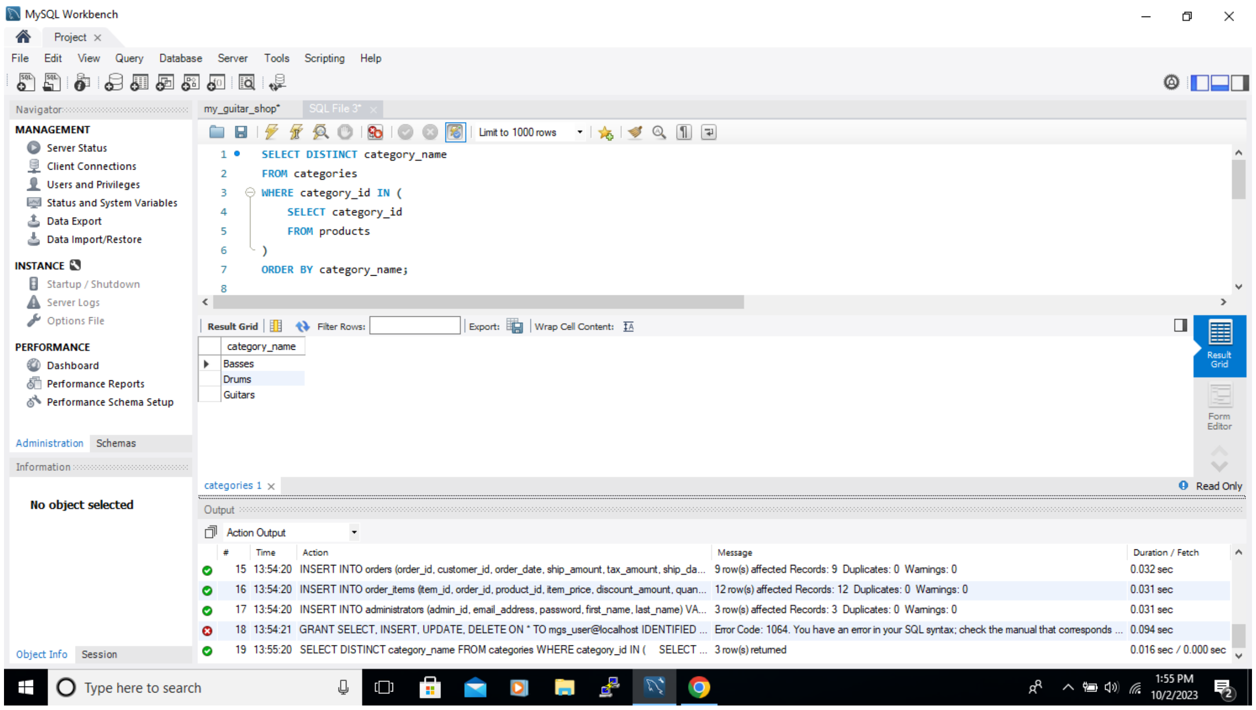
Lab 7: How to Code Subqueries

Step 1: Write a SELECT statement that returns the same result set as this SELECT statement, but don't use a join. Instead, use a subquery in a WHERE clause that uses the IN keyword. SELECT DISTINCT category_name FROM categories c JOIN products p ON c.category_id = p.category_id ORDER BY category_name

