

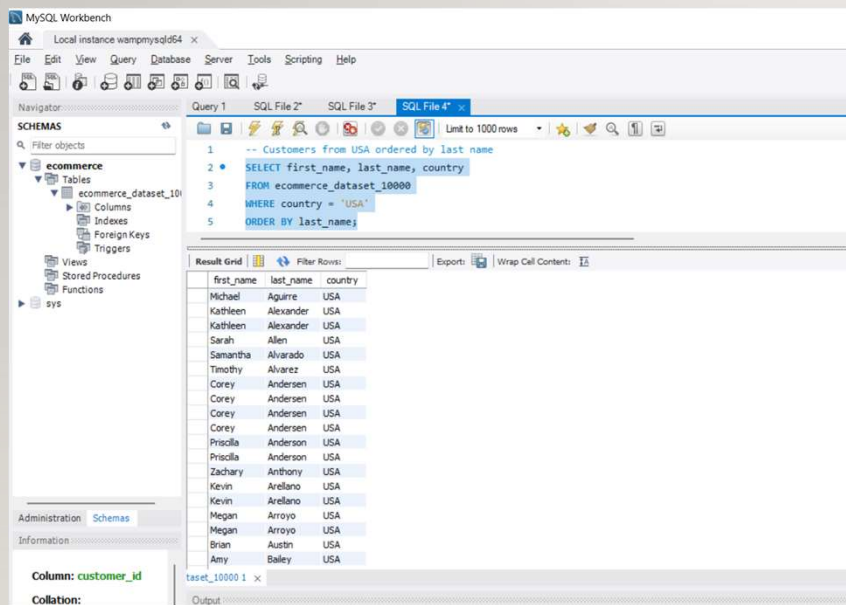
SCREENSHOTS OF OUTPUTS

RETRIEVAL OF ALL COLUMNS - SELECT

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view with 'ecommerce' expanded, containing 'ecommerce_dataset_10000'. The central 'Query Editor' shows a SQL query: `SELECT * FROM ecommerce_dataset_10000;`. The 'Result Grid' at the bottom displays the query results in a table format. The table has 14 columns: customer_id, first_name, last_name, gender, age_group, signup_date, country, product_id, product_name, category, quantity, unit_price, order_id, and order_date. The results show 10 rows of data, including customer details and product information.

customer_id	first_name	last_name	gender	age_group	signup_date	country	product_id	product_name	category	quantity	unit_price	order_id	order_date
CUST4463	Christopher	White	Male	Adults	2023-08-24	China	PROD103	Levi's Jeans	Apparel	4	59	ORD10001	2024-08-
CUST2790	Shelby	Sutton	Other	Adults	2025-07-18	Canada	PROD108	Fitbit Versa 3	Electronics	5	229	ORD10005	2023-04-
CUST2451	Barbara	Hansen	Female	Adults	2024-11-10	UK	PROD103	Levi's Jeans	Apparel	2	59	ORD10007	2024-01-
CUST2364	Mary	Guzman	Female	Adults	2023-10-30	China	PROD102	Sony Headphones	Electronics	5	199	ORD10008	2022-11-
CUST1438	Michelle	Vargas	Male	Adults	2023-07-11	UK	PROD109	Kindle Paperwhite	Books	1	129	ORD10012	2024-05-
CUST2488	Christy	Davis	Female	Adults	2024-12-08	China	PROD100	iPhone 14	Electronics	2	999	ORD10013	2023-11-
