```
SQL>
SQL> /*
SQL> CIS 353 - Database Design Project
SQL> Kyle Jacobson
SQL> Farid Karadsheh
SQL> Daniel Shamburger
SQL> Chesten VanPelt
SQL> */
SQL>
SQL> -- Drop the tables (in case they already exist in the system).
SQL>
SQL> DROP TABLE Warehouse CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Purchasing_Coordinator CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Store CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Orders CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Product CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Supplier CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Inventory CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Sells CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Represents CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Notes CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Sales_History CASCADE CONSTRAINTS;

Table dropped.
```

```
SQL> DROP TABLE Portion CASCADE CONSTRAINTS;

Table dropped.

SQL> DROP TABLE Manager CASCADE CONSTRAINTS;

Table dropped.

SQL>
SQL> -- Create the tables for the project.
SQL>
SQL> -- Create the warehouse table.
SQL> CREATE TABLE  Warehouse (
  2  wareNum         INTEGER,
  3  wareName        VARCHAR(25) NOT NULL,
  4  address         VARCHAR(50) NOT NULL,
  5  CONSTRAINT wIC1 PRIMARY KEY (wareNum)
  6  );

Table created.

SQL>
SQL> -- Create the purchasing coordinator table.
SQL> CREATE TABLE Purchasing_Coordinator (
  2  customerID      INTEGER PRIMARY KEY,
  3  purName         VARCHAR(25) NOT NULL,
  4  purPhoneNum     VARCHAR(25) NOT NULL
  5  );

Table created.

SQL>
SQL> -- Create the store table.
SQL> CREATE TABLE Store (
  2  storeID         INTEGER PRIMARY KEY,
  3  storeName       VARCHAR(25) NOT NULL,
  4  storeLocation     VARCHAR(60) NOT NULL
  5  );

Table created.

SQL>
SQL> -- Create the order table.
SQL> CREATE TABLE Orders (
  2  orderID         INTEGER PRIMARY KEY,
  3  wareNum         INTEGER NOT NULL,
  4  customerID      INTEGER NOT NULL,
  5  storeID         INTEGER NOT NULL,
  6  o_date          DATE NOT NULL,
  7  CONSTRAINT ordIC1 FOREIGN KEY (wareNum) REFERENCES Warehouse (wareNum),
  8  CONSTRAINT ordIC2 FOREIGN KEY (customerID) REFERENCES Purchasing_Coordinator (customerID),
  9  CONSTRAINT ordIC3 FOREIGN KEY (storeID) REFERENCES Store (storeID)
 10  );
```

```
Table created.

SQL>
SQL> -- Create the product table.
SQL> CREATE TABLE Product (
  2  productID         INTEGER PRIMARY KEY,
  3  proName           VARCHAR(50) NOT NULL,
  4  description      VARCHAR(200) NOT NULL,
  5  weight             DECIMAL NOT NULL,
  6  CONSTRAINT proIC1 CHECK (weight > 0)
  7  );

Table created.

SQL>
SQL> -- Create the supplier table.
SQL> CREATE TABLE Supplier (
  2  supplierID     INTEGER PRIMARY KEY,
  3  supName        VARCHAR(25) NOT NULL,
  4  supLocation    VARCHAR(50) NOT NULL
  5  );

Table created.

SQL>
SQL> -- Create the sells table.
SQL> CREATE TABLE Sells (
  2  supplierID     INTEGER NOT NULL,
  3  productID        INTEGER NOT NULL,
  4  primary key    (supplierID, productID),
  5  CONSTRAINT seIC1 FOREIGN KEY (supplierID) REFERENCES Supplier (supplierID) ON DELETE CASCADE,
  6  CONSTRAINT seIC2 FOREIGN KEY (productID) REFERENCES Product (productID) ON DELETE CASCADE
  7  );

Table created.

SQL> -- Create the represents table.
SQL> CREATE TABLE Represents (
  2  customerID     INTEGER NOT NULL,
  3  storeID          INTEGER NOT NULL,
  4  primary key    (storeID, customerID),
  5  CONSTRAINT repIC1 FOREIGN KEY (customerID) REFERENCES Purchasing_Coordinator (customerID) ON
DELETE CASCADE,
  6  CONSTRAINT repIC2 FOREIGN KEY (storeID) REFERENCES Store (storeID) ON DELETE CASCADE
  7  );

Table created.