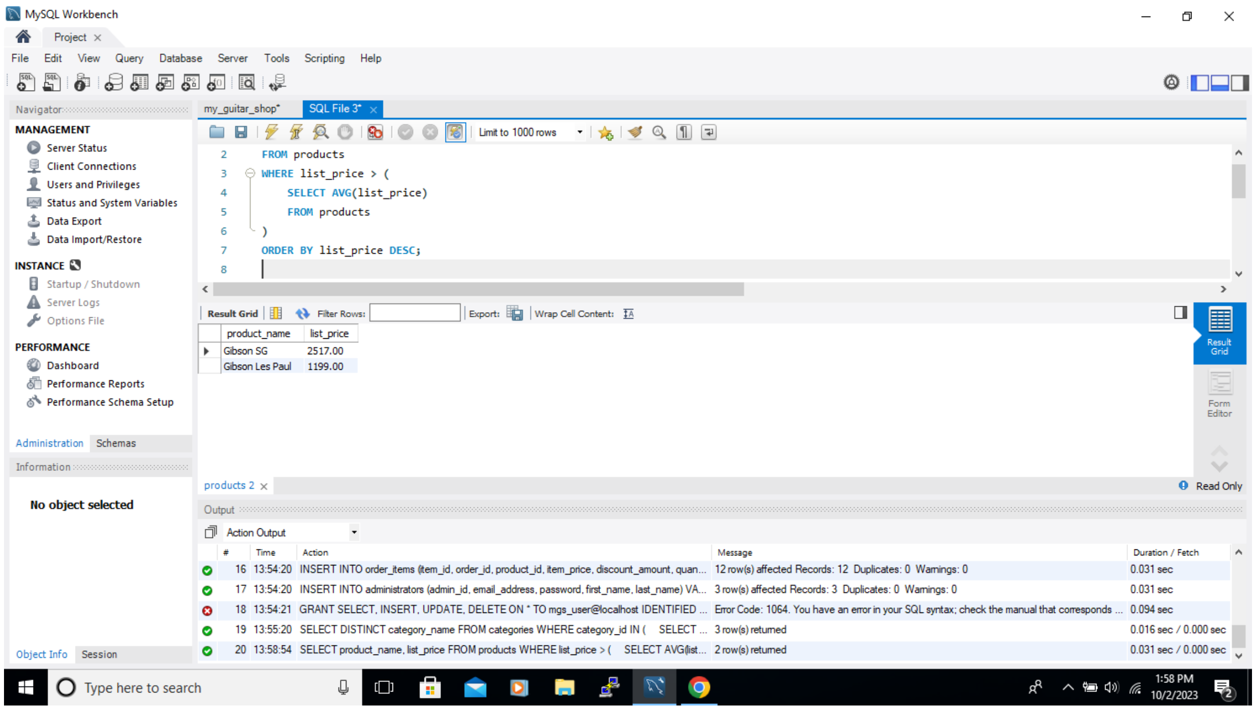


```
SELECT DISTINCT category_name
FROM categories
WHERE category_id IN (
    SELECT category_id
    FROM products
)
ORDER BY category_name;
```

category_name
Basses
Drums
Guitars

```
#   Time      Action                                                                            Message                                                                         Duration / Fetch
15  13:54:20  INSERT INTO orders (order_id, customer_id, order_date, ship_amount, tax_amount, ship_da...   9 row(s) affected Records: 9  Duplicates: 0  Warnings: 0                        0.032 sec
16  13:54:20  INSERT INTO order_items (item_id, order_id, product_id, item_price, discount_amount, quan...  12 row(s) affected Records: 12  Duplicates: 0  Warnings: 0                      0.031 sec
17  13:54:20  INSERT INTO administrators (admin_id, email_address, password, first_name, last_name) VA...  3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0                        0.031 sec
18  13:54:21  GRANT SELECT, INSERT, UPDATE, DELETE ON * TO mgs_user@localhost IDENTIFIED ...           Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds ...  0.094 sec
19  13:55:20  SELECT DISTINCT category_name FROM categories WHERE category_id IN (     SELECT ...      3 row(s) returned                                                               0.016 sec / 0.000 sec
```

Step 2: Write a SELECT statement that answers this question: Which products have a list price that's greater than the average list price for all products? Return the product_name and list_price columns for each product. Sort the result set by the list_price column in descending sequence.

