

1. Створіть базу даних для керування бібліотекою книг згідно зі структурою, наведеною нижче. Використовуйте DDL-команди для створення необхідних таблиць та їх зв'язків.

### Структура БД

a) Назва схеми — "LibraryManagement"

b) Таблиця "authors":

`author_id` (INT, автоматично зростаючий PRIMARY KEY)

`author_name` (VARCHAR)

c) Таблиця "genres":

`genre_id` (INT, автоматично зростаючий PRIMARY KEY)

`genre_name` (VARCHAR)

d) Таблиця "books":

`book_id` (INT, автоматично зростаючий PRIMARY KEY)

`title` (VARCHAR)

`publication_year` (YEAR)

`author_id` (INT, FOREIGN KEY зв'язок з "Authors")

`genre_id` (INT, FOREIGN KEY зв'язок з "Genres")

e) Таблиця "users":

`user_id` (INT, автоматично зростаючий PRIMARY KEY)

`username` (VARCHAR)

`email` (VARCHAR)

f) Таблиця "borrowed\_books":

`borrow_id` (INT, автоматично зростаючий PRIMARY KEY)

`book_id` (INT, FOREIGN KEY зв'язок з "Books")

`user_id` (INT, FOREIGN KEY зв'язок з "Users")

`borrow_date` (DATE)

`return_date` (DATE)

Відповідь:

```
CREATE DATABASE IF NOT EXISTS LibraryManagement;
USE LibraryManagement;
```

```
CREATE TABLE authors (
  author_id INT AUTO_INCREMENT PRIMARY KEY,
  author_name VARCHAR(200)
);
```

```
CREATE TABLE genres (
  genre_id INT AUTO_INCREMENT PRIMARY KEY,
  genre_name VARCHAR(200)
);
```

```
CREATE TABLE books (
  book_id INT AUTO_INCREMENT PRIMARY KEY,
  title VARCHAR(200),
  publication_year YEAR,
  author_id INT,
  genre_id INT,
  FOREIGN KEY (author_id) REFERENCES authors(author_id),
```

```
FOREIGN KEY (genre_id) REFERENCES genres(genre_id)
);
```

```
CREATE TABLE users (
  user_id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(200),
  email VARCHAR(200)
);
```

```
CREATE TABLE borrowed_books (
  borrow_id INT AUTO_INCREMENT PRIMARY KEY,
  book_id INT,
  user_id INT,
  borrow_date DATE,
  return_date DATE,
  FOREIGN KEY (book_id) REFERENCES books(book_id),
  FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

Administration Schemas Query 11 Limit to 1000 rows

SCHEMAS

Filter objects

- hw\_3
- LibraryManagement
  - Tables
    - authors
    - books
    - borrowed\_books
    - genres
    - users
  - Views
  - Stored Procedures
  - Functions
- mydb
- sys