SQL>
SQL> -- Create the notes table.
SQL> CREATE TABLE Notes (
  2  orderID          INTEGER,
  3  notes            VARCHAR(100),
```

```
  4  primary key    (orderID, notes),
  5  CONSTRAINT notIC1 FOREIGN KEY (orderID) REFERENCES Orders (orderID) ON DELETE CASCADE
  6  );

Table created.

SQL>
SQL> -- Create the sales history table.
SQL> CREATE TABLE Sales_History (
  2  numSold         INTEGER NOT NULL,
  3  revenue         INTEGER NOT NULL,
  4  productID       INTEGER NOT NULL,
  5  saleYear        SMALLINT NOT NULL,
  6  primary key    (productID, saleYear),
  7  CONSTRAINT s_hIC1 FOREIGN KEY(productID) REFERENCES Product (productID),
  8  CONSTRAINT s_hIC2 CHECK ((numSold > 0) AND (revenue > 0))
  9  );

Table created.

SQL>
SQL> -- Creates a relational table linking products to warehouse.
SQL> CREATE TABLE Inventory (
  2  wareNum         INTEGER NOT NULL,
  3  productID       INTEGER NOT NULL,
  4  quantity        INTEGER NOT NULL,
  5  primary key    (productID, wareNum),
  6  CONSTRAINT invIC1 FOREIGN KEY(productID) REFERENCES Product (productID) ON DELETE CASCADE,
  7  CONSTRAINT invIC2 FOREIGN KEY(wareNum) REFERENCES Warehouse(wareNum) ON DELETE CASCADE
  8  );

Table created.

SQL>
SQL> -- Creates a relational table linking products to order.
SQL> CREATE TABLE Portion (
  2  orderID         INTEGER NOT NULL,
  3  productID       INTEGER NOT NULL,
  4  pQuantity       INTEGER NOT NULL,
  5  primary key    (productID, orderID),
  6  CONSTRAINT porIC1 FOREIGN KEY(productID) REFERENCES Product (productID),
  7  CONSTRAINT porIC2 FOREIGN KEY(OrderID) REFERENCES Orders (OrderID) ON DELETE CASCADE
  8  );

Table created.

SQL>
SQL> -- Create the manager table.
SQL> CREATE TABLE Manager (
  2  managerID      INTEGER PRIMARY KEY,
  3  wareNum        INTEGER NOT NULL,
  4  manPhoneNum    VARCHAR(25) NOT NULL,
  5  manName        VARCHAR(30),
```

```
  6  CONSTRAINT manIC1 FOREIGN KEY(wareNum) REFERENCES Warehouse(wareNum) ON DELETE
CASCADE
  7  );

Table created.

SQL>
SQL> SET FEEDBACK OFF
SQL>
SQL> alter session set NLS_DATE_FORMAT = 'YYYY-MM-DD';
SQL>
SQL> -- Inserts data into the warehouse table: (wareNum, wareName, address)
SQL> INSERT INTO Warehouse VALUES (0, 'West Grand Rapids', '123 Bridge St. NW, Grand Rapids, MI 49504');
SQL> INSERT INTO Warehouse VALUES (1, 'East Grand Rapids', '4125 28th St. NE, Grand Rapids, MI 49506');
SQL> INSERT INTO Warehouse VALUES (2, 'Central Detroit', '4567 Main St. S, Detroit, MI 48206');
SQL>
SQL> -- Inserts data into the purchasing coordinator table: (customerID, purName, purPhoneNum)
SQL> INSERT INTO Purchasing_Coordinator VALUES (0, 'John Smith', '616-617-0083');
SQL> INSERT INTO Purchasing_Coordinator VALUES (1, 'Jackie Chan', '231-367-5309');
SQL> INSERT INTO Purchasing_Coordinator VALUES (2, 'Sandra Bullock', '616-375-4081');
SQL> INSERT INTO Purchasing_Coordinator VALUES (3, 'Prahdeep Singh', '231-997-4602');
SQL> INSERT INTO Purchasing_Coordinator VALUES (4, 'Emma Stone', '421-697-3371');
SQL>
SQL> -- Inserts data into the store table: (storeID, storeName, storeLocation)