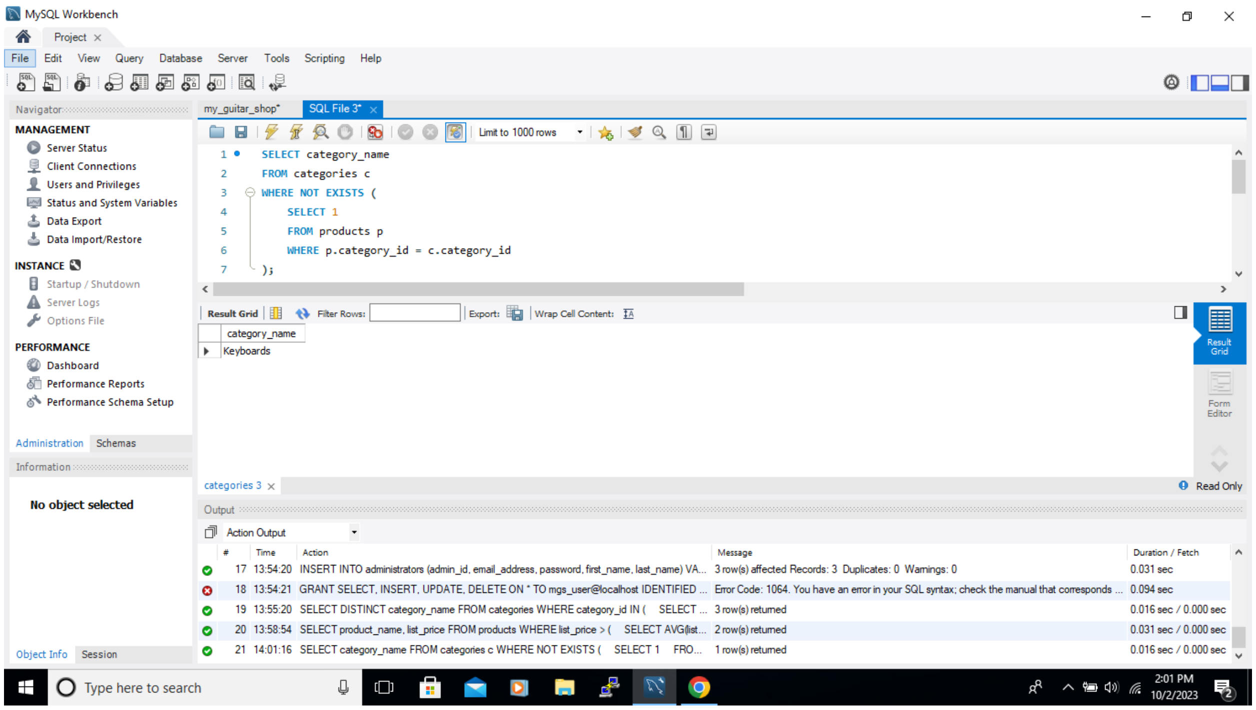


```
FROM products
WHERE list_price > (
    SELECT AVG(list_price)
    FROM products
)
ORDER BY list_price DESC;
```

product_name	list_price
Gibson SG	2517.00
Gibson Les Paul	1199.00

#	Time	Action	Message	Duration / Fetch
16	13:54:20	INSERT INTO order_items (item_id, order_id, product_id, item_price, discount_amount, quan...	12 row(s) affected Records: 12 Duplicates: 0 Warnings: 0	0.031 sec
17	13:54:20	INSERT INTO administrators (admin_id, email_address, password, first_name, last_name) VA...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.031 sec
18	13:54:21	GRANT SELECT, INSERT, UPDATE, DELETE ON * TO mgs_user@localhost IDENTIFIED ...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds ...	0.094 sec
19	13:55:20	SELECT DISTINCT category_name FROM categories WHERE category_id IN (SELECT ...	3 row(s) returned	0.016 sec / 0.000 sec
20	13:58:54	SELECT product_name, list_price FROM products WHERE list_price > (SELECT AVG(list...	2 row(s) returned	0.031 sec / 0.000 sec

Step 3: Write a SELECT statement that returns the category_name column from the Categories table. Return one row for each category that has never been assigned to any product in the Products table. To do that, use a subquery introduced with the NOT EXISTS operator.

