Homework Problem Set 5: SQL SELECT, Part II

# Overview

In this lab, we will explore advanced aspects of the SQL SELECT statements such as data aggregations, window functions, and complex queries.

## Learning Objectives

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

* Use aggregate functions with the GROUP BY and HAVING clauses.
* Demonstrate use of the WITH statement to reduce query complexity.
* Use window functions to apply a function to a partition of data.
* Select the appropriate window function for the problem at hand.
* Write SQL SELECT queries to solve a variety of problems.
* 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:

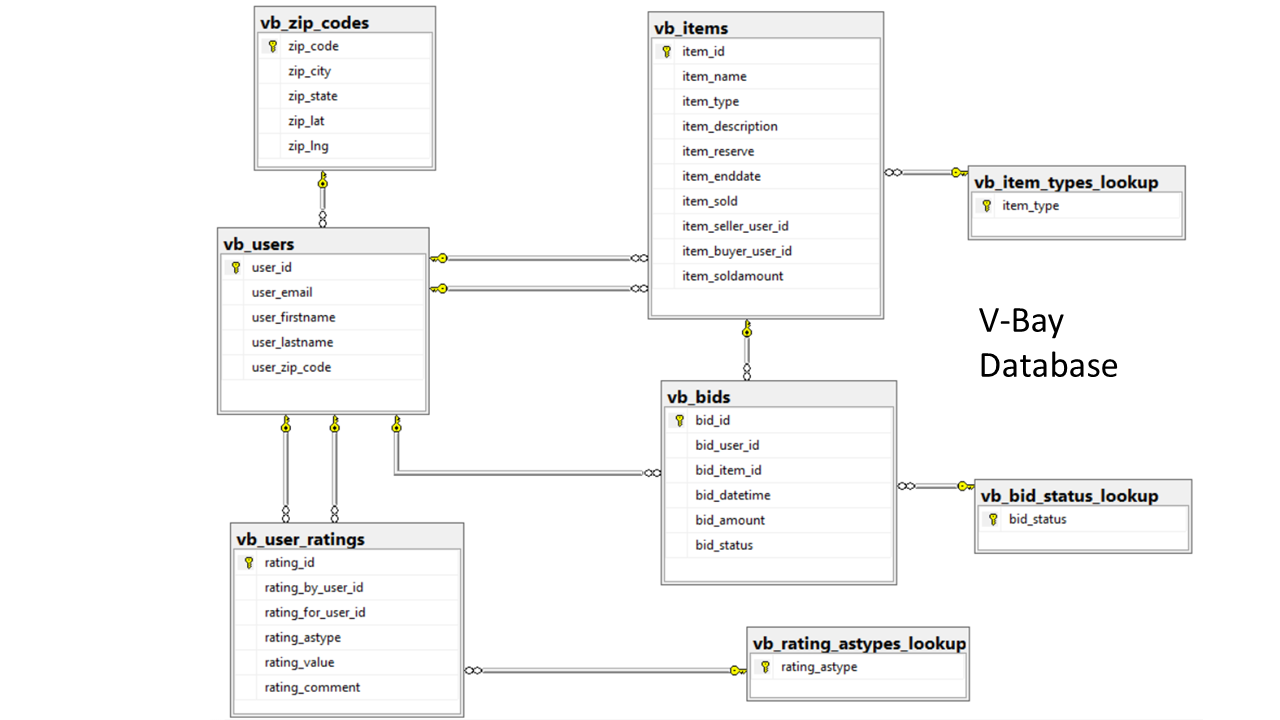
* Microsoft SQL Server DBMS,
* Provision the **vbay** database using the database provisioner application <https://localhost:5000>.
* Azure Data Studio connected to SQL Server with an open query window.
* Please review the first lab if you require assistance with these tools.

