

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

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
1 SELECT DISTINCT category_name
2 FROM categories
3 WHERE category_id IN (
4     SELECT category_id
5     FROM products
6 )
7 ORDER BY category_name;
```

The Results window shows the output of the query, which is a table with one column, `category_name`, and three rows: `Basses`, `Drums`, and `Guitars`.

The Action Output window shows the execution log, including the following messages:

- 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.

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
2 FROM products
3 WHERE list_price > (
4     SELECT AVG(list_price)
5     FROM products
6 )
7 ORDER BY list_price DESC;
8
```

The Result Grid shows the following data:

product_name	list_price
Gibson SG	2517.00
Gibson Les Paul	1199.00

The Action Output pane shows the following log entries:

#	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.

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

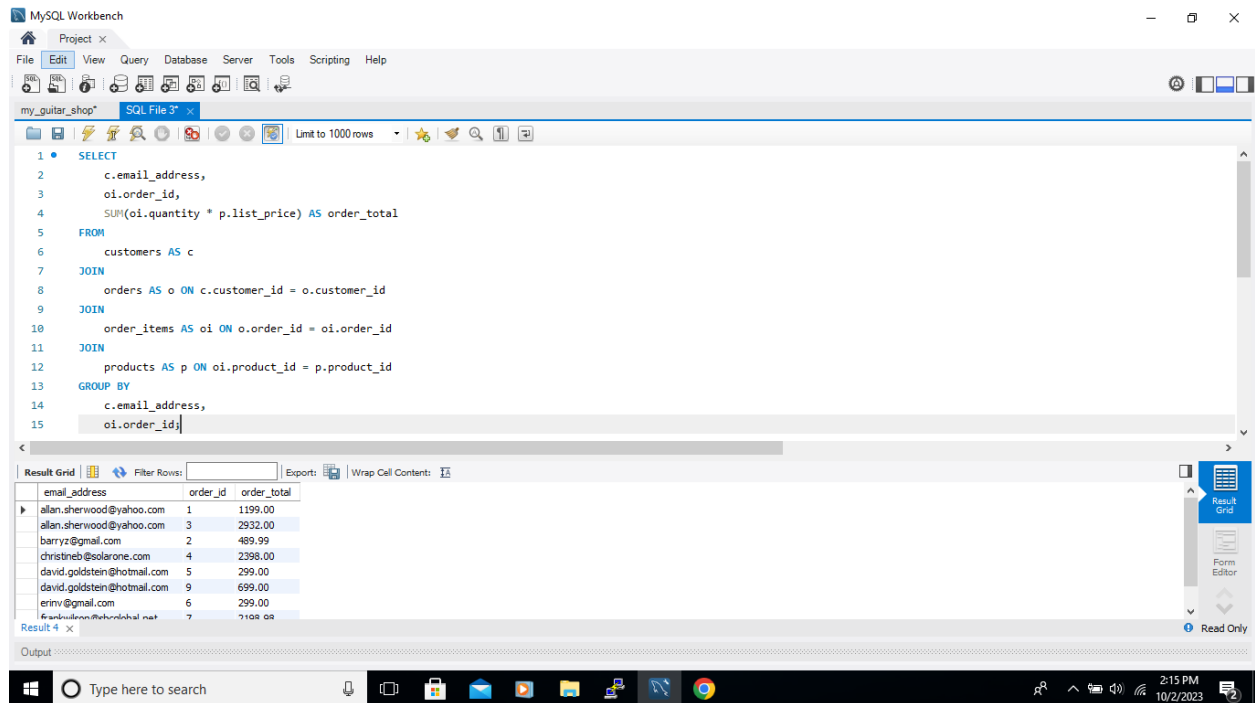
```
1 SELECT category_name
2 FROM categories c
3 WHERE NOT EXISTS (
4     SELECT 1
5     FROM products p
6     WHERE p.category_id = c.category_id
7 );
```

The Results window shows the output of the query, which is a single column named 'category_name' with one row containing the value 'Keyboards'.

