1. **Write the query that will generate a combined list of customers (from tables CUSTOMER and CUSTOMER\_2) that do not include the duplicate customer records. (Note that only the customer named Juan Ortega shows up in both customer tables.)**

SELECT CUST\_LNAME, CUST\_FNAME FROM CUSTOMER

UNION

SELECT CUST\_LNAME, CUST\_FNAME FROM CUSTOMER\_2;

1. **Write the query that will show only the duplicate customer records.**

We have shown both Oracle and MS Access query formats:

In Oracle:

SELECT CUST\_LNAME, CUST\_FNAME FROM CUSTOMER

INTERSECT

SELECT CUST\_LNAME, CUST\_FNAME FROM CUSTOMER\_2;

In MS Access:

SELECT C.CUST\_LNAME, C.CUST\_FNAME

FROM CUSTOMER AS C, CUSTOMER\_2 AS C2

WHERE C.CUST\_LNAME=C2.CUST\_LNAME AND C.CUST\_FNAME=C2.CUST\_FNAME;

Because Access doesn’t support the INTERSECT SQL operator, you need to list only the rows in which all the attributes match.

1. **Write the query that will generate only the records that are unique to the CUSTOMER\_2 table.**

We have shown both Oracle and MS Access query formats:

In Oracle:

SELECT CUST\_LNAME, CUST\_FNAME FROM CUSTOMER\_2

MINUS

SELECT CUST\_LNAME, CUST\_FNAME FROM CUSTOMER;

In MS Access:

SELECT C2.CUST\_LNAME, C2.CUST\_FNAME

FROM CUSTOMER\_2 AS C2

WHERE C2.CUST\_LNAME + C2.CUST\_FNAME NOT IN

 (SELECT C1.CUST\_LNAME + C1.CUST\_FNAME FROM CUSTOMER C1);

Because Access doesn’t support the MINUS SQL operator, you need to list only the rows that are in CUSTOMER\_2 that do not have a matching row in CUSTOMER.

1. **Write the query to show the invoice number, the customer number, the customer name, the invoice date, and the invoice amount for all customers with a customer balance of $1,000 or more.**

This command will run in Oracle and in MS Access:

SELECT INV\_NUM, CUSTOMER.CUST\_NUM, CUST\_LNAME, CUST\_FNAME, INV\_DATE, INV\_AMOUNT

FROM INVOICE INNER JOIN CUSTOMER ON INVOICE.CUST\_NUM=CUSTOMER.CUST\_NUM

WHERE CUST\_BALANCE>=1000;

1. **Write the query that will show the invoice number, the invoice amount, the average invoice amount, and the difference between the average invoice amount and the actual invoice amount.**

There are at least two ways to do this query.

SELECT INV\_NUM, AVG\_INV, (INV\_AMOUNT - AVG\_INV) AS DIFF

FROM INVOICE, (SELECT AVG(INV\_AMOUNT) AS AVG\_INV FROM INVOICE)

GROUP BY INV\_NUM, AVG\_INV, INV\_AMOUNT- AVG\_INV

Another way to write this query is:

SELECT INV\_NUM, INV\_AMOUNT,

 (SELECT AVG(INV\_AMOUNT) FROM INVOICE) AS AVG\_INV,

 (INV\_AMOUNT-(SELECT AVG(INV\_AMOUNT) FROM INVOICE)) AS DIFF

FROM INVOICE

GROUP BY INV\_NUM, INV\_AMOUNT;

The preceding code examples will run in both Oracle and MS Access.

1. **Modify the CUSTOMER table to included two new attributes: CUST\_DOB and CUST\_AGE. Customer 1000 was born on March 15, 1979, and customer 1001 was born on December 22, 1988.**

In Oracle:

ALTER TABLE CUSTOMER ADD (CUST\_DOB DATE) ADD (CUST\_AGE NUMBER);

The SQL code required to enter the date values is:

UPDATE CUSTOMER

SET CUST\_DOB = ’15-MAR-1979’

WHERE CUST\_NUM = 1000;

UPDATE CUSTOMER

SET CUST\_DOB = ‘2-DEC-1988’

WHERE CUST\_NUM = 1001;

1. **Assuming you completed problem 10, write the query that will list the names and ages of your customers.**

In Oracle:

SELECT CUST\_LNAME, CUST\_FNAME, ROUND((SYSDATE-CUST\_DOB)/365,0) AS AGE

FROM CUSTOMER;

In MS Access:

SELECT CUST\_LNAME, CUST\_FNAME, ROUND((DATE()-CUST\_DOB)/365,0) AS AGE

FROM CUSTOMER;

1. **Write the query that will list the average age of your customers. (Assume that the CUSTOMER table has been modified to include the CUST\_DOB and the derived CUST\_AGE attribute.)**

SELECT AVG(CUST\_AGE) FROM CUSTOMER;

1. **Write the trigger to update the CUST\_BALANCE in the CUSTOMER table when a new invoice record is entered. (Assume that the sale is a credit sale.) Test the trigger using the following new INVOICE record:**

**8005, 1001, ’27-APR-12’, 225.40**

**Name the trigger** **trg\_updatecustbalance**.

CREATE OR REPLACE TRIGGER TRG\_UPDATECUSTBALANCE

AFTER INSERT ON INVOICE

FOR EACH ROW

BEGIN

 UPDATE CUSTOMER

SET CUST\_BALANCE = CUST\_BALANCE + :NEW.INV\_AMOUNT

 WHERE CUST\_NUM = :NEW.CUST\_NUM;

END;

To test the trigger you do the following:

SELECT \* FROM CUSTOMER;

INSERT INTO INVOICE VALUES (8005,1001,’27-APR-12’,225.40);

SELECT \* FROM CUSTOMER;

 MS SQL SERVER SOLUTION

create trigger trg\_update\_custbal

on invoice

after insert

as

update customer

set customer.cust\_balance= customer.cust\_balance+

(select inv\_amount from inserted)

where customer.cust\_num = (select cust\_num from inserted);

TO TEST

select \* from CUSTOMER;

select \* from INVOICE;

insert into INVOICE values (8005,1001, '2013-03-01', 190);

select \* from CUSTOMER; select \* from INVOICE;

1. **Write a procedure to add a new customer to the CUSTOMER table. Use the following values in the new record:**

**1002, ‘Rauthor’, ‘Peter’, 0.00**

**Name the procedure prc\_cust\_add. Run a query to see if the record has been added.**

CREATE OR REPLACE PROCEDURE PRC\_CUST\_ADD

(W\_CN IN NUMBER, W\_CLN IN VARCHAR, W\_CFN IN VARCHAR, W\_CBAL IN NUMBER) AS

BEGIN

 INSERT INTO CUSTOMER (CUST\_NUM, CUST\_LNAME, CUST\_FNAME, CUST\_BALANCE)

 VALUES (W\_CN, W\_CLN, W\_CFN, W\_CBAL);

END;

To test the procedure:

EXEC PRC\_CUST\_ADD(1002,’Rauthor’,’Peter’,0.00);

SELECT \* FROM CUSTOMER;

MS SQL SERVER SOLUTION

create PROCEDURE prv\_cust\_add @w\_cn decimal(18,0), @w\_cln varchar(20), @w\_cfn varchar(20), @w\_cbal decimal(18,0) as insert into customer (cust\_num, cust\_lname, cust\_fname,cust\_balance) values (@w\_cn,@w\_cln,@w\_cfn,@w\_cbal);

TO TEST

select \* from CUSTOMER;

exec prv\_cust\_add 1002,'Gold','Arthur',100; select \* from CUSTOMER;