```

1 CREATE DATABASE IF NOT EXISTS LibraryManagement;
2 USE LibraryManagement;
3
4 CREATE TABLE authors (
5   author_id INT AUTO_INCREMENT PRIMARY KEY,
6   author_name VARCHAR(200)
7 );
8
9 CREATE TABLE genres (
10  genre_id INT AUTO_INCREMENT PRIMARY KEY,
11  genre_name VARCHAR(200)
12 );
13
14 CREATE TABLE books (
15  book_id INT AUTO_INCREMENT PRIMARY KEY,
16  title VARCHAR(200),
17  publication_year YEAR,
18  author_id INT,
19  genre_id INT,
20  FOREIGN KEY (author_id) REFERENCES authors(author_id),
21  FOREIGN KEY (genre_id) REFERENCES genres(genre_id)
22 );
23
24 CREATE TABLE users (
25  user_id INT AUTO_INCREMENT PRIMARY KEY,
26  username VARCHAR(200),
27  email VARCHAR(200)
28 );
29
30 CREATE TABLE borrowed_books (
31  borrow_id INT AUTO_INCREMENT PRIMARY KEY,
32  book_id INT,
33  user_id INT,
34  borrow_date DATE,
35  return_date DATE,
36  FOREIGN KEY (book_id) REFERENCES books(book_id),
37  FOREIGN KEY (user_id) REFERENCES users(user_id)
38 );

```

75% 3:38

Action Output

	Time	Action	Response	Duration / Fetch Time
3	22:49:43	CREATE TABLE authors ( author_id INT AUTO_INCREMENT PRIMARY KEY, author_name VARCHAR(200) )	0 row(s) affected	0.013 sec
4	22:49:43	CREATE TABLE genres ( genre_id INT AUTO_INCREMENT PRIMARY KEY, genre_name VARCHAR(200) )	0 row(s) affected	0.0075 sec
5	22:49:43	CREATE TABLE books ( book_id INT AUTO_INCREMENT PRIMARY KEY, title VARCHAR(200), publication_year YEAR, author_id INT, genre_id INT, F...	0 row(s) affected	0.013 sec
6	22:49:43	CREATE TABLE users ( user_id INT AUTO_INCREMENT PRIMARY KEY, username VARCHAR(200), email VARCHAR(200) )	0 row(s) affected	0.015 sec
7	22:49:43	CREATE TABLE borrowed_books ( borrow_id INT AUTO_INCREMENT PRIMARY KEY, book_id INT, user_id INT, borrow_date DATE, return_date DATE,...	0 row(s) affected	0.019 sec

2. Заповніть таблиці простими вигуманими тестовими даними. Достатньо одного-двох рядків у кожену таблицю.

Відповідь:

```
INSERT INTO authors (author_name) VALUES
('Jane Austen'),
('George Orwell');
```

```
INSERT INTO genres (genre_name) VALUES
('Novel'),
('Historical Fiction');
```

```
INSERT INTO users (username, email) VALUES
('John Smith', 'john@example.com'),
('Emily Johnson', 'emily@example.com');
```

```
INSERT INTO books (title, publication_year, author_id, genre_id) VALUES
('Pride and Prejudice', 1813, 1, 1),
('1984', 1949, 2, 2);
```

```
INSERT INTO borrowed_books (book_id, user_id, borrow_date, return_date) VALUES
(1, 1, '2024-03-28', '2024-04-10'),
(2, 2, '2024-03-25', '2024-04-05');
```

```
39
40  INSERT INTO authors (author_name) VALUES
41    ('Jane Austen'),
42    ('George Orwell');
43
44  INSERT INTO genres (genre_name) VALUES
45    ('Novel'),
46    ('Historical Fiction');
47
48  INSERT INTO users (username, email) VALUES
49    ('John Smith', 'john@example.com'),
50    ('Emily Johnson', 'emily@example.com');
51
52  INSERT INTO books (title, publication_year, author_id, genre_id) VALUES
53    ('Pride and Prejudice', 1813, 1, 1),
54    ('1984', 1949, 2, 2);
55
56  INSERT INTO borrowed_books (book_id, user_id, borrow_date, return_date) VALUES
57    (1, 1, '2024-03-28', '2024-04-10'),
58    (2, 2, '2024-03-25', '2024-04-05');
59
```

100% 1:59

Action Output

	Time	Action	Response	Duration / Fetch Time
8	23:42:52	INSERT INTO authors (author_name) VAL...	2 row(s) affected Records: 2 Duplicates: 0...	0.0025 sec
9	23:42:52	INSERT INTO genres (genre_name) VALU...	2 row(s) affected Records: 2 Duplicates: 0...	0.0031 sec
10	23:42:52	INSERT INTO users (username, email) VA...	2 row(s) affected Records: 2 Duplicates: 0...	0.0033 sec
11	23:42:52	INSERT INTO books (title, publication_ye...	2 row(s) affected Records: 2 Duplicates: 0...	0.0054 sec
12	23:42:52	INSERT INTO borrowed_books (book_id,...	2 row(s) affected Records: 2 Duplicates: 0...	0.0035 sec

3. Перейдіть до бази даних, з якою працювали у темі 3. Напишіть запит за допомогою операторів FROM та INNER JOIN, що об'єднує всі таблиці даних, які ми завантажили з файлів: order\_details, orders, customers, products, categories, employees, shippers, suppliers. Для цього ви маєте знайти спільні ключі.  
Перевірте правильність виконання запиту.

Відповідь:

SELECT

