

NOVA

IMS

Information
Management
School

3

COMMON DATA TYPES AND SQL

Data Science for Marketing

© 2021-2022 Nuno António

Acreditações e Certificações



Summary

- 
1. Types of data
 2. Common semi-structured data formats
 3. Databases and SQL

3.1

Types of data

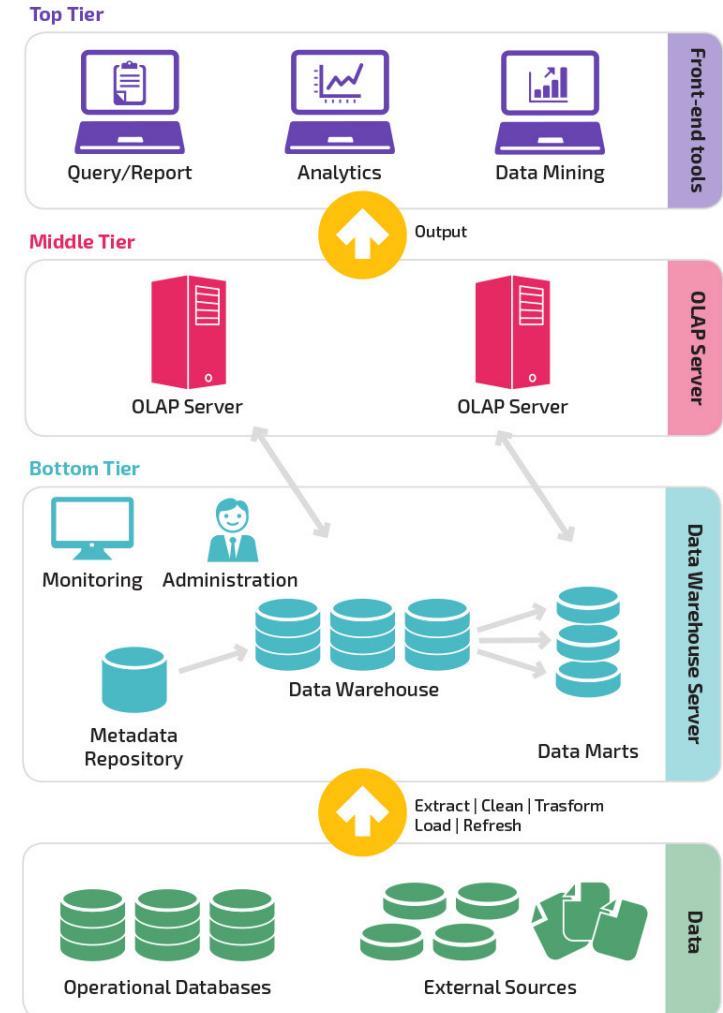
Common data types and SQL

Transactional data

- Records in a database or other type of data structure (e.g., CSV file) that capture any type of transaction, for example, a purchase
- Typically have a unique ID and are stored in a relational database, that is why many time relational databases are called transactional databases

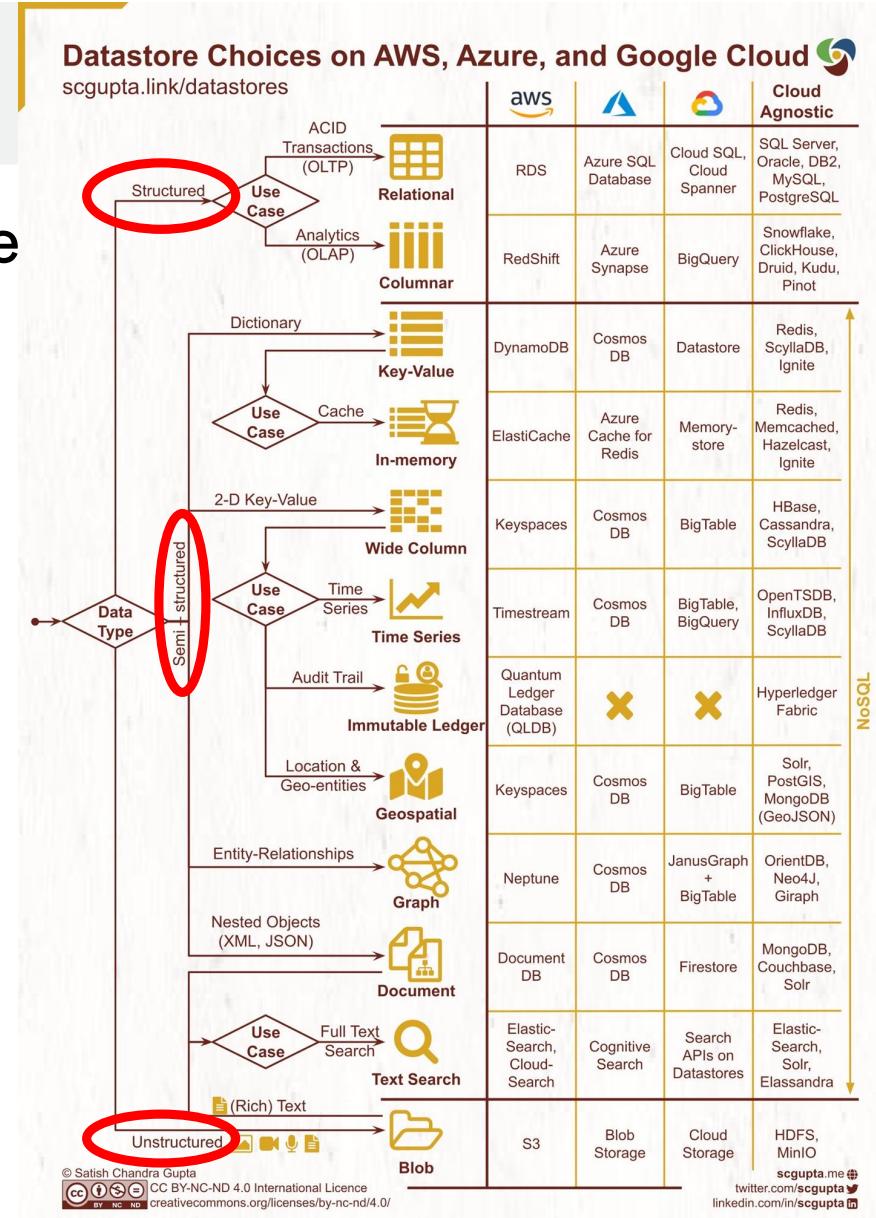
Datawarehouse

- Repository of data collected from multiple sources
- Stored under a unified schema (depends)
- Usually residing at a single site
- Usually modeled into a multidimensional database (aka "data cube")



