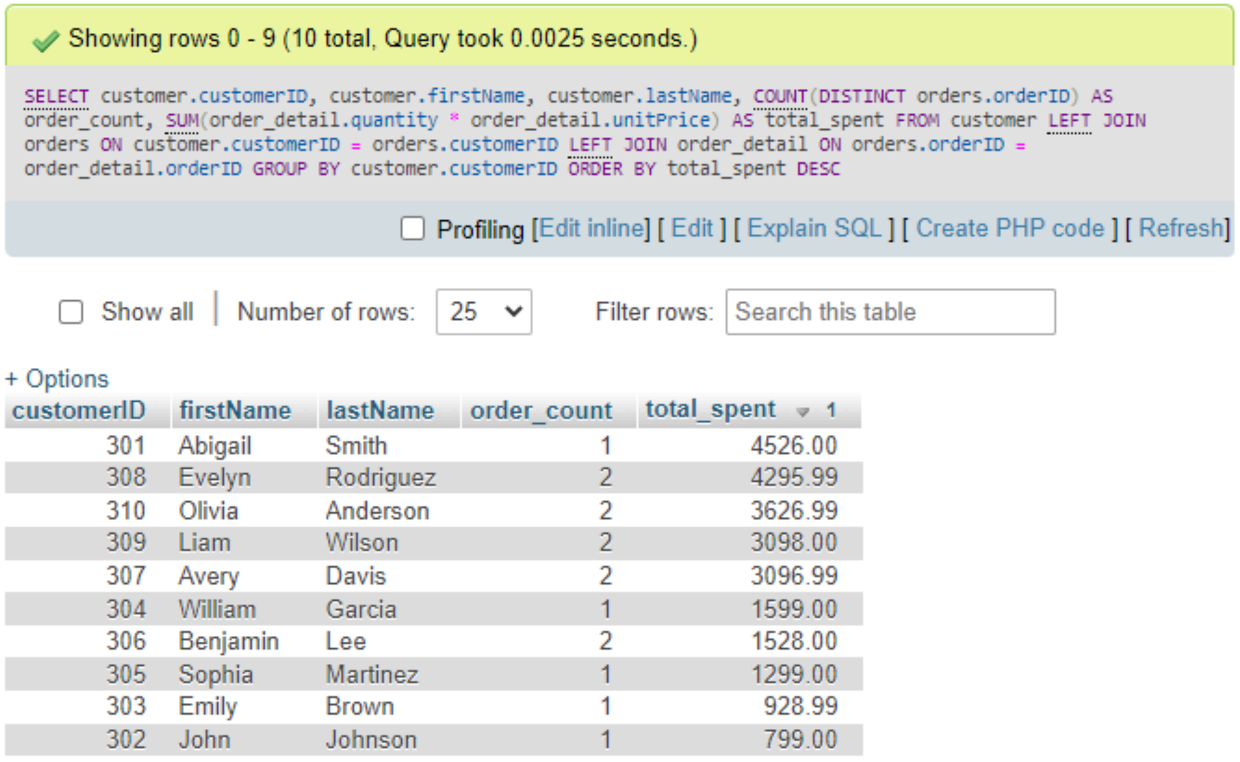
Implementation in MySQL:

Query 1: Retrieve information about all customers and their total amount spent in descending order



Showing rows 0 - 9 (10 total, Query took 0.0025 seconds.)

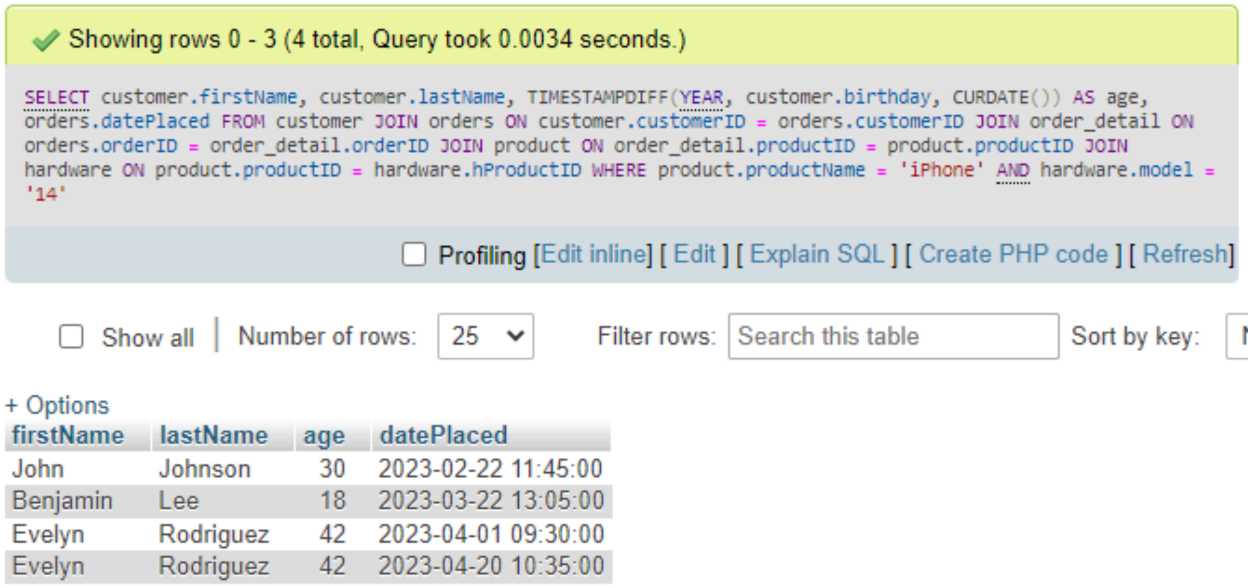
```
SELECT customer.customerID, customer.firstName, customer.lastName, COUNT(DISTINCT orders.orderID) AS
order_count, SUM(order_detail.quantity * order_detail.unitPrice) AS total_spent FROM customer LEFT JOIN
orders ON customer.customerID = orders.customerID LEFT JOIN order_detail ON orders.orderID =
order_detail.orderID GROUP BY customer.customerID ORDER BY total_spent DESC
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25
Filter rows: Search this table
+ Options

customerID	firstName	lastName	order_count	total_spent 1
301	Abigail	Smith	1	4526.00
308	Evelyn	Rodriguez	2	4295.99
310	Olivia	Anderson	2	3626.99
309	Liam	Wilson	2	3098.00
307	Avery	Davis	2	3096.99
304	William	Garcia	1	1599.00
306	Benjamin	Lee	2	1528.00
305	Sophia	Martinez	1	1299.00
303	Emily	Brown	1	928.99
302	John	Johnson	1	799.00

