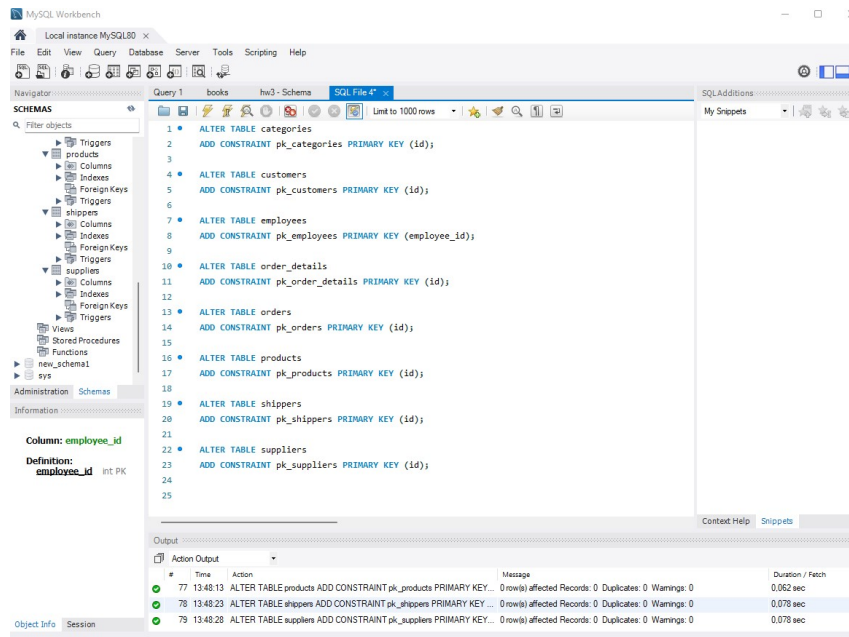
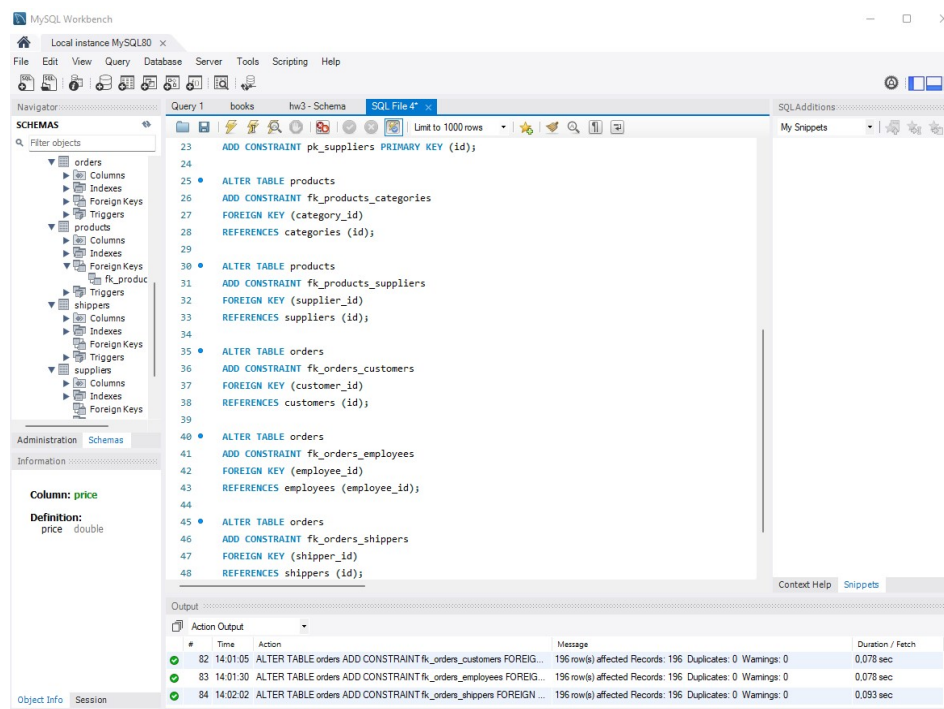