```
order_details.id AS order_details_id,  
order_details.quantity,  
orders.id AS order_id,  
orders.date AS order_date,  
customers.*,  
products.id AS products_id,  
products.name AS products_name,  
products.unit AS products_unit,  
products.price AS products_price,  
categories.*,  
employees.*,  
shippers.*,  
suppliers.*
```

FROM order\_details

INNER JOIN orders ON order\_details.order\_id = orders.id

INNER JOIN customers ON orders.customer\_id = customers.id

INNER JOIN products ON order\_details.product\_id = products.id

INNER JOIN categories ON products.category\_id = categories.id

INNER JOIN employees ON orders.employee\_id = employees.employee\_id

INNER JOIN shippers ON orders.shipper\_id = shippers.id

INNER JOIN suppliers ON products.supplier\_id = suppliers.id

The screenshot shows a database client interface with a SQL query editor and a results grid. The query is a complex JOIN statement combining multiple tables. The results grid displays 11 columns: order\_details\_id, quantity, order\_id, order\_date, name, contact, address, city, postal\_code, country, products\_id, products\_name, products\_unit, products\_price, id, name, and description. The first few rows of data are visible, showing details for various products and their suppliers.

order_details_id	quantity	order_id	order_date	name	contact	address	city	postal_code	country	products_id	products_name	products_unit	products_price	id	name	description
163	5	10308	1996-09-18	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitucion 2222	Mexico D.F.	05021	Mexico	70	Outback Lager	24 - 355 ml bottles	15	1	Beverages	Soft drink
162	1	10308	1996-09-18	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitucion 2222	Mexico D.F.	05021	Mexico	69	Gudbrandsdalsost	10 kg pkg.	36	4	Dairy Products	Cheeses
314	24	10365	1996-11-27	Antonio Moreno Taqueria	Antonio Moreno	Matadero 2312	Mexico D.F.	05023	Mexico	11	Queso Cabrales	1 kg pkg.	21	4	Dairy Products	Cheeses
360	20	10383	1996-12-16	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	56	Gnocchi di nonna Alice	24 - 250 g pkgs.	36	5	Grains/Cereals	Breads, c
286	25	10355	1996-11-15	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	57	Ravioli Angelo	24 - 250 g pkgs.	19.5	5	Grains/Cereals	Breads, c

Result 11

Action Output

Time	Action	Response	Duration / Fetch Time
02:24:58	SELECT	order_details.id AS order_details_id, order_details.quantity, orders.id AS order_id, orders.date AS order_date, customers.*, products.id AS products_id, products.name AS products_name, products.unit AS products_unit, products.price AS products_price, categories.*, employees.*, shippers.*, suppliers.*	518 row(s) returned 0.150 sec / 0.018 sec

#### 4. Виконайте запити, перелічені нижче.

Визначте, скільки рядків ви отримали (за допомогою оператора COUNT).



Не забувайте робити скриншоти результатів і запитів

Відповідь:

```
SELECT COUNT(*) AS row_count
FROM order_details
```

The screenshot shows the SQL Developer interface. The query editor at the top contains the following SQL code:

```
23 SELECT
24   COUNT(*) AS row_count
25 FROM
26   order_details
27
```

Below the query editor, the 'Result Grid' is displayed, showing a single row with the value 518 in the 'row\_count' column. The 'Action Output' pane at the bottom shows the execution details:

Time	Action	Response	Duration / Fetch Time
02:40:15	SELECT order_details.id AS order_details_id, order_details.quantity, orders.id AS order_id, orders.date AS order_date, customers.*, products.id AS products_id, products.n...	518 row(s) returned	0.072 sec / 0.014 sec
02:40:15	SELECT COUNT(*) AS row_count FROM order_details LIMIT 0, 1000	1 row(s) returned	0.0011 sec / 0.00001...

Змініть декілька операторів INNER на LEFT чи RIGHT. Визначте, що відбувається з кількістю рядків. Чому? Напишіть відповідь у текстовому файлі.

Відповідь:

```
SELECT COUNT(*) AS row_count
FROM order_details
LEFT JOIN orders ON order_details.order_id = orders.id
RIGHT JOIN customers ON orders.customer_id = customers.id
LEFT JOIN products ON order_details.product_id = products.id
RIGHT JOIN categories ON products.category_id = categories.id
INNER JOIN employees ON orders.employee_id = employees.employee_id
INNER JOIN shippers ON orders.shipper_id = shippers.id
INNER JOIN suppliers ON products.supplier_id = suppliers.id;
```

The screenshot shows the SQL Developer interface with a complex query in the query editor:

```
28 SELECT COUNT(*) AS row_count
29 FROM order_details
30 LEFT JOIN orders ON order_details.order_id = orders.id
31 RIGHT JOIN customers ON orders.customer_id = customers.id
32 LEFT JOIN products ON order_details.product_id = products.id
33 RIGHT JOIN categories ON products.category_id = categories.id
34 INNER JOIN employees ON orders.employee_id = employees.employee_id
35 INNER JOIN shippers ON orders.shipper_id = shippers.id
36 INNER JOIN suppliers ON products.supplier_id = suppliers.id;
37
```

The 'Result Grid' shows a single row with the value 518. The 'Action Output' pane shows the execution details for multiple steps:

Time	Action	Response	Duration / Fetch Time
02:40:15	SELECT COUNT(*) AS row_count FROM order_details LIMIT 0, 1000	1 row(s) returned	0.0011 sec / 0.00001...
02:44:41	SELECT order_details.id AS order_details_id, order_details.quantity, orders.id AS order_id, orders.date AS order_date, customers.*, products.id AS products_id, products.n...	518 row(s) returned	0.091 sec / 0.012 sec
02:44:41	SELECT COUNT(*) AS row_count FROM order_details INNER JOIN orders ON order_details.order_id = orders.id LEFT JOIN customers ON orders.customer_id = customers.id RIGH...	1 row(s) returned	0.121 sec / 0.00009...
02:45:40	SELECT order_details.id AS order_details_id, order_details.quantity, orders.id AS order_id, orders.date AS order_date, customers.*, products.id AS products_id, products.n...	518 row(s) returned	0.013 sec / 0.003 sec
02:47:47	SELECT order_details.id AS order_details_id, order_details.quantity, orders.id AS order_id, orders.date AS order_date, customers.*, products.id AS products_id, products.n...	518 row(s) returned	0.096 sec / 0.014 sec
02:47:47	SELECT COUNT(*) AS row_count FROM order_details LIMIT 0, 1000	1 row(s) returned	0.0034 sec / 0.00001...
02:47:47	SELECT COUNT(*) AS row_count FROM order_details LEFT JOIN orders ON order_details.order_id = orders.id RIGHT JOIN customers ON orders.customer_id = customers.id LEFT JOIN...	1 row(s) returned	0.132 sec / 0.000011...

27	
28	SELECT *
29	FROM order_details
30	RIGHT JOIN orders ON order_details.order_id = orders.id
31	RIGHT JOIN customers ON orders.customer_id = customers.id
32	RIGHT JOIN products ON order_details.product_id = products.id
33	RIGHT JOIN categories ON products.category_id = categories.id
34	RIGHT JOIN employees ON orders.employee_id = employees.employee_id
35	RIGHT JOIN shippers ON orders.shipper_id = shippers.id
36	RIGHT JOIN suppliers ON products.supplier_id = suppliers.id;
37	

62:32	
-------	--

Time	Action	Response	Duration / Fetch Time
02:54:20	SELECT * FROM order_details LEFT JOIN orders ON order_details.order_id = orders.id LEFT JOIN customers ON orders.customer_id = customers.id LEFT JOIN products ON order_details.p...	518 row(s) returned	0.0046 sec / 0.0075...
02:54:51	SELECT * FROM order_details INNER JOIN orders ON order_details.order_id = orders.id INNER JOIN customers ON orders.customer_id = customers.id INNER JOIN products ON order_det...	518 row(s) returned	0.137 sec / 0.023 sec