SQL> INSERT INTO Store VALUES (0, 'McDonalds', '417 Michigan St. NE, Grand Rapids, MI 49503');
SQL> INSERT INTO Store VALUES (1, 'Burger King', '2155 Gratiot Ave., Detroit, MI 48207');
SQL> INSERT INTO Store VALUES (2, 'Wahlburgers', '569 Monroe St., Detroit, MI 48226');
SQL> INSERT INTO Store VALUES (3, 'Qdoba', '25 Michigan St. NE Suite 1100, Grand Rapids, MI 49503');
SQL> INSERT INTO Store VALUES (4, 'Spectrum Health', '100 Michigan St. NE, Grand Rapids, Michigan, 49503');
SQL> INSERT INTO Store VALUES (5, 'Shell Gas Station', '4661 Woodward Ave., Detroit, MI 48201');
SQL> INSERT INTO Store VALUES (6, 'McDonalds', '4235 Woodward Ave., Detroit, MI 48201');
SQL>
SQL> -- Inserts data into the order table: (orderID, wareNum, customerID, storeID, o_date)
SQL> INSERT INTO Orders VALUES (0, 0, 0, 0, '2019-07-12');
SQL> INSERT INTO Orders VALUES (1, 1, 1, 1, '2019-07-13');
SQL> INSERT INTO Orders VALUES (2, 1, 0, 1, '2019-07-14');
SQL> INSERT INTO Orders VALUES (3, 0, 1, 3, '2019-07-15');
SQL> INSERT INTO Orders VALUES (4, 2, 4, 4, '2019-07-16');
SQL> INSERT INTO Orders VALUES (5, 2, 4, 4, '2019-09-16');
SQL> INSERT INTO Orders VALUES (6, 1, 4, 3, '2019-07-16');
SQL> INSERT INTO Orders VALUES (7, 1, 4, 3, '2019-09-16');
SQL>
SQL> -- Inserts data into the Product table: (productID, proName, description, weight)
SQL> INSERT INTO Product VALUES (0, 'Ultimate Dish Soap', 'Commercial dish soap cleaner.', 1);
SQL> INSERT INTO Product VALUES (1, 'Concentrated HE Bleach', 'Commercial sanitizing solution and whitener.',
5);
SQL> INSERT INTO Product VALUES (2, 'Roma Tomatoes', 'Farm fresh red-Roma tomatoes.', 25);
SQL> INSERT INTO Product VALUES (3, '5-quart Colander', 'Stainless steel micro-perforated 5-quart colander.', 4);
SQL> INSERT INTO Product VALUES (4, '6 in. Strainer', 'A fine mesh stainless-steel strainer.', 1);
SQL> INSERT INTO Product VALUES (5, 'Double Coffee Brewer and Warmer', 'An electronic coffee brewer with hot
water intake, two glass decanters, a warmer and a brewer.', 45);
SQL> INSERT INTO Product VALUES (6, 'Disinfecting Wipes', 'Clorox Disinfecting Wipes is an all-purpose wipe that
cleans and disinfects with antibacterial power killing 99.9% of viruses and bacteria in a Crisp Lemon scent.', 3);
SQL>
```

```
SQL> -- Inserts data into the supplier table: (supplierID, supName, supLocation)
SQL> INSERT INTO Supplier VALUES (0, 'Dawn', '123 1st St., New York, NY 22222');
SQL> INSERT INTO Supplier VALUES (1, 'Walmart', '1221 Broadway,  Oakland, CA 94607');
SQL> INSERT INTO Supplier VALUES (2, 'Shelton Farms', '1832 S 11th St., Niles, MI 49120');
SQL> INSERT INTO Supplier VALUES (3, 'Bellemain', '736 28th St. SE, Grand Rapids, MI 49502');
SQL> INSERT INTO Supplier VALUES (4, 'Cuisinart', '4536 West Blvd., Atlanta, GA 12355');
SQL> INSERT INTO Supplier VALUES (5, 'BUNN', '1400 Adlai Stevenson Dr., Springfield, IL 62703');
SQL> INSERT INTO Supplier VALUES (6, 'Walmart', '5064 S Merrimac Ave., Chicago, IL 60638');
SQL>
SQL> -- Inserts data into the notes table: (orderID, note)