CUST4174	Daniel	Gray	Other	Adults	2022-11-10	China	PROD113	Wilson Tennis Racket	Sports	2	149	ORD10017	2022-12-
CUST3410	Kenneth	Garner	Female	Adults	2022-01-02	Canada	PROD108	Fitbit Versa 3	Electronics	3	229	ORD10019	2024-10-
CUST4051	Edward	Molina	Female	Adults	2024-01-29	Australia	PROD112	Barbie Dreamhouse	Toys	5	199	ORD10021	2025-07-
CUST4430	Michael	Thomas	Male	Adults	2024-02-14	India	PROD109	Kindle Paperwhite	Books	3	129	ORD10033	2024-03-
CUST3021	Mitchell	Rogers	Other	Adults	2024-01-19	Australia	PROD101	Samsung Galaxy S23	Electronics	4	899	ORD10034	2025-06-

USE OF - WHERE , ORDERBY & GROUP BY



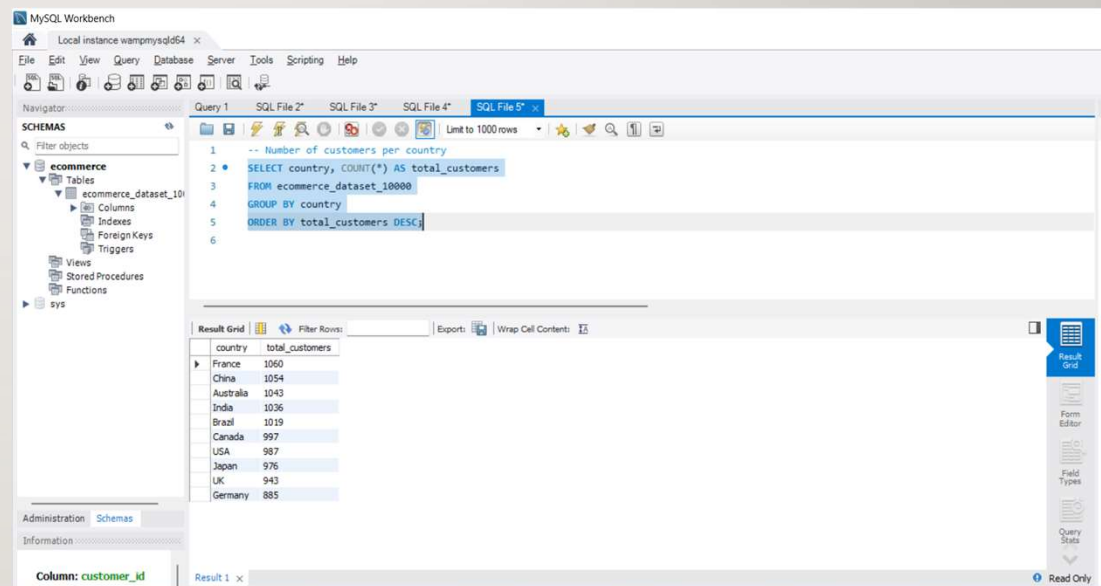
The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
-- Customers from USA ordered by last name
SELECT first_name, last_name, country
FROM ecommerce_dataset_10000
WHERE country = 'USA'
ORDER BY last_name;
```

The result grid displays the following data:

first_name	last_name	country
Michael	Aguirre	USA
Kathleen	Alexander	USA
Kathleen	Alexander	USA
Sarah	Allen	USA
Samantha	Alvarado	USA
Timothy	Alvarez	USA
Corey	Andersen	USA
Corey	Andersen	USA
Corey	Andersen	USA
Corey	Andersen	USA
Corey	Andersen	USA
Priscilla	Anderson	USA
Priscilla	Anderson	USA
Zachary	Anthony	USA
Kevin	Arellano	USA
Kevin	Arellano	USA
Megan	Arroyo	USA
Megan	Arroyo	USA
Brian	Austin	USA
Amy	Bailey	USA

Customers from USA ordered by last name



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

