Homework Problem Set 4: SQL SELECT, Part I

# Overview

In this lab, we will explore learn how to write SQL queries using the SELECT statement and use SQL to answer questions and solve problems in the data.

## Learning Objectives

Upon completion of the lab, you should be able to:

* Write single table SELECT queries.
* Use the JOIN clause to combine multiple tables in a query into a single output.
* Use relational and logical expressions in the WHERE clause.
* Use the CASE statement to display columnar output based on various conditions.
* Use the DISTINCT, ORDER BY, and TOP clauses.
* Read database schemas (internal data models).

## What You Will Need

To complete this lab, you will need the learn-databases environment up and running, specifically:

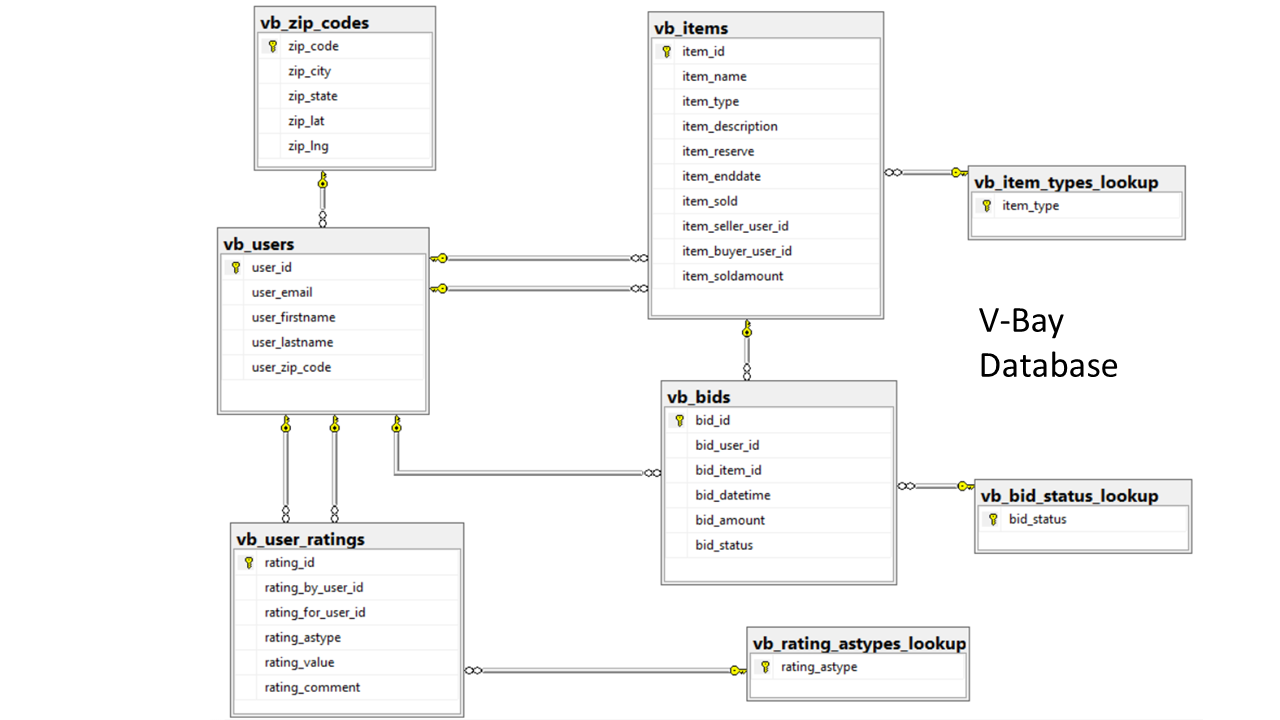
* Microsoft SQL Server DBMS.
* Provision the **vbay** database using the database provisioner application <https://localhost:5000>.
* Azure Data Studio connected to SQL Server with an open query window in the **vbay** database.
* Please review the first lab if you require assistance with these tools.

