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).

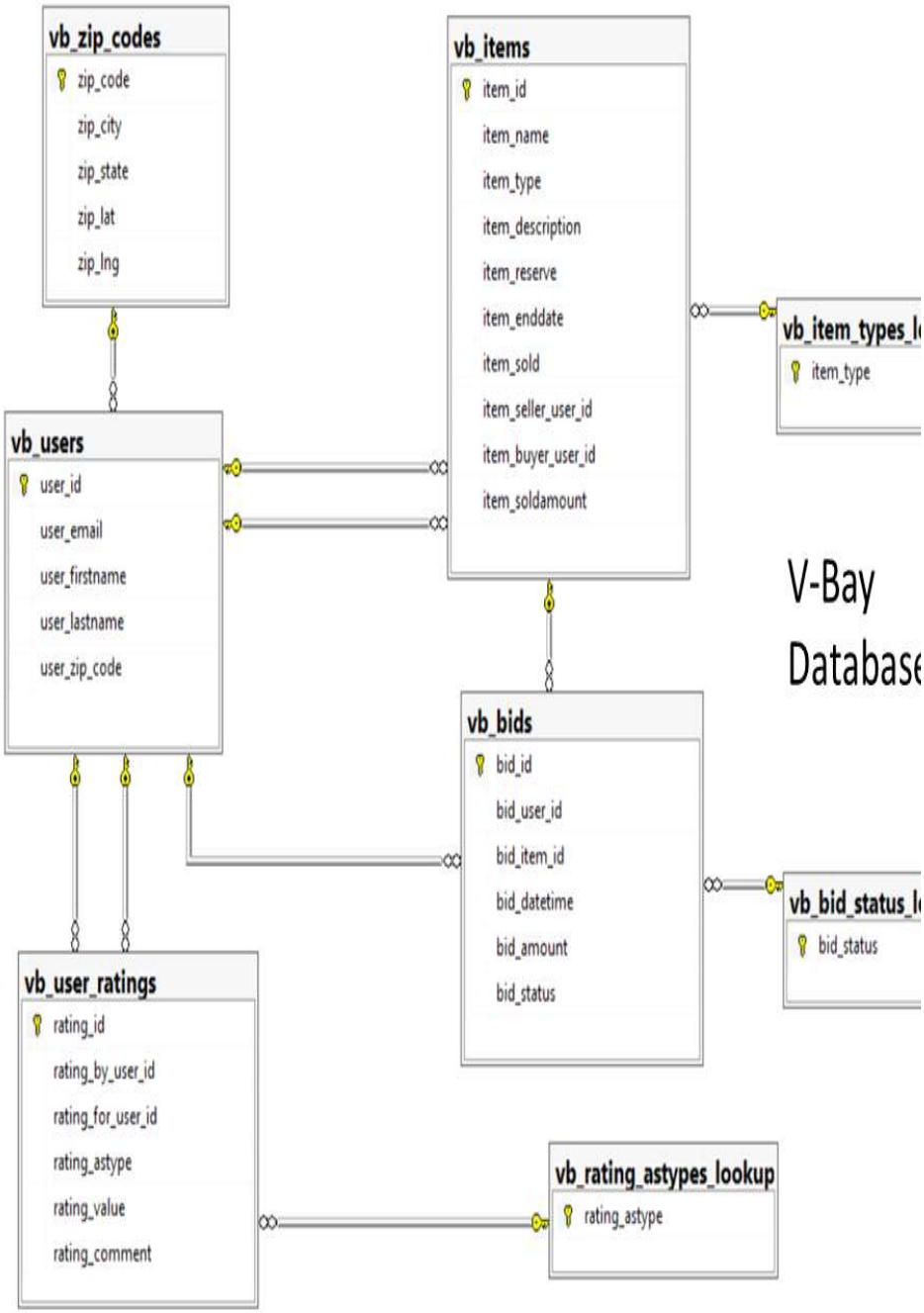


Figure 1. The internal data model for vBay!. This represents the tables, primary keys, and foreign keys.

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:

- Figure out the tables you will need.
- Figure out how those tables should be joined
- Which rows should be filtered?
- Are there any groupings?
- Do those grouping need to be filtered?
- Which columns should be projected?
- 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.