SQL> INSERT INTO Notes VALUES (0, 'Deliver after 3pm.');
SQL> INSERT INTO Notes VALUES (4, 'Payment is to be collected on arrival.');
SQL> INSERT INTO Notes VALUES (1, 'Pick up a return after dropping off order.');
SQL> INSERT INTO Notes VALUES (0, 'Please include an additional copy of the invoice.');
SQL>
SQL> -- Inserts data into the sells table: (supplierID, productID)
SQL> INSERT INTO Sells VALUES(0, 0);
SQL> INSERT INTO Sells VALUES(0, 1);
SQL> INSERT INTO Sells VALUES(1, 2);
SQL> INSERT INTO Sells VALUES(1, 3);
SQL> INSERT INTO Sells VALUES(1, 5);
SQL> INSERT INTO Sells VALUES(1, 6);
SQL> INSERT INTO Sells VALUES(2, 2);
SQL> INSERT INTO Sells VALUES(2, 3);
SQL> INSERT INTO Sells VALUES(2, 4);
SQL> INSERT INTO Sells VALUES(2, 6);
SQL> INSERT INTO Sells VALUES(3, 5);
SQL> INSERT INTO Sells VALUES(3, 6);
SQL> INSERT INTO Sells VALUES(4, 3);
SQL> INSERT INTO Sells VALUES(4, 4);
SQL> INSERT INTO Sells VALUES(4, 6);
SQL> INSERT INTO Sells VALUES(5, 1);
SQL> INSERT INTO Sells VALUES(5, 6);
SQL> INSERT INTO Sells VALUES(6, 3);
SQL> INSERT INTO Sells VALUES(6, 4);
SQL> INSERT INTO Sells VALUES(6, 6);
SQL> INSERT INTO Sells VALUES(6, 1);
SQL>
SQL> --Inserts data into portion (orderID, productID , pQuantity)
SQL> INSERT INTO Portion VALUES(6, 6, 2);
SQL> INSERT INTO Portion VALUES(3, 4, 3);
SQL> INSERT INTO Portion VALUES(7, 6, 5);
SQL>
SQL> -- Inserts data into the inventory table: (wareNum, productID, quantity)
SQL> INSERT INTO Inventory VALUES(0, 0, 12);
SQL> INSERT INTO Inventory VALUES(0, 1, 2);
SQL> INSERT INTO Inventory VALUES(0, 3, 7);
SQL> INSERT INTO Inventory VALUES(0, 4, 2);
SQL> INSERT INTO Inventory VALUES(1, 5, 7);
SQL> INSERT INTO Inventory VALUES(1, 3, 36);
SQL> INSERT INTO Inventory VALUES(1, 4, 8);
SQL> INSERT INTO Inventory VALUES(1, 6, 14);
SQL> INSERT INTO Inventory VALUES(2, 1, 7);
SQL> INSERT INTO Inventory VALUES(2, 3, 3);
```

```
SQL> INSERT INTO Inventory VALUES(2, 4, 4);
SQL> INSERT INTO Inventory VALUES(2, 5, 7);
SQL> INSERT INTO Inventory VALUES(1, 2, 3);
SQL>
SQL> -- Inserts data into the Sales_History table: (numSold, revenue, productID, saleYear)
SQL> INSERT INTO Sales_History VALUES(5, 100, 0, 2019);
SQL> INSERT INTO Sales_History VALUES(22, 200, 1, 2019);
SQL> INSERT INTO Sales_History VALUES(16, 160, 2, 2019);
SQL> INSERT INTO Sales_History VALUES(34, 220, 3, 2019);
SQL> INSERT INTO Sales_History VALUES(76, 332, 4, 2019);
SQL> INSERT INTO Sales_History VALUES(14, 245, 5, 2019);
SQL>
SQL> -- Inserts data into the Manager table: (managerID, wareNum, manPhoneNum, manName)
SQL> INSERT INTO Manager VALUES(0, 0, '616-234-8930', 'Matt Champion');
SQL> INSERT INTO Manager VALUES(1, 1, '616-345-1390', 'Kevin Abstract');
SQL> INSERT INTO Manager VALUES(2, 2, '586-321-5690', 'JOBA');
SQL>
SQL> -- Inserts data into the Represents table: (customerID, storeID)
SQL> INSERT INTO Represents VALUES(1, 2);
SQL> INSERT INTO Represents VALUES(2, 3);
SQL>
SQL> SET FEEDBACK ON
SQL> COMMIT;

Commit complete.

SQL>
SQL> -- Performs various queries for sorting the PROJECT database information.