## The Database: vBay!

vBay! is a knock-off of a popular auction website with a very similar name 😉. A very high-level conceptual data model of the business processes supporting vBay are:

* Users are buyers and/or sellers.
* Users post items for sale as sellers.
* Users place bids on items as buyers.
* The highest bid “wins” the item, and, therefore, that user buys it.
* Users rate each other as buyers and sellers.

Here is the internal model for vBay! with foreign keys so that you can see the metadata business rules that support the data model. For example, the **vb\_items** table has an FK **item\_seller\_user\_id** (the ID of the user selling the item) as well as an FK **item\_seller\_buyer\_id** (the ID of the user who bought the item).



*Figure 1. The internal data model for vBay!. This represents the tables, primary keys, and foreign keys.*

## A Word About Query Writing

It goes without saying that query writing is a critical database skill. Being able to take high-level questions about your data such as “Which items have no bids?” or “Which items have not met their reserve?” and answer them with the SQL SELECT statement is a highly sought-after skillset, which some feel borders on black-magic sorcery. The truth is, it takes knowledge of SQL and an understanding of the internal model of your database to write good queries. It is more than just a technical skill; you must be able to read the data model and understand the underlying business processes that create the information you see before you. It takes practice to master this skill, and you should consider this lab a start on your journey.

# Walkthrough

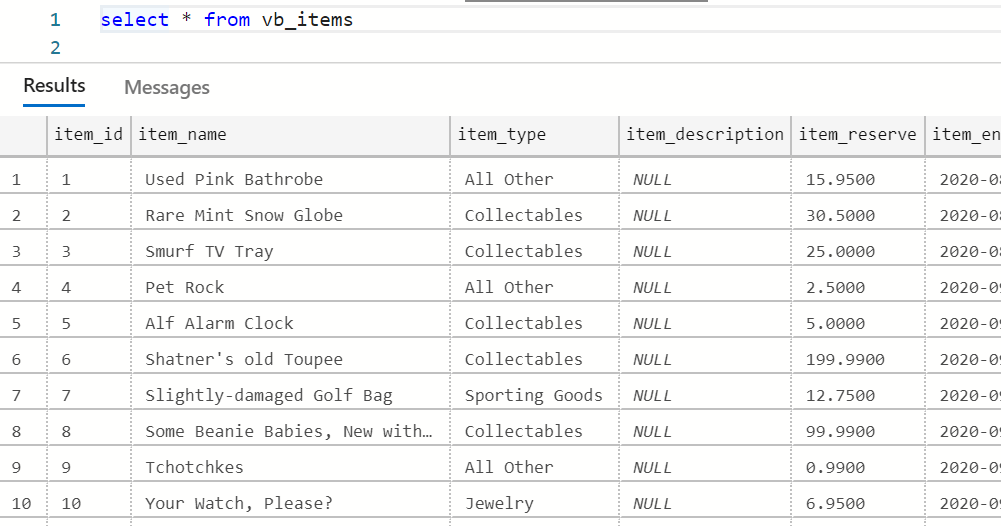
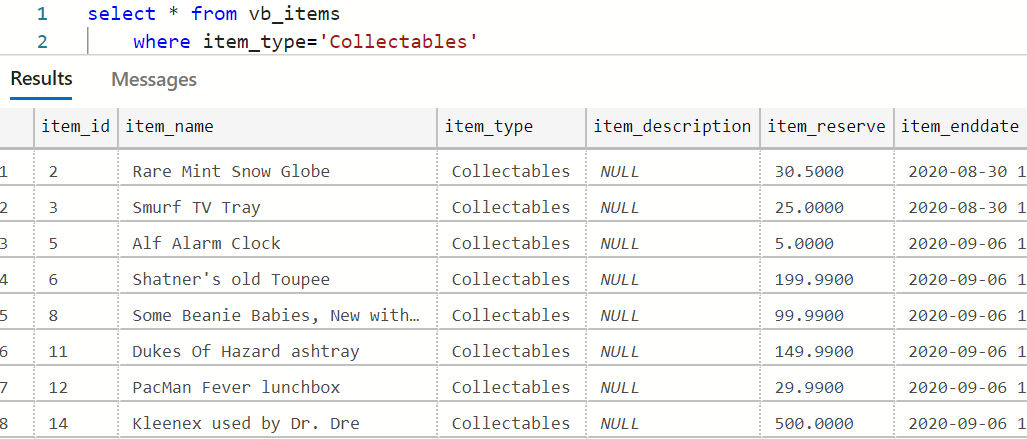
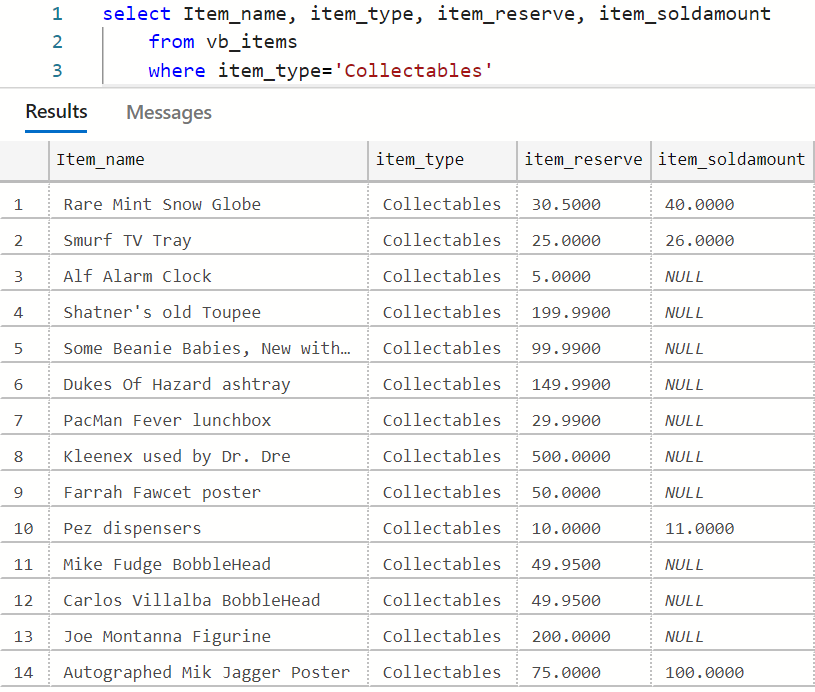
Let’s walk through the query-writing process, focusing on how to break down a question about your data into the corresponding SQL statement. Here we will focus on the process. The general process follows the order the query gets processed, not the order in which it is written:

1. Figure out the tables you will need.
2. Figure out how those tables should be joined
3. Which rows should be filtered?
4. Which columns should be projected?
5. How should the output be limited or sorted?

## Query 1: Collectables

Write a query to show items that are collectables. The reserve price and sold amounts should be included.

Here are the steps. Notice we write and then rewrite the query at each step.

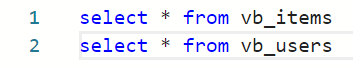
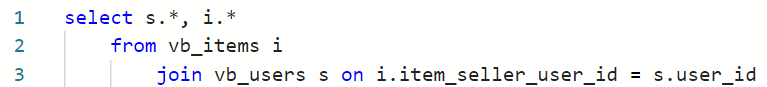
1. Figure out the tables you will need.  **vb\_items**
2. Figure out how those tables should be joined. **No join required—single table.**
3. Which rows should be filtered? **Rows—only Collectables:**
4. Which columns should be projected? **Item\_name, item\_type, item\_reserve, item\_soldamount**  
   
5. How should the output be limited or sorted? **Unsure, but will sort by item\_name to that it’s easy to locate items.**  
   select Item_name, item_type, item_reserve, item_soldamount
       from vb_items
       where item_type='Collectables'
       order by item_name 

It’s just that simple! Let’s try another!