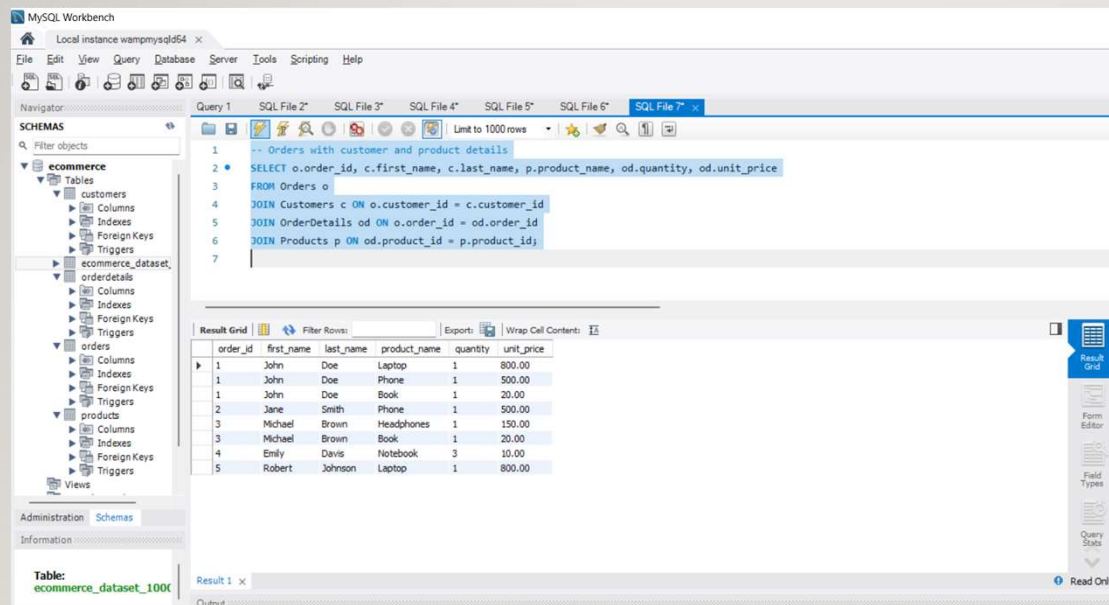
```
-- Number of customers per country
SELECT country, COUNT(*) AS total_customers
FROM ecommerce_dataset_10000
GROUP BY country
ORDER BY total_customers DESC;
```

The result grid displays the following data:

country	total_customers
France	1060
China	1054
Australia	1043
India	1036
Brazil	1019
Canada	997
USA	987
Japan	976
UK	943
Germany	885

Number of customers per country

USE OF JOINS



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

```
1 -- Orders with customer and product details
2 SELECT o.order_id, c.first_name, c.last_name, p.product_name, od.quantity, od.unit_price
3 FROM Orders o
4 JOIN Customers c ON o.customer_id = c.customer_id
5 JOIN OrderDetails od ON o.order_id = od.order_id
6 JOIN Products p ON od.product_id = p.product_id
7
```

The result grid displays the following data:

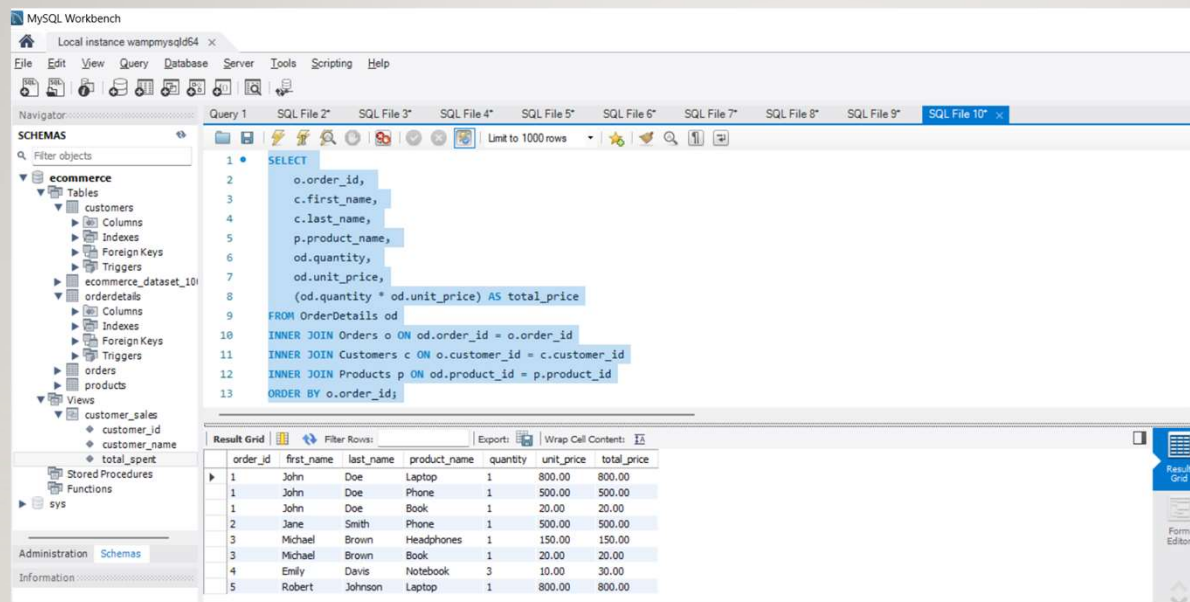
order_id	first_name	last_name	product_name	quantity	unit_price
1	John	Doe	Laptop	1	800.00
1	John	Doe	Phone	1	500.00
1	John	Doe	Book	1	20.00
2	Jane	Smith	Phone	1	500.00
3	Michael	Brown	Headphones	1	150.00
3	Michael	Brown	Book	1	20.00
4	Emily	Davis	Notebook	3	10.00
5	Robert	Johnson	Laptop	1	800.00

This query creates a combined view of:

- Which customer placed the order
- What products were ordered
- The quantity and price per product

Basically, it gives you a complete sales order report from multiple tables.

