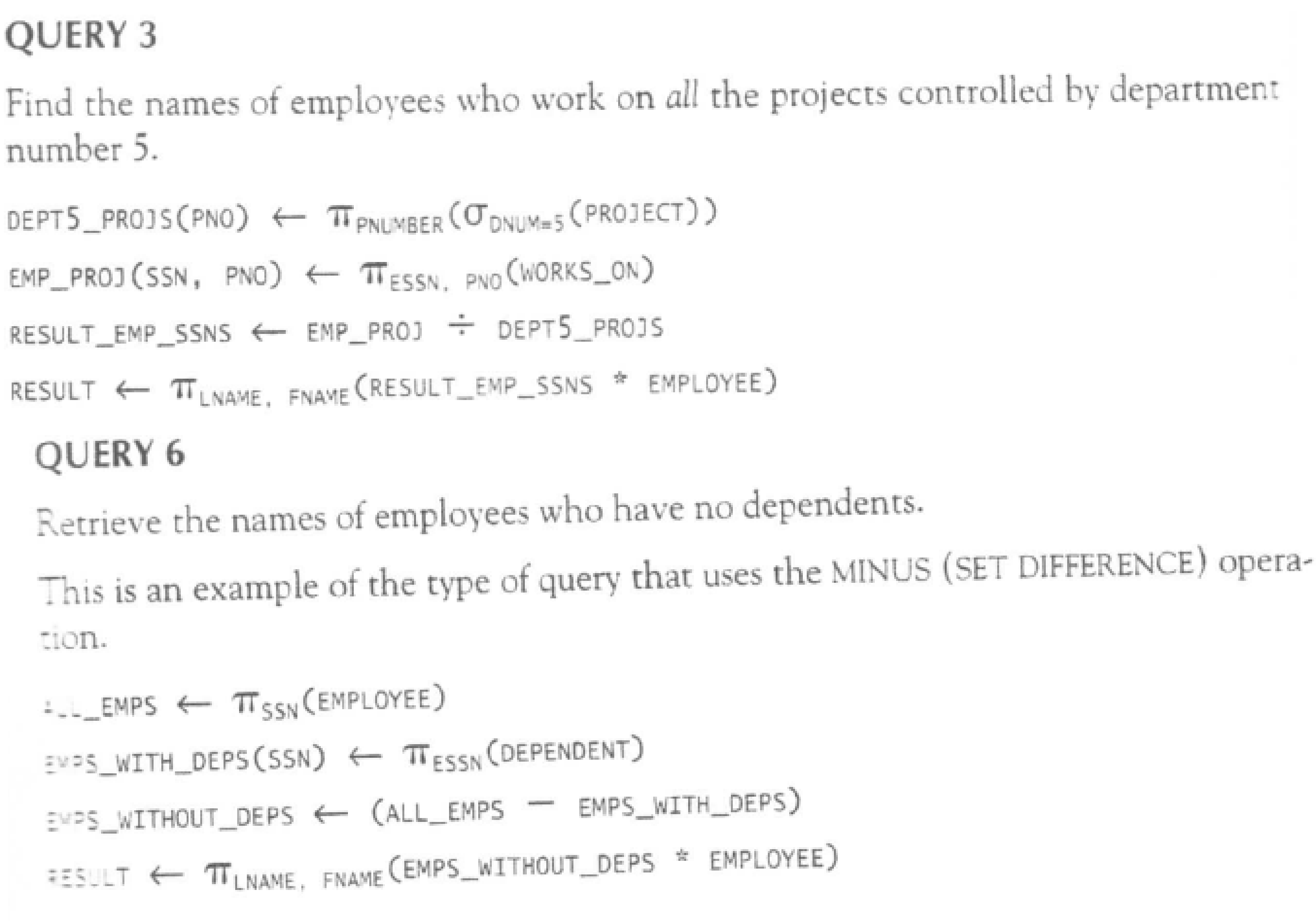
For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

$\text{STAFFORD\_PROJS} \leftarrow \sigma_{\text{PLOCATION='Stafford'}}(\text{PROJECT})$

$\text{CONTR\_DEPT} \leftarrow (\text{STAFFORD\_PROJS} \bowtie_{\text{DNUM=DNUMBER}} \text{DEPARTMENT})$

$\text{PROJ\_DEPT\_MGR} \leftarrow (\text{CONTR\_DEPT} \bowtie_{\text{MGRSSN=SSN}} \text{EMPLOYEE})$

$\text{RESULT} \leftarrow \pi_{\text{PNUMBER, DNUM, LNAME, ADDRESS, BDATE}}(\text{PROJ\_DEPT\_MGR})$



QUERY 3

Find the names of employees who work on *all* the projects controlled by department number 5.