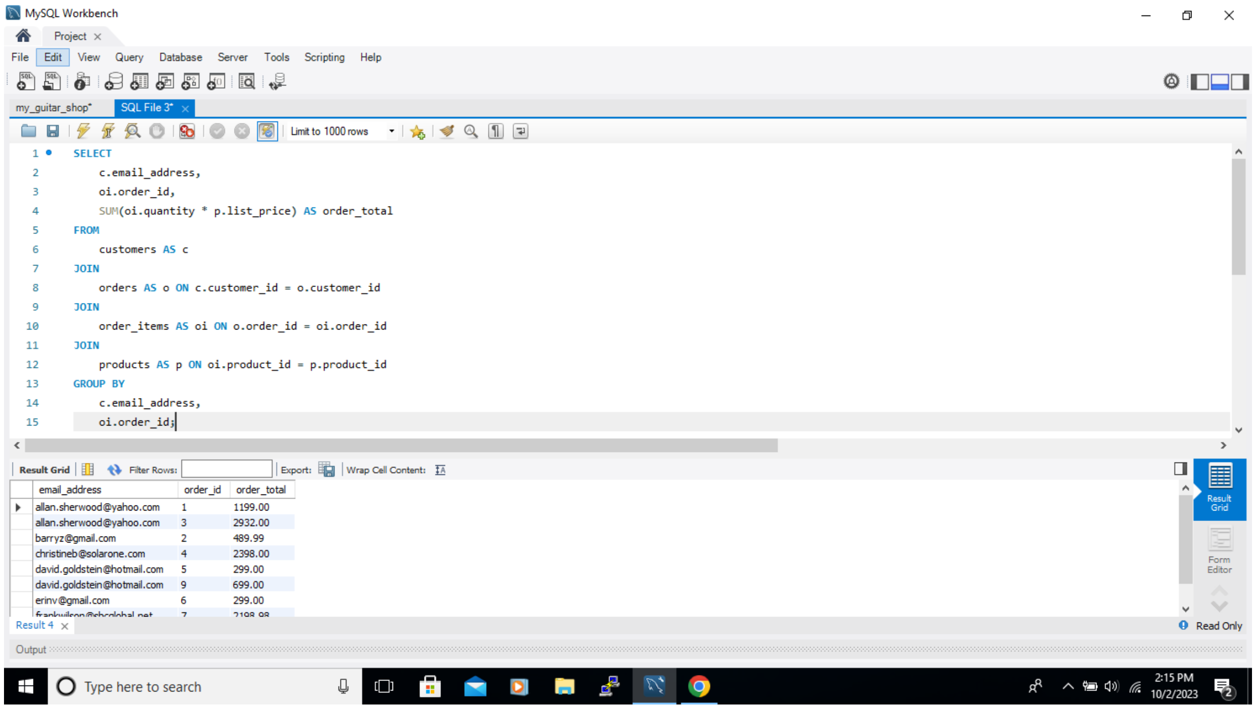


```
SELECT category_name
FROM categories c
WHERE NOT EXISTS (
    SELECT 1
    FROM products p
    WHERE p.category_id = c.category_id
);
```

```
category_name
Keyboards
```

```
#   Time      Action                                                                        Message                                                                   Duration / Fetch
17  13:54:20  INSERT INTO administrators (admin_id, email_address, password, first_name, last_name) VA...  3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0        0.031 sec
18  13:54:21  GRANT SELECT, INSERT, UPDATE, DELETE ON * TO mgs_user@localhost IDENTIFIED ...  Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds ...  0.094 sec
19  13:55:20  SELECT DISTINCT category_name FROM categories WHERE category_id IN (    SELECT ...  3 row(s) returned  0.016 sec / 0.000 sec
20  13:58:54  SELECT product_name, list_price FROM products WHERE list_price > (    SELECT AVG(list...  2 row(s) returned  0.031 sec / 0.000 sec
21  14:01:16  SELECT category_name FROM categories c WHERE NOT EXISTS (    SELECT 1    FRO...  1 row(s) returned  0.016 sec / 0.000 sec
```