- Figure out the tables you will need. **vb_bids , vb_items**

```
select * from vb_items
select * from vb_bids
```

- Figure out how those tables should be joined. **Join on PK/FK**

```
select *
    from vb_items
        join vb_bids on item_id=bid_
```

- Which rows should be filtered? **Only valid bids**

```
select *
    from vb_items
        join vb_bids on item_id=bid_
    where bid_status = 'ok'
```

- 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.**

```
select item_name, item_reserve, min(bid_amount) a
    max(bid_amount) as max_bid, item_soldamount
    from vb_items
        join vb_bids on item_id=bid_item_id
    where bid_status = 'ok'
    group by item_name, item_reserve, item_soldam
```

- Do those grouping need to be filtered? **No**
- Which columns should be projected? **Already done**
- How should the output be limited or sorted? **Sort by reserve in descending order**

```
select item_name, item_reserve, min(bid_amount)
    max(bid_amount) as max_bid, item_soldamount
    from vb_items
        join vb_bids on item_id=bid_item_id
    where bid_status = 'ok'
    group by item_name, item_reserve, item_solda
    order by item_reserve desc
```

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:

- Figure out the tables you will need. **vb_users, vb_bids**

```
select * from vb_users
select * from vb_bids
```

- Figure out how those tables should be joined. **Left join from users to bids.** This way it includes users with no bids.

```
select *
    from vb_users s
        left join vb_bids b
            on b.bid_user_id = s.u
```

- Which rows should be filtered? **Only valid bids**

```
select *
    from vb_users s
        left join vb_bids b
            on b.bid_user_id = s.u
    where b.bid_status = 'ok'
```

- 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.

```
select s.user_email, s.user_firstname, s.user_lastname, count(*) a
    from vb_users s
        left join vb_bids b
            on b.bid_user_id = s.user_id
    where b.bid_status = 'ok'
    group by s.user_email, s.user_firstname, s.user_lastname
```

- Do these groupings need to be filtered? **Filtered, no. Further categorized, yes.**
- 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.**

```
select s.user_email, s.user_firstname, s.user_lastname, count(*) a
    case when count(*) between 0 and 1 then 'Low'
        when count(*) between 2 and 4 then 'Moderate'
        else 'High' end as user_bid_activity
    from vb_users s
        left join vb_bids b
            on b.bid_user_id = s.user_id
    where b.bid_status = 'ok'
    group by s.user_email, s.user_firstname, s.user_lastname
```

- How should the output be limited or sorted? **No need**.

Step 2: Produce activity report from that.

- 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:**

```
with user_bids as (
    select s.user_email, s.user_firstname, s.user_lastname, count(*)
        case when count(*) between 0 and 1 then 'Low'
            when count(*) between 2 and 4 then 'Moderate'
            else 'High' end as user_bid_activity
        from vb_users s
            left join vb_bids b
                on b.bid_user_id = s.user_id
        where b.bid_status = 'ok'
        group by s.user_email, s.user_firstname, s.user_lastname
)
select * from user_bids
```

- Figure out how those tables should be joined. **No joins.**
- Which rows should be filtered? **No filters**
- Are there any groupings? **Yes, group by user_bid_activity and count rows.**

```sql
with user_bids as (
    select s.user_email, s.user_firstname, s.user_lastname, count(*)
        case when count(*) between 0 and 1 then 'Low'
            when count(*) between 2 and 4 then 'Moderate'
            else 'High' end as user_bid_activity
        from vb_users s
            left join vb_bids b
                on b.bid_user_id = s.user_id
        where b.bid_status = 'ok'
        group by s.user_email, s.user_firstname, s.user_lastname
)
select user_bid_activity, count(*) as user_count
    from user_bids
    group by user_bid_activity
```