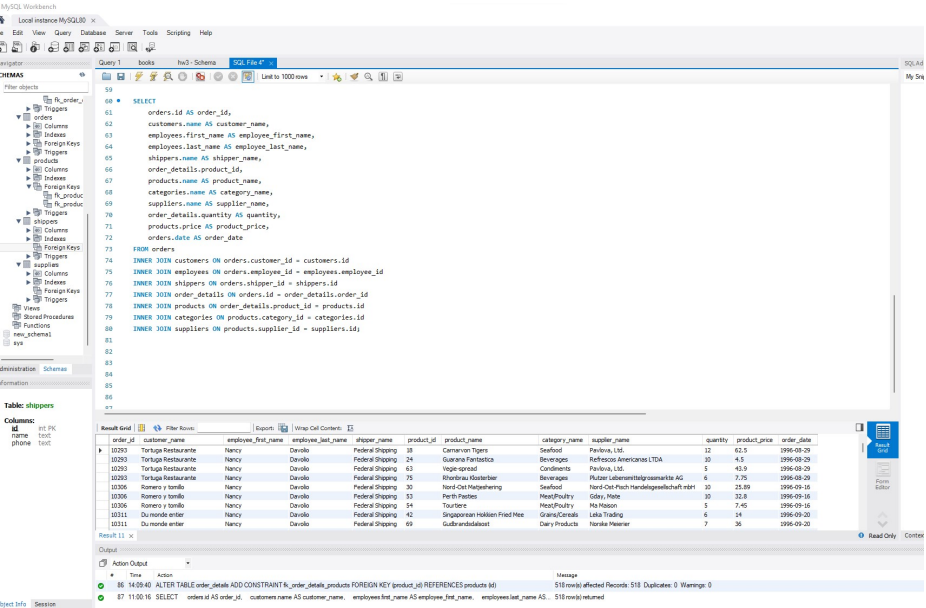
### 3.1. Спочатку ми додаємо первинні ключі до кожної з таблиць:



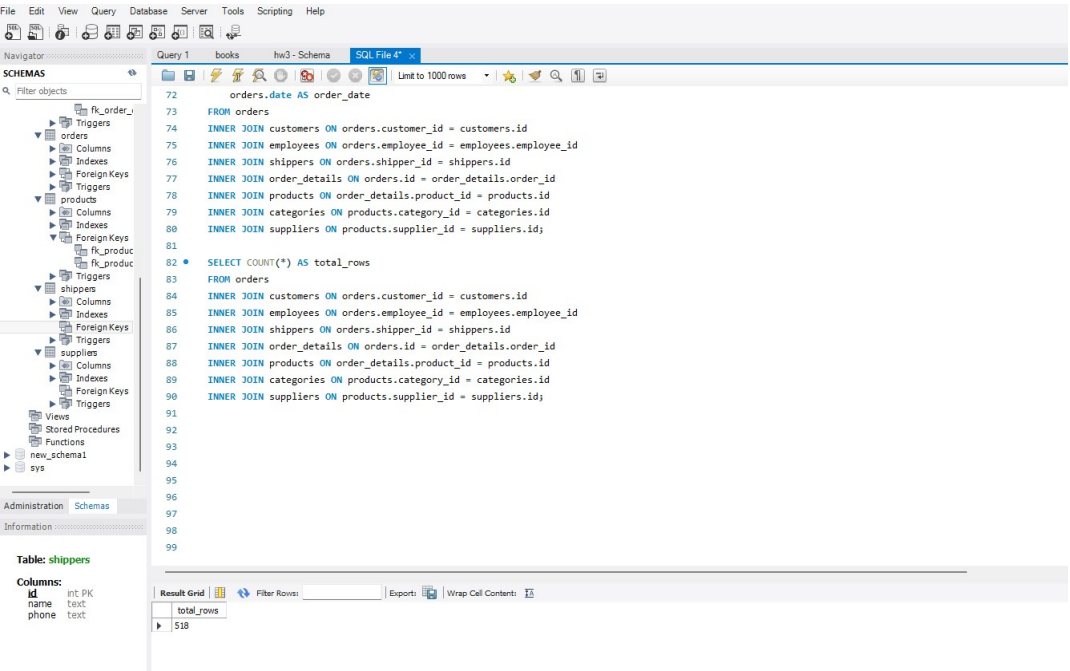
### 3.2. Щоб створити зв'язки між таблицями, ми використовуємо ALTER TABLE із додаванням зовнішнього ключа (FOREIGN KEY).