INNER JOINS



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' panel displays a tree view of the 'ecommerce' database, including tables like 'customers', 'orderdetails', 'orders', and 'products'. The main query editor displays the following SQL query:

```
1 SELECT
2   o.order_id,
3   c.first_name,
4   c.last_name,
5   p.product_name,
6   od.quantity,
7   od.unit_price,
8   (od.quantity * od.unit_price) AS total_price
9 FROM OrderDetails od
10 INNER JOIN Orders o ON od.order_id = o.order_id
11 INNER JOIN Customers c ON o.customer_id = c.customer_id
12 INNER JOIN Products p ON od.product_id = p.product_id
13 ORDER BY o.order_id;
```

Below the query editor, the 'Result Grid' shows the output of the query. It contains 5 rows of data, each representing an order line item. The columns are: order_id, first_name, last_name, product_name, quantity, unit_price, and total_price.

order_id	first_name	last_name	product_name	quantity	unit_price	total_price
1	John	Doe	Laptop	1	800.00	800.00
1	John	Doe	Phone	1	500.00	500.00
1	John	Doe	Book	1	20.00	20.00
2	Jane	Smith	Phone	1	500.00	500.00
3	Michael	Brown	Headphones	1	150.00	150.00
3	Michael	Brown	Book	1	20.00	20.00
4	Emily	Davis	Notebook	3	10.00	30.00
5	Robert	Johnson	Laptop	1	800.00	800.00

- Shows only orders that exist in all tables.
- Total price per order line is calculated.

RIGHT JOIN

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'ecommerce' database selected. The main editor window shows a SQL query in 'Query 1':

```
1 SELECT
2     Execute the selected portion of the script or everything, if there is no selection
3
4     o.total_amount,
5     c.first_name,
6     c.last_name
7 FROM Customers c
8 RIGHT JOIN Orders o ON c.customer_id = o.customer_id
9 ORDER BY o.order_id;
```

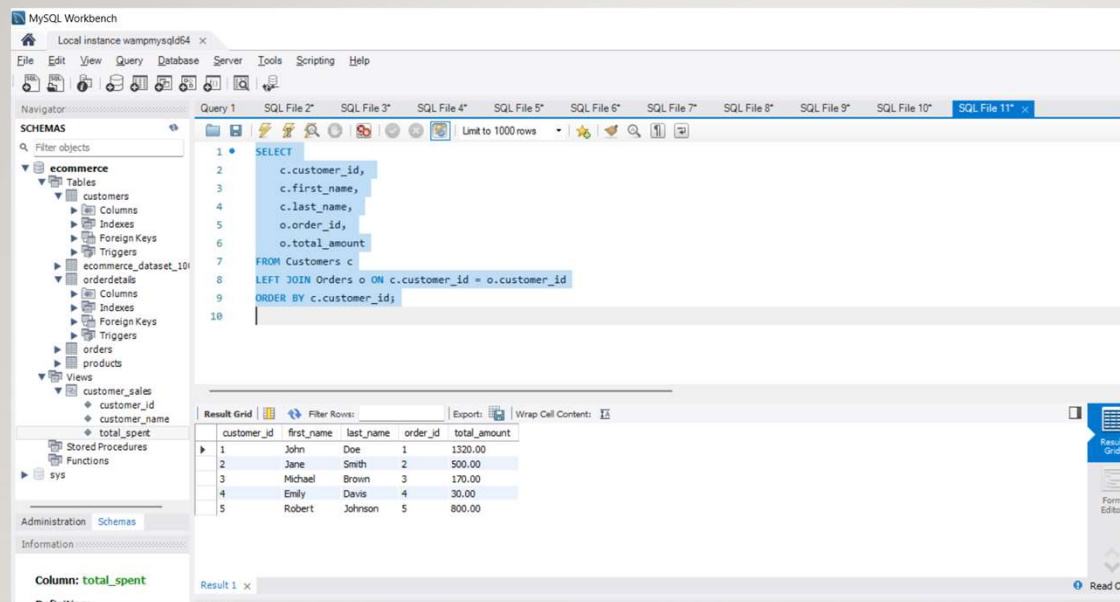
Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The columns are 'order_id', 'total_amount', 'first_name', and 'last_name'. The results are as follows:

order_id	total_amount	first_name	last_name
1	1320.00	John	Doe
2	500.00	Jane	Smith
3	170.00	Michael	Brown
4	30.00	Emily	Davis
5	800.00	Robert	Johnson

The bottom status bar indicates 'Column: total_spent' and 'Result 1 x'.