The Output window shows the execution log, including the following messages:

#	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.



The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that joins the customers, orders, order_items, and products tables to calculate the order total for each customer. The query is as follows:

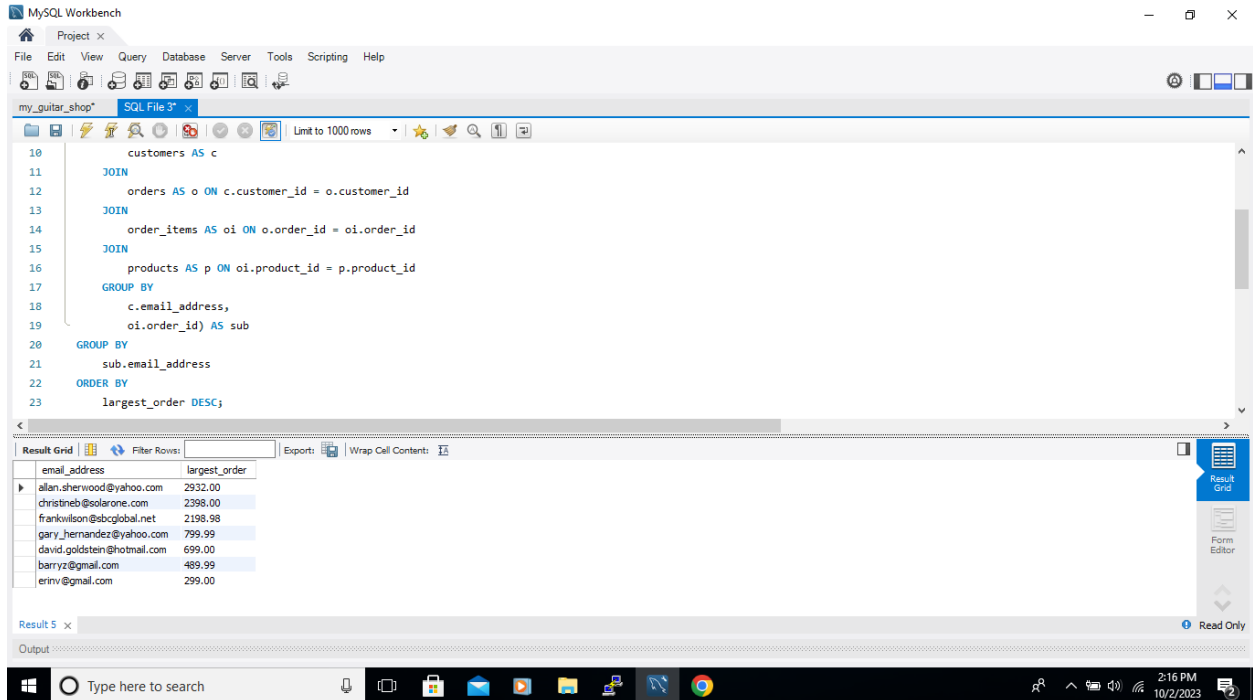
```
1 SELECT
2   c.email_address,
3   oi.order_id,
4   SUM(oi.quantity * p.list_price) AS order_total
5 FROM
6   customers AS c
7 JOIN
8   orders AS o ON c.customer_id = o.customer_id
9 JOIN
10  order_items AS oi ON o.order_id = oi.order_id
11 JOIN
12  products AS p ON oi.product_id = p.product_id
13 GROUP BY
14   c.email_address,
15   oi.order_id
```

The results are displayed in the Result Grid below the query editor. The grid shows 7 rows of data, each representing a customer and their order total.

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
erinn@gmail.com	6	299.00

The bottom of the screenshot shows the Windows taskbar with the Start button, a search bar, and several application icons. The system clock indicates the time is 2:15 PM on 10/2/2023.