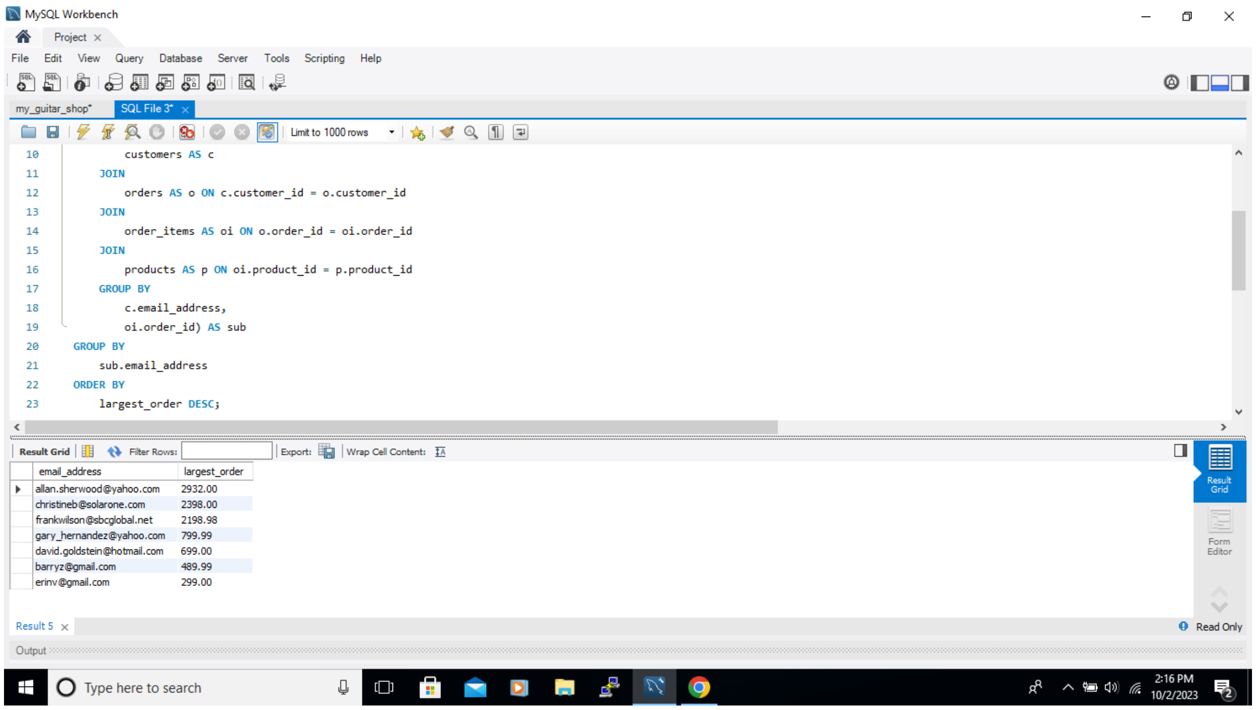
Step 4: Write a SELECT statement that returns three columns: email_address, order_id, and the order total for each customer. To do this, you can group the result set by the email_address and order_id columns. In addition, you must calculate the order total from the columns in the Order_Items table.



```sql
SELECT
    c.email_address,
    oi.order_id,
    SUM(oi.quantity * p.list_price) AS order_total
FROM
    customers AS c
JOIN
    orders AS o ON c.customer_id = o.customer_id
JOIN
    order_items AS oi ON o.order_id = oi.order_id
JOIN
    products AS p ON oi.product_id = p.product_id
GROUP BY
    c.email_address,
    oi.order_id;
```

email_address	order_id	order_total
allan.sherwood@yahoo.com	1	1199.00
allan.sherwood@yahoo.com	3	2932.00
barryz@gmail.com	2	489.99
christineb@solarone.com	4	2398.00
david.goldstein@hotmail.com	5	299.00
david.goldstein@hotmail.com	9	699.00
erinv@gmail.com	6	299.00
frankwilson@sbcglobal.net	7	2198.98