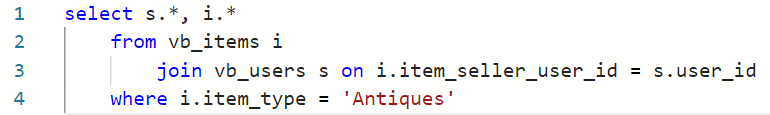
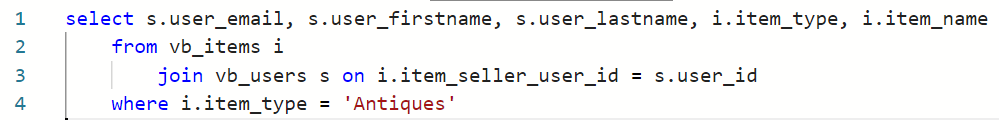
## Query 2: Sellers of Antiques

The marketing department would like to reach out to those vBay! customers who are sellers of antiques. There is a webinar about antique appraisals they feel customers might be interested in attending and would like to send mass-marketing emails to these individuals.

This time we will write the query but only show the output at the end to save space.

1. Figure out the tables you will need.   
     
   After some exploration of the tables, we will need data from both of these. **vb\_items** tells us who is selling the item and whether or not it’s an antique. **vb\_users** has names and emails needed for the email blast.
2. Figure out how those tables should be joined.  
   From the output of the previous step, we can see that there is a column **item\_seller\_user\_id** in **vb\_items**. The internal model shows this is a foreign key referencing **vb\_users**,so we write:

We alias the users table as s for seller. In this output we can see the names of each of the individuals selling the items.

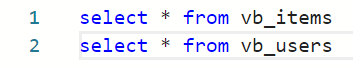
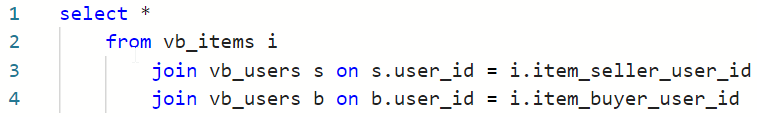
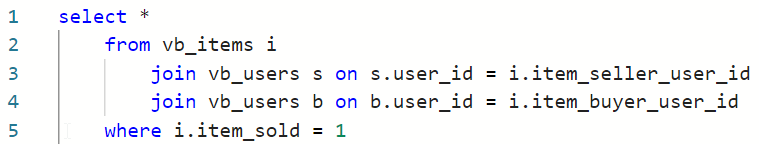
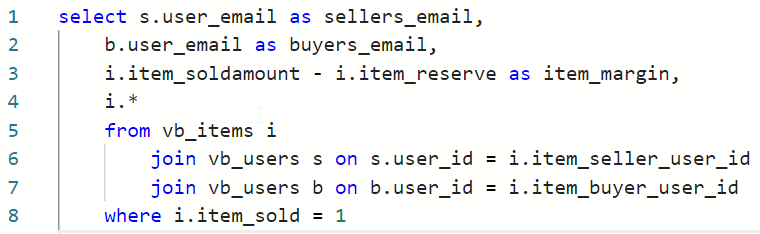
1. Which rows should be filtered?  
   Next, we filter out the rows to show only antiques:  
   
2. Which columns should be projected?  
   There are way more columns than we need, so let’s include only relevant information needed for the email blast:  
   
3. How should the output be limited or sorted?  
   No sorting necessary, so:  
   select s.user_email, s.user_firstname, s.user_lastname, i.item_type, i.item_name
       from vb_items i 
           join vb_users s on i.item_seller_user_id = s.user_id
       where i.item_type = 'Antiques'

You can now download this output and send the file to the marketing department!

## Query 3: Seller’s Report

For the website you need to make a seller’s report. This report should show all information about items that are sold and include the buyer’s and seller’s emails in the output. Furthermore, it should include the seller’s margin, which is the difference between the sold amount and the item reserve. Sort with largest margins first.