- Shows all orders.
- If any order does not have a matching customer, customer columns will be NULL.

LEFT JOIN



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the 'ecommerce' database, including tables like 'customers', 'orderdetails', 'orders', and 'products'. The main query editor displays the following SQL query:

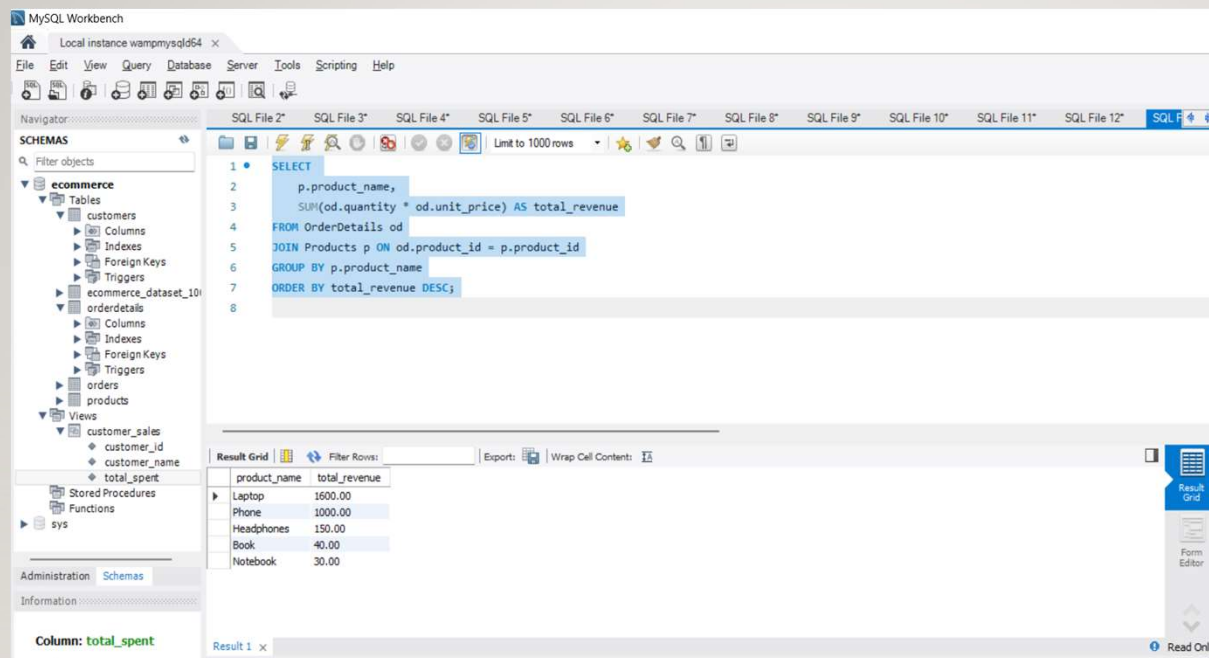
```
1 SELECT
2   c.customer_id,
3   c.first_name,
4   c.last_name,
5   o.order_id,
6   o.total_amount
7 FROM Customers c
8 LEFT JOIN Orders o ON c.customer_id = o.customer_id
9 ORDER BY c.customer_id;
```

Below the query editor, the 'Result Grid' shows the output of the query. The grid has five columns: 'customer_id', 'first_name', 'last_name', 'order_id', and 'total_amount'. The results are as follows:

customer_id	first_name	last_name	order_id	total_amount
1	John	Doe	1	1320.00
2	Jane	Smith	2	500.00
3	Michael	Brown	3	170.00
4	Emily	Davis	4	30.00
5	Robert	Johnson	5	800.00

- Shows all customers.
- If a customer has no orders, order_id and total_amount will be NULL.