- Do those grouping need to be filtered? **No**
- Which columns should be projected? **Same**
- How should the output be limited or sorted? **Let's sort by user_count.**

```sql
with user_bids as (
    select s.user_email, s.user_firstname, s.user_lastname, count(*)
        case when count(*) between 0 and 1 then 'Low'
            when count(*) between 2 and 4 then 'Moderate'
            else 'High' end as user_bid_activity
        from vb_users s
            left join vb_bids b
                on b.bid_user_id = s.user_id
        where b.bid_status = 'ok'
        group by s.user_email, s.user_firstname, s.user_lastname
)
select user_bid_activity, count(*) as user_count
    from user_bids
    group by user_bid_activity
    order by user_count
```

- 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_sol
    from vb_items
    where item_type='Collectables'
    order by item_name
```

Results Messages

	Item_name	item_type	item_reserve	ite
1	Alf Alarm Clock	Collectables	5.0000	NU
2	Autographed Mik Jagger Poster	Collectables	75.0000	10
3	Carlos Villalba BobbleHead	Collectables	49.9500	NU
4	Dukes Of Hazard ashtray	Collectables	149.9900	NU
5	Farrah Fawcet poster	Collectables	50.0000	NU
6	Joe Montanna Figurine	Collectables	200.0000	NU
7	Kleenex used by Dr. Dre	Collectables	500.0000	NU
8	Mike Fudge BobbleHead	Collectables	49.9500	NU
9	PacMan Fever lunchbox	Collectables	29.9900	NU
10	Pez dispensers	Collectables	10.0000	11
11	Rare Mint Snow Globe	Collectables	30.5000	40
12	Shatner's old Toupee	Collectables	199.9900	NU
13	Smurf TV Tray	Collectables	25.0000	26
14	Some Beanie Babies, New with…	Collectables	99.9900	NU

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.

- 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.

```sql
SELECT COUNT(DISTINCT item_type) AS [Number of Item Types]
FROM vb_items;


SELECT item_type,
COUNT(item_name) AS [Number of Items],
min(item_reserve) AS [Min Reserve Price],
max(item_reserve) AS [Max Reserve Price],
avg(item_reserve) AS [Avg Reserve Price]
FROM vb_items
GROUP BY item_type
ORDER BY item_type;
```

*Untitled - ...

File Edit Format V

pwalsh03

	Number of Item Types
1	8

File Edit Format View Help

pwalsh03

Windows (CRLF) UTF-8

	item_type	Number of Items	Min Reserve Price	Max Reserve Price	Avg Reserve Price
1	All Other	4	0.99	10000000.00	2500004.86
2	Antiques	6	9.00	250.00	81.5833
3	Books	3	4.50	10.99	8.48
4	Collectables	14	5.00	500.00	105.3828
5	Electronics	1	15.00	15.00	15.00
6	Jewelry	2	6.95	599.99	303.47
7	Sporting Goods	2	12.50	12.75	12.625
8	Tickets	2	5.00	750.00	377.50

- 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.

```sql
WITH item_counts AS (
SELECT  item_type,
        min(item_reserve) AS [Min Reserve Price],
        max(item_reserve) AS [Max Reserve Price],
        avg(item_reserve) AS [Avg Reserve Price]
FROM vb_items
WHERE item_type = 'Antiques' OR item_type = 'Collectables'
GROUP BY item_type)