Here’s a very high-level breakdown of the steps, one last time:

1. Tables:  
   
2. Joins: Now users are needed in both the buyer and seller’s role, so we must join twice (again reviewing the FKs in the internal data model):  
   
3. Where: Only sold items:  
   
4. Column projections:  
   
5. Sort with highest margins first, with final output:  
   select s.user_email as sellers_email, 
       b.user_email as buyers_email,
       i.item_soldamount - i.item_reserve as item_margin,
       i.*
       from vb_items i
           join vb_users s on s.user_id = i.item_seller_user_id
           join vb_users b on b.user_id = i.item_buyer_user_id
       where i.item_sold = 1
       order by item_margin DESC
       
   

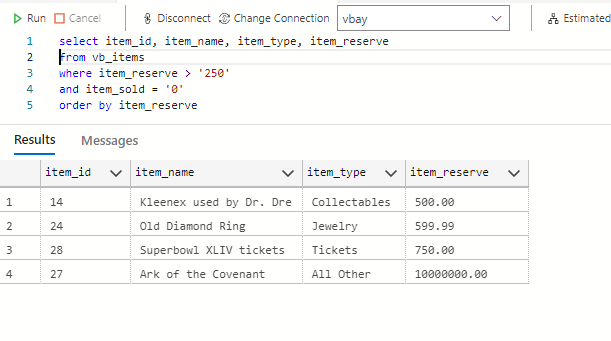
# Questions

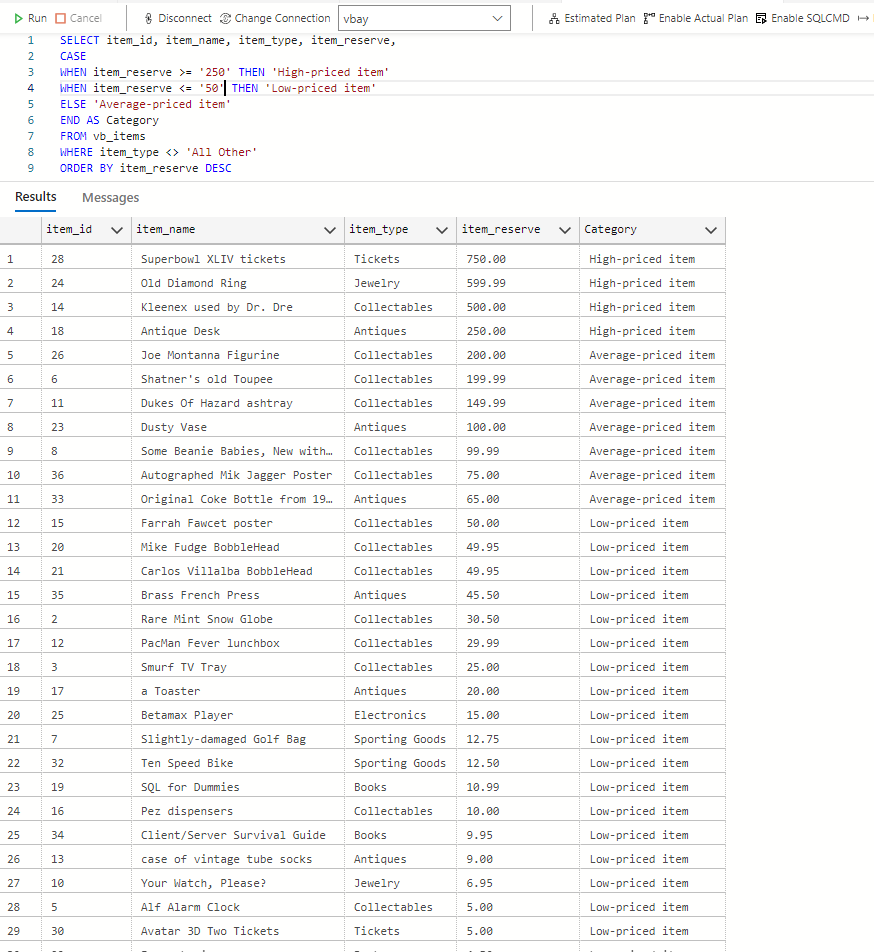
Answer these questions using the problem set submission template. You will need to consult the logical model in the overview section for details. For any screen shots provided, please follow the guidelines for submitting a screen shot.