SQL>
SQL> -- Query: Complete analyzation of the Warehouse table.
SQL> -- Description: Finds all of the given information that has been inserted into Warehouse table.
SQL> SELECT *
  2  FROM Warehouse;

   WARENUM WARENAME
---------- -------------------------
ADDRESS
--------------------------------------------------
         0 West Grand Rapids
123 Bridge St. NW, Grand Rapids, MI 49504

         1 East Grand Rapids
4125 28th St. NE, Grand Rapids, MI 49506

         2 Central Detroit
4567 Main St. S, Detroit, MI 48206


3 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Purchasing_Coordinator table.
SQL> -- Description: Finds all of the given information that has been inserted into Purchasing_Coordinator table.
SQL> SELECT *
```

```
  2  FROM Purchasing_Coordinator;

CUSTOMERID PURNAME                  PURPHONENUM
---------- ------------------------ ------------------------
         0 John Smith               616-617-0083
         1 Jackie Chan              231-367-5309
         2 Sandra Bullock           616-375-4081
         3 Prahdeep Singh           231-997-4602
         4 Emma Stone               421-697-3371

5 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Store table.
SQL> -- Description: Finds all of the given information that has been inserted into Store table.
SQL> SELECT *
  2  FROM Store;

   STOREID STORENAME
---------- -------------------------
STORELOCATION
-------------------------------------------------------------
         0 McDonalds
417 Michigan St. NE, Grand Rapids, MI 49503

         1 Burger King
2155 Gratiot Ave., Detroit, MI 48207

         2 Wahlburgers
569 Monroe St., Detroit, MI 48226


   STOREID STORENAME
---------- -------------------------
STORELOCATION
-------------------------------------------------------------
         3 Qdoba
25 Michigan St. NE Suite 1100, Grand Rapids, MI 49503

         4 Spectrum Health
100 Michigan St. NE, Grand Rapids, Michigan, 49503

         5 Shell Gas Station
4661 Woodward Ave., Detroit, MI 48201


   STOREID STORENAME
---------- -------------------------
STORELOCATION
-------------------------------------------------------------
         6 McDonalds
4235 Woodward Ave., Detroit, MI 48201
```

7 rows selected.

```
SQL>
SQL> -- Query: Complete analyzation of the Orders table.
SQL> -- Description: Finds all of the given information that has been inserted into Orders table.
SQL> SELECT *
  2  FROM Orders;

   ORDERID        WARENUM CUSTOMERID           STOREID O_DATE
---------- ---------- ---------- ---------- ----------
         0          0          0          0 2019-07-12
         1          1          1          1 2019-07-13
         2          1          0          1 2019-07-14
         3          0          1          3 2019-07-15
         4          2          4          4 2019-07-16
         5          2          4          4 2019-09-16
         6          1          4          3 2019-07-16
         7          1          4          3 2019-09-16

8 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Product table.
SQL> -- Description: Finds all of the given information that has been inserted into Product table.
SQL> SELECT *
  2  FROM Product;

 PRODUCTID PRONAME
---------- --------------------------------------------------
DESCRIPTION
--------------------------------------------------------------------------------
         WEIGHT
----------
         0 Ultimate Dish Soap
Commercial dish soap cleaner.
         1

         1 Concentrated HE Bleach
Commercial sanitizing solution and whitener.
         5

 PRODUCTID PRONAME
---------- --------------------------------------------------
DESCRIPTION
--------------------------------------------------------------------------------
         WEIGHT
----------

         2 Roma Tomatoes
Farm fresh red-Roma tomatoes.
         25

         3 5-quart Colander
Stainless steel micro-perforated 5-quart colander.
```

```
 PRODUCTID PRONAME
---------- --------------------------------------------------
DESCRIPTION
--------------------------------------------------------------------------------
            WEIGHT
----------
            4

            4 6 in. Strainer
A fine mesh stainless-steel strainer.
            1

            5 Double Coffee Brewer and Warmer

 PRODUCTID PRONAME
---------- --------------------------------------------------
DESCRIPTION
--------------------------------------------------------------------------------
            WEIGHT
----------
An electronic coffee brewer with hot water intake, two glass decanters, a warmer
 and a brewer.