SELECT v.item_name,
        v.item_type,
        v.item_reserve,
        i.[Min Reserve Price],
        i.[Max Reserve Price],
        i.[Avg Reserve Price]
FROM vb_items v
JOIN item_counts i ON v.item_type=i.item_type
ORDER BY v.item_name;
```

*Untitled - ..
File Edit Form
pwalsh03

	item_name	item_type	item_reserve	Min Reserve Price	Max Reserve Price	Avg Reserve Price
1	a Toaster	Antiques	20.00	9.00	250.00	81.5833
2	Alf Alarm Clock	Collectables	5.00	5.00	500.00	105.3828
3	Antique Desk	Antiques	250.00	9.00	250.00	81.5833
4	Autographed Mik Jagger Poster	Collectables	75.00	5.00	500.00	105.3828
5	Brass French Press	Antiques	45.50	9.00	250.00	81.5833
6	Carlos Villalba BobbleHead	Collectables	49.95	5.00	500.00	105.3828
7	case of vintage tube socks	Antiques	9.00	9.00	250.00	81.5833
8	Dukes Of Hazard ashtray	Collectables	149.99	5.00	500.00	105.3828
9	Dusty Vase	Antiques	100.00	9.00	250.00	81.5833
10	Farrah Fawcet poster	Collectables	50.00	5.00	500.00	105.3828
11	Joe Montanna Figurine	Collectables	200.00	5.00	500.00	105.3828
12	Kleenex used by Dr. Dre	Collectables	500.00	5.00	500.00	105.3828
13	Mike Fudge BobbleHead	Collectables	49.95	5.00	500.00	105.3828
14	Original Coke Bottle from 19…	Antiques	65.00	9.00	250.00	81.5833
15	PacMan Fever lunchbox	Collectables	29.99	5.00	500.00	105.3828
16	Pez dispensers	Collectables	10.00	5.00	500.00	105.3828
17	Rare Mint Snow Globe	Collectables	30.50	5.00	500.00	105.3828
18	Shatner's old Toupee	Collectables	199.99	5.00	500.00	105.3828
19	Smurf TV Tray	Collectables	25.00	5.00	500.00	105.3828
20	Some Beanie Babies, New with…	Collectables	99.99	5.00	500.00	105.3828

*Untitled
File Edit For
pwalsh03
Windows (CR

- 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.

```sql
-- ratings and an average rating of 4.75.
SELECT user_firstname,
        user_lastname,
        COUNT(rating_for_user_id) AS [Number of Ratings],
        avg(CAST(rating_value AS DECIMAL(3,2))) AS [Avg Seller Rating]
FROM vb_users
JOIN vb_user_ratings ON user_id=rating_for_user_id
WHERE rating_astype='Seller'
GROUP BY user_firstname, user_lastname
ORDER BY user_firstname;
```

*Untitled - ...
File Edit Forma
pwalsh03

	user_firstname	user_lastname	Number of Ratings	Avg Seller Rating
1	Abby	Kuss	3	4.333333
2	Anita	Job	1	3.000000
3	Barb	Barion	2	3.500000
4	Carrie	Dababbi	4	4.750000
5	Les	Ismoore	2	2.500000
6	Martin	Eyezing	2	2.500000
7	Rose	Abov-Duresst	3	1.000000
8	Ty	Anott	2	2.500000
9	Victor	Rhee	1	4.000000

*Untitled -
File Edit Forr
pwalsh03
Windows (CRL

- 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.

```sql
-- name of the item and the number of bids, making sure the item wi
SELECT item_name, COUNT(b.bid_item_id) AS [Number of bids]
FROM vb_items AS i
INNER JOIN vb_bids AS b ON i.item_id=b.bid_item_id
WHERE item_type = 'Collectables'
GROUP BY item_name
HAVING COUNT(*) > 1
ORDER BY [Number of bids] DESC;
```

*Untitled
File Edit Fo
pwalsh03

Results Messages

	item_name	Number of bids
1	Dukes Of Hazard ashtray	9
2	Autographed Mik Jagger Poster	6
3	Shatner's old Toupee	5
4	Rare Mint Snow Globe	3
5	Farrah Fawcet poster	3
6	Pez dispensers	2

*Untitled -
File Edit Form
pwalsh03
Windows (CRLF

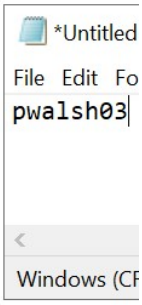
- Generate a valid bidding history for any given item of your choice. Display the item ID, item name, a number representing the order the bid was placed, the bid amount, and the bidder's name. Here's an example showing the first three bids on item 11:

item_id	item_name	bid_order	bid_amount	bidder
11	Dukes Of Hazard ashtray	1	150.0000	Dan Del
11	Dukes Of Hazard ashtray	2	175.0000	Al Fres
11	Dukes Of Hazard ashtray	3	200.0000	Carrie

```sql
WITH bidders AS (
    SELECT user_id, user_firstname + ' ' + user_lastname AS bidder
    FROM vb_users
)