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.



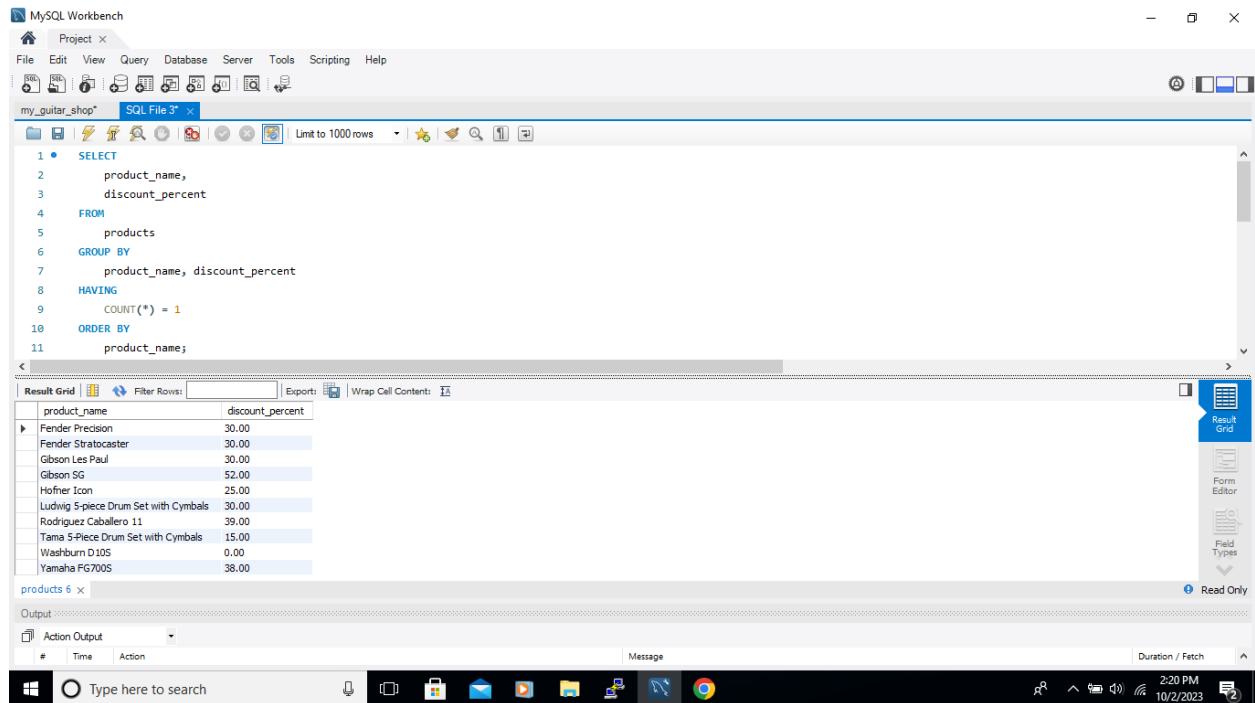
The screenshot shows the MySQL Workbench interface. The SQL Editor at the top contains a query that joins the customers, orders, order_items, and products tables. It groups the results by customer email address and the order ID of the largest order for each customer, then sorts the results by the largest order in descending order.

```
10  customers AS c
11  JOIN
12  orders AS o ON c.customer_id = o.customer_id
13  JOIN
14  order_items AS oi ON o.order_id = oi.order_id
15  JOIN
16  products AS p ON oi.product_id = p.product_id
17  GROUP BY
18  c.email_address,
19  oi.order_id) AS sub
20  GROUP BY
21  sub.email_address
22  ORDER BY
23  largest_order DESC;
```

The Result Grid at the bottom displays the query results, showing the email address and the largest order for each customer.

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.



