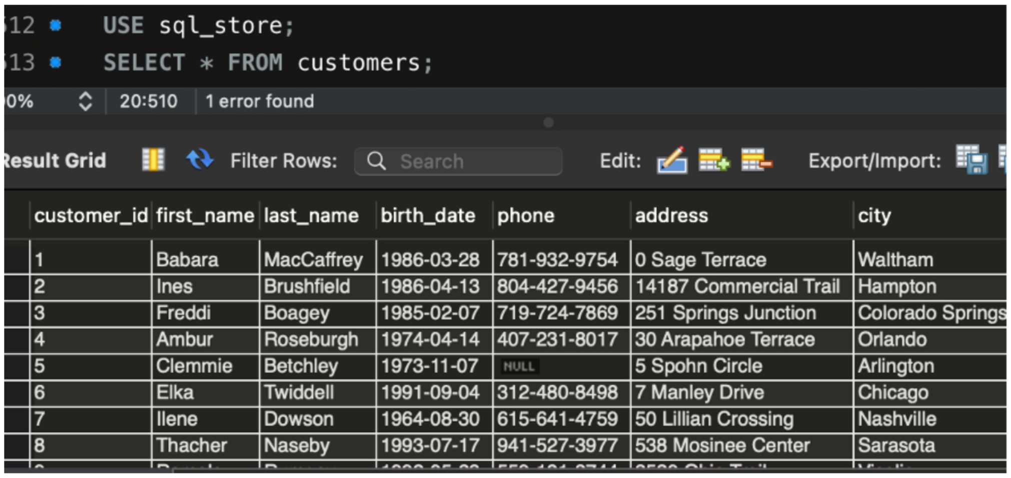
Querying SQL store & inventory databases

Querying SQL store and inventory databases involves using SQL (Structured Query Language) to retrieve, update, and manage data related to products, stock levels, sales, and transactions.



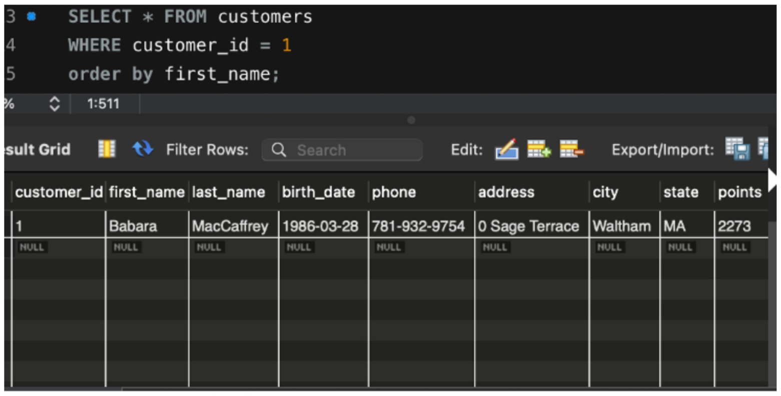
```
USE sql_store;
SELECT * FROM customers;
```

customer_id	first_name	last_name	birth_date	phone	address	city
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota

Select all from

customers.

The query selects all customer information from the customer table in sql_store database.

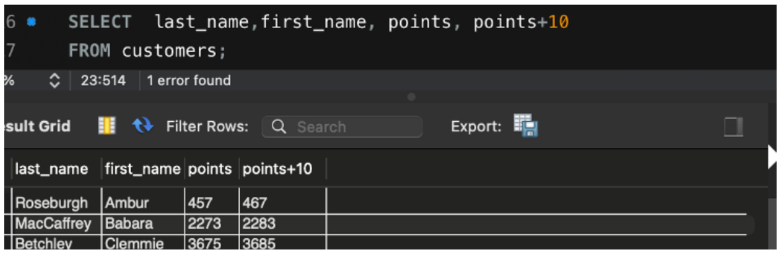


```
SELECT * FROM customers
WHERE customer_id = 1
order by first_name;
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Details of the customer with ID =1.

The query selects all customer information from the customers table for customer with ID=1

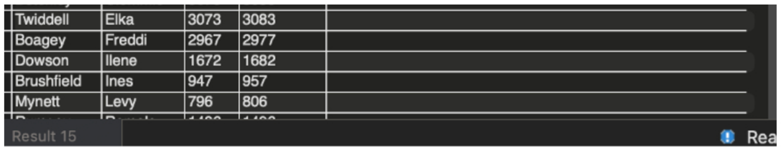