Other data types

- Time-series data (e.g., stock exchange data)
- Data streams (e.g., video surveillance and sensor data)
- Spatial data (e.g., maps)
- Video
- Images
- Audio
- Semi-structured data (e.g., XML or JSON files)
- Other unstructured data (e.g., text or website content)



3.2

Common semi-structured data formats

Common data types and SQL

CSV

CSV files

- CSV stands for *Comma Separated Values*
- Typically, a CSV file is composed of:
 - One “header” row with the name of the variables/fields
 - Many rows of data (also known as records or observations)
- In header row or in the data rows, the fields are separated by commas
- In some cases, the separator could be other characters, such as a semi-colon (;), pipe (|), tab, or other characters
- In same cases, when the separator character is the tab, files are also called TSV files

Example

```
Name, Age, TotalPurchased, Recency, Frequency
Mario Costa, 48, 1000.00, 30, 20
Anne Marie, 27, 2132.00, 25, 10
Elisabeth James, 34, 3334.50, 10, 15
Paul Phillips, 54, 783.00, 5, 6
```

Header
First row

Data
Subsequent rows

XML

Origin

- XML stands for *Extensible Markup Language*
- Subset of SGML (Standard Generalized Markup Language) – ISO standard defined in 1986
- Not owned by any company or institution
- XML standards are defined by the W3C consortium (<http://www.w3.org>)

Why

- To structure data:
 - When data are out of context don't make sense or can be incorrectly interpreted
 - Structure data can be read and used by other applications
- To create a language to describe data (metadata)
- To help:
 - Enterprise integration
 - B2B ecommerce
 - Information distribution

What it solved

Organized data in a way that:

- Is simultaneously “human” and “machine” readable
- Defines the structure and the content
- Emphasizes relationships
- Separates structure from presentation
- Open and extensible

What is a “Markup language”?

Agency XYZ
Number 12345

02/12/2013

Room	5	20.00
SPA	1	99.70

199.70

BOOKING WITHOUT MARKUP

```
<Booking ID="12345">
<From>Agency XYZ</From>
<Arrival>02/12/2013</Arrival>
<Items>
  <Item>
    <Designation>Room</Designation>
    <Quantity>5</Quantity>
    <UnitPrice>20.00</UnitPrice>
  </Item>
  <Item>
    <Designation>SPA</Designation>
    <Quantity>1</Quantity>
    <UnitPrice>99.70</UnitPrice>
  </Item>
</Items>
<Total Currency="EUR">199.70</Total>
</Booking>
```

BOOKING WITH MARKUP

What is a “Markup language”?

- Is a markup because it describes data in an open and auto-descriptive way
- Is a language because it uses its own tags and enables the creation of new ones
- Is a “text file”

```
<Booking ID="12345">
<From>Agency XYZ</From>
<Arrival>02/12/2013</Arrival>
<Items>
  <Item>
    <Designation>Room</Designation>
    <Quantity>5</Quantity>
    <UnitPrice>20.00</UnitPrice>
  </Item>
  <Item>
    <Designation>SPA</Designation>
    <Quantity>1</Quantity>
    <UnitPrice>99.70</UnitPrice>
  </Item>
</Items>
<Total Currency="EUR">199.70</Total>
</Booking>
```

BOOKING WITH MARKUP

B2B Example: Booking.com reservation

```
<bookings hotelId="12123">
  <booking id="JU3XPE" distributorId="1234" currency="EUR"
    date="2014-03-24T11:53:11" totalAmount="856.0000" paidAmount="0.0000"
    dueAmount="856.0000" origin="Booking.com" originId="9192"
    lastModificationDate="2014-03-24T11:53:11" paxCount="2">
    <rooms>
      <room id="38673" uniqueId="01" status="Create">
        <stays>
          <stay from="2014-06-13" to="2014-06-21" quantity="1"
            unitPrice="107.0000" rateCode="BAR3" />
        </stays>
        <guests>
          <guest firstName="Brian" lastName="Black" title="MR" ageRange="0" />
          <guest firstName="Carol" lastName="Black" title="MRS" ageRange="0" />
        </guests>
      </room>
    </rooms>
    <customer firstName="Brian" lastName="Black" title="MR">
      <contact email="brian@aol.com" phone="4412345667">
        <address country="GB" />
      </contact>
    </customer>
  </booking>
</bookings>
```

JSON

What is

- JSON stands for *JavaScript Object Notation*
- A lightweight data-interchange format
- A syntax for storing and exchanging data
- Easier-to use alternative to XML
- A subset of JavaScript
- A “text file” of .json type
- Open standards defined in <http://www.json.org>

Data types

- **Number:** real, integer or floating
- **String:** any kind of text
- **Boolean:** true or false value
- **Null:** variable with out value
- **Object:** a collection of key-value pairs, separated by comma and enclosed in curly brackets
- **Array:** order sequence of values, separated by command and enclosed in brackets

Example JSON VS. XML

```
{  
    "bookings": [  
        {"id":1234,  
         "name":"John",  
         "arrival":"2018-02-01",  
         "nights":3  
        },  
        {"id":1235,  
         "name":"Mary",  
         "arrival":"2018-02-02",  
         "nights":4}  
    ]  
}
```

JSON

```
<bookings>  
    <booking>  
        <id>1234</id>  
        <name>John</name>  
        <arrival>2018-02-01</arrival>  
        <nights>3</nights>  
    </booking>  
    <booking>  
        <id>1235</id>  
        <name>Mary</name>  
        <arrival>2018-02-02</arrival>  
        <nights>4</nights>  
    </booking>  
</bookings>
```

XML

JSON VS XML

JSON

- Simpler to read and write
- Supports array data structure
- Easily read by “humans”
- Supports structure data through arrays and objects
- Native object support

XML

- Does not support arrays
- Tends to be more complicated for “human” reading
- Relies on XML schemas to define data types
- Objects are expressed by conventions (attributes and elements)

3.3

Databases and SQL

Common data types and SQL

Introduction

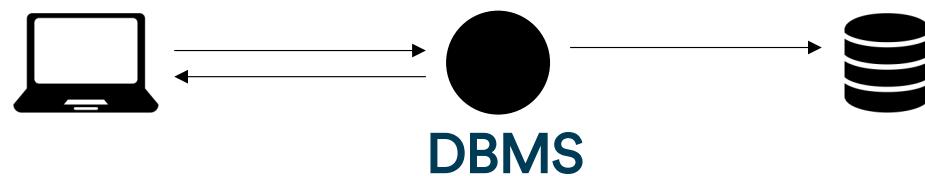
Databases and SQL

What is a database

- A database is an organized collection of data, composed of tables, queries, reports, views, and other objects
- Data is organized to model aspects of reality in a way that support processes requiring information, such as understand each customer spending

Database management systems (DBMS)

DBMS are the software that interfaces with applications, users, and the database itself to capture and analyze the data