```
SELECT customer.customerID,  customer.firstName, customer.lastName, COUNT(DISTINCT
orders.orderID) AS order_count, SUM(order_detail.quantity * order_detail.unitPrice) AS
total_spent
FROM customer
   LEFT JOIN orders ON customer.customerID = orders.customerID
   LEFT JOIN order_detail ON orders.orderID = order_detail.orderID
GROUP BY customer.customerID
ORDER BY total_spent DESC;
```

Query 2: Show a set of names and age of the customers who have placed an order for an iPhone 14, and the date they placed the order



✓ Showing rows 0 - 3 (4 total, Query took 0.0034 seconds.)

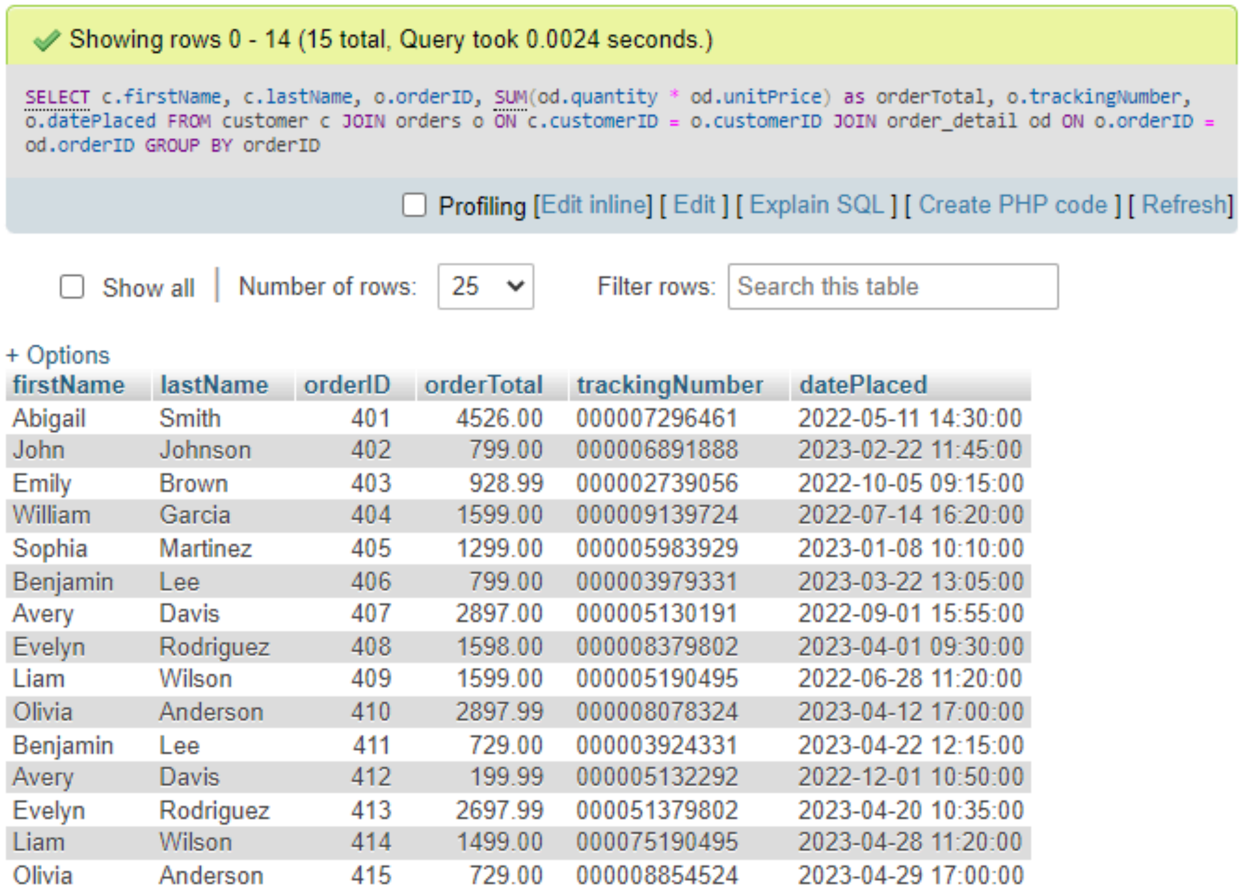
```
SELECT customer.firstName, customer.lastName, TIMESTAMPDIFF(YEAR, customer.birthday, CURDATE()) AS age,
orders.datePlaced FROM customer JOIN orders ON customer.customerID = orders.customerID JOIN order_detail ON
orders.orderID = order_detail.orderID JOIN product ON order_detail.productID = product.productID JOIN
hardware ON product.productID = hardware.hProductID WHERE product.productName = 'iPhone' AND hardware.model =
'14'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25
Filter rows: Search this table
Sort by key:
+ Options

firstName	lastName	age	datePlaced
John	Johnson	30	2023-02-22 11:45:00
Benjamin	Lee	18	2023-03-22 13:05:00
Evelyn	Rodriguez	42	2023-04-01 09:30:00
Evelyn	Rodriguez	42	2023-04-20 10:35:00

SELECT customer.firstName, customer.lastName, TIMESTAMPDIFF(YEAR, customer.birthday, CURDATE()) AS age, orders.datePlaced
FROM customer
JOIN orders ON customer.customerID = orders.customerID
JOIN order_detail ON orders.orderID = order_detail.orderID
JOIN product ON order_detail.productID = product.productID
JOIN hardware ON product.productID = hardware.hProductID
WHERE product.productName = 'iPhone' AND hardware.model = '14';

Query 3 (Join 1): Retrieves the names of all customers who have placed orders and the total price of each order, along with the date of the order.



Showing rows 0 - 14 (15 total, Query took 0.0024 seconds.)

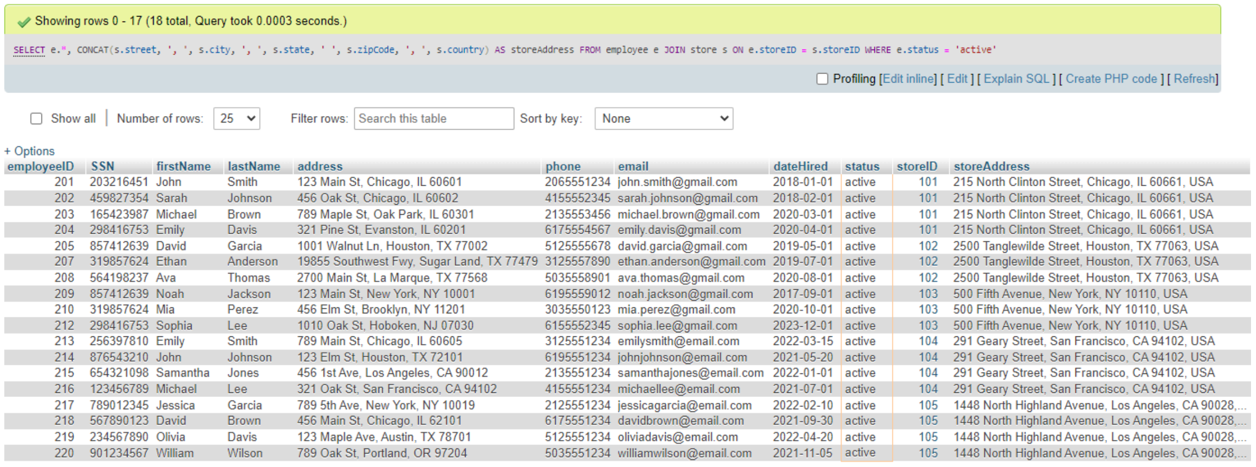
```
SELECT c.firstName, c.lastName, o.orderID, SUM(od.quantity * od.unitPrice) as orderTotal, o.trackingNumber,
o.datePlaced FROM customer c JOIN orders o ON c.customerID = o.customerID JOIN order_detail od ON o.orderID =
od.orderID GROUP BY orderID
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
+ Options

firstName	lastName	orderID	orderTotal	trackingNumber	datePlaced
Abigail	Smith	401	4526.00	000007296461	2022-05-11 14:30:00
John	Johnson	402	799.00	000006891888	2023-02-22 11:45:00
Emily	Brown	403	928.99	000002739056	2022-10-05 09:15:00
William	Garcia	404	1599.00	000009139724	2022-07-14 16:20:00
Sophia	Martinez	405	1299.00	000005983929	2023-01-08 10:10:00
Benjamin	Lee	406	799.00	000003979331	2023-03-22 13:05:00
Avery	Davis	407	2897.00	000005130191	2022-09-01 15:55:00
Evelyn	Rodriguez	408	1598.00	000008379802	2023-04-01 09:30:00
Liam	Wilson	409	1599.00	000005190495	2022-06-28 11:20:00
Olivia	Anderson	410	2897.99	000008078324	2023-04-12 17:00:00
Benjamin	Lee	411	729.00	000003924331	2023-04-22 12:15:00
Avery	Davis	412	199.99	000005132292	2022-12-01 10:50:00
Evelyn	Rodriguez	413	2697.99	000051379802	2023-04-20 10:35:00
Liam	Wilson	414	1499.00	000075190495	2023-04-28 11:20:00
Olivia	Anderson	415	729.00	000008854524	2023-04-29 17:00:00

SELECT c.firstName, c.lastName, o.orderID, SUM(od.quantity * od.unitPrice) as orderTotal, o.trackingNumber, o.datePlaced
FROM customer c
JOIN orders o ON c.customerID = o.customerID
JOIN order_detail od ON o.orderID = od.orderID
GROUP BY orderID;

Query 4 (Join 2): Retrieves the details of all active employees and their corresponding store location IDs and address.



Showing rows 0 - 17 (18 total, Query took 0.0003 seconds.)

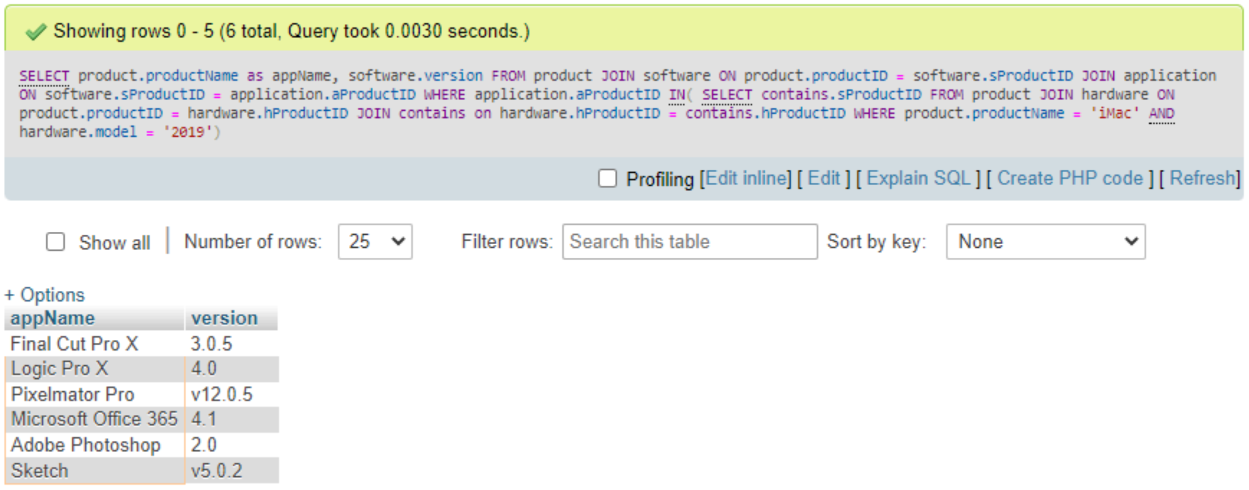
```
SELECT e.*, CONCAT(s.street, ', ', s.city, ', ', s.state, ' ', s.zipCode, ', ', s.country) AS storeAddress FROM employee e JOIN store s ON e.storeID = s.storeID WHERE e.status = 'active'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options

employeeID	SSN	firstName	lastName	address	phone	email	dateHired	status	storeID	storeAddress
201	203216451	John	Smith	123 Main St, Chicago, IL 60601	2065551234	john.smith@gmail.com	2018-01-01	active	101	215 North Clinton Street, Chicago, IL 60661, USA
202	459827354	Sarah	Johnson	456 Oak St, Chicago, IL 60602	4155552345	sarah.johnson@gmail.com	2018-02-01	active	101	215 North Clinton Street, Chicago, IL 60661, USA
203	165423987	Michael	Brown	789 Maple St, Oak Park, IL 60301	2135553456	michael.brown@gmail.com	2020-03-01	active	101	215 North Clinton Street, Chicago, IL 60661, USA
204	298416753	Emily	Davis	321 Pine St, Evanston, IL 60201	6175554567	emily.davis@gmail.com	2020-04-01	active	101	215 North Clinton Street, Chicago, IL 60661, USA
205	857412639	David	Garcia	1001 Walnut Ln, Houston, TX 77002	5125555678	david.garcia@gmail.com	2019-05-01	active	102	2500 Tanglewilde Street, Houston, TX 77063, USA
207	319857624	Ethan	Anderson	19855 Southwest Fwy, Sugar Land, TX 77479	3125557890	ethan.anderson@gmail.com	2019-07-01	active	102	2500 Tanglewilde Street, Houston, TX 77063, USA
208	564198237	Ava	Thomas	2700 Main St, La Marque, TX 77568	5035558901	ava.thomas@gmail.com	2020-08-01	active	102	2500 Tanglewilde Street, Houston, TX 77063, USA
209	857412639	Noah	Jackson	123 Main St, New York, NY 10001	6195559012	noah.jackson@gmail.com	2017-09-01	active	103	500 Fifth Avenue, New York, NY 10110, USA
210	319857624	Mia	Perez	456 Elm St, Brooklyn, NY 11201	3035550123	mia.perez@gmail.com	2020-10-01	active	103	500 Fifth Avenue, New York, NY 10110, USA
212	298416753	Sophia	Lee	1010 Oak St, Hoboken, NJ 07030	6155552345	sophia.lee@gmail.com	2023-12-01	active	103	500 Fifth Avenue, New York, NY 10110, USA
213	256397810	Emily	Smith	789 Main St, Chicago, IL 60605	3125551234	emilysmith@email.com	2022-03-15	active	104	291 Geary Street, San Francisco, CA 94102, USA
214	876543210	John	Johnson	123 Elm St, Houston, TX 72101	6195551234	johnjohnson@email.com	2021-05-20	active	104	291 Geary Street, San Francisco, CA 94102, USA
215	654321098	Samantha	Jones	456 1st Ave, Los Angeles, CA 90012	2135551234	samanthajones@email.com	2022-01-01	active	104	291 Geary Street, San Francisco, CA 94102, USA
216	123456789	Michael	Lee	321 Oak St, San Francisco, CA 94102	4155551234	michaellee@email.com	2021-07-01	active	104	291 Geary Street, San Francisco, CA 94102, USA
217	789012345	Jessica	Garcia	789 5th Ave, New York, NY 10019	2125551234	jessicagarcia@email.com	2022-02-10	active	105	1448 North Highland Avenue, Los Angeles, CA 90028,...
218	567890123	David	Brown	456 Main St, Chicago, IL 62101	6175551234	davidbrown@email.com	2021-09-30	active	105	1448 North Highland Avenue, Los Angeles, CA 90028,...
219	234567890	Olivia	Davis	123 Maple Ave, Austin, TX 78701	5125551234	oliviadavis@email.com	2022-04-20	active	105	1448 North Highland Avenue, Los Angeles, CA 90028,...
220	901234567	William	Wilson	789 Oak St, Portland, OR 97204	5035551234	williamwilson@email.com	2021-11-05	active	105	1448 North Highland Avenue, Los Angeles, CA 90028,...

SELECT e.*, CONCAT(s.street, ', ', s.city, ', ', s.state, ' ', s.zipCode, ', ', s.country) AS storeAddress
FROM employee e
JOIN store s ON e.storeID = s.storeID
WHERE e.status = 'active'

Query 5 (Join 3): Retrieves the name and version of all applications that can run on an iMac 2019 model.



Showing rows 0 - 5 (6 total, Query took 0.0030 seconds.)

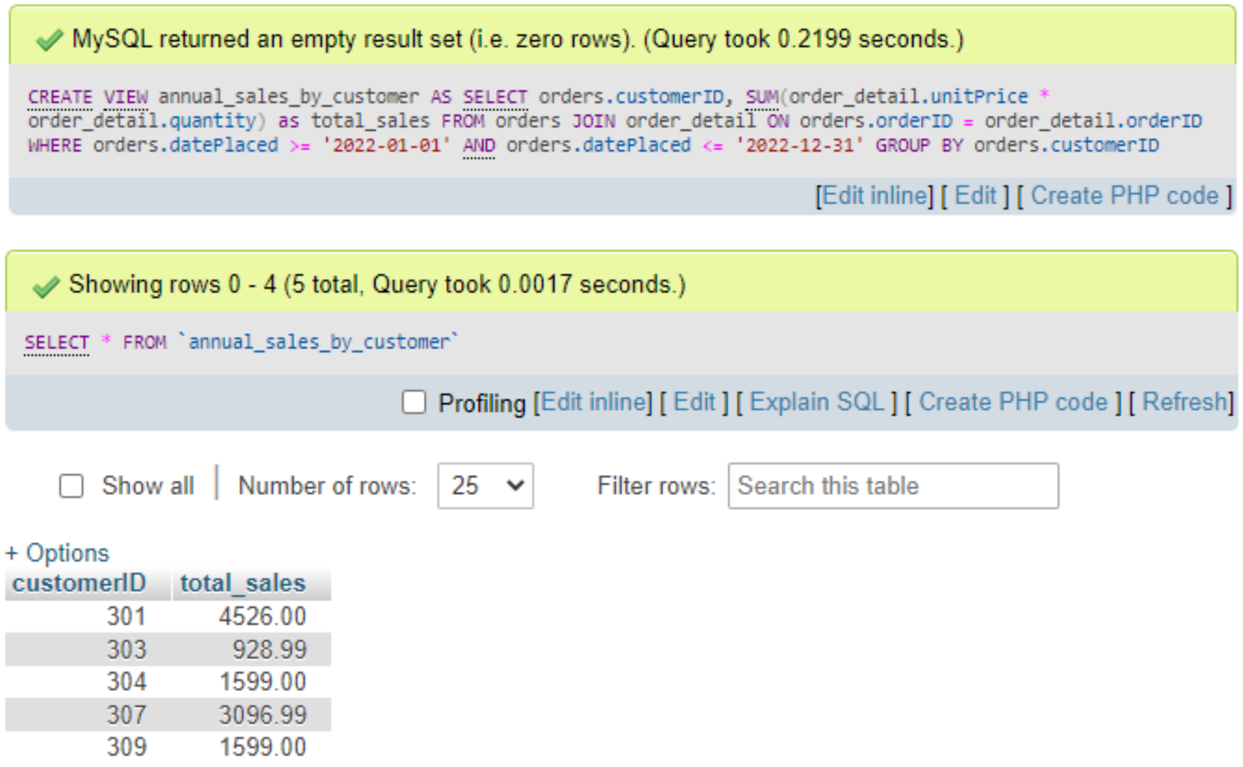
```
SELECT product.productName as appName, software.version FROM product JOIN software ON product.productID = software.sProductID JOIN application ON software.sProductID = application.aProductID WHERE application.aProductID IN( SELECT contains.sProductID FROM product JOIN hardware ON product.productID = hardware.hProductID JOIN contains on hardware.hProductID = contains.hProductID WHERE product.productName = 'iMac' AND hardware.model = '2019')
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

appName	version
Final Cut Pro X	3.0.5
Logic Pro X	4.0
Pixelmator Pro	v12.0.5
Microsoft Office 365	4.1
Adobe Photoshop	2.0
Sketch	v5.0.2

```
SELECT product.productName as appName, software.version
FROM product
JOIN software ON product.productID = software.sProductID
JOIN application ON software.sProductID = application.aProductID
WHERE application.aProductID IN(
    SELECT contains.sProductID
    FROM product
    JOIN hardware ON product.productID = hardware.hProductID
    JOIN contains on hardware.hProductID = contains.hProductID
    WHERE product.productName = 'iMac' AND hardware.model = '2019');
```

First View: Shows the customerID and the total sales for each.



MySQL returned an empty result set (i.e. zero rows). (Query took 0.2199 seconds.)

```
CREATE VIEW annual_sales_by_customer AS SELECT orders.customerID, SUM(order_detail.unitPrice *
order_detail.quantity) as total_sales FROM orders JOIN order_detail ON orders.orderID = order_detail.orderID
WHERE orders.datePlaced >= '2022-01-01' AND orders.datePlaced <= '2022-12-31' GROUP BY orders.customerID
```

Showing rows 0 - 4 (5 total, Query took 0.0017 seconds.)

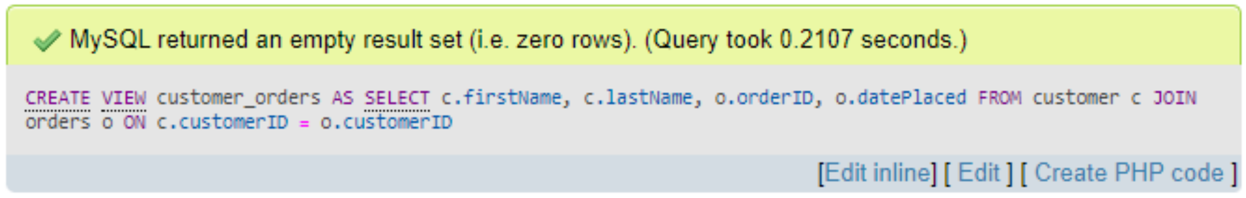
```
SELECT * FROM `annual_sales_by_customer`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
+ Options

customerID	total_sales
301	4526.00
303	928.99
304	1599.00
307	3096.99
309	1599.00

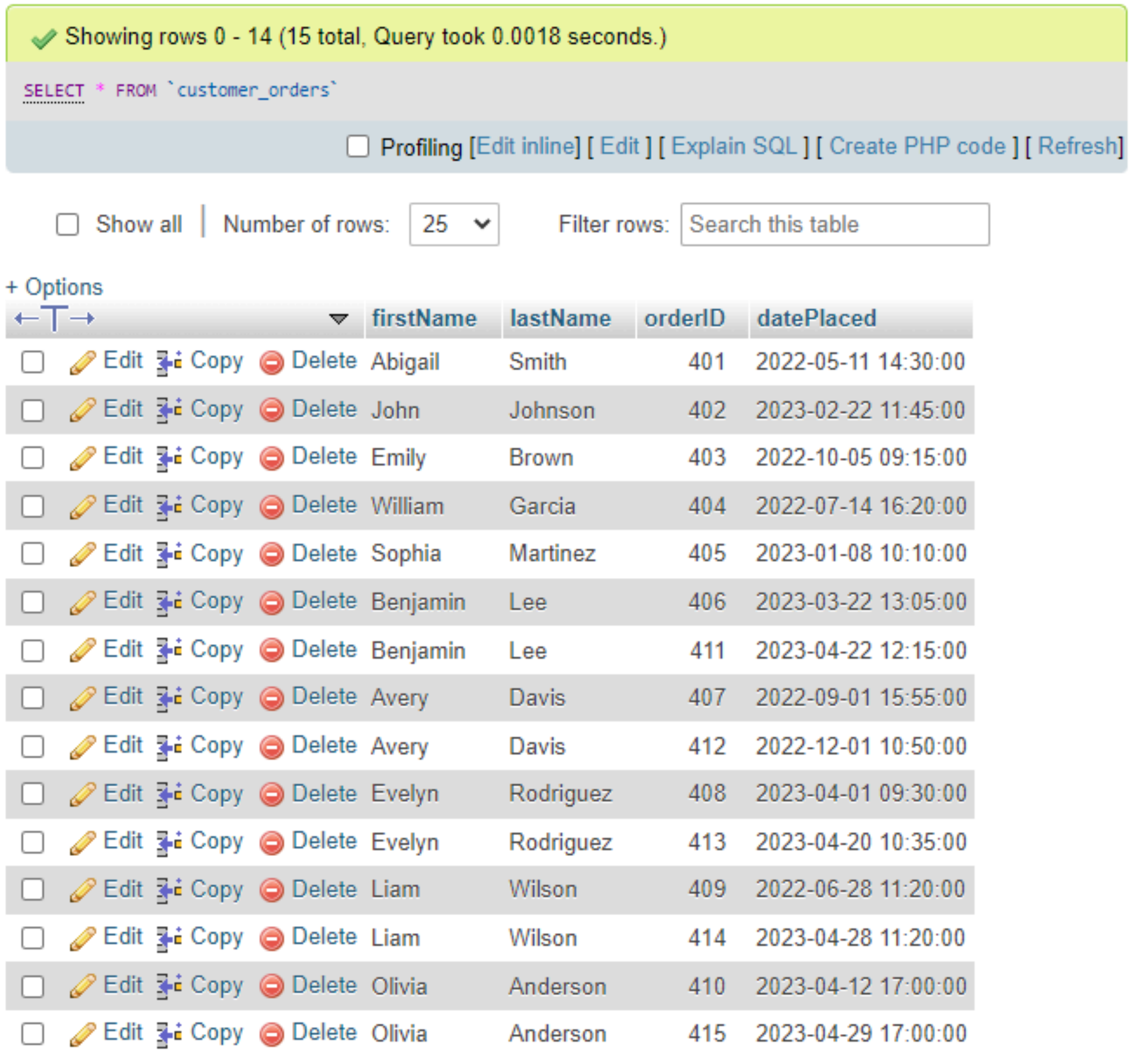
```
CREATE VIEW annual_sales_by_customer AS
SELECT orders.customerID, SUM(order_detail.unitPrice * order_detail.quantity) as total_sales
FROM orders
JOIN order_detail ON orders.orderID = order_detail.orderID
WHERE orders.datePlaced >= '2022-01-01' AND orders.datePlaced <= '2022-12-31'
GROUP BY orders.customerID;
```

Second View: Returns the customer name, order number, and order date for each order in the system.



MySQL returned an empty result set (i.e. zero rows). (Query took 0.2107 seconds.)

```
CREATE VIEW customer_orders AS SELECT c.firstName, c.lastName, o.orderID, o.datePlaced FROM customer c JOIN
orders o ON c.customerID = o.customerID
```



Showing rows 0 - 14 (15 total, Query took 0.0018 seconds.)

```
SELECT * FROM `customer_orders`
```

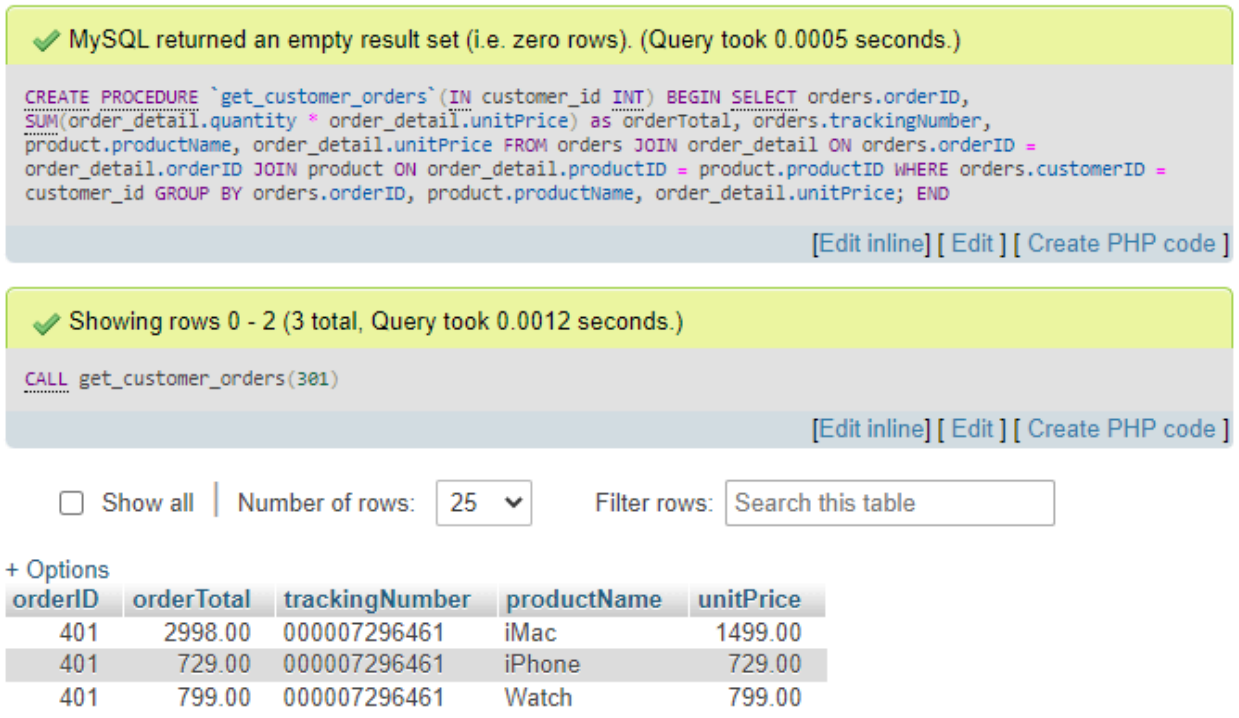
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options

firstName	lastName	orderID	datePlaced
Abigail	Smith	401	2022-05-11 14:30:00
John	Johnson	402	2023-02-22 11:45:00
Emily	Brown	403	2022-10-05 09:15:00
William	Garcia	404	2022-07-14 16:20:00
Sophia	Martinez	405	2023-01-08 10:10:00
Benjamin	Lee	406	2023-03-22 13:05:00
Benjamin	Lee	411	2023-04-22 12:15:00
Avery	Davis	407	2022-09-01 15:55:00
Avery	Davis	412	2022-12-01 10:50:00
Evelyn	Rodriguez	408	2023-04-01 09:30:00
Evelyn	Rodriguez	413	2023-04-20 10:35:00
Liam	Wilson	409	2022-06-28 11:20:00
Liam	Wilson	414	2023-04-28 11:20:00
Olivia	Anderson	410	2023-04-12 17:00:00
Olivia	Anderson	415	2023-04-29 17:00:00

CREATE VIEW customer_orders AS
SELECT c.firstName, c.lastName, o.orderID, o.datePlaced
FROM customer c
JOIN orders o ON c.customerID = o.customerID;

Ashley Cao, Remi Henry, Haley Iniguez, Piero Quintana, Emanuel Tafese

First Stored Procedure: Retrieves the orders for a given customer, where we are able to view order ID, price, tracking number, detail, and the name of each product. We join the product table as it correlates to the same product ID from both the product and orders table. The procedure results in a set of orders made by the specified customer, along with the associated product name and price.



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)

```
CREATE PROCEDURE `get_customer_orders`(IN customer_id INT) BEGIN SELECT orders.orderID,
SUM(order_detail.quantity * order_detail.unitPrice) as orderTotal, orders.trackingNumber,
product.productName, order_detail.unitPrice FROM orders JOIN order_detail ON orders.orderID =
order_detail.orderID JOIN product ON order_detail.productID = product.productID WHERE orders.customerID =
customer_id GROUP BY orders.orderID, product.productName, order_detail.unitPrice; END
```

Showing rows 0 - 2 (3 total, Query took 0.0012 seconds.)

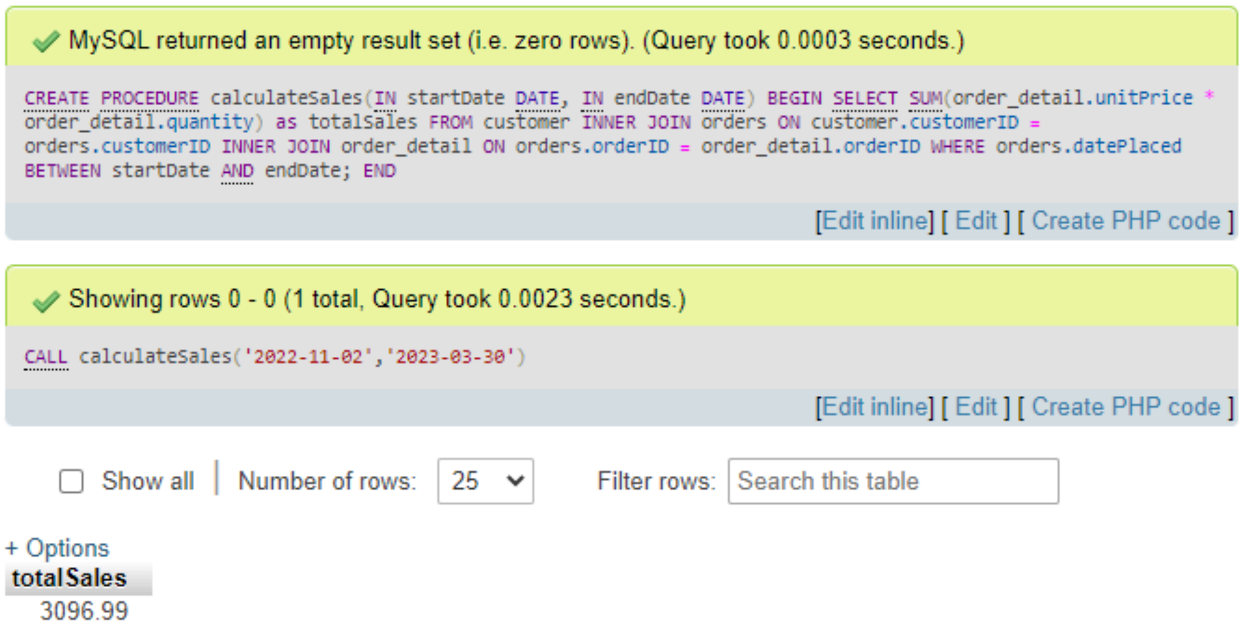
```
CALL get_customer_orders(301)
```

orderID	orderTotal	trackingNumber	productName	unitPrice
401	2998.00	000007296461	iMac	1499.00
401	729.00	000007296461	iPhone	729.00
401	799.00	000007296461	Watch	799.00

```
DELIMITER //
CREATE PROCEDURE `get_customer_orders`(IN customer_id INT)
BEGIN
  SELECT orders.orderID, SUM(order_detail.quantity * order_detail.unitPrice) as orderTotal,
orders.trackingNumber, product.productName, order_detail.unitPrice
  FROM orders
  JOIN order_detail ON orders.orderID = order_detail.orderID
  JOIN product ON order_detail.productID = product.productID
  WHERE orders.customerID = customer_id
  GROUP BY orders.orderID, product.productName, order_detail.unitPrice;
END //
DELIMITER ;

CALL get_customer_orders(301);
```

Second Stored Procedure: Calculates the total sales for a given period (YYYY-MM-DD)



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```
CREATE PROCEDURE calculateSales(IN startDate DATE, IN endDate DATE) BEGIN SELECT SUM(order_detail.unitPrice * order_detail.quantity) as totalSales FROM customer INNER JOIN orders ON customer.customerID = orders.customerID INNER JOIN order_detail ON orders.orderID = order_detail.orderID WHERE orders.datePlaced BETWEEN startDate AND endDate; END
```

✓ Showing rows 0 - 0 (1 total, Query took 0.0023 seconds.)

```
CALL calculateSales('2022-11-02','2023-03-30')
```

[Edit inline] [Edit] [Create PHP code]
Show all
Number of rows: 25
Filter rows: Search this table
+ Options

totalSales
3096.99

```
DELIMITER //
CREATE PROCEDURE calculateSales(IN startDate DATE, IN endDate DATE)
BEGIN
  SELECT SUM(order_detail.unitPrice * order_detail.quantity) as totalSales
  FROM customer
  INNER JOIN orders ON customer.customerID = orders.customerID
  INNER JOIN order_detail ON orders.orderID = order_detail.orderID
  WHERE orders.datePlaced BETWEEN startDate AND endDate;
END //
DELIMITER ;

CALL calculateSales('2022-11-02','2023-03-30');
```