## 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.*

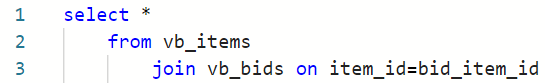
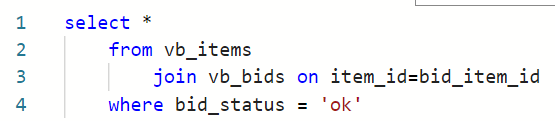
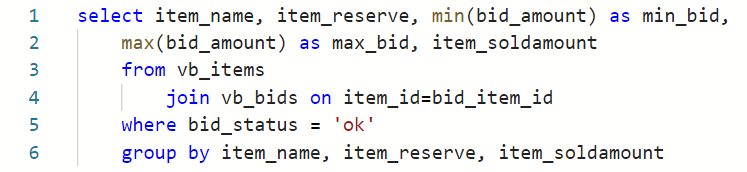
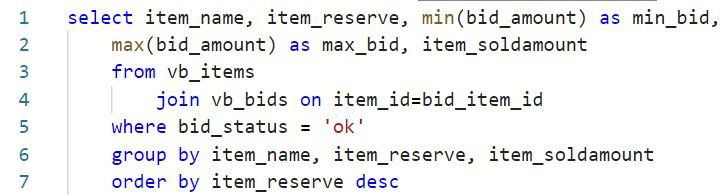
# Walkthrough

Let’s walk through the query-writing process once more, 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. Are there any groupings?
5. Do those grouping need to be filtered?
6. Which columns should be projected?
7. How should the output be limited or sorted?

## Query 1: Highest and Lowest Bids per Item

For all items, include the name of the item, the reserve price, the lowest bid, highest bid, and sold amount. Put the items with the largest reserve first.

1. Figure out the tables you will need. **vb\_bids , vb\_items**  
   
2. Figure out how those tables should be joined. **Join on PK/FK**
3. Which rows should be filtered? **Only valid bids**  
   
4. Are there any groupings? **Yes, minimum and maximum bids grouped by item. We must include all columns in the projection part of an aggregate function in the group by clause as well.**
5. Do those grouping need to be filtered? **No**
6. Which columns should be projected? **Already done**
7. How should the output be limited or sorted? **Sort by reserve in descending order**  
   

## Query 2: Classifying Bidders’ Activity

This query is a major step up and a lot more complicated.

vBay! would like to classify their users based on the numbers of valid bids they have placed.

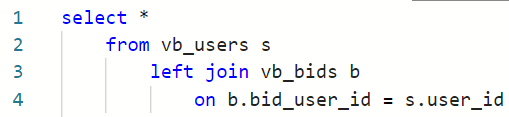
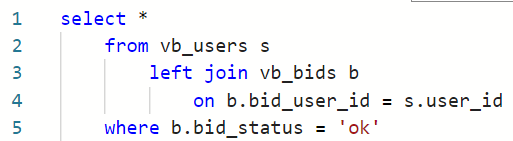
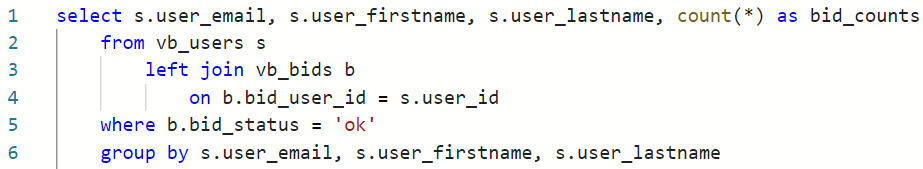
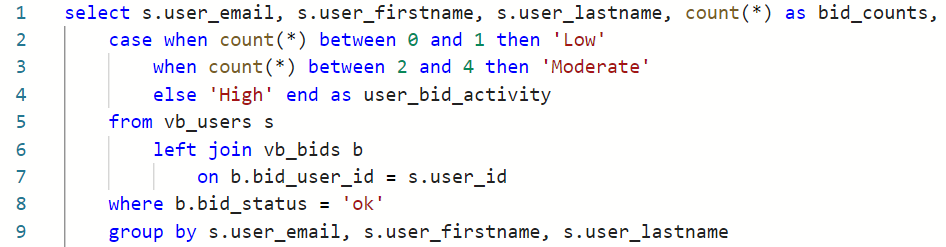
Low Activity = 0 or 1 bids  
Moderate Activity = 2 to 4 bids  
High Activity = 5 or more bids

Then they would like to produce a report counting the number of users who fall into low, moderate, and high activity categories.

