**Manipulating the Database Structure**

Your first big project as a VaporGames database developer is to create the foundation for some new features. This includes manipulating the structure of the database and inserting new data.

For this assignment, write queries that will answer the questions below. Save every query to a SQL script named ManipulateDBStructure. Use -- to write a comment above each query, noting the question it refers to. Additionally, take a screenshot of the query and result set for each question, and insert it below the corresponding question.

1. Your first task is to eliminate the STOREFRONT table. To accomplish this without losing any existing data, complete the following:  
   * Alter the PRODUCTLIST table to include a PRICE and DESCRIPTION column
   * Write a series of update statements to move the price and description data stored within STOREFRONT to the new columns in PRODUCTLIST
   * Drop the STOREFRONT table
2. VaporGames would like to improve their user experience by adding chat features as a new service. Your second task is to design a new table to accommodate this feature. Users who receive a message should be able to see the sender’s USERID, the date the message was sent, and the contents of the message. To implement this, complete the following:  
   * Create a CHATLOG table with this design:
     + Include a column named CHATID with the data type NUMBER(3)
     + Include a column named RECEIVERID with the data type NUMBER(3)
     + Include a column named SENDERID with the data type NUMBER(3)
     + Include a column named DATESENT with the data type DATE
     + Include a column named CONTENT with the data type VARCHAR2(250)
     + Include a Primary Key constraint on the CHATID column
     + Include a Foreign Key constraint on RECEIVERID that references the USERID column in the USERBASE table
     + Include a Foreign Key constraint on SENDERID that references the USERID column in the USERBASE table
   * Insert 10 - 15 rows of sample data into the CHATLOG table.
3. A screenshot of a computer

   AI-generated content may be incorrect.The third task you are assigned is to help improve user interaction by accommodating a friends list feature. Accomplish this by completing the following:  
   * Create a FRIENDSLIST table with this design:
     + Include a column named USERID with the data type NUMBER(3)
     + Include a column named FRIENDID with the data type NUMBER(3)
     + Include a Primary Key constraint on the USERID and FRIENDID columns
     + Include a Foreign Key constraint on USERID that references the USERID in the USERBASE table
     + Include a Foreign Key constraint on FRIENDID that references the USERID column in the USERBASE table
   * Insert 10 - 15 rows of sample data into the FRIENDSLIST table.
4. The next step in the user interaction overhaul is providing a wish list feature. Users should be able to change the position of each item in the list. Implement this feature by completing the following:  
   * Create a WISHLIST table with this design:
     + Include a column named USERID with the data type NUMBER(3)
     + Include a column named PRODUCTCODE with the data type VARCHAR2(5)
     + Include a column named POSITION with the data type NUMBER(3)
     + Include a Primary Key constraint on the USERID and PRODUCTCODE columns
     + Include a Foreign Key constraint on USERID that references the USERID column in the USERBASE table
     + Include a Foreign Key constraint on PRODUCTCODE that references the PRODUCTCODE column in the PRODUCTLIST table
   * Insert 10 - 15 rows of sample data into the WISHLIST table.

1. Users have voted to add a user profile page and VaporGames has accepted this request, including it in your task list. Users would like to save the path to an image file to use as a profile picture and be able to describe themselves in an “About Me” section. Implement this feature by completing the following:  
   * Create a USERPROFILE table with this design:
     + Include a column named USERID with the data type NUMBER(3)
     + Include a column named IMAGEFILE with the data type VARCHAR2(250)
     + Include a column named DESCRIPTION with the data type VARCHAR2(250)
     + Include a Primary Key constraint on the USERID column
     + Include a Foreign Key constraint on USERID that references the USERID column in the USERBASE table
   * Insert 10 - 15 rows of sample data into the USERPROFILE table.
2. Another item on your agenda is to help with account retrieval by storing responses to security questions. Accomplish this task by completing the following:  
   * Create a SECURITYQUESTION table with this design:
     + Include a column named QUESTIONID with the data type NUMBER
     + Include a column named USERID with the data type NUMBER(3)
     + Include a column named QUESTION with the data type VARCHAR2(250)
     + Include a column named ANSWER with the data type VARCHAR2(250)
     + Include a Primary Key constraint on the QUESTIONID column
     + Include a Foreign Key constraint on USERID that references the USERID column in the USERBASE table
   * Insert 10 - 15 rows of sample data into the SECURITYQUESTION table.
3. VaporGames has decided to implement guidelines to promote a healthy and respectful environment on the platform with these new user interaction updates. Each guideline should have a number to reference by, a title that denotes the type of rule, a description explaining the conditions of the rule, and a numerical point system indicating its importance (a higher point number means a more important rule). Implement this feature by completing the following:  
   * Create a COMMUNITYRULES table with this design:
     + Include a column named RULENUM with the data type NUMBER(3)
     + Include a column named TITLE with the data type VARCHAR2(250)
     + Include a column named DESCRIPTION with the data type VARCHAR2(250)
     + Include a column named SEVERITYPOINT with the data type NUMBER(4)
     + Include a Primary Key constraint on the RULENUM column
   * Insert 10 - 15 rows of sample data into the COMMUNITYRULES table.
4. Along with the new guidelines, VaporGames would like a way to track when users have violated the rules and if they received a punishment for the violation. Accomplish this task by competing the following:  
   * Create an INFRACTIONS table with this design:
     + Include a column named INFRACTIONID with the data type NUMBER
     + Include a column named USERID with the data type NUMBER(3)
     + Include a column named RULENUM with the data type NUMBER(3)
     + Include a column named DATEASSIGNED with the data type DATE
     + Include a column named PENALTY with the data type VARCHAR2(250)
     + Include a Primary Key constraint on the INFRACTIONID column
     + Include a Foreign Key constraint on USERID that references the USERID column in the USERBASE table
     + Include a Foreign Key constraint on RULENUM that references the RULENUM column in the COMMUNITYRULES table
   * Insert 10 - 15 rows of sample data into the INFRACTIONS table.
5. The last table you are requested to design for the user interaction overhaul is a user support ticketing system. Users should be able to provide a contact email address and a quick description of what they need support on. The ticketing system should also keep track of the date the ticket was submitted, when it was last updated, and the current status of the ticket (‘NEW’, ’IN PROGRESS’, ’CLOSED’). Accomplish this task by completing the following:  
   * Create a USERSUPPORT table with this design:
     + Include a column named TICKETID with the data type NUMBER
     + Include a column named EMAIL with the data type VARCHAR2(250)
     + Include a column named ISSUE with the data type VARCHAR2(250)
     + Include a column named DATESUBMITTED with the data type DATE
     + Include a column named DATEUPDATED with the data type DATE
     + Include a column named STATUS with the data type VARCHAR2(250)
     + Include a Primary Key constraint on the TICKETID column
   * Insert 10 - 15 rows of sample data into the USERSUPPORT table.
6. Your final task is to generate views to display some useful information. Create views that do the following:  
   * Create a view that displays every unique QUESTION from the SECURITYQUESTION table.
   * Create a view that displays the TICKETID, EMAIL, ISSUE, and DATEUPDATED only for tickets with a STATUS of ‘NEW’ or ‘IN PROGRESS’, sorted by the earliest DATEUPDATED.