02:55:14	SELECT * FROM order_details RIGHT JOIN orders ON order_details.order_id = orders.id RIGHT JOIN customers ON orders.customer_id = customers.id RIGHT JOIN products ON order_det...	Error Code: 2013. Lost connection to MySQL server d...	30.003 sec

Зміна операторів INNER JOIN на LEFT або RIGHT JOIN впливає на те, як включаються рядки з таблиць у результат запиту. При LEFT JOIN у результаті залишаються всі рядки з першої (лівої) таблиці, навіть якщо в них немає відповідностей у другій (правій) таблиці. При RIGHT JOIN у результаті залишаються всі рядки з другої (правої) таблиці, навіть якщо вони не мають відповідностей у першій (лівій) таблиці. Це може змінювати кількість рядків у вихідному наборі даних залежно від наявності відповідностей між таблицями.

Оберіть тільки ті рядки, де employee\_id > 3 та ≤ 10.

Відповідь:

WHERE employees.employee\_id>3 AND employees.employee\_id<=10;

Find

Replace

Found match

Q- supplier

Done

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

SELECT

order\_details.id AS order\_details\_id,

order\_details.quantity,

orders.id AS order\_id,

orders.date AS order\_date,

customers.\*,

products.id AS products\_id,

products.name AS products\_name,

products.unit AS products\_unit,

products.price AS products\_price,

categories.\*,

employees.\*,

shippers.\*,

suppliers.\*

FROM order\_details

INNER JOIN orders ON order\_details.order\_id = orders.id

INNER JOIN customers ON orders.customer\_id = customers.id

INNER JOIN products ON order\_details.product\_id = products.id

INNER JOIN categories ON products.category\_id = categories.id

INNER JOIN employees ON orders.employee\_id = employees.employee\_id

INNER JOIN shippers ON orders.shipper\_id = shippers.id

INNER JOIN suppliers ON products.supplier\_id = suppliers.id

WHERE employees.employee\_id>3 AND employees.employee\_id<=10;

1:24

Result Grid

Filter Rows: Search

Export: CSV

Read Only

order_details_id	quantity	order_id	order_date	id	name	contact	address	city	postal_code	country	products_id	products_name	products_unit	products_price	id	name	descriptio
163	5	10308	1996-09-18	2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitucion 2222	Mexico D.F.	05021	Mexico	70	Outback Lager	24 - 355 ml bottles	15	1	Beverages	Soft drinks
162	1	10308	1996-09-18	2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitucion 2222	Mexico D.F.	05021	Mexico	69	Gudbrandsdalsost	10 kg pkg.	36	4	Dairy Products	Cheeses
358	20	10383	1996-12-16	4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	13	Konbu	2 kg box	6	8	Seafood	Seaweed
360	20	10383	1996-12-16	4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	56	Gnocchi di nonna Alice	24 - 250 g pkgs.	38	5	Grains/Cereals	Breads, cr
286	25	10355	1996-11-15	4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	57	Ravioli Angelo	24 - 250 g pkgs.	19.5	5	Grains/Cereals	Breads, cr

Result 32

Action Output

Time

Action

Response

Duration / Fetch Time

1

03:10:32

SELECT order\_details.id AS order\_details\_id, order\_details.quantity, orders.id AS order\_id, orders.date AS order\_date, customers.\*, products.id AS products\_id, products.name...

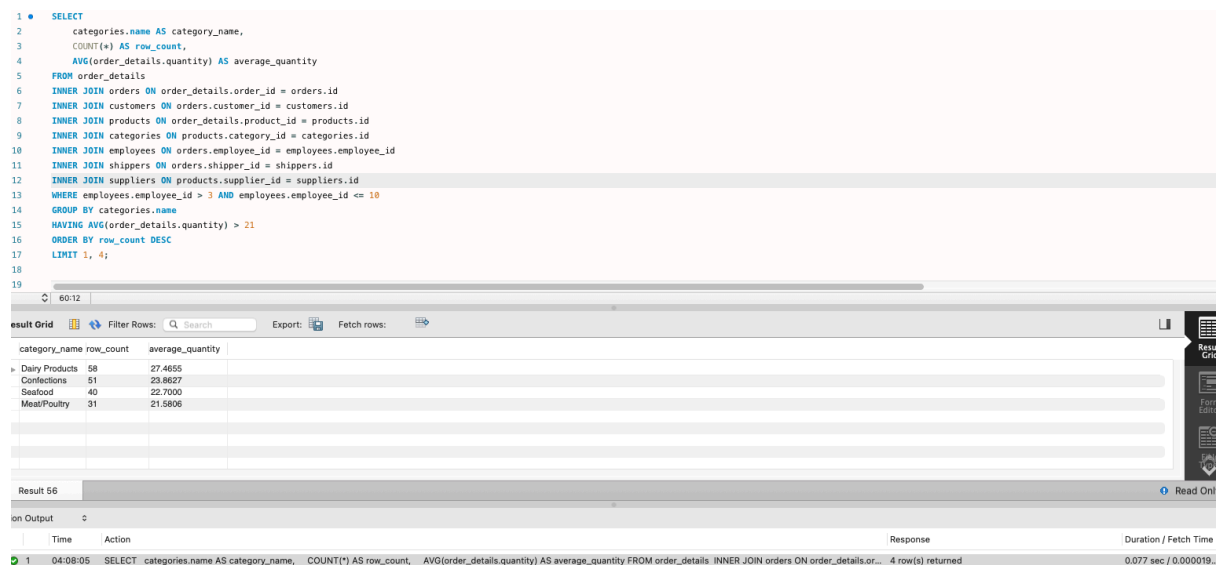
317 row(s) returned

0.105 sec / 0.0046 sec

Згрупуйте за іменем категорії (GROUP BY categories.name),  
порахуйте кількість рядків у групі (COUNT(\*) AS row\_count),  
середню кількість товару (кількість товару знаходиться в  
order\_details.quantity)(AVG(order\_details.quantity) AS average\_quantity)  
Відфільтруйте рядки, де середня кількість товару більша за 21.(HAVING  
AVG(order\_details.quantity) > 21)  
Відсортуйте рядки за спаданням кількості рядків.(ORDER BY row\_count DESC)  
Виведіть на екран (оберіть) чотири рядки з пропущеним першим рядком.(LIMIT  
1, 4;)

Відповідь:

```
SELECT
    categories.name AS category_name,
    COUNT(*) AS row_count,
    AVG(order_details.quantity) AS average_quantity
FROM order_details
INNER JOIN orders ON order_details.order_id = orders.id
INNER JOIN customers ON orders.customer_id = customers.id
INNER JOIN products ON order_details.product_id = products.id
INNER JOIN categories ON products.category_id = categories.id
INNER JOIN employees ON orders.employee_id = employees.employee_id
INNER JOIN shippers ON orders.shipper_id = shippers.id
INNER JOIN suppliers ON products.supplier_id = suppliers.id
WHERE employees.employee_id > 3 AND employees.employee_id <= 10
GROUP BY categories.name
HAVING AVG(order_details.quantity) > 21
ORDER BY row_count DESC
LIMIT 1, 4;
```



The screenshot shows a database query editor with the following SQL query:

```
1 SELECT
2     categories.name AS category_name,
3     COUNT(*) AS row_count,
4     AVG(order_details.quantity) AS average_quantity
5 FROM order_details
6 INNER JOIN orders ON order_details.order_id = orders.id
7 INNER JOIN customers ON orders.customer_id = customers.id
8 INNER JOIN products ON order_details.product_id = products.id
9 INNER JOIN categories ON products.category_id = categories.id
10 INNER JOIN employees ON orders.employee_id = employees.employee_id
11 INNER JOIN shippers ON orders.shipper_id = shippers.id
12 INNER JOIN suppliers ON products.supplier_id = suppliers.id
13 WHERE employees.employee_id > 3 AND employees.employee_id <= 10
14 GROUP BY categories.name
15 HAVING AVG(order_details.quantity) > 21
16 ORDER BY row_count DESC
17 LIMIT 1, 4;
```

The results are displayed in a table grid with the following data:

category_name	row_count	average_quantity
Dairy Products	58	27.4655
Confections	51	23.8627
Seafood	40	22.7000
Meat/Poultry	31	21.5806

The bottom of the screenshot shows a log of the query execution:

```
1 04:08:05 SELECT categories.name AS category_name, COUNT(*) AS row_count, AVG(order_details.quantity) AS average_quantity FROM order_details INNER JOIN orders ON order_details.or... 4 row(s) returned 0.077 sec / 0.000019s
```