```
SELECT  last_name,first_name, points, points+10
FROM customers;
```

last_name	first_name	points	points+10
Roseburgh	Ambur	457	467
MacCaffrey	Babara	2273	2283
Betchley	Clemmie	3675	3685

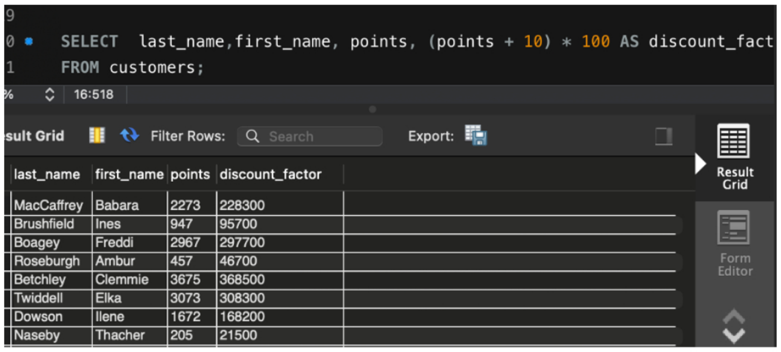
Select some columns from the customer table

The query selects firstname, lastname, points, points+10 information from the customer table



Twiddell	Elka	3073	3083
Boagey	Freddi	2967	2977
Dowson	Ilene	1672	1682
Brushfield	Ines	947	957
Mynett	Levy	796	806

Result 15

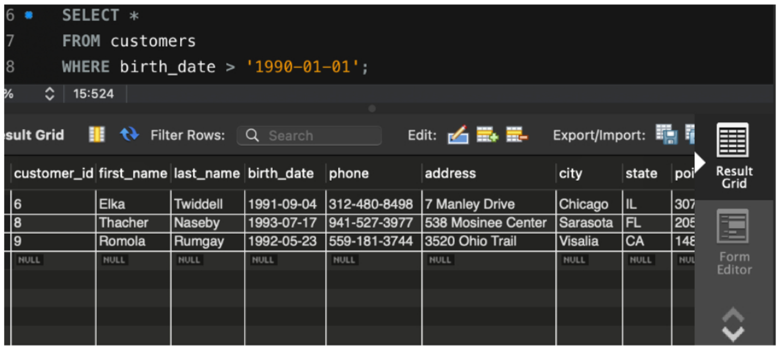


```
SELECT  last_name,first_name, points, (points + 10) * 100 AS discount_fact
FROM customers;
```

last_name	first_name	points	discount_factor
MacCaffrey	Babara	2273	228300
Brushfield	Ines	947	95700
Boagey	Freddi	2967	297700
Roseburgh	Ambur	457	46700
Betchley	Clemmie	3675	368500
Twiddell	Elka	3073	308300
Dowson	Ilene	1672	168200
Naseby	Thacher	205	21500

Calculate Discount factor

The query selects the last name, first name, points, and a calculated discount factor ((points plus 10) multiplied by 100) from the customers table. This information can be used for marketing or sales strategies to offer personalized discounts or rewards.

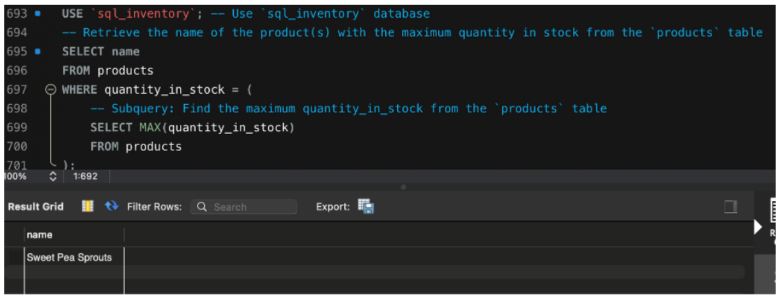