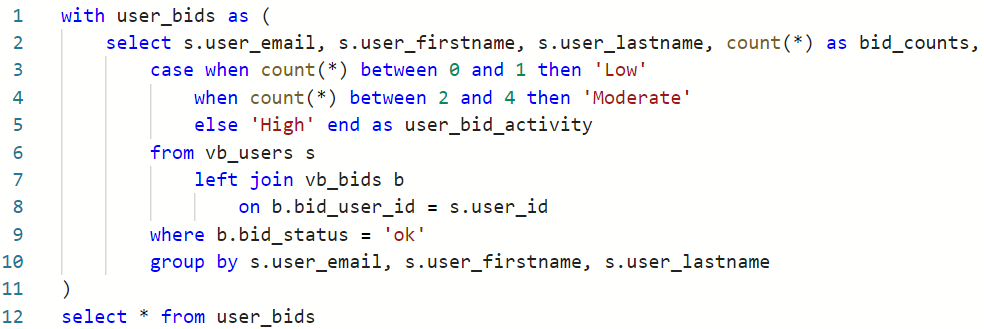
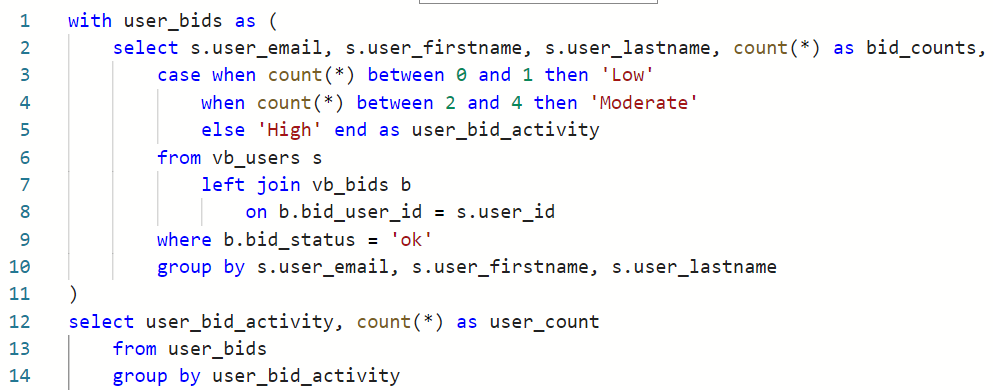
There are several ways to write this query, but we will break it down into two steps:

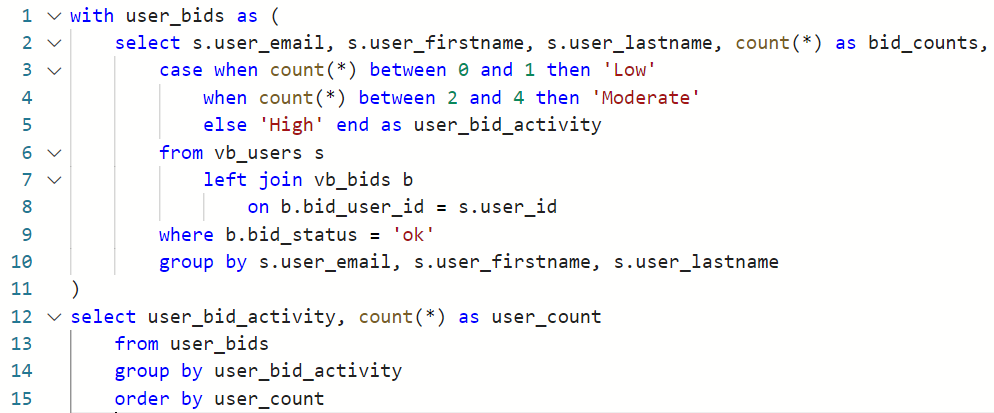
Step 1: Produce user list with count of bids and activities.  
Step 2: Produce activity report from that.

Step 1: Produce user list with count of bids:

1. Figure out the tables you will need. **vb\_users, vb\_bids**  
    
2. Figure out how those tables should be joined. **Left join from users to bids.** This way it includes users with no bids.  
   