### 3.3. Запит, який об'єднує всі зазначені таблиці, використовуючи INNER JOIN і спільні ключі



### 4.1. Кількість рядків у об'єднаних таблицях



#### 4.2. Заміна декількох операторів INNER на LEFT чи RIGHT.

hw3

- Tables
  - borrowed\_books
  - books
  - authors
  - categories
    - Columns
      - id
      - name
      - description
    - Indexes
    - PRIMARY
    - ForeignKeys
    - Triggers
  - customers
    - Columns
      - employee\_id
      - last\_name
      - first\_name
      - birthdate
      - photo
      - notes
    - Indexes
    - ForeignKeys
- administration
  - Schemas
- information
  - Columns:
    - employee\_id
    - last\_name
    - first\_name
    - birthdate
    - photo
    - notes

```

83 FROM orders
84 INNER JOIN customers ON orders.customer_id = customers.id
85 INNER JOIN employees ON orders.employee_id = employees.employee_id
86 INNER JOIN shippers ON orders.shipper_id = shippers.id
87 INNER JOIN order_details ON orders.id = order_details.order_id
88 INNER JOIN products ON order_details.product_id = products.id
89 INNER JOIN categories ON products.category_id = categories.id
90 INNER JOIN suppliers ON products.supplier_id = suppliers.id;
91
92 SELECT COUNT(*) AS total_rows
93 FROM orders;
94 LEFT JOIN customers ON orders.customer_id = customers.id
95 RIGHT JOIN employees ON orders.employee_id = employees.employee_id
96 LEFT JOIN shippers ON orders.shipper_id = shippers.id
97 LEFT JOIN order_details ON orders.id = order_details.order_id
98 RIGHT JOIN products ON order_details.product_id = products.id
99 LEFT JOIN categories ON products.category_id = categories.id
100 INNER JOIN suppliers ON products.supplier_id = suppliers.id;
101
102 SELECT *
103 FROM employees
104 WHERE employee_id > 3 AND employee_id <= 10;
105
106 SELECT
107     categories.name AS category_name,
108     COUNT(*) AS total_rows,
109     AVG(order_details.quantity) AS average_quantity
110 FROM order_details
111 INNER JOIN products ON order_details.product_id = products.id
112 INNER JOIN categories ON products.category_id = categories.id
  
```

Result Grid Filter Rows:  | Export: Wrap Cell Contents: ☐

total_rows	
1	518

Result 76 x

Output

Action Output

#	Time	Action	Message
162	16:21:23	SELECT order_details FROM order_details LEFT JOIN orders ON order_details.order_id = orders.id WHERE orders.id IS NULL LIMIT 0, 1000	0 rows(s) returned
163	16:22:28	SELECT COUNT(*) AS total_rows FROM orders LEFT JOIN customers ON orders.customer_id = customers.id RIGHT JOIN employees ON orders.employee_id = employees.id	1 row(s) returned

#### 4.3. Рядки, де employee id > 3 та ≤ 10

The screenshot displays the Microsoft Access interface, showing the database schema and the results of a SQL query.

**Database Schema:**

- categories**: Includes Columns, Indexes, PRIMARY, Foreign Keys, Triggers.
- customers**: Includes Columns, Indexes, Foreign Keys, Triggers.
- employees**: Includes Columns, Indexes, Foreign Keys, Triggers.
- order\_details**: Includes Columns, Indexes, Foreign Keys, Triggers.
- orders**: Includes Columns, Indexes, Foreign Keys, Triggers.
- Schemas**: A section for managing database schemas.