```
SELECT *
FROM customers
WHERE birth_date > '1990-01-01';
```

customer_id	first_name	last_name	birth_date	phone	address	city	state	poi
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	307
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	148
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NUL

Customers born after 1990-01-01

The query selects all columns from the customers table for customers whose birth date is after January 1, 1990. It identifies and retrieves information about customers who are relatively young or born after a specific date, which can be useful for targeted marketing or demographic analysis.

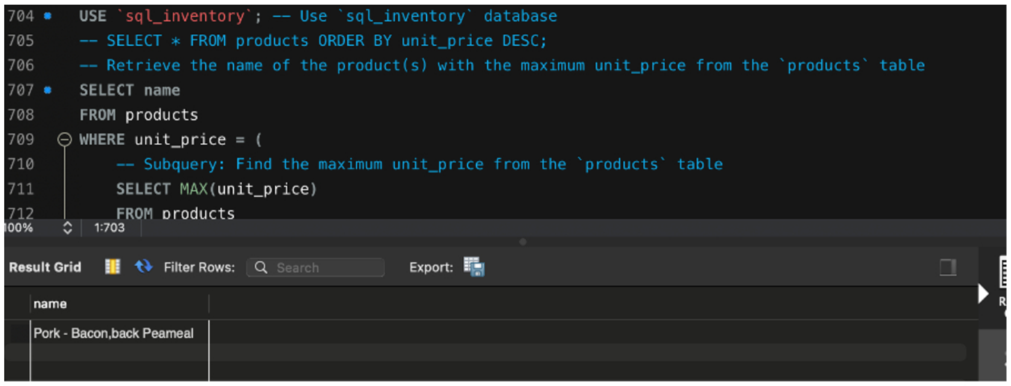


```
USE `sql_inventory`; -- Use `sql_inventory` database
-- Retrieve the name of the product(s) with the maximum quantity in stock from the `products` table
SELECT name
FROM products
WHERE quantity_in_stock = (
    -- Subquery: Find the maximum quantity_in_stock from the `products` table
    SELECT MAX(quantity_in_stock)
    FROM products
);
```

name
Sweet Pea Sprouts

Product with the most quantity in stock

This query retrieves the names of products that have the highest quantity in stock. This helps to identify the most well-stocked items for inventory management or restocking decisions.

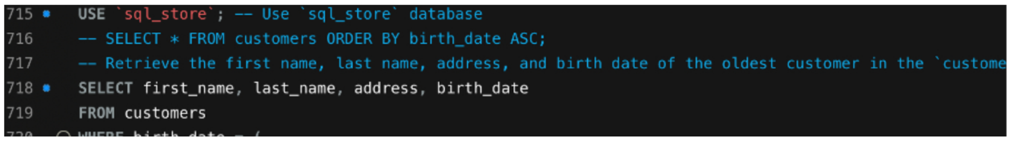


```
USE `sql_inventory`; -- Use `sql_inventory` database
-- SELECT * FROM products ORDER BY unit_price DESC;
-- Retrieve the name of the product(s) with the maximum unit_price from the `products` table
SELECT name
FROM products
WHERE unit_price = (
    -- Subquery: Find the maximum unit_price from the `products` table
    SELECT MAX(unit_price)
    FROM products
```

name
Pork - Bacon,back Peameal

Most expensive products

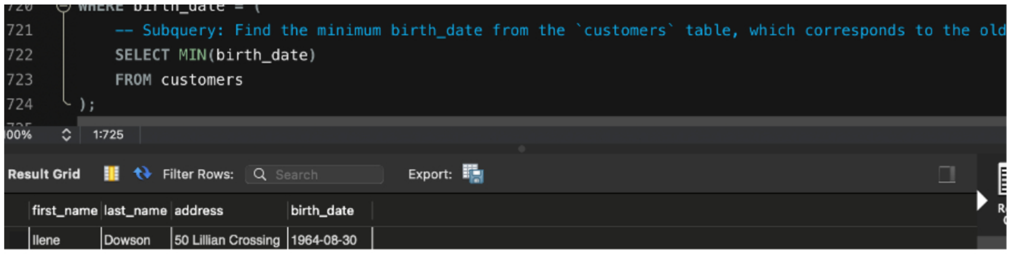
This query retrieves the names of products with the highest unit price. This is useful for identifying premium or high-value items in the inventory.