AGGREGATE FUNCTIONS - SUM



The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view of the 'ecommerce' database, including tables like 'customers', 'orderdetails', 'products', and 'orders'. The main editor window contains a SQL query that uses the SUM aggregate function to calculate the total revenue for each product. The query is as follows:

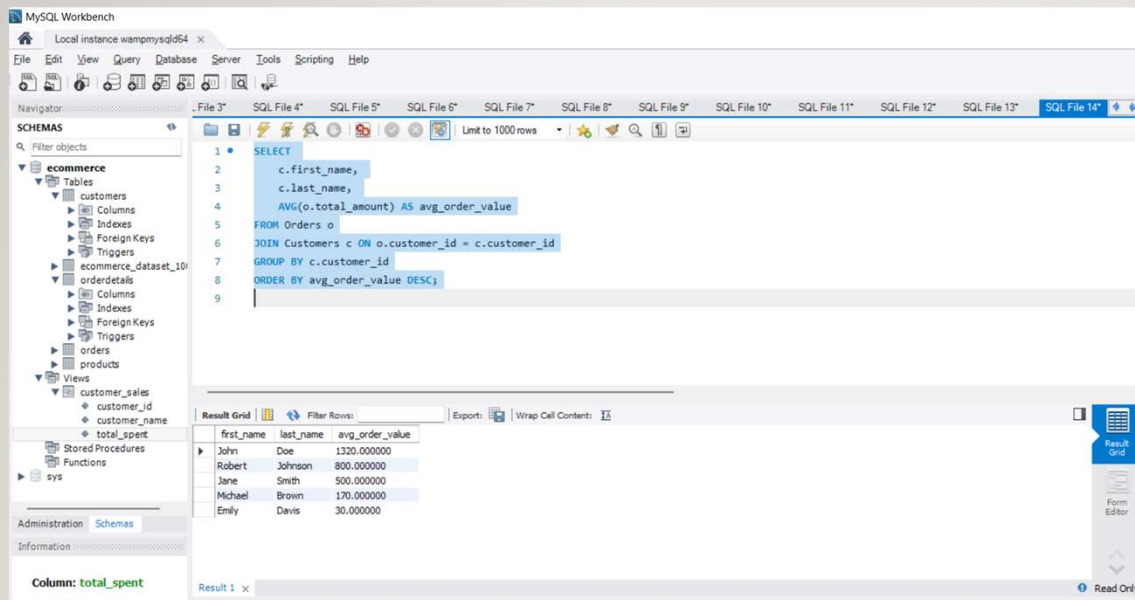
```
1 SELECT
2   p.product_name,
3   SUM(od.quantity * od.unit_price) AS total_revenue
4 FROM OrderDetails od
5 JOIN Products p ON od.product_id = p.product_id
6 GROUP BY p.product_name
7 ORDER BY total_revenue DESC;
```

Below the query editor, the 'Result Grid' shows the output of the query, displaying the product names and their corresponding total revenues:

product_name	total_revenue
Laptop	1600.00
Phone	1000.00
Headphones	150.00
Book	40.00
Notebook	30.00

For each product, calculates the total revenue by multiplying quantity and unit_price, then summing it up.

AGGREGATE FUNCTIONS - AVG



The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view of the 'ecommerce' database, including tables like 'customers', 'orders', and 'products'. The main editor window contains an SQL query that selects customer names and calculates the average order value. The query is as follows:

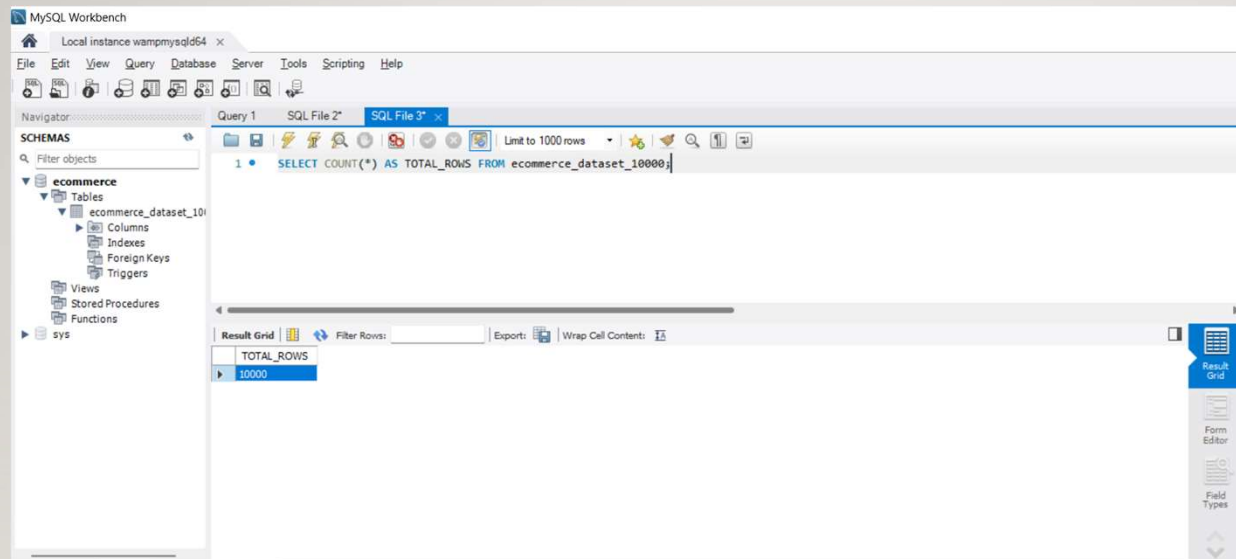
```
1 SELECT
2   c.first_name,
3   c.last_name,
4   AVG(o.total_amount) AS avg_order_value
5 FROM Orders o
6 JOIN Customers c ON o.customer_id = c.customer_id
7 GROUP BY c.customer_id
8 ORDER BY avg_order_value DESC;
```

Below the query editor, the 'Result Grid' shows the output of the query, displaying the first name, last name, and the calculated average order value for each customer, sorted in descending order.

first_name	last_name	avg_order_value
John	Doe	1320.000000
Robert	Johnson	800.000000
Jane	Smith	500.000000