3. Which rows should be filtered? **Only valid bids**
4. Are there any groupings? **Yes, we need to show user information with a count of bids. Count(\*) makes sense because we need to include rows with nulls.** Also, because we group by these columns, they should appear in the projection.  
   
5. Do these groupings need to be filtered? **Filtered, no. Further categorized, yes.**
6. Which columns should be projected? **The ones we have already plus a case statement based on the counts to produce Low, Moderate, and High activity.**
7. How should the output be limited or sorted? **No need**.

Step 2: Produce activity report from that.

1. What tables do we need? **The output from the previous query is the “table” we wish to use, so we use the WITH statement to name the first query:**  
   
2. Figure out how those tables should be joined. **No joins.**
3. Which rows should be filtered? **No filters**
4. Are there any groupings? **Yes, group by user\_bid\_activity and count rows.**
5. Do those grouping need to be filtered? **No**
6. Which columns should be projected? **Same**
7. How should the output be limited or sorted? **Let’s sort by user\_count.**



1. 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 

# 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 yourself and justify your reasoning. For each, include the SQL as a screen shot with the output of the query.

1. How many item types are there? Perform an analysis of each item type. For each item type, provide the count of items in that type and the minimum, average, and maximum item reserve prices for that type. Sort the output by item type.   
   Table

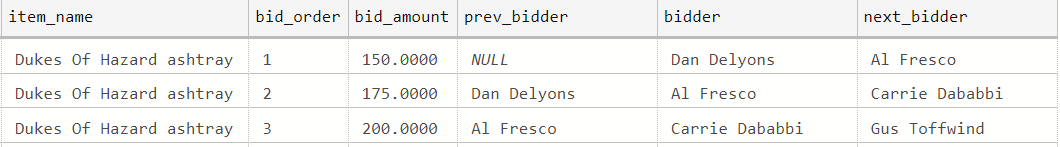
   Description automatically generated
2. Perform an analysis of each item in the “Antiques” and “Collectables” item types. For each item, display the name, item type, and item reserve. Include the minimum, maximum, and average item reserve over each item type so that the current item reserve can be compared to these values.  
   Table

   Description automatically generated
3. Write a query to include the names, counts (number of ratings), and average seller ratings (as a decimal) of users. For reference, User Carrie Dababbi has four seller ratings and an average rating of 4.75.   
   Graphical user interface, application, table

   Description automatically generated
4. Create a list of “Collectable” item types with more than one bid. Include the name of the item and the number of bids, making sure the item with the most bids appear first.  
   Graphical user interface

   Description automatically generated
5. Generate a valid bidding history for any given item of your choice. Display the item ID, item name, a number representing the order the bid was placed, the bid amount, and the bidder’s name. Here’s an example showing the first three bids on item 11: Table

   Description automatically generated  
     
    Graphical user interface, table

   Description automatically generated
6. Rewrite your query in the previous question to include the names of the next and previous bidders, like this example, again showing the first three bids for item 11.  
     
   Graphical user interface, application, table

   Description automatically generated
7. Find the names and emails of the users who give out the worst ratings (lower than the overall average rating) to either buyers or sellers (no need to differentiate whether the user rated a buyer or seller), and include only those users who have submitted more than one rating.  
   Graphical user interface

   Description automatically generated
8. Produce a report of the KPI (key performance indicator) user bids per item. Show the user’s name and email, total number of valid bids, total count of items bid upon, and then the ratio of bids to items. As a check, Anne Dewey’s bids per item ratio is 1.666666.

I am unable to get the correct ratio due to my inability to properly aggregate total of bids per item, and how many bids a person submitted. I know to get total bids, you have to use aggregate functions on the number of records in the bid table . All my answers turns out to a ratio of 1.

Graphical user interface, table

Description automatically generated

1. Among items not sold, show highest bidder name and the highest bid for each item. Make sure to include only valid bids.

Table

Description automatically generated

1. Write a query with output similar to Question 3, but also includes the overall average seller rating and the difference between each user’s average rating and the overall average. For reference, the overall average seller rating should be 3.2.

Graphical user interface, application, table

Description automatically generated