SELECT i.item_id,
        i.item_name,
        -- b.bid_id,
        RANK() OVER (
            PARTITION BY item_name
            ORDER BY b.bid_amount) AS bid_order,
        b.bid_amount,
        bidder
FROM vb_items AS i
INNER JOIN vb_bids AS b ON i.item_id=b.bid_item_id
INNER JOIN bidders ON b.bid_user_id=bidders.user_id
WHERE bid_status = 'ok';
```

	item_id	item_name	bid_order	bid_amount	bidder
1	5	Alf Alarm Clock	1	5.01	Les Ismoore
2	18	Antique Desk	1	251.00	Ray Ovlight
3	18	Antique Desk	2	252.00	Barb Barion
4	18	Antique Desk	3	253.00	Ray Ovlight
5	18	Antique Desk	4	254.00	Barb Barion
6	18	Antique Desk	5	255.00	Ray Ovlight
7	36	Autographed Mik Jagger Poster	1	80.00	Abby Kuss
8	36	Autographed Mik Jagger Poster	2	85.00	Anne Dewey
9	36	Autographed Mik Jagger Poster	3	90.00	Abby Kuss
10	36	Autographed Mik Jagger Poster	4	95.00	Anne Dewey
11	36	Autographed Mik Jagger Poster	5	100.00	Abby Kuss
12	13	case of vintage tube socks	1	9.01	Anne Dewey
13	13	case of vintage tube socks	2	9.02	Victor Rhee
14	34	Client/Server Survival Guide	1	11.00	Al Fresco
15	11	Dukes Of Hazard ashtray	1	150.00	Dan Delyons



- 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.

item_name	bid_order	bid_amount	prev_bidder	bidder	nex
Dukes Of Hazard ashtray	1	150.0000	*NULL*	Dan Delyons	Al
Dukes Of Hazard ashtray	2	175.0000	Dan Delyons	Al Fresco	Ca
Dukes Of Hazard ashtray	3	200.0000	Al Fresco	Carrie Dababbi	Gu

```sql
WITH bidders AS (
    SELECT user_id, user_firstname + ' ' + user_lastname AS bidder
    FROM vb_users
)

SELECT i.item_id,
        i.item_name,
        -- b.bid_id,
        RANK() OVER (
            PARTITION BY item_name
            ORDER BY b.bid_amount) AS bid_order,
        b.bid_amount,
        bidder,
        LAG(bidder) OVER (
            PARTITION BY item_name
            ORDER BY b.bid_amount) AS previous_bidder,
        LEAD(bidder) OVER (
            PARTITION BY item_name
            ORDER BY b.bid_amount) AS next_bidder
FROM vb_items AS i
INNER JOIN vb_bids AS b ON i.item_id=b.bid_item_id
INNER JOIN bidders ON b.bid_user_id=bidders.user_id
WHERE bid_status = 'ok';
```

*Untitled - ...
File Edit Forma
pwalsh03
Windows (CRLF)

	item_id	item_name	bid_order	bid_amount	bidder	previous_bidder	next_bidder
16	11	Dukes Of Hazar…	2	175.00	Al Fresco	Dan Delyons	Carrie Dababbi
17	11	Dukes Of Hazar…	3	200.00	Carrie Dab…	Al Fresco	Gus Toffwind
18	11	Dukes Of Hazar…	4	225.00	Gus Toffwi…	Carrie Dababbi	Isabelle Gunne…
19	11	Dukes Of Hazar…	5	250.00	Isabelle G…	Gus Toffwind	Dan Delyons
20	11	Dukes Of Hazar…	6	275.00	Dan Delyons	Isabelle Gunnering	Carrie Dababbi
21	11	Dukes Of Hazar…	7	300.00	Carrie Dab…	Dan Delyons	Isabelle Gunne…
22	11	Dukes Of Hazar…	8	325.00	Isabelle G…	Carrie Dababbi	NULL
23	23	Dusty Vase	1	101.00	Oliver Stu…	NULL	Hank Erchief
24	23	Dusty Vase	2	102.00	Hank Erchi…	Oliver Stuffismis…	Jean Poole
25	23	Dusty Vase	3	103.00	Jean Poole	Hank Erchief	Oliver Stuffis…
26	23	Dusty Vase	4	104.00	Oliver Stu…	Jean Poole	Hank Erchief
27	23	Dusty Vase	5	105.00	Hank Erchi…	Oliver Stuffismis…	Jean Poole
28	23	Dusty Vase	6	106.00	Jean Poole	Hank Erchief	NULL
29	15	Farrah Fawcet …	1	505.00	Ray Ovlight	NULL	Victor Rhee
30	15	Farrah Fawcet …	2	510.00	Victor Rhee	Ray Ovlight	Ray Ovlight
31	15	Farrah Fawcet …	3	515.00	Ray Ovlight	Victor Rhee	NULL

*Untitled
File Edit Fo
pwalsh03
Windows (CF

- 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.