Michael	Brown	170.000000
Emily	Davis	30.000000

For each customer, calculates the average order amount across all their orders.

AGGREGATE FUNCTIONS - COUNT



Count total rows

SUBQUERIES

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'ecommerce' database selected. The main editor window shows a SQL query using a subquery to identify customers who have spent more than \$500. The query is as follows:

```
1 -- Customers who spent more than $500
2 SELECT first_name, last_name
3 FROM Customers
4 WHERE customer_id IN (
5     SELECT o.customer_id
6     FROM Orders o
7     JOIN OrderDetails od ON o.order_id = od.order_id
8     GROUP BY o.customer_id
9     HAVING SUM(od.quantity * od.unit price) > 500
10 )
```

Below the query editor, the 'Result Grid' shows the output of the query, displaying the first and last names of the customers who meet the criteria:

first_name	last_name
John	Doe
Robert	Johnson

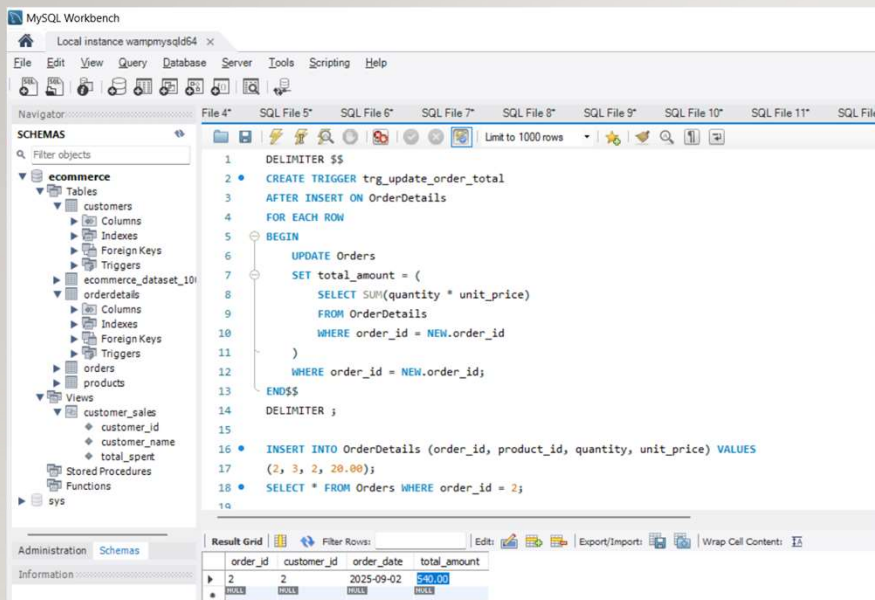
Customers who spent more than \$500

VIEWS

The screenshot displays a database management interface with a Navigator pane on the left and a SQL editor on the right. The Navigator pane shows a tree structure for the 'ecommerce' database, including tables (customers, orderdetails, orders, products), views (customer_sales), and stored procedures. The 'customer_sales' view is expanded, showing its columns: customer_id, customer_name, and total_spent. The SQL editor shows a query to create the 'Customer_Sales' view. The query uses a SELECT statement with columns c.customer_id, CONCAT(c.first_name, ' ', c.last_name) AS customer_name, and SUM(od.quantity * od.unit_price) AS total_spent. It joins the Customers table (c) with the Orders table (o) on c.customer_id = o.customer_id, and the OrderDetails table (od) on o.order_id = od.order_id. The results are grouped by c.customer_id.