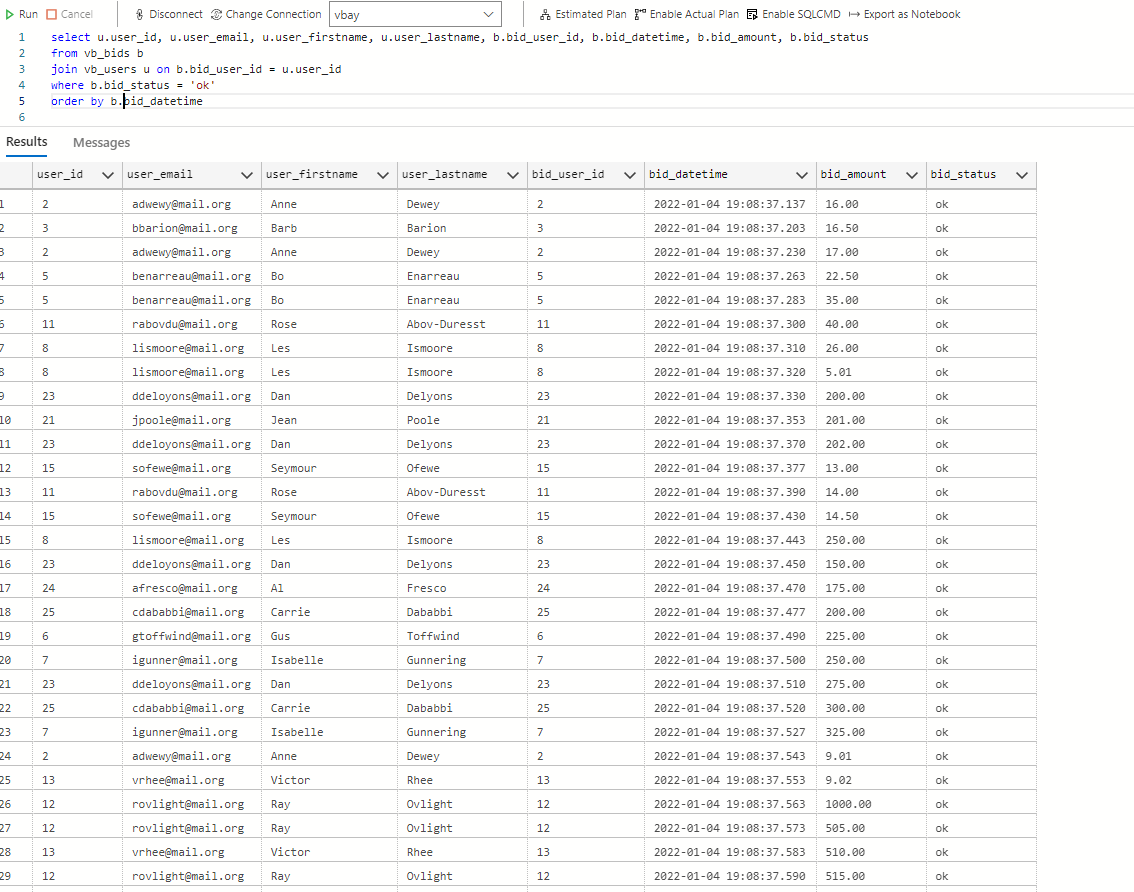
Write the following as SQL queries. If the query is ambiguous, fill in the gaps and justify your reasoning. For each, include the SQL as a screen shot with the output of the query.