            45

            6 Disinfecting Wipes
Clorox Disinfecting Wipes is an all-purpose wipe that cleans and disinfects with
 antibacterial power killing 99.9% of viruses and bacteria in a Crisp Lemon scen

 PRODUCTID PRONAME
---------- --------------------------------------------------
DESCRIPTION
--------------------------------------------------------------------------------
            WEIGHT
----------
t.
            3


7 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Supplier table.
SQL> -- Description: Finds all of the given information that has been inserted into Supplier table.
SQL> SELECT *
  2  FROM Supplier;

SUPPLIERID SUPNAME
---------- -------------------------
SUPLOCATION
--------------------------------------------------
            0 Dawn
123 1st St., New York, NY 22222
```

```
         1 Walmart
1221 Broadway,  Oakland, CA 94607

         2 Shelton Farms
1832 S 11th St., Niles, MI 49120


SUPPLIERID SUPNAME
---------- -------------------------
SUPLOCATION
--------------------------------------------------
         3 Bellemain
736 28th St. SE, Grand Rapids, MI 49502

         4 Cuisinart
4536 West Blvd., Atlanta, GA 12355

         5 BUNN
1400 Adlai Stevenson Dr., Springfield, IL 62703


SUPPLIERID SUPNAME
---------- -------------------------
SUPLOCATION
--------------------------------------------------
         6 Walmart
5064 S Merrimac Ave., Chicago, IL 60638


7 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Inventory table.
SQL> -- Description: Finds all of the given information that has been inserted into Inventory table.
SQL> SELECT *
  2  FROM Inventory;

   WARENUM  PRODUCTID   QUANTITY
---------- ---------- ----------
         0          0         12
         0          1          2
         0          3          7
         0          4          2
         1          5          7
         1          3         36
         1          4          8
         1          6         14
         2          1          7
         2          3          3
         2          4          4

   WARENUM  PRODUCTID   QUANTITY
---------- ---------- ----------
         2          5          7
```

```
         1          2          3

13 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Sells table.
SQL> -- Description: Finds all of the given information that has been inserted into Sells table.
SQL> SELECT *
  2  FROM Sells;

SUPPLIERID  PRODUCTID
---------- ----------
         0          0
         0          1
         1          2
         1          3
         1          5
         1          6
         2          2
         2          3
         2          4
         2          6
         3          5

SUPPLIERID  PRODUCTID
---------- ----------
         3          6
         4          3
         4          4
         4          6
         5          1
         5          6
         6          1
         6          3
         6          4
         6          6

21 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Represents table.
SQL> -- Description: Finds all of the given information that has been inserted into Represents table.
SQL> SELECT *
  2  FROM Represents;

CUSTOMERID    STOREID
---------- ----------
         1          2
         2          3

2 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Notes table.
```

```
SQL> -- Description: Finds all of the given information that has been inserted into Notes table.
SQL> SELECT *
  2  FROM Notes;

   ORDERID
----------
NOTES
--------------------------------------------------------------------------------
         0
Deliver after 3pm.

         0
Please include an additional copy of the invoice.

         1
Pick up a return after dropping off order.


   ORDERID
----------
NOTES
--------------------------------------------------------------------------------
         4
Payment is to be collected on arrival.


4 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Sales_History table.
SQL> -- Description: Finds all of the given information that has been inserted into Sales_History table.
SQL> SELECT *
  2  FROM Sales_History;

   NUMSOLD      REVENUE  PRODUCTID   SALEYEAR
---------- ---------- ---------- ----------
         5        100          0       2019
        22        200          1       2019
        16        160          2       2019
        34        220          3       2019
        76        332          4       2019
        14        245          5       2019

6 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Portion table.
SQL> -- Description: Finds all of the given information that has been inserted into Portion table.
SQL> SELECT *
  2  FROM Portion;

   ORDERID  PRODUCTID  PQUANTITY
---------- ---------- ----------
         6          6          2
```

```
         3          4          3
         7          6          5

3 rows selected.

SQL>
SQL> -- Query: Complete analyzation of the Managers table.
SQL> -- Description: Finds all of the given information that has been inserted into Managers table.
SQL> SELECT *
  2  FROM Manager;

 MANAGERID     WARENUM MANPHONENUM                    MANNAME
---------- ---------- ------------------------ ------------------------------
         0          0 616-234-8930             Matt Champion
         1          1 616-345-1390             Kevin Abstract
         2          2 586-321-5690             JOBA

3 rows selected.