**SQL Query:**

```
SELECT *  
FROM employees  
WHERE employee_id > 3 AND employee_id <= 10;
```

**Query Results:**

employee_id	last_name	first_name	birthdate	photo	notes
4	Peacock	Margaret	1958-09-19	Emp024.jpg	Margaret holds a BA in English literature from C...
5	Buchanan	Steven	1955-03-04	Emp025.jpg	Steven Buchanan graduated from St. Andrews ...
6	Suyama	Michael	1962-07-02	Emp026.jpg	Michael is a graduate of Susan University (M...
7	King	Robert	1960-05-29	Emp027.jpg	Robert King served in the Peace Corps and tra...
8	Cabotian	Laura	1958-01-09	Emp028.jpg	Laura received a BA in psychology from the Uni...
9	Doddsworth	Anne	1969-07-02	Emp029.jpg	Anne has a BA degree in English from St. Lane...
10	West	Adam	1929-09-19	Emp010.jpg	An old chum.

**Output:**

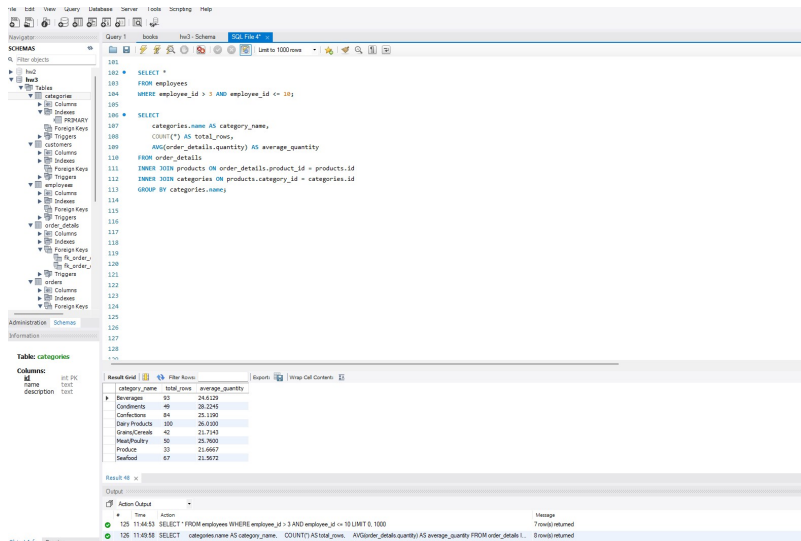
Action Output

Time Action Message

124 11:43:19 SELECT orders.id FROM orders LEFT JOIN employees ON orders.employee\_id = employees.employee\_id WHERE employees.employee... 0 records returned

125 11:44:53 SELECT \* FROM employees WHERE employee\_id > 3 AND employee\_id <= 10 LIMIT 0, 1000 7 records returned

#### 4.4. Групування за іменем категорії, рахуємо кількість рядків у групі та середню кількість товару (кількість товару знаходиться в order\_details.quantity)

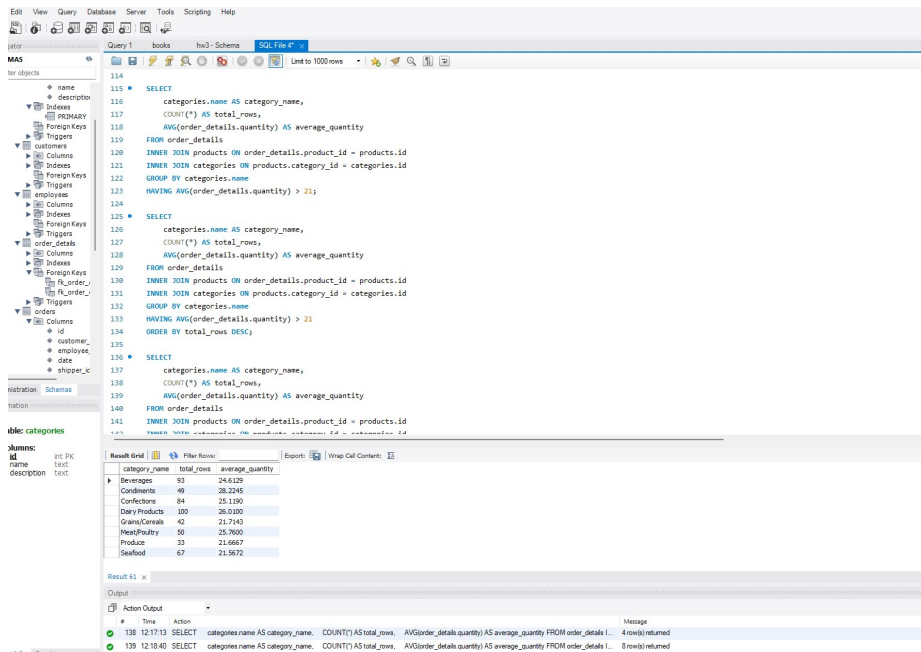


```
SELECT *
FROM employees
WHERE employee_id > 1 AND employee_id <= 10;

SELECT
  categories.name AS category_name,
  COUNT(*) AS total_rows,
  AVG(order_details.quantity) AS average_quantity
FROM order_details
INNER JOIN products ON order_details.product_id = products.id
INNER JOIN categories ON products.category_id = categories.id
GROUP BY categories.name;
```