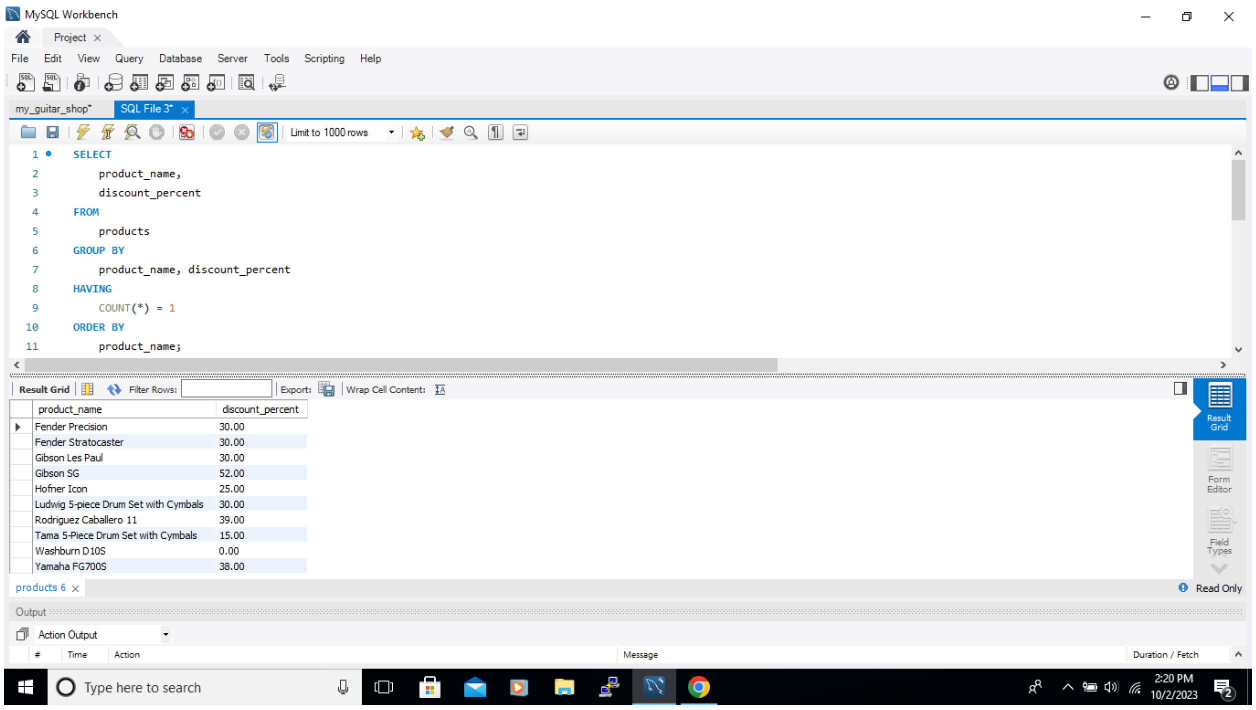
Write a second SELECT statement that uses the first SELECT statement in its FROM clause. The main query should return two columns: the customer's email address and the largest order for that customer. To do this, you can group the result set by the email_address. Sort the result set by the largest order in descending sequence.



```
            customers AS c
        JOIN
            orders AS o ON c.customer_id = o.customer_id
        JOIN
            order_items AS oi ON o.order_id = oi.order_id
        JOIN
            products AS p ON oi.product_id = p.product_id
        GROUP BY
            c.email_address,
            oi.order_id) AS sub
    GROUP BY
        sub.email_address
    ORDER BY
        largest_order DESC;
```

email_address	largest_order
allan.sherwood@yahoo.com	2932.00
christineb@solarone.com	2398.00
frankwilson@sbcglobal.net	2198.98
gary_hernandez@yahoo.com	799.99
david.goldstein@hotmail.com	699.00
barryz@gmail.com	489.99
erinv@gmail.com	299.00