SQL>
SQL> -- Query: A join involving at least four relations.
SQL> -- Description: Finds Warehouses, Stores, and Suppliers that are all on the same order, and are located in
“Grand Rapids”.
SQL> SELECT    O.orderID, W.address, S.storeLocation, Sup.supLocation
  2  FROM        Warehouse W, Store S, Supplier Sup, Order O, Product P, Portion Por, Sells Sel
  3  WHERE       O.storeID = S.StoreID AND
  4  S.storeLocation LIKE ‘%Grand Rapids%’    AND
  5  O.wareNum = W.wareNum AND
  6  W.address LIKE ‘%Grand Rapids%’ AND
  7  O.orderID = Por.orderID AND
  8  Por.productID = P.productID AND
  9  P.productID = Sel.productID AND
 10  Sel.supplierID = Sup.supplierID AND
 11  Sup.supLocation LIKE ‘%Grand Rapids%’
 12  ORDER BY O.orderID
 13
SQL> -- Query: A self-join.
SQL> -- Description: Finds pairs of unique orders that came from the same warehouse and went to the same store.
SQL> SELECT    O1.orderID, O1.wareNum, O1.storeID, O2.orderID, O2.wareNum, O2.storeID
  2  FROM        Orders O1, Orders O2
  3  WHERE       O1.wareNum = O2.wareNum AND
  4  O1.storeID = O2.storeID AND
  5  O1.orderID > O2.orderID
  6  ORDER BY            O1.orderID;

  ORDERID     WARENUM     STOREID     ORDERID     WARENUM     STOREID
---------- ---------- ---------- ---------- ---------- ----------
         2          1          1          1          1          1
         5          2          4          4          2          4
         7          1          3          6          1          3

3 rows selected.

SQL>
```

```
SQL> -- Query: UNION, INTERSECT, and/or MINUS.
SQL> -- Description: Finds products that there are more than 4 in any warehouse, and whose weight is greater than
10
SQL> SELECT    P.productID, P.weight, I.quantity
  2  FROM          Product P, Inventory I
  3  WHERE     I.quantity > 4 AND
  4                        I.productID = P.productID
  5  INTERSECT
  6  SELECT        P.productID, P.weight, I.quantity
  7  FROM          Product P, Inventory I
  8  WHERE     I.productID = P.productID AND
  9                        P.weight  > 10;

 PRODUCTID      WEIGHT   QUANTITY
---------- ---------- ----------
         5         45          7

1 row selected.

SQL>
SQL> -- Query: SUM, AVG, MAX and/or MIN.
SQL> -- Description: Finds average weight of products for each warehouse.
SQL> SELECT    W.wareNum, AVG(P.weight)
  2  FROM          Warehouse W, Product P, Inventory I
  3  WHERE         I.wareNum = W.wareNum AND
  4  I.productID = P.productID
  5  GROUP BY              W.wareNum
  6  HAVING        AVG(P.weight) > 0
  7  ORDER BY              W.wareNum;

   WARENUM AVG(P.WEIGHT)
---------- --------------
         0           2.75
         1           15.6
         2          13.75

3 rows selected.

SQL>
SQL> -- Query: GROUP BY, HAVING, and ORDER BY, all appearing in the same query.
SQL> -- Description: Finds suppliers that supply more than 2 products.
SQL> SELECT    DISTINCT S.supplierID, S.supName, COUNT(DISTINCT P.productID)
  2  FROM          Supplier S, Product P, Sells Sel
  3  WHERE         P.productID = Sel.productID AND
  4                   Sel.supplierID = S.supplierID
  5  GROUP BY              S.supplierID, S.supName
  6  HAVING        COUNT(*) > 2
  7  ORDER BY              S.supplierID;

SUPPLIERID SUPNAME                COUNT(DISTINCTP.PRODUCTID)
---------- ------------------------- --------------------------
         1 Walmart                                4
         2 Shelton Farms                                  4
         4 Cuisinart                              3
```

```
         6 Walmart                          4

4 rows selected.

SQL>
SQL> -- Query: A correlated subquery.
SQL> -- Description: Finds the heaviest item and shows what warehouse it's in.
SQL> SELECT    I1.wareNum, I1.productID, P1.weight
  2  FROM          Product P1, Inventory I1
  3  WHERE     I1.productID = P1.productID AND
  4  P1.weight =
  5                          (SELECT    MAX(P2.weight)
  6                          FROM     Product P2, Inventory I2
  7  WHERE       I1.wareNum = I2.wareNum)
  8  ORDER BY I1.wareNum;

   WARENUM  PRODUCTID           WEIGHT
---------- ---------- ----------
         1          5          45
         2          5          45

2 rows selected.

SQL>
SQL> -- Query: A non-correlated subquery.
SQL> -- Description: Finds the name of every product that has less than 15 number of sold units.
SQL> SELECT    P.productID, P.proName
  2  FROM          Product P
  3  WHERE     P.productID NOT IN
  4                          (SELECT SH.productID
  5  FROM Sales_History SH
  6  WHERE SH.numSold > 15 )
  7  ORDER BY P.productID;

 PRODUCTID PRONAME
---------- --------------------------------------------------
         0 Ultimate Dish Soap
         5 Double Coffee Brewer and Warmer
         6 Disinfecting Wipes

3 rows selected.

SQL>
SQL> -- Query: A relational DIVISION query.
SQL> -- Description: Finds the name and product ID of every product that is supplied by Walmart.
SQL> SELECT    P.proName, P.productID
  2  FROM          Product P
  3  WHERE     NOT EXISTS((SELECT S.supName
  4                          FROM     Supplier S
  5                          WHERE            S.supName = 'Walmart')
  6                          MINUS
  7                          (SELECT    S.supName
  8                          FROM    Supplier S, Sells Sel
  9  WHERE       P.productID = Sel.productID AND
```

```
 10                 Sel.supplierID = S.supplierID AND
 11  S.supName = 'Walmart'))
 12  ORDER BY              P.proName;

PRONAME                                            PRODUCTID
-------------------------------------------------- ----------
5-quart Colander                                   3
6 in. Strainer                                     4
Concentrated HE Bleach                                     1
Disinfecting Wipes                                 6
Double Coffee Brewer and Warmer                            5
Roma Tomatoes                                      2

6 rows selected.

SQL>
SQL> -- Query: An outer join query.
SQL> -- Description: Finds the managerID, manName, manPhoneNum for every manager. Also shows the address of
the warehouse they manage.
SQL> SELECT    M.managerID, M.manName, M.manPhoneNum, W.wareNum
  2  FROM          Manager M LEFT OUTER JOIN Warehouse W ON M.wareNum = W.wareNum;

 MANAGERID MANNAME                        MANPHONENUM            WARENUM
---------- ------------------------------ ------------------------ ----------
         0 Matt Champion                  616-234-8930                0
         1 Kevin Abstract                 616-345-1390                1
         2 JOBA                           586-321-5690                2

3 rows selected.

SQL>
SQL> -- Insert DELETE/UPDATE statements that violate integrity constraints for testing our queries.
SQL>
SQL> INSERT INTO Inventory VALUES('a', 20, 7);
INSERT INTO Inventory VALUES('a', 20, 7)
                             *
ERROR at line 1:
ORA-01722: invalid number


SQL> INSERT INTO Inventory VALUES(1, 20, 7);
INSERT INTO Inventory VALUES(1, 20, 7)
*
ERROR at line 1:
ORA-02291: integrity constraint (JACOKYLE.INVIC1) violated - parent key not
found


SQL> INSERT INTO Product VALUES (5, 'BUNN', 'Double Coffee Brewer and Warmer', 'An electronic coffee brewer
with hot water intake, two glass decanters, a warmer and a brewer.', -2);
INSERT INTO Product VALUES (5, 'BUNN', 'Double Coffee Brewer and Warmer', 'An electronic coffee brewer with
hot water intake, two glass decanters, a warmer and a brewer.', -2)
            *
ERROR at line 1:
```

ORA-00913: too many values

```
SQL> INSERT INTO Sales_History VALUES(-1, -24, 5, 2018);
INSERT INTO Sales_History VALUES(-1, -24, 5, 2018)
*
ERROR at line 1:
ORA-02290: check constraint (JACOKYLE.S_HIC2) violated


SQL>
SQL> COMMIT;

Commit complete.

SQL> SPOOL OFF
```