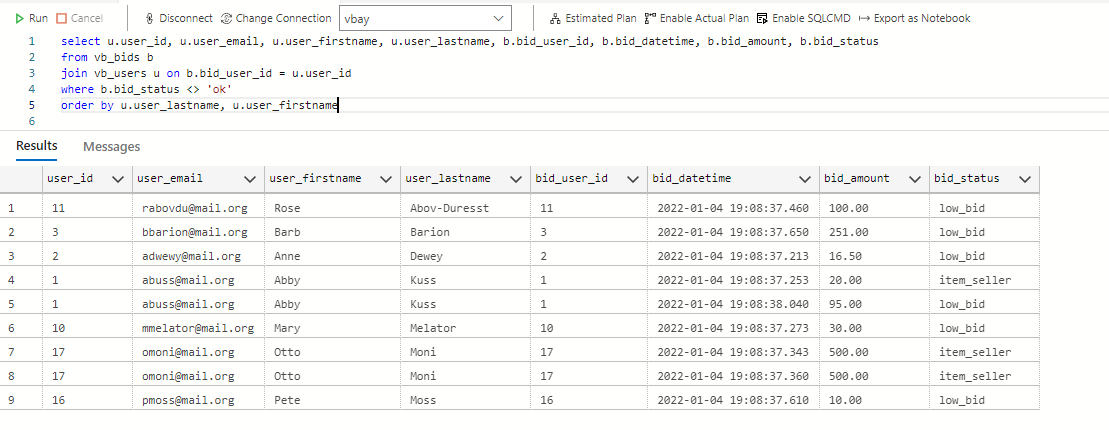
1. Sales would like to send mailings to users who live in a Zip code that starts with “13,” for example, 13244, so that they can be notified of their new contact in that region.  
   Table

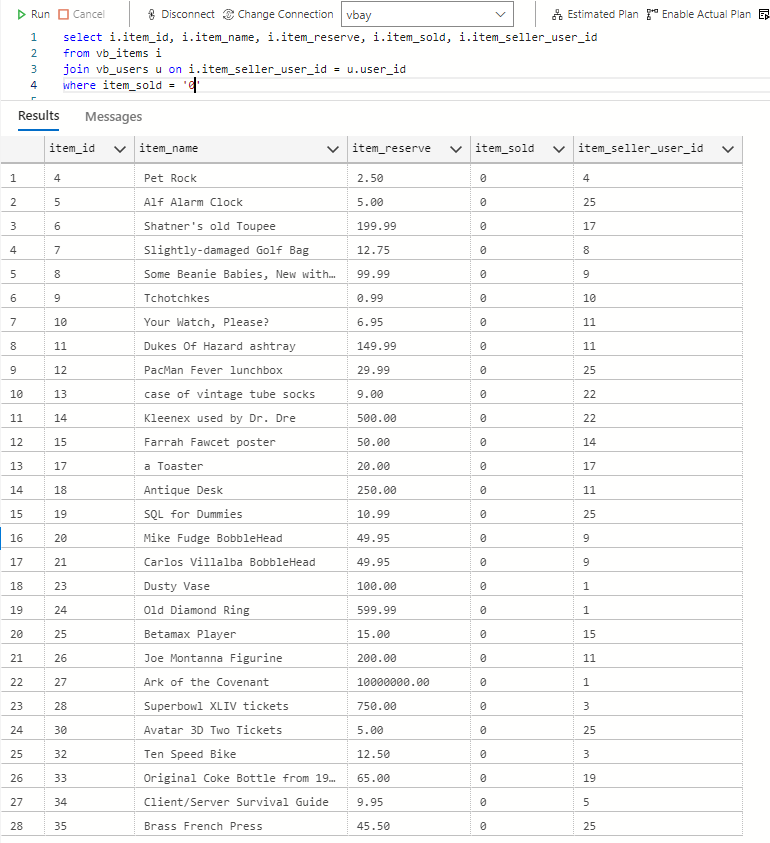
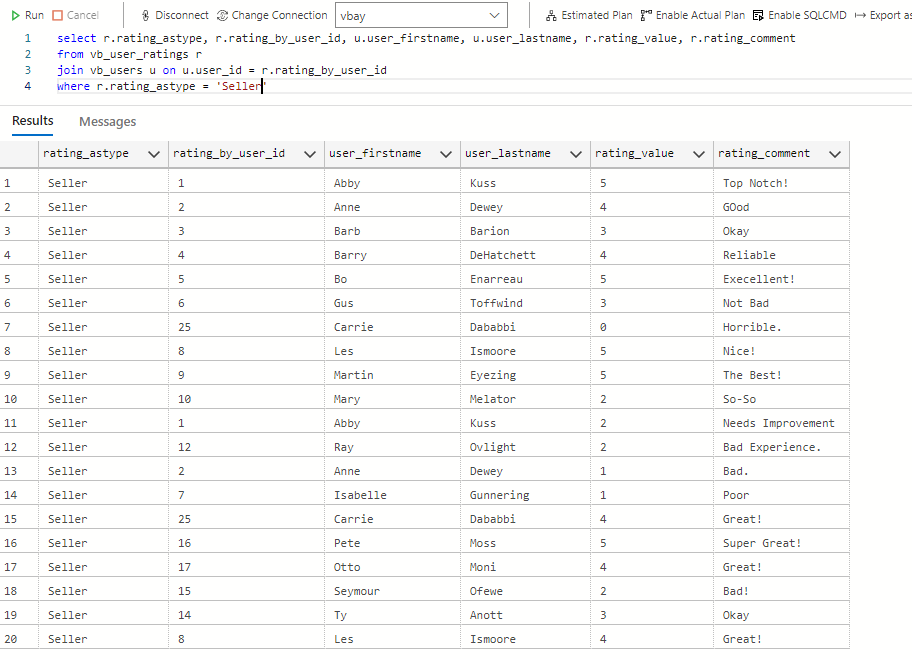
   Description automatically generated
2. Find all the users from the state of New York. Print their names and emails along with their city, state, and Zip code. Sort by city, then by user’s last /first name.  
   Table

   Description automatically generated
3. High-priced items: Return the ID, name, type, and reserve of items that have not been sold and have a reserve of 250 or higher. Sort the output so that the largest reserve items are first.  
   
4. Reserve item categories. Include the ID, name, type, and reserve price of the item. Do not include items of type “All Other”. Create a category column based on item reserve price.  
   When the item is 250 or more, it is a high-priced item.   
   When the item is 50 or less, it is a low-priced item.  
   Everything else is an average-priced item.



1. Bidder list. Write a query that displays the valid user bids (bid status of ‘ok’) for a given item\_id. This would commonly be displayed on the website for the chosen item. You select the item ID to display and show the bid ID, bid user’s name, bid user email, bid date, and bid amount. Put the most recent bids at the top.  
   
2. The bad bidder list. Write query to help the security audit team find fraudulent activity. For any bid that does not have a status of ‘ok’, include the date of the bid, name, email, and ID of the bidder and the name and ID of the item bid upon. Also include the amount of the bid and bid status. Sort the output by the user name (last, then first) and then by bid date for users with multiple bad bids.



1. Produce a report of items that do not contain a bid. Include the item ID, item name, item type, seller’s name, and item reserve.  
   
2. Produce a list of seller ratings. Include the name of the user who gave the rating, the name of the user the rating was for, the rating value, and rating comment. Include ratings of only sellers.  
   
3. For items that were sold, generate a report that includes the locations (city and state) of the buyer and seller. Include item ID, item name, item type item sold amount, name of seller, seller’s city/state, name of buyer, and the buyer’s city /state.  
   Graphical user interface, text

   Description automatically generated
4. Users with no activity. Find the names and emails of any users who have never posted an item for bid or have never bought the item or have never placed a bid.

Table

Description automatically generated