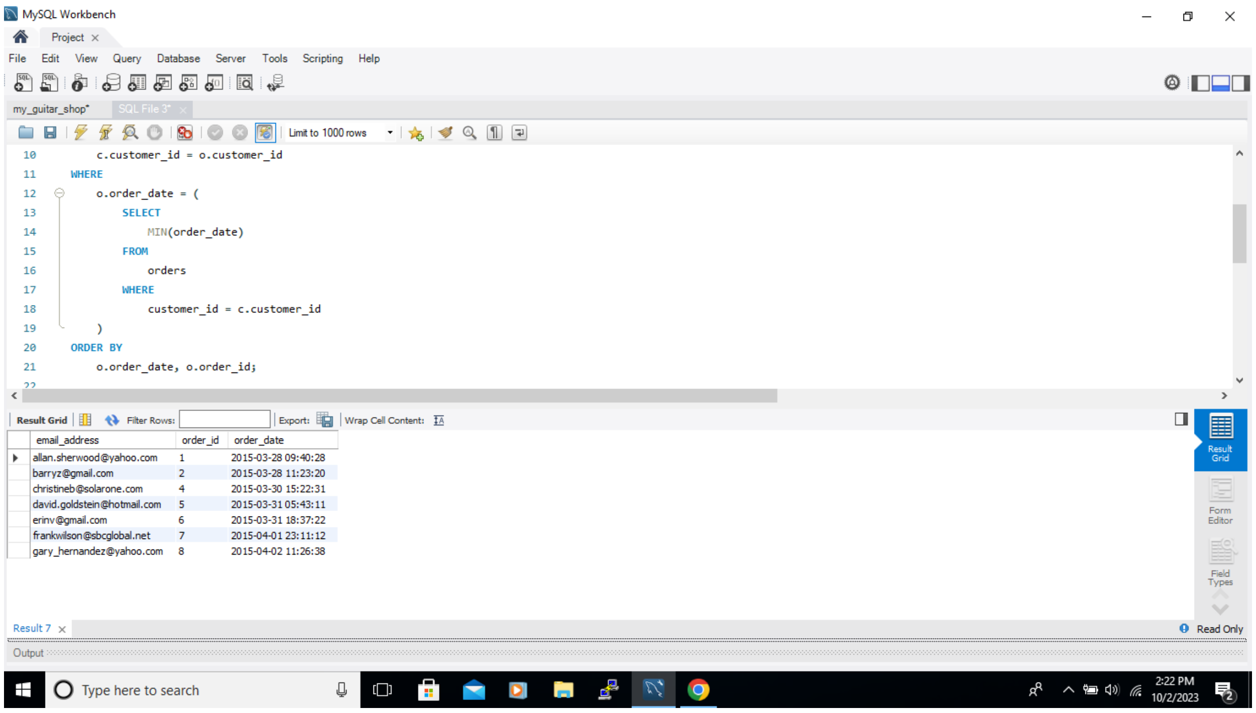
Step 5: Write a SELECT statement that returns the name and discount percent of each product that has a unique discount percent. In other words, don't include products that have the same discount percent as another product. Sort the result set by the product_name column.



```
SELECT
    product_name,
    discount_percent
FROM
    products
GROUP BY
    product_name, discount_percent
HAVING
    COUNT(*) = 1
ORDER BY
    product_name;
```

product_name	discount_percent
Fender Precision	30.00
Fender Stratocaster	30.00
Gibson Les Paul	30.00
Gibson SG	52.00
Hofner Icon	25.00
Ludwig 5-piece Drum Set with Cymbals	30.00
Rodriguez Caballero 11	39.00
Tama 5-Piece Drum Set with Cymbals	15.00
Washburn D10S	0.00
Yamaha FG700S	38.00

Step 6: Use a correlated subquery to return one row per customer, representing the customer's oldest order (the one with the earliest date). Each row should include these three columns: email_address, order_id, and order_date. Sort the result set by the order_date and order_id columns.



```
    c.customer_id = o.customer_id
WHERE
    o.order_date = (
        SELECT
            MIN(order_date)
        FROM
            orders
        WHERE
            customer_id = c.customer_id
    )
ORDER BY
    o.order_date, o.order_id;
```

email_address	order_id	order_date
allan.sherwood@yahoo.com	1	2015-03-28 09:40:28
barryz@gmail.com	2	2015-03-28 11:23:20
christineb@solarone.com	4	2015-03-30 15:22:31
david.goldstein@hotmail.com	5	2015-03-31 05:43:11
erinv@gmail.com	6	2015-03-31 18:37:22
frankwilson@sbcglobal.net	7	2015-04-01 23:11:12
gary_hernandez@yahoo.com	8	2015-04-02 11:26:38