category_name	total_rows	average_quantity
Beverages	93	24.6129
Confections	49	26.2245
Confections	84	25.1190
Dairy Products	100	26.0100
GrainsCereals	42	21.7143
Meat/Seafood	50	25.7600
Produce	33	21.6667
Seafood	67	21.5072

#### 4.5. Фільтруємо рядки, де середня кількість товару більша за 21.



```
SELECT
  categories.name AS category_name,
  COUNT(*) AS total_rows,
  AVG(order_details.quantity) AS average_quantity
FROM order_details
INNER JOIN products ON order_details.product_id = products.id
INNER JOIN categories ON products.category_id = categories.id
GROUP BY categories.name
HAVING AVG(order_details.quantity) > 21;

SELECT
  categories.name AS category_name,
  COUNT(*) AS total_rows,
  AVG(order_details.quantity) AS average_quantity
FROM order_details
INNER JOIN products ON order_details.product_id = products.id
INNER JOIN categories ON products.category_id = categories.id
GROUP BY categories.name
HAVING AVG(order_details.quantity) > 21
ORDER BY total_rows DESC;
```

category_name	total_rows	average_quantity
Beverages	93	24.6129
Confections	49	26.2245
Confections	84	25.1190
Dairy Products	100	26.0100
GrainsCereals	42	21.7143
Meat/Seafood	50	25.7600
Produce	33	21.6667
Seafood	67	21.5072

4.6. Сортуюмо рядки за спаданням кількості рядків.

The screenshot shows a SQL query in SQL Studio. The query is as follows:

```
120 INNER JOIN products ON order_details.product_id = products.id
121 INNER JOIN categories ON products.category_id = categories.id
122 GROUP BY categories.name
123 HAVING AVG(order_details.quantity) > 21;
124
125 SELECT
126     categories.name AS category_name,
127     COUNT(*) AS total_rows,
128     AVG(order_details.quantity) AS average_quantity
129 FROM order_details
130 INNER JOIN products ON order_details.product_id = products.id
131 INNER JOIN categories ON products.category_id = categories.id
132 GROUP BY categories.name
133 HAVING AVG(order_details.quantity) > 21
134 ORDER BY total_rows DESC;
135
136 SELECT
137     categories.name AS category_name,
138     COUNT(*) AS total_rows,
139     AVG(order_details.quantity) AS average_quantity
140 FROM order_details
141 INNER JOIN products ON order_details.product_id = products.id
142 INNER JOIN categories ON products.category_id = categories.id
143 GROUP BY categories.name
144 HAVING AVG(order_details.quantity) > 21
145 ORDER BY total_rows DESC
146 LIMIT 4 OFFSET 1;
```

The results table shows the following data:

category_name	total_rows	average_quantity
Dairy Products	100	26.0100
Beverages	93	24.6129
Confections	84	25.1190
Seafood	67	21.5672
Meat/Poultry	50	25.7600
Condiments	49	28.2245
Grains/Cereals	42	21.7143
Produce	33	21.6667

4.7. Виводимо на екран чотири рядки з пропущеним першим рядком.

The screenshot shows the same SQL query as in 4.6, but with the LIMIT clause changed to LIMIT 4 OFFSET 1. The results table shows the following data:

category_name	total_rows	average_quantity
Beverages	93	24.6129
Confections	84	25.1190
Seafood	67	21.5672
Meat/Poultry	50	25.7600