CSCI 5020 Assignment 7

Chapter 8 How to work with data types and functions

4. Write a SELECT statement that returns these columns from the Invoices table:

   The invoice_number column
   The invoice_date column
   The invoice_date column plus 30 days
   The payment_date column

   A column named days_to_pay that shows the number of days between the invoice date and the payment date

   The number of the invoice_date's month
   The four-digit year of the invoice_date
   The last day of the invoice date's month

   When you've got this working, add a WHERE clause that retrieves just the invoices for the month of May based on the invoice date, not the number of the invoice month.

5. Write a SELECT statement that returns these columns from the Invoices table:

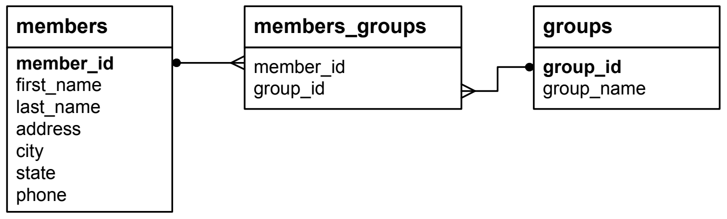
   The invoice_number column

   The balance due (invoice total minus payment total minus credit total) with commas, a decimal point, and two decimal positions

   A column named "Balance Rank" that uses the RANK function to return a column that ranks the balance due in descending order.

Chapter 10 How to create tables, sequences, and indexes

1. Write an ALTER TABLE statement that adds two new check constraints to the Invoices table of the AP schema. The first should allow (1) payment_date to be null only if payment_total is zero and (2) payment_date to be not null only if payment_total is greater than zero. The second constraint should prevent the sum of payment_total and credit_total from being greater than invoice_total.
2. Add an index to the AP schema for the zip code field in the Vendors table.
3. Write the CREATE TABLE statements needed to implement the following design in the EX schema:



   These tables provide for members of an association, and each member can be registered in one or more groups within the association. There should be one row for each member in the Members table and one row for each group in the Groups table. The member ID and group ID columns are the primary keys for the Members and Groups tables. And the Members_Groups table relates each member to one or more groups.

When you create the tables, be sure to include the key constraints. Also, include any null or default constraints that you think are necessary.

4. Write INSERT statements that add two rows to the Members table for member IDs 1 and 2, two rows to the Groups table for group IDs 1 and 2, and three rows to the Group_Membership table: one row for member 1 and group 2; one for member 2 and group 1; and one for member 2 and group 2. Then, write a SELECT statement that joins the three tables and retrieves the group name, member last name, and member first name.

7. Write an ALTER TABLE statement that adds two new columns to the Members table: one column for annual dues that provides for three digits to the left of the decimal point and two to the right; and one column for the payment date. The annual dues column should have a default value of 52.50.

Chapter 11 How to create views

1. Create a view that defines a view named open_items that shows the invoices that haven't been paid. This view should return four columns from the Vendors and Invoices tables: vendor_name, invoice_number, invoice_total, and balance_due (invoice_total – payment_total – credit_total). However, a row should only be returned when the balance due is greater than zero, and the rows should be in sequence by vendor_name. Then, run the script to create the view, and use SQL Developer to review the data that it returns. (You may have to click on the Refresh button in the Connections window after you click on the Views node to show the view you just created.)

2. Write a SELECT statement that returns all of the columns in the open_items view that you created in exercise 1, with one row for each invoice that has a balance due of $1000 or more.

Chapter 12 How to manage database security

1. Write a script that creates a database role named payment_entry in the AP schema. This new role should have SELECT and UPDATE privileges for the Vendors table; SELECT and UPDATE privileges for the Invoices table; and SELECT, UPDATE, and INSERT privileges for the Invoice_Line_Items table. This role should also have the right to create a session.

2. Write a script that creates a user named Tom with a password of "temp" and assigns the payment_entry role to him. Next, use SQL*Plus to test whether the username and password are able to connect to the database. Then, use SQL*Plus to run a SELECT statement that selects the vendor_id column from all rows in the Vendors table.

3. Use SQL Developer to test whether the user and role that you created in exercises 1 and 2 work correctly. To start, create a new connection for the used named Tom. Then, write a SELECT statement to select all of the columns for all of the rows in the Vendors table, and use the Tom connection to run it. (It should succeed, though you may need to qualify the table name with the schema name.) Finally, write a DELETE statement that attempts to delete one of the rows in the Vendors table, and use the Tom connection to run it. (It should fail due to insufficient privileges.)

Write each statement in separate script files (*.sql). Zip all the files into a single file and submit it to the Dropbox.