DBMS

Relational

**Non
Relational**

Relational DBMS

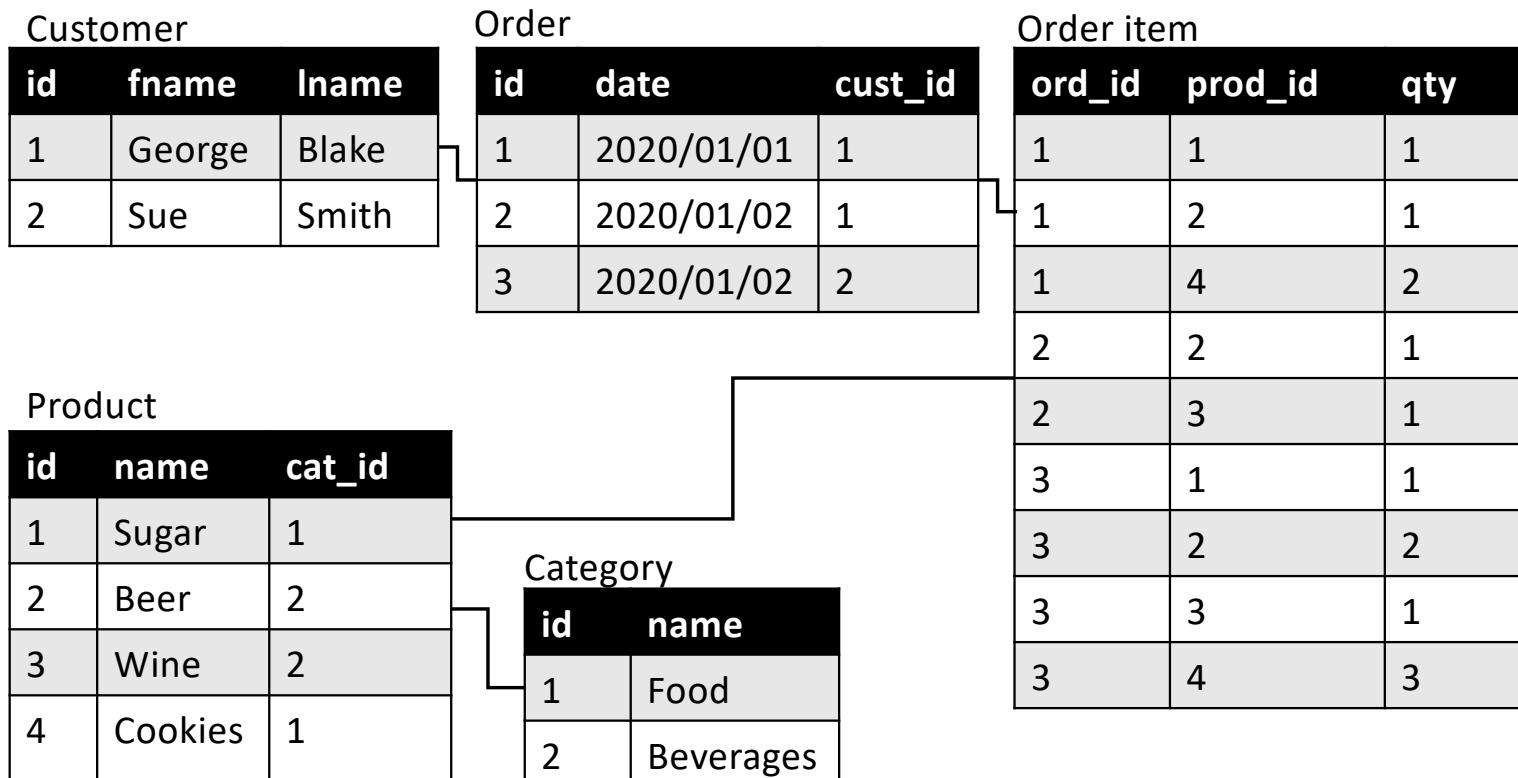


Most applications in business environments (e.g., ERP or CRM systems) store data in relational databases

Common DBMS documentation links

- Microsoft SQL: <https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15>
- MySQL: <https://dev.mysql.com/doc/>
- Oracle: <https://docs.oracle.com/en/database/oracle/oracle-database/>
- Postgresql: <https://www.postgresql.org/docs/>

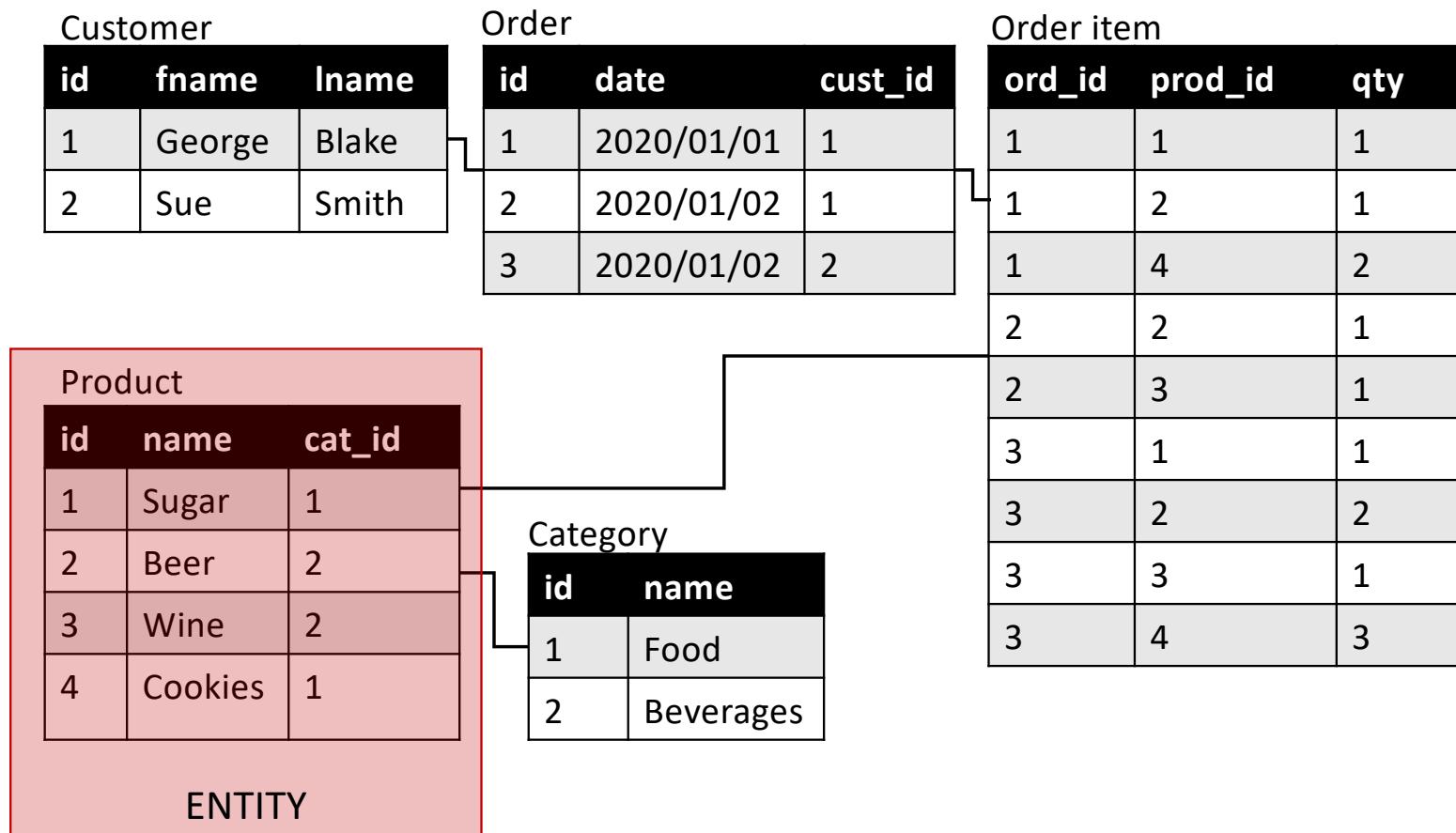
Relational database



ERD – Entity Relationship Diagram

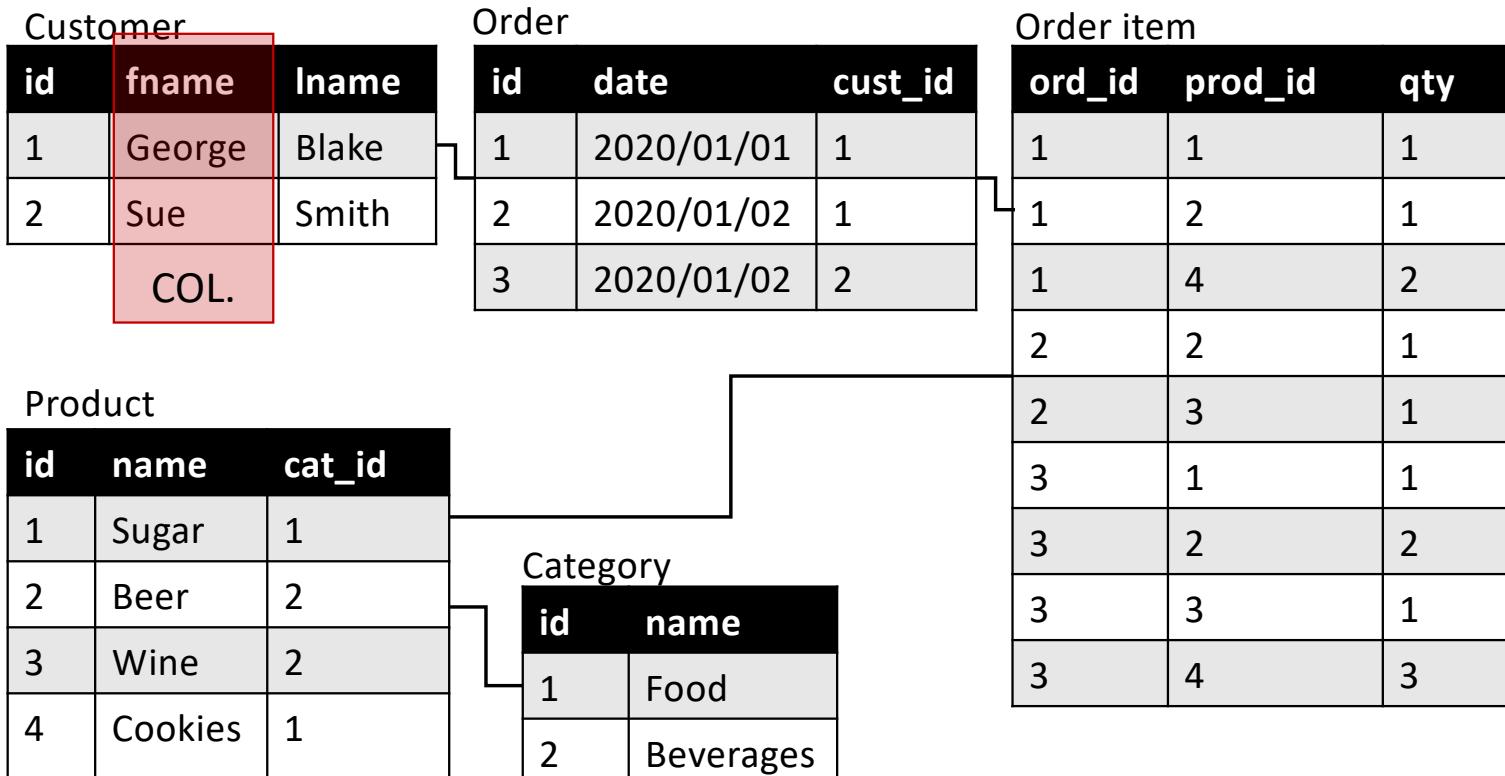
Redundant data is used to link records in different tables

Relational database



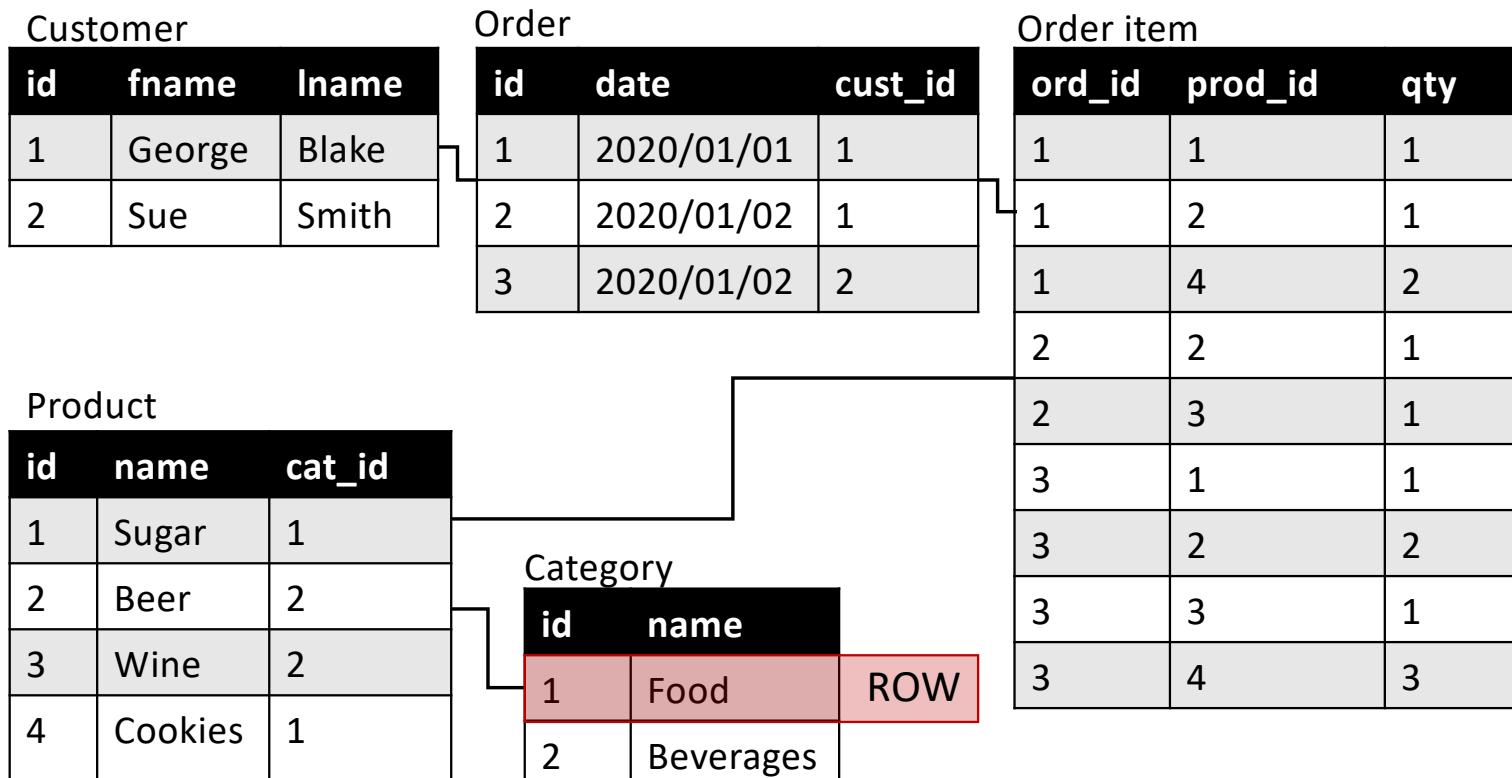
Entity: Something of interest to the database users (e.g., customers, products, etc.)

Relational database



Column (OR field): An individual piece of data

Relational database



Row (record): A set of columns that together describe an entity or some action to an entity

Relational database

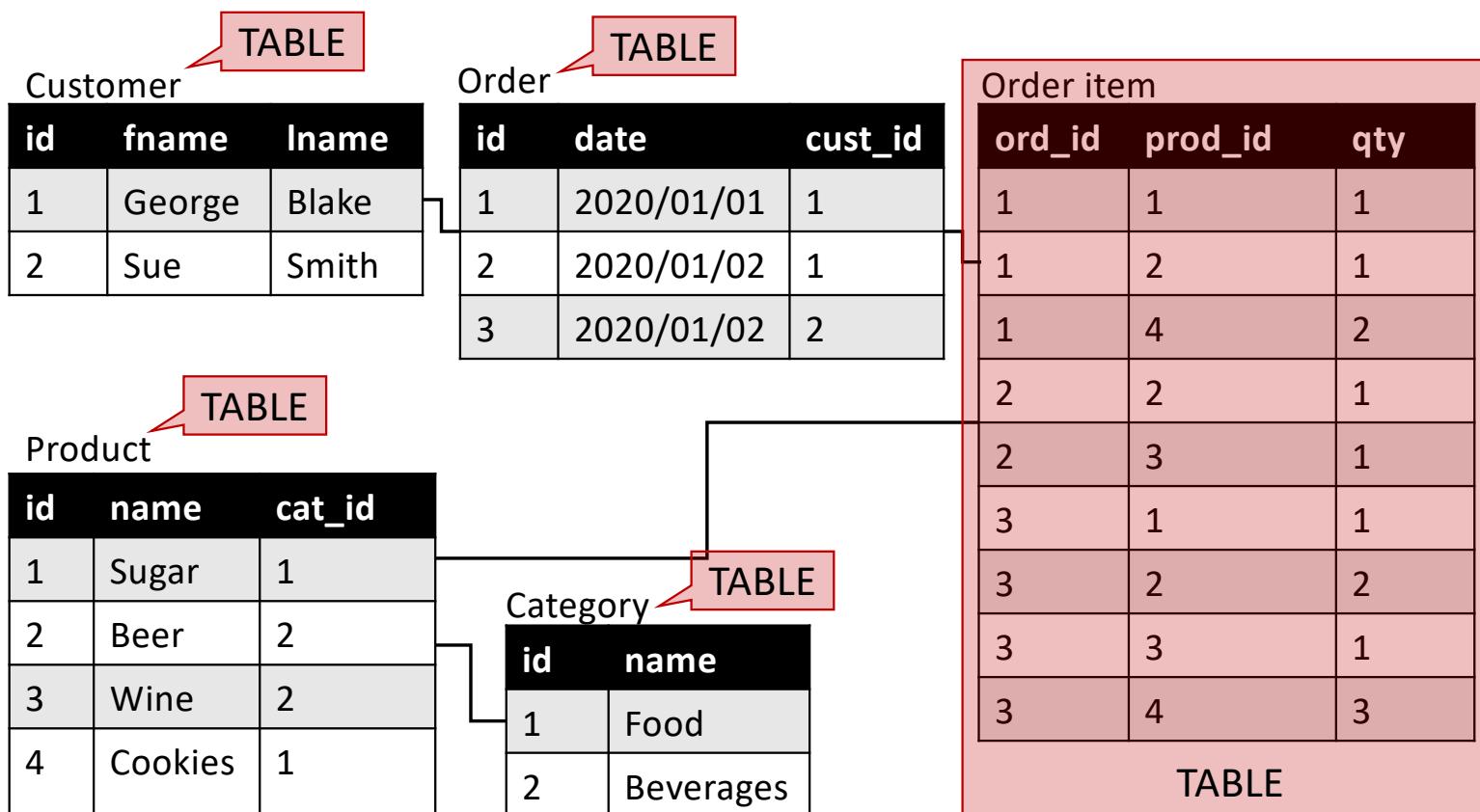
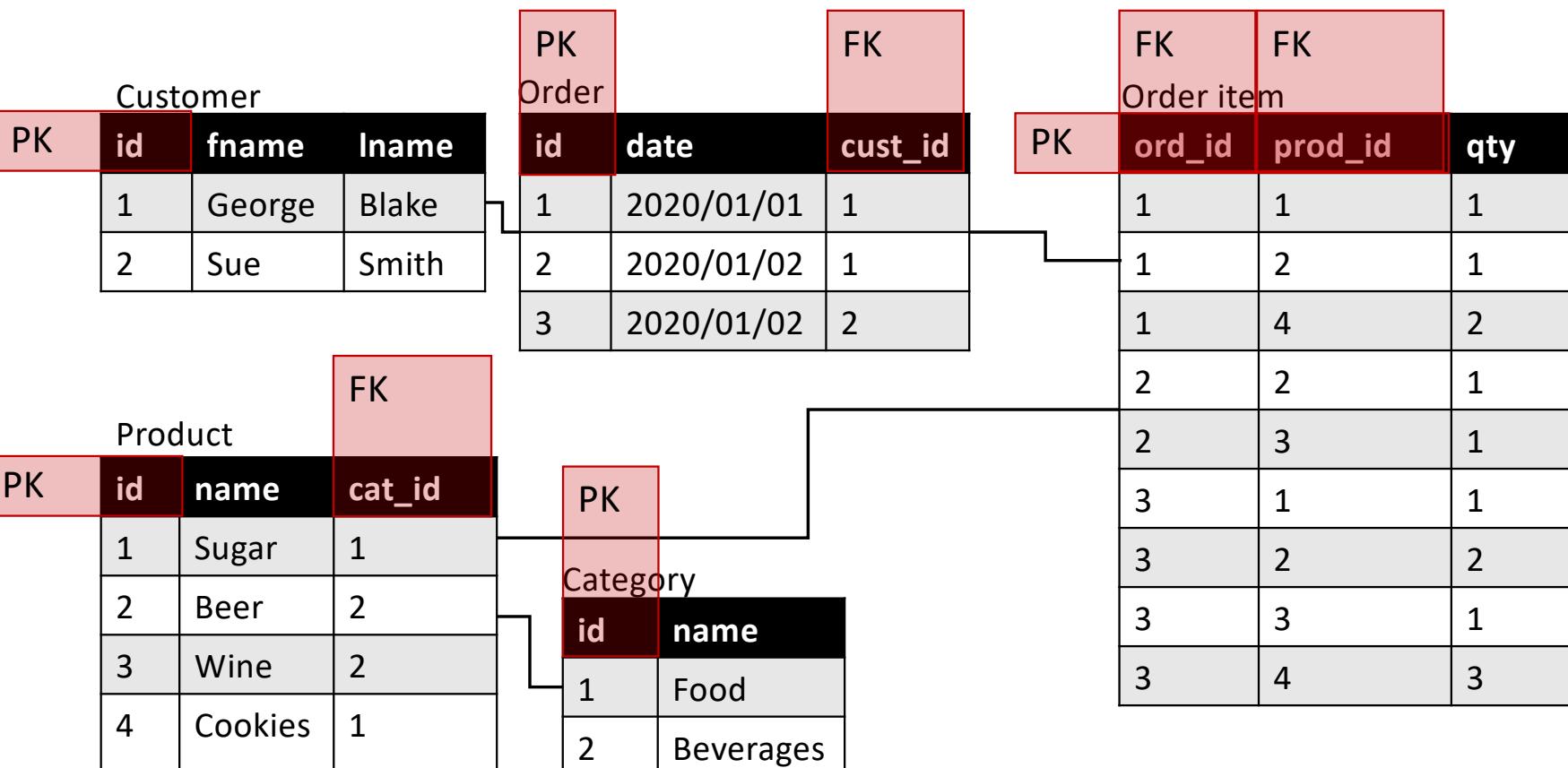


Table: A set of rows, held either in memory (nonpersistent) or on permanent storage (persistent)

Relational database



Primary key: One or more columns that can be used as a unique identifier for each row in a table

Foreign key: One or more columns that can be used to identify a single row in another table

Structured Query Language (SQL)

Databases and SQL



medium.com

SQL

- Standard language for querying and manipulating data
 - Developed in the 1970's by IBM
 - Several ANSI standards were introduced since 1986
 - Vendors support different subsets

Benefits of a standardized language

- Reduces training costs
- Improves productivity
- Application portability
- Application longevity
- Reduces vendor dependency
- Cross-systems communication

SQL components

- **Schema:** The structure that contains the descriptions of objects (tables, views, constraints, etc.)
- **Catalog:** A set of schemas that constitute the description of a database
- **Data Definition Language (DDL):** Commands that define the database: create, alter, delete tables and their attributes
- **Data Manipulation Language (DML):** Commands to maintain and query a data base
- **Data Control Language (DCL):** Commands that control a database, including administering privileges and committing data

SQL query example

Get all customers who last name is “Blake”

```
SELECT *  
FROM Customer  
WHERE lname='Blake'
```

Customer		
id	fname	lname
1	George	Blake
2	Sue	Smith

DBMS

Relational

Non
Relational

Non relational database

Customer		
id	fname	lname
1	George	Blake
2	Sue	Smith

Order		
id	date	cust_id
1	2020/01/01	1
2	2020/01/02	1
3	2020/01/02	2

Order item		
ord_id	prod_id	qty
1	1	1
1	2	1
1	4	2
2	2	1
2	3	1
3	1	1
3	2	2
3	3	1
3	4	3

Videos	
Name	
a	
b	

Product		
id	name	cat_id
1	Sugar	1
2	Beer	2
3	Wine	2
4	Cookies	1

Category	
id	name
1	Food
2	Beverages

There are no relations between entities
 Usually, data is not structured

Relational vs Non relational

	Relational	Non relational
Data size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many times
Transactions	ACID	None
Structure	Schema-on-write	Schema-on-read
Integrity	High	Low
Scaling	Nonlinear	Linear

[White, T. (2015)]

ACID:

- Atomicity: indivisible
- Consistency: any transaction can only change data in allowed ways
- Isolation: determines which parts of data can be visible to other users (e.g., purchase orders headers can only be visible after all lines are entered)
- Durability: guarantees that all committed transactions will survive permanently

Exercise: create database

Databases and SQL

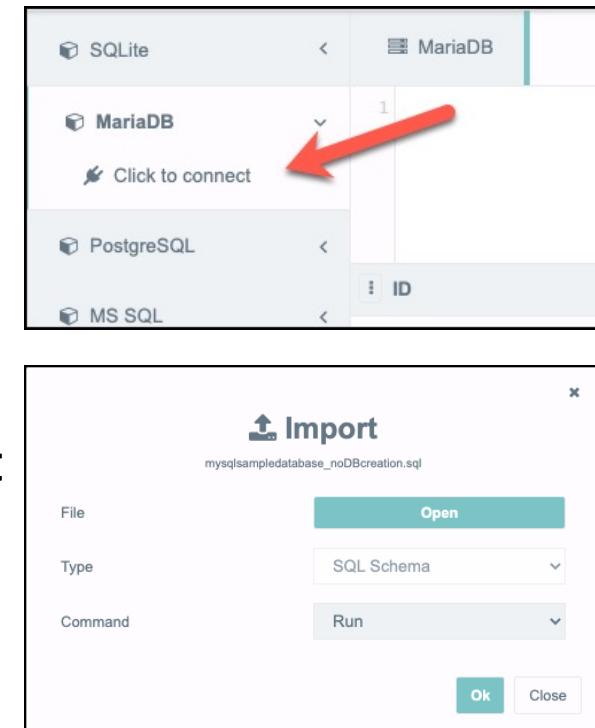
Online SQL

Instead of installing a local DBMS, we will use **Sqliteonline**, but there are several alternatives:

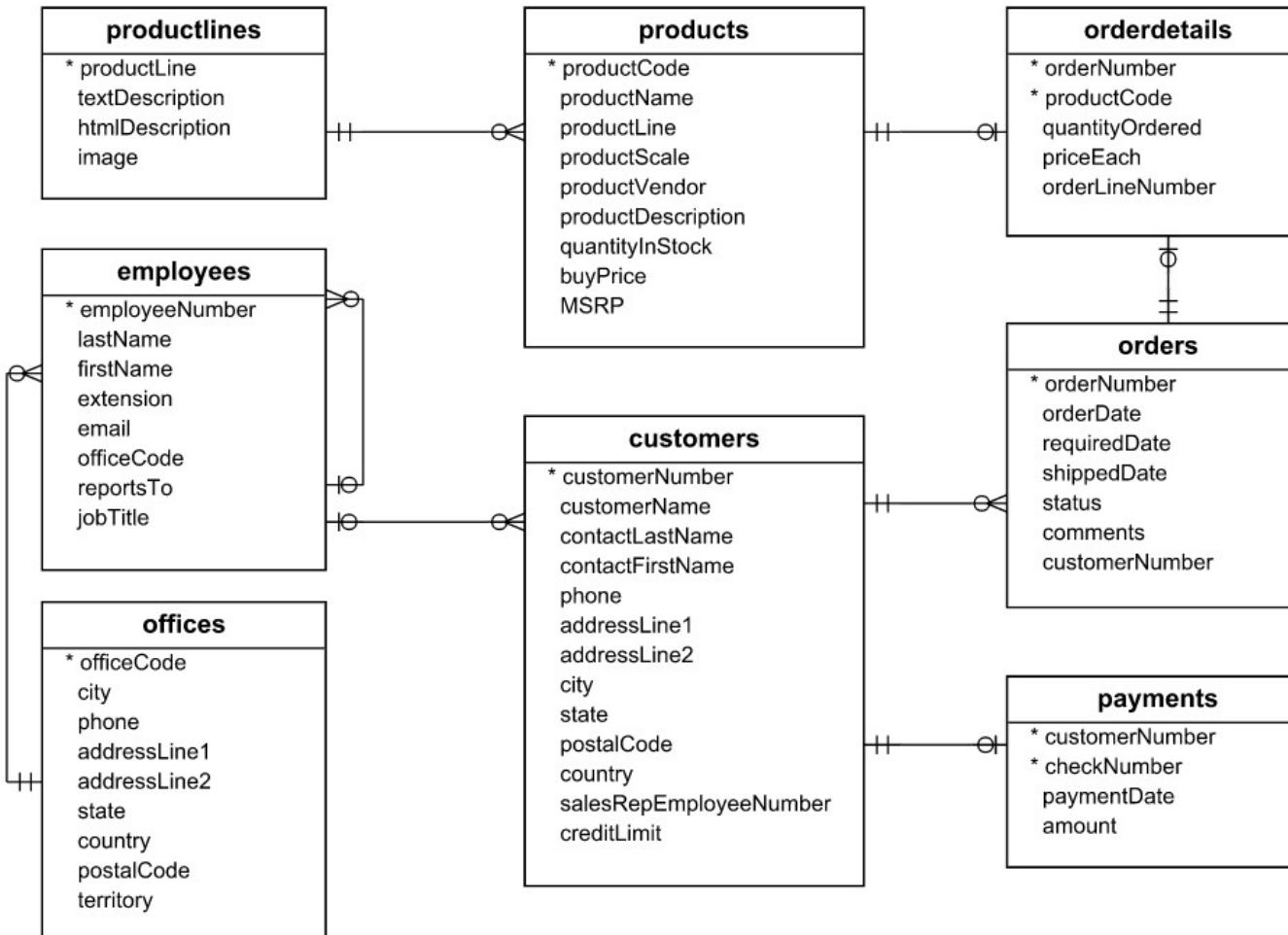
- <https://sqliteonline.com>: user-friendly interface for data science (SQLite, MariaDB/MySQL, PostgreSQL, MSSQL)
- https://www.tutorialspoint.com/execute_sql_online.php: SQLite platform for coding
- <http://sqlfiddle.com>: multi-SQL compatible platform for database training
- <https://www.db-fiddle.com>: user-friendly interface for multi-SQL database training
- If you prefer to install a local version, you could install, for example, MySQL community edition (free) from <https://www.mysql.com/products/community>

Database creation

1. With a text editor open the script file *mysqlsampledatabase_noDBcreation.sql* (if you want to create a new DB in a local DBMS use the file *mysqlsampledatabase.sql*)
2. See the SQL code in the script file. It not only creates tables, but also populates it with data
3. Open SQLiteonline, select MariaDB (compatible with MySQL) and click in “Click to connect”. It will open a database with a table named “Demo”
4. Click the “Import” button and select the file mentioned in point 1. After, click “Ok” for the script to run.
5. All tables will be created and populated with data.



Entity Relationship Diagram (ERD)



Exercise: queries and operators

Databases and SQL

Reading data from a database



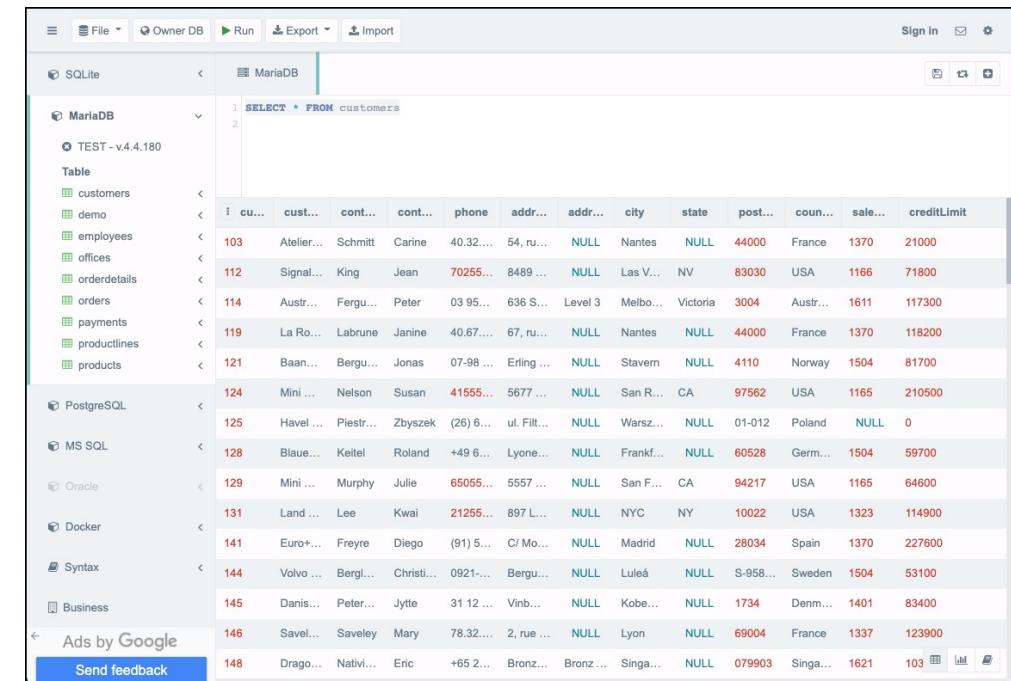
SELECT <list of columns>
FROM <table(s) or view(s) from which to read data>
WHERE <conditions to include rows>
GROUP BY <categorize results by>
HAVING <conditions under which a group will be included>
ORDER BY <criteria to sort results>
<...>

Selecting all columns

Select all columns from the table
customers

Selecting all columns

select * from customers



The screenshot shows a database interface with a sidebar containing database connections and tables. The main area displays the results of a query:

```
1 SELECT * FROM customers
2
```

The results table has the following structure:

	customerNumber	customerName	contactName	contactTitle	phone	address	address2	city	state	postalCode	country	salesRepEmployeeNumber	creditLimit
103	Atelier... Schmitt	Carine	40.32...	54, ru...	NULL	Nantes	NULL	44000	France	1370	21000		
112	Signal...	King Jean	70255...	8489...	NULL	Las V...	NV	83030	USA	1166	71800		
114	Austr...	Fergus...	Peter	03 95...	636 S...	Level 3	Melbo...	Victoria	3004	Austr...	1611	117300	
119	La Ro...	Labrune	Janine	40.67...	67, ru...	NULL	Nantes	NULL	44000	France	1370	118200	
121	Baan...	Bergu...	Jonas	07-98 ...	Erling ...	NULL	Stavern	NULL	4110	Norway	1504	81700	
124	Mini ...	Nelson Susan	41555...	5677 ...	NULL	San R...	CA	97562	USA	1165	210500		
125	Havel ...	Piestr...	Zbyszek	(26) 6...	ul. Fil...	NULL	Warsz...	NULL	01-012	Poland	NULL	0	
128	Blau...	Keitel Roland	Roland	+49 6...	Lyone...	NULL	Frankf...	NULL	60528	Germ...	1504	59700	
129	Mini ...	Murphy Julie	Julie	65055...	5557 ...	NULL	San F...	CA	94217	USA	1165	64600	
131	Land ...	Lee Kwai	Kwai	21255...	897 L...	NULL	NYC	NY	10022	USA	1323	114900	
141	Euro+...	Freyre Diego	Diego	(91) 5...	C/ Mo...	NULL	Madrid	NULL	28034	Spain	1370	227600	
144	Volvo ...	Bergl...	Christi...	0921...	Bergu...	NULL	Luleå	NULL	S-958...	Sweden	1504	53100	
145	Danis...	Peter...	Jytte	31 12 ...	Vinb...	NULL	Kobe...	NULL	1734	Denm...	1401	83400	
146	Savel...	Saveley Mary	Mary	78.32...	2, rue ...	NULL	Lyon	NULL	69004	France	1337	123900	
148	Drago...	Nativi...	Eric	+65 2...	Bronz...	Bronz ...	Singa...	NULL	079903	Singa...	1621	103	

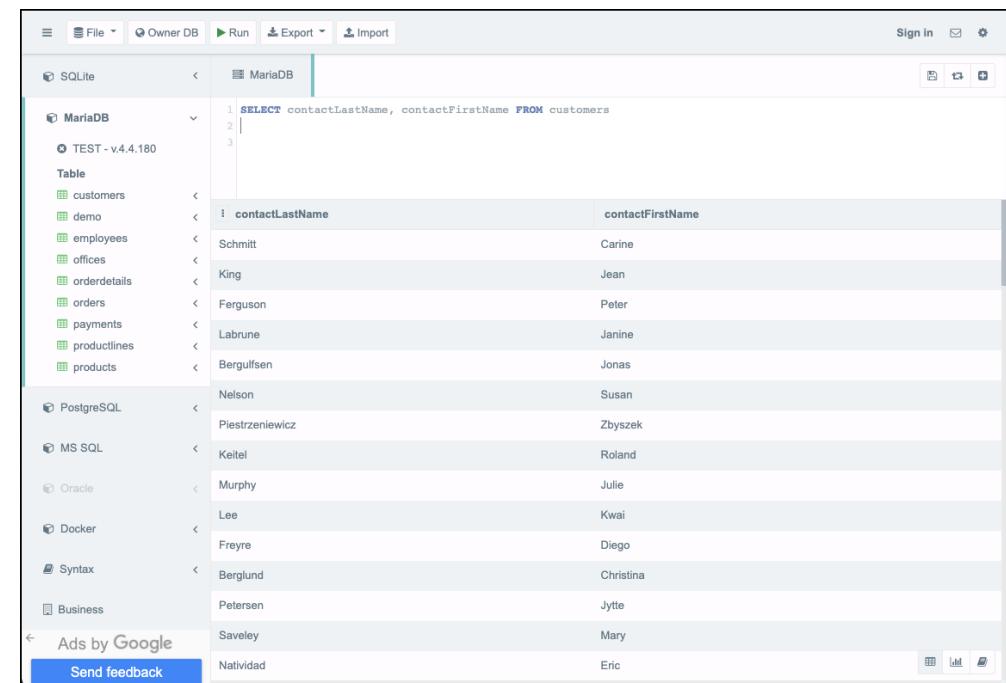
Selecting specific columns



Select the contact's last name
and the contact first name from
the table *customers*

Selecting specific columns

select contactLastName,
contactFirstName from
customers



The screenshot shows a SQL editor interface with the following details:

- Toolbar:** File, Owner DB, Run, Export, Import.
- Database:** SQLite (selected), MariaDB (selected).
- Query:**

```
1 SELECT contactLastName, contactFirstName FROM customers
```
- Table:** customers (selected). Other tables listed: demo, employees, offices, orderdetails, orders, payments, productlines, products.
- Results:** A grid showing contactLastName and contactFirstName for various rows. The first few rows are:

