1. **Write the SQL code that will create the table structure for a table named EMP\_1. This table is a subset of the EMPLOYEE table. The basic EMP\_1 table structure is summarized in the table below. (Note that the JOB\_CODE is the FK to JOB.)**

|  |  |
| --- | --- |
| **ATTRIBUTE (FIELD) NAME** | **DATA DECLARATION** |
| EMP\_NUM | CHAR(3) |
| EMP\_LNAME | VARCHAR(15) |
| EMP\_FNAME | VARCHAR(15) |
| EMP\_INITIAL | CHAR(1) |
| EMP\_HIREDATE | DATE |
| JOB\_CODE | CHAR(3) |

CREATE TABLE EMP\_1 (

EMP\_NUM CHAR(3) PRIMARY KEY,

EMP\_LNAME VARCHAR(15) NOT NULL,

EMP\_FNAME VARCHAR(15) NOT NULL,

EMP\_INITIAL CHAR(1),

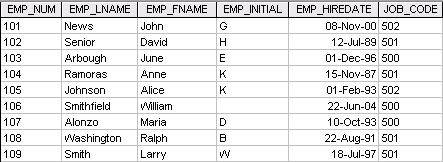
EMP\_HIREDATE DATE,

JOB\_CODE CHAR(3),

FOREIGN KEY (JOB\_CODE) REFERENCES JOB);

1. **Having created the table structure in Problem 1, write the SQL code to enter the first two rows for the table shown in Figure P7.2.**

Figure P7.2 The contents of the EMP\_1 table



INSERT INTO EMP\_1 VALUES (‘101’, ‘News’, ‘John’, ‘G’, ’08-Nov-00’, ‘502’);

INSERT INTO EMP\_1 VALUES (‘102’, ‘Senior’, ‘David’, ‘H’, ’12-Jul-89’, ‘501’);

1. **Write the SQL code to change the job code to 501 for the person whose employee number (EMP\_NUM) is 107. After you have completed the task, examine the results, and then reset the job code to its original value.**

UPDATE EMP\_1

SET JOB\_CODE = ‘501’

WHERE EMP\_NUM = ‘107’;

1. **Write the SQL code to delete the row for the person named William Smithfield, who was hired on June 22, 2004, and whose job code classification is 500. (Hint: Use logical operators to include all of the information given in this problem.)**

DELETE FROM EMP\_1

WHERE EMP\_LNAME = 'Smithfield'

AND EMP\_FNAME = 'William'

AND EMP\_HIREDATE = '22-June-04'

AND JOB\_CODE = '500';

1. **Write the SQL code to create a copy of EMP\_1, naming the copy EMP\_2. Then write the SQL code that will add the attributes EMP\_PCT and PROJ\_NUM to its structure. The EMP\_PCT is the bonus percentage to be paid to each employee. The new attribute characteristics are:**

**EMP\_PCTNUMBER(4,2)**

**PROJ\_NUMCHAR(3)**

**(Note: If your SQL implementation allows it, you may use DECIMAL(4,2) rather than NUMBER(4,2).)**

There are two way to get this job done. The two possible solutions are shown next.

Solution A:

CREATE TABLE EMP\_2 (

EMP\_NUM CHAR(3) NOT NULL UNIQUE,

EMP\_LNAME VARCHAR(15) NOT NULL,

EMP\_FNAME VARCHAR(15) NOT NULL,

EMP\_INITIAL CHAR(1),

EMP\_HIREDATE DATE NOT NULL,

JOB\_CODE CHAR(3) NOT NULL,

PRIMARY KEY (EMP\_NUM),

FOREIGN KEY (JOB\_CODE) REFERENCES JOB);

INSERT INTO EMP\_2 SELECT \* FROM EMP\_1;

ALTER TABLE EMP\_2

ADD (EMP\_PCT NUMBER (4,2)),

ADD (PROJ\_NUM CHAR(3));

Solution B:

CREATE TABLE EMP\_2 AS SELECT \* FROM EMP\_1;

ALTER TABLE EMP\_2

ADD (EMP\_PCT NUMBER (4,2)),

ADD (PROJ\_NUM CHAR(3));

1. **Using a single command sequence, write the SQL code that will change the project number (PROJ\_NUM) to 25 for all employees whose job classification (JOB\_CODE) is 502 or higher. When you finish Problems 10 and 11, the EMP\_2 table will contain the data shown in Figure P7.11. (You may assume that the table has been saved again at this point.)**

Figure P7.11 The EMP\_2 table contents after the modification



UPDATE EMP\_2

SET PROJ\_NUM = '25'

WHERE JOB\_CODE > = '502'

1. **Write the SQL code that will change the PROJ\_NUM to 14 for those employees who were hired before January 1, 1994 and whose job code is at least 501. (You may assume that the table will be restored to its condition preceding this question.)**

UPDATE EMP\_2

SET PROJ\_NUM = '14'

WHERE EMP\_HIREDATE <= ' 01-Jan-94'

AND JOB\_CODE >= '501';

1. **Write the two SQL command sequences required to:**

There are many ways to accomplish both tasks. We are illustrating the shortest way to do the job next.

* 1. **Create a temporary table named TEMP\_1 whose structure is composed of the EMP\_2 attributes EMP\_NUM and EMP\_PCT.**

The SQL code shown in problem 13b contains the solution for problem 13a.

* 1. **Copy the matching EMP\_2 values into the TEMP\_1 table.**

CREATE TABLE TEMP\_1 AS SELECT EMP\_NUM, EMP\_PCT FROM EMP\_2;

An alternate way would be to create the table and then, use an INSERT with a sub-select to populate the rows.

CREATE TABLE TEMP\_1 AS (

EMP\_NUM CHAR(3),

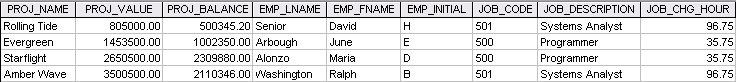
EMP\_PCT NUMBER(4,2));

INSERT INTO TEMP\_1

SELECT EMP\_NUM, EMP\_PCT FROM EMP\_2;

1. **Using the EMPLOYEE, JOB, and PROJECT tables in the** Ch07\_ConstructCo **database (see Figure P7.1), write the SQL code that will produce the results shown in Figure P7.16.**

Figure P7.16 The query results for Problem 16



SELECT PROJ\_NAME, PROJ\_VALUE, PROJ\_BALANCE, EMPLOYEE.EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMPLOYEE.JOB\_CODE, JOB.JOB\_DESCRIPTION, JOB.JOB\_CHG\_HOUR

FROM PROJECT, EMPLOYEE, JOB

WHERE EMPLOYEE.EMP\_NUM = PROJECT.EMP\_NUM

AND JOB.JOB\_CODE = EMPLOYEE.JOB\_CODE;

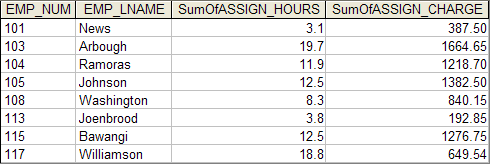
1. **Write the SQL code to calculate the ASSIGN\_CHARGE values in the ASSIGNMENT table in the** Ch07\_ConstructCo **database. (See Figure P7.1.) Note that ASSIGN\_CHARGE is a derived attribute that is calculated by multiplying ASSIGN\_CHG\_HR by ASSIGN\_HOURS.**

UPDATE ASSIGNMENT

SET ASSIGN\_CHARGE = ASSIGN\_CHG\_HR \* ASSIGN\_HOURS;

1. **Using the data in the ASSIGNMENT table, write the SQL code that will yield the total number of hours worked for each employee and the total charges stemming from those hours worked. The results of running that query are shown in Figure P7.22.**

Figure P7.22 Total hours and charges by employee



SELECT ASSIGNMENT.EMP\_NUM, EMPLOYEE.EMP\_LNAME, Sum(ASSIGNMENT.ASSIGN\_HOURS) AS SumOfASSIGN\_HOURS, Sum(ASSIGNMENT.ASSIGN\_CHARGE) AS SumOfASSIGN\_CHARGE

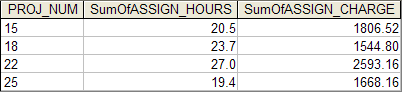
FROM EMPLOYEE, ASSIGNMENT

WHERE EMPLOYEE.EMP\_NUM = ASSIGNMENT.EMP\_NUM

GROUP BY ASSIGNMENT.EMP\_NUM, EMPLOYEE.EMP\_LNAME;

1. **Write a query to produce the total number of hours and charges for each of the projects represented in the ASSIGNMENT table. The output is shown in Figure P7.23.**

Figure P7.23 Total hour and charges by project



SELECT ASSIGNMENT.PROJ\_NUM,

Sum(ASSIGNMENT.ASSIGN\_HOURS) AS SumOfASSIGN\_HOURS,

Sum(ASSIGNMENT.ASSIGN\_CHARGE) AS SumOfASSIGN\_CHARGE

FROM ASSIGNMENT

GROUP BY ASSIGNMENT.PROJ\_NUM

1. **Write the SQL code to generate the total hours worked and the total charges made by all employees. The results are shown in Figure P7.24. (Hint: This is a nested query. If you use Microsoft Access, you can generate the result by using the query output shown in Figure P7.22 as the basis for the query that will produce the output shown in Figure P7.24.)**

Figure P7.24 Total hours and charges, all employees

FigP7-24-Total-Hours-and-Charges

Solution A:

SELECT Sum(SumOfASSIGN\_HOURS) AS SumOfASSIGN\_HOURS,

Sum(SumOfASSIGN\_CHARGE) AS SumOfASSIGN\_CHARGE

FROM Q23;

or

SELECT Sum(SumOfASSIGN\_HOURS) AS SumOfASSIGN\_HOURS,

Sum(SumOfASSIGN\_CHARGE as SumOfASSIGN\_CHARGE

FROM (SELECT ASSIGNMENT.PROJ\_NUM,

Sum(ASSIGNMENT.ASSIGN\_HOURS) AS SumOfASSIGN\_HOURS,

Sum(ASSIGNMENT.ASSIGN\_CHARGE) AS

SumOfASSIGN\_CHARGE

FROM ASSIGNMENT

GROUP BY ASSIGNMENT.PROJ\_NUM

);

Solution B:

SELECT Sum(SumOfASSIGN\_HOURS) AS SumOfASSIGN\_HOURS,

Sum(SumOfASSIGN\_CHARGE) AS SumOfASSIGN\_CHARGE

FROM Q22;

or

SELECT Sum(SumOfASSIGN\_HOURS) AS SumOfASSIGN\_HOURS,

Sum(SumOfASSIGN\_CHARGE) AS SumOfASSIGN\_CHARGE

FROM (SELECT ASSIGNMENT.EMP\_NUM, EMPLOYEE.EMP\_LNAME, Sum(ASSIGNMENT.ASSIGN\_HOURS) AS SumOfASSIGN\_HOURS, Sum(ASSIGNMENT.ASSIGN\_CHARGE) AS

SumOfASSIGN\_CHARGE

FROM EMPLOYEE, ASSIGNMENT

WHERE EMPLOYEE.EMP\_NUM = ASSIGNMENT.EMP\_NUM

GROUP BY ASSIGNMENT.EMP\_NUM, EMPLOYEE.EMP\_LNAME

);

1. **Write a query to count the number of invoices.**

SELECT COUNT(\*) FROM INVOICE;

1. **Write a query to count the number of customers with a customer balance over $500.**

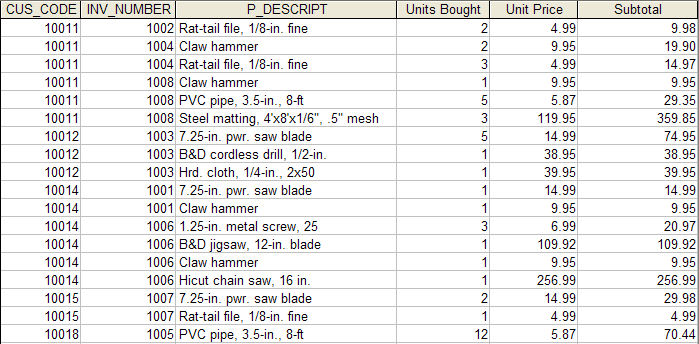
SELECT COUNT(\*)

FROM CUSTOMER

WHERE CUS\_BALANCE >500;

1. **Using the output shown in Figure P7.29 as your guide, generate the listing of customer purchases, including the subtotals for each of the invoice line numbers. (*Hint*: Modify the query format used to produce the listing of customer purchases in Problem 18, delete the INV\_DATE column, and add the derived (computed) attribute LINE\_UNITS \* LINE\_PRICE to calculate the subtotals.)**

**FIGURE P7.29 Summary of Customer Purchases with Subtotals**



SELECT INVOICE.CUS\_CODE, INVOICE.INV\_NUMBER, PRODUCT.P\_DESCRIPT,

LINE.LINE\_UNITS AS [Units Bought], LINE.LINE\_PRICE AS [Unit Price],

LINE.LINE\_UNITS\*LINE.LINE\_PRICE AS Subtotal

FROM CUSTOMER, INVOICE, LINE, PRODUCT

WHERE CUSTOMER.CUS\_CODE = INVOICE.CUS\_CODE

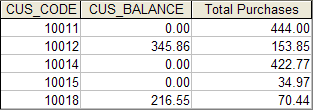
AND INVOICE.INV\_NUMBER = LINE.INV\_NUMBER

AND PRODUCT.P\_CODE = LINE.P\_CODE

ORDER BY INVOICE.CUS\_CODE, INVOICE.INV\_NUMBER, PRODUCT.P\_DESCRIPT;

1. **Modify the query used in Problem 29 to produce the summary shown in Figure P7.30.**

**FIGURE P7.30 Customer Purchase Summary**



SELECT INVOICE.CUS\_CODE, CUSTOMER.CUS\_BALANCE,

Sum(LINE.LINE\_UNITS\*LINE.LINE\_PRICE) AS [Total Purchases]

FROM CUSTOMER, INVOICE, LINE

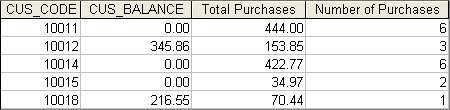
WHERE INVOICE.INV\_NUMBER = LINE.INV\_NUMBER

AND CUSTOMER.CUS\_CODE = INVOICE.CUS\_CODE

GROUP BY INVOICE.CUS\_CODE, CUSTOMER.CUS\_BALANCE;

1. **Modify the query in Problem 30 to include the number of individual product purchases made by each customer. (In other words, if the customer’s invoice is based on three products, one per LINE\_NUMBER, you would count three product purchases. If you examine the original invoice data, you will note that customer 10011 generated three invoices, which contained a total of six lines, each representing a product purchase.) Your output values must match those shown in Figure P7.31.**

**FIGURE P7.31 Customer Total Purchase Amounts and Number of Purchases**



SELECT INVOICE.CUS\_CODE, CUSTOMER.CUS\_BALANCE,

Sum(LINE.LINE\_UNITS\*LINE.LINE\_PRICE) AS [Total Purchases],

Count(\*) AS [Number of Purchases]

FROM CUSTOMER, INVOICE, LINE

WHERE INVOICE.INV\_NUMBER = LINE.INV\_NUMBER

AND CUSTOMER.CUS\_CODE = INVOICE.CUS\_CODE

GROUP BY INVOICE.CUS\_CODE, CUSTOMER.CUS\_BALANCE;