The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

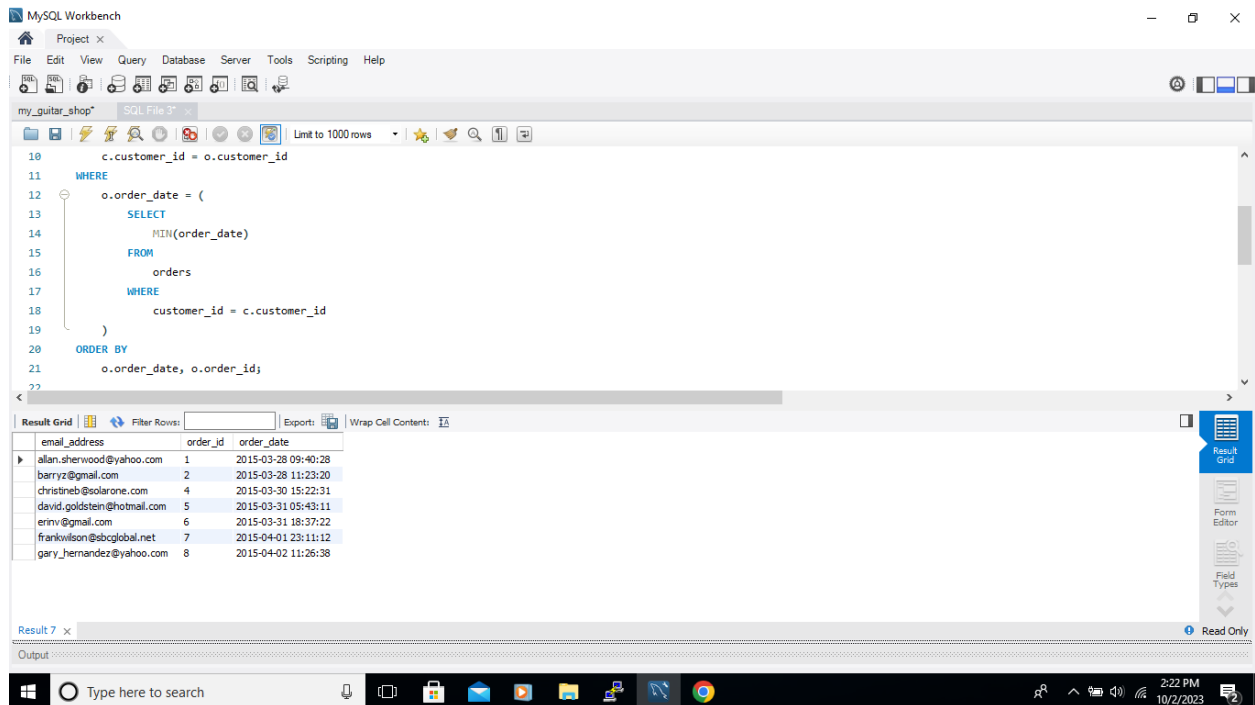
```
1 SELECT
2   product_name,
3   discount_percent
4 FROM
5   products
6 GROUP BY
7   product_name, discount_percent
8 HAVING
9   COUNT(*) = 1
10 ORDER BY
11  product_name;
```

The Result Grid displays the following data:

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

The interface also shows a sidebar with 'Result Grid', 'Form Editor', and 'Field Types' options. The bottom status bar indicates the time is 2:20 PM on 10/2/2023.

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.



The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that uses a correlated subquery to find the oldest order for each customer. The query is as follows:

```
10 c.customer_id = o.customer_id
11 WHERE
12 o.order_date = (
13     SELECT
14         MIN(o.order_date)
15     FROM
16         orders
17     WHERE
18         customer_id = c.customer_id
19 )
20 ORDER BY
21 o.order_date, o.order_id;
```

The results are displayed in the Result Grid below the query editor. The grid shows three columns: email_address, order_id, and order_date. The data is sorted by order_date and 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
christneb@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

The bottom of the screenshot shows the Windows taskbar with the search bar and system clock indicating 2:22 PM on 10/2/2023.