```sql
-- than the overall average rating) to either buyers or sellers (no ne
-- differentiate whether the user rated a buyer or seller), and includ
-- users who have submitted more than one rating.
SELECT u.user_id,
        u.user_firstname + ' ' + u.user_lastname AS username,
        u.user_email,
        r.rating_value,
        COUNT(*) OVER (PARTITION BY r.rating_by_user_id) AS occurs
FROM vb_users u
JOIN vb_user_ratings r ON u.user_id=r.rating_by_user_id
WHERE rating_value < (SELECT avg(CAST(rating_value AS DECIMAL)) AS avg_rating FROM vb_user_ratings)
ORDER BY occurs DESC
```

File Edit Format View Help
pwalsh03

10 Windows (CRLF) UTF

	user_id	username	user_email	rating_value	occurs
1	19	Oliver Stuffismission	ostuff@mail.org	1	2
2	19	Oliver Stuffismission	ostuff@mail.org	1	2

*Untitled
File Edit For
pwalsh03

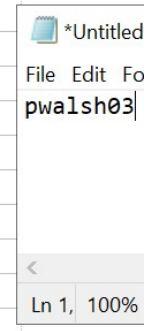
- Produce a report of the KPI (key performance indicator) user bids per item. Show the user's name and email, total number of valid bids, total count of items bid upon, and then the ratio of bids to items. As a check, Anne Dewey's bids per item ratio is 1.666666.

```sql
WITH bids_per_item AS (
    SELECT u.user_firstname + ' ' + u.user_lastname AS bidder_name,
            u.user_email,
            COUNT(DISTINCT b.bid_item_id) AS item_count,
            COUNT(*) AS total_bids
    FROM vb_bids AS b
    JOIN vb_users AS u ON b.bid_user_id = u.user_id
    WHERE b.bid_status = 'ok'
    GROUP BY
        u.user_firstname,
        u.user_lastname,
        u.user_email
)
SELECT bidder_name,
        user_email,
        total_bids,
        item_count,
        CAST(total_bids AS DECIMAL) / CAST(item_count AS DECIMAL) AS bids_per_item_ratio
FROM bids_per_item;
```

*Untitled - Note
File Edit Format
pwalsh03

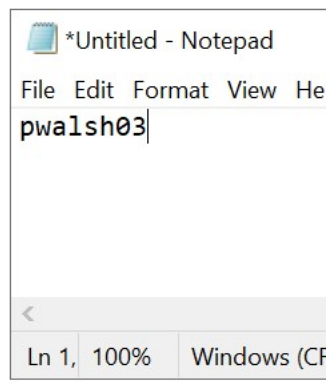
Ln 1, 100% Wir

	bidder_name	user_email	total_bids	item_count	bids_per_item_ratio
1	Abby Kuss	abuss@mail.org	3	1	3.0000000000000000000
2	Anne Dewey	adwewy@mail.org	5	3	1.6666666666666666666
3	Al Fresco	afresco@mail.org	3	3	1.0000000000000000000
4	Barb Barion	bbarion@mail.org	3	2	1.5000000000000000000
5	Barry DeHatchett	bdehatchett@mail.org	5	1	5.0000000000000000000
6	Bo Enarreau	benarreau@mail.org	2	2	1.0000000000000000000
7	Carrie Dababbi	cdababbi@mail.org	2	1	2.0000000000000000000
8	Dan Delyons	ddeloyons@mail.org	4	2	2.0000000000000000000
9	Gus Toffwind	gtoffwind@mail.org	2	2	1.0000000000000000000
10	Hank Erchief	herchief@mail.org	2	1	2.0000000000000000000
11	Isabelle Gunner…	igunner@mail.org	7	2	3.5000000000000000000
12	Jean Poole	jpoole@mail.org	3	2	1.5000000000000000000
13	Les Ismoore	lismoore@mail.org	3	3	1.0000000000000000000



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