```
USE `sql_store`; -- Use `sql_store` database
-- SELECT * FROM customers ORDER BY birth_date ASC;
-- Retrieve the first name, last name, address, and birth date of the oldest customer in the `custome
SELECT first_name, last_name, address, birth_date
FROM customers
```

Oldest customer

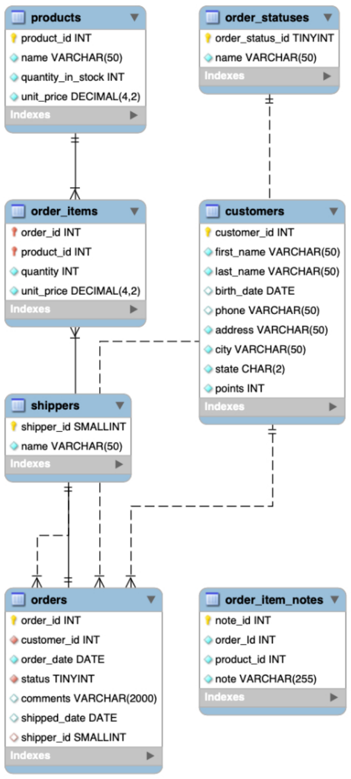
This query retrieves the details of the oldest



```
WHERE birth_date = (
    -- Subquery: Find the minimum birth_date from the `customers` table, which corresponds to the old
    SELECT MIN(birth_date)
    FROM customers
);
```

first_name	last_name	address	birth_date
Ilene	Dowson	50 Lillian Crossing	1964-08-30

customers based on the minimum birth date, which helps in identifying the most senior customers for personalised outreach or recognition.



The Entity-Relationship (ER) diagram shows a clear and visual representation of the sql_store database structure, including its entities, attributes, and relationships.

Database Description

In the sql_store database, there are 7 tables and 5 relationships. See below for the details.

1. The **customers** table is designed to store information about customers including customer_id, first_name, last_name, birth_date, phone, address, city, state, points. customer_id is the primary key of the table and ensures each customer has a unique identifier. It is a foreign key in the orders table.

   · The relationship is one-to-many: One customer can have many orders, but each order is associated with only one customer

2. The **orders** table stores information on customer orders. It is used to manage and track customer orders. order_id is the primary key of the table and ensures each order has a unique identifier. It has other columns including customer_id, order_date, status, comments, shipped_date, and shipper_id. The customer_id, status, order_statuses_order_status_id, shipper_id are foreign keys. Foreign key constraints enforce referential integrity by maintaining accurate relationships between orders and associated entities such as customers, order_status, order_statuses, and shippers.

In addition to the one-to-many relationship between the customers and orders tables, there are relationships

· many-to-one relationship between orders and order_statuses tables. One order status can be associated with many orders, but each order has only one status.
· many-to-one between orders and shippers tables. One shipper can handle many orders, but each order can be shipped by only one shipper.
· one-to-many relationship between the orders and order_items tables. One order can have many order items.

3. The **order_item** table has a composite key consisting or product_id and order_id to ensure that each product within an order is unique. In addition to the one-to-many relationship between orders and order_item, there is

· one-to-many relationship between order_item and products table. One product can appear in many order items, but each order item is associated with only one product.

4. The **products** table stores information on products including name and quantity. It is a foreign key in the order_items table.

5. The **shippers** table stores information about the shipping companies that handle the delivery of orders. Its primary key is shipper_id.

6. The **order_statuses** table holds different statuses that an order can have, such as pending, shipped, or delivered. Its primary key is order_status_id.

7.The **order_item_notes** table stores notes or comments related to specific order items. Its primary key is note_id. It does not have a relationship with other tables in this database