$\text{DEPT5\_PROJS(PNO)} \leftarrow \pi_{\text{PNUMBER}}(\sigma_{\text{DNUM=5}}(\text{PROJECT}))$

$\text{EMP\_PROJ(SSN, PNO)} \leftarrow \pi_{\text{ESSN, PNO}}(\text{WORKS\_ON})$

$\text{RESULT\_EMP\_SSNS} \leftarrow \text{EMP\_PROJ} \div \text{DEPT5\_PROJS}$

$\text{RESULT} \leftarrow \pi_{\text{LNAME, FNAME}}(\text{RESULT\_EMP\_SSNS} * \text{EMPLOYEE})$

QUERY 6

Retrieve the names of employees who have no dependents.

This is an example of the type of query that uses the MINUS (SET DIFFERENCE) operation.

$\text{ALL\_EMPS} \leftarrow \pi_{\text{SSN}}(\text{EMPLOYEE})$

$\text{EMPS\_WITH\_DEPS(SSN)} \leftarrow \pi_{\text{ESSN}}(\text{DEPENDENT})$

$\text{EMPS\_WITHOUT\_DEPS} \leftarrow (\text{ALL\_EMPS} - \text{EMPS\_WITH\_DEPS})$

$\text{RESULT} \leftarrow \pi_{\text{LNAME, FNAME}}(\text{EMPS\_WITHOUT\_DEPS} * \text{EMPLOYEE})$

1. Select upper(FIRST_NAME) from Worker;
2. Select distinct DEPARTMENT from Worker;
3. Select substring(FIRST_NAME,1,3) from Worker;
4. Select * from Worker where FIRST_NAME like '%a%’;
5. Select * from Worker where FIRST_NAME like '_____h’;
6. SELECT DEPARTMENT, count(WORKER_ID) No_Of_Workers FROM worker GROUP BY DEPARTMENT ORDER BY No_Of_Workers DESC;
7. SELECT DISTINCT W.FIRST_NAME, T.WORKER_TITLE FROM Worker W JOIN Title T ON W.WORKER_ID = T.WORKER_REF_ID AND T.WORKER_TITLE in ('Manager’);
8. SELECT * FROM Worker WHERE MOD (WORKER_ID, 2) <> 0;
9. SELECT * FROM Worker ORDER BY Salary DESC LIMIT 10;
   SELECT TOP 10 * FROM Worker ORDER BY Salary DESC;

10. SELECT TOP 1 Salary FROM (SELECT DISTINCT TOP n Salary FROM Worker ORDER BY Salary DESC LIMIT 5
}
ORDER BY Salary ASC;

11, Select distinct W.WORKER_ID, W.FIRST_NAME, W.Salary from Worker W, Worker W1 where W.Salary = W1.Salary and W.WORKER_ID != W1.WORKER_ID;

12. Select max(Salary) from Worker where Salary not in (Select max(Salary) from Worker);

13. SELECT * FROM WORKER WHERE WORKER_ID <= (SELECT count(WORKER_ID)/2 from Worker);

14. SELECT DEPARTMENT, COUNT(WORKER_ID) as 'Number of Workers' FROM Worker GROUP BY DEPARTMENT HAVING COUNT(WORKER_ID) < 5;

15. SELECT t.DEPARTMENT,t.FIRST_NAME,t.Salary from(SELECT max(Salary) as TopSalary, DEPARTMENT from Worker group by DEPARTMENT) as TempNew Join Worker t on TempNew.DEPARTMENT=t.DEPARTMENT and TempNew.TopSalary=t.Salary;