```sql
SELECT
    i.item_id,
    i.item_name,
    b.bid_amount AS highest_bid,
    u.user_firstname AS bidder_firstname,
    u.user_lastname AS bidder_lastname
FROM
    vb_items i
LEFT JOIN (
    SELECT
        bid_item_id,
        MAX(bid_amount) AS bid_amount,
        bid_user_id
    FROM
        vb_bids
    WHERE
        bid_status = 'ok'
    GROUP BY
        bid_item_id, bid_user_id
) b ON i.item_id = b.bid_item_id
LEFT JOIN vb_users u ON b.bid_user_id = u.user_id
WHERE
    i.item_sold = 0;
```



	item_id	item_name	highest_bid	bidder_firstname	bidder_lastname
1	4	Pet Rock	NULL	NULL	NULL
2	5	Alf Alarm Clock	5.01	Les	Ismoore
3	6	Shatner's old Toupee	201.00	Jean	Poole
4	6	Shatner's old Toupee	202.00	Dan	Delyons
5	7	Slightly-damaged Golf Bag	14.00	Rose	Abov-Duresst
6	7	Slightly-damaged Golf Bag	14.50	Seymour	Ofewe
7	8	Some Beanie Babies, New …	250.00	Les	Ismoore
8	9	Tchotchkes	NULL	NULL	NULL
9	10	Your Watch, Please?	NULL	NULL	NULL
10	11	Dukes Of Hazard ashtray	225.00	Gus	Toffwind
11	11	Dukes Of Hazard ashtray	325.00	Isabelle	Gunnering
12	11	Dukes Of Hazard ashtray	275.00	Dan	Delyons
13	11	Dukes Of Hazard ashtray	175.00	Al	Fresco
14	11	Dukes Of Hazard ashtray	300.00	Carrie	Dababbi
15	12	PacMan Fever lunchbox	NULL	NULL	NULL
16	13	case of vintage tube soc…	9.01	Anne	Dewey
17	13	case of vintage tube soc	9.02	Victor	Rhee

*Untitled

File Edit Fo

pwalsh03

Ln 1, 100%

- 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.

```
WITH overall_avg AS (
  SELECT avg(CAST(rating_value AS DECIMAL(3,2))) AS overall_avg_rating
  FROM vb_user_ratings
  WHERE rating_astype='Seller'
)
SELECT user_firstname,
    user_lastname,
    COUNT(rating_for_user_id) AS [Number of Ratings],
    avg(CAST(rating_value AS DECIMAL(3,2))) AS [Avg Seller Rating],
    overall_avg.overall_avg_rating AS [Overall Avg Seller Rating],
    avg(CAST(rating_value AS DECIMAL(3,2))) - overall_avg.overall_avg_rating AS [Rating Difference]
FROM vb_users
  JOIN vb_user_ratings ON user_id=rating_for_user_id
  CROSS JOIN overall_avg
WHERE
  rating_astype='Seller'
GROUP BY
  user_firstname,
  user_lastname,
  overall_avg.overall_avg_rating
ORDER BY user_firstname;
```

*Untitled - Notepad

File Edit Format View Help

pwalsh03

	user_firstname	user_lastname	Number of Ratings	Avg Seller Rating	Overall Avg Seller Rating	Rating Difference
1	Abby	Kuss	3	4.333333	3.200000	1.133333
2	Anita	Job	1	3.000000	3.200000	-0.200000
3	Barb	Barion	2	3.500000	3.200000	0.300000
4	Carrie	Dababbi	4	4.750000	3.200000	1.550000
5	Les	Ismoore	2	2.500000	3.200000	-0.700000
6	Martin	Eyezing	2	2.500000	3.200000	-0.700000
7	Rose	Abov-Duresst	3	1.000000	3.200000	-2.200000
8	Ty	Anott	2	2.500000	3.200000	-0.700000
9	Victor	Rhee	1	4.000000	3.200000	0.800000