```
1 • CREATE VIEW Customer_Sales AS
2 SELECT c.customer_id, CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
3        SUM(od.quantity * od.unit_price) AS total_spent
4 FROM Customers c
5 JOIN Orders o ON c.customer_id = o.customer_id
6 JOIN OrderDetails od ON o.order_id = od.order_id
7 GROUP BY c.customer_id;
8
```

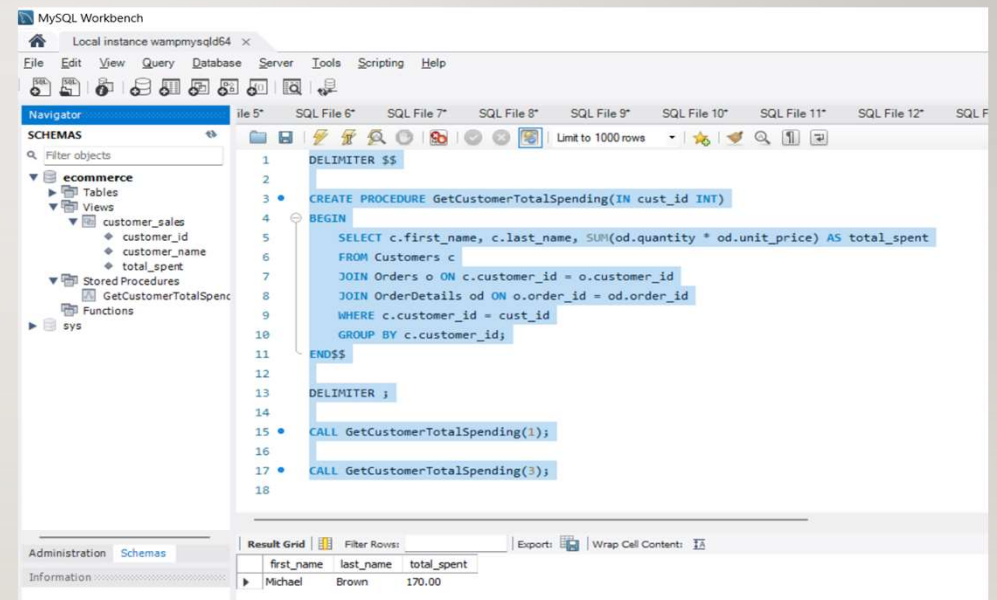
TRIGGERS & STORED PROCEDURE



The screenshot shows the MySQL Workbench interface with a script editor containing a trigger definition. The trigger, named `trg_update_order_total`, is an `AFTER INSERT` trigger on the `OrderDetails` table. It uses a `FOR EACH ROW` logic to update the `total_amount` in the `Orders` table for the corresponding order. The script includes `DELIMITER $$`, `CREATE TRIGGER`, `UPDATE`, `INSERT INTO`, and `SELECT` statements. The `Result Grid` at the bottom shows a single row with the following data:

order_id	customer_id	order_date	total_amount
2	2	2025-09-02	540.00

- After a new order detail is added, the trigger recalculates the `total_amount` for the corresponding order.
- `NEW.order_id` refers to the order of the newly inserted row.



The screenshot shows the MySQL Workbench interface with a script editor containing a stored procedure definition. The procedure, named `GetCustomerTotalSpending`, takes a customer ID as input and returns the total spending for that customer. The script includes `DELIMITER $$`, `CREATE PROCEDURE`, `SELECT`, `JOIN`, `WHERE`, `GROUP BY`, `END$$`, and `CALL` statements. The `Result Grid` at the bottom shows a single row with the following data:

first_name	last_name	total_spent
Michael	Brown	170.00

- Accepts a customer ID as input.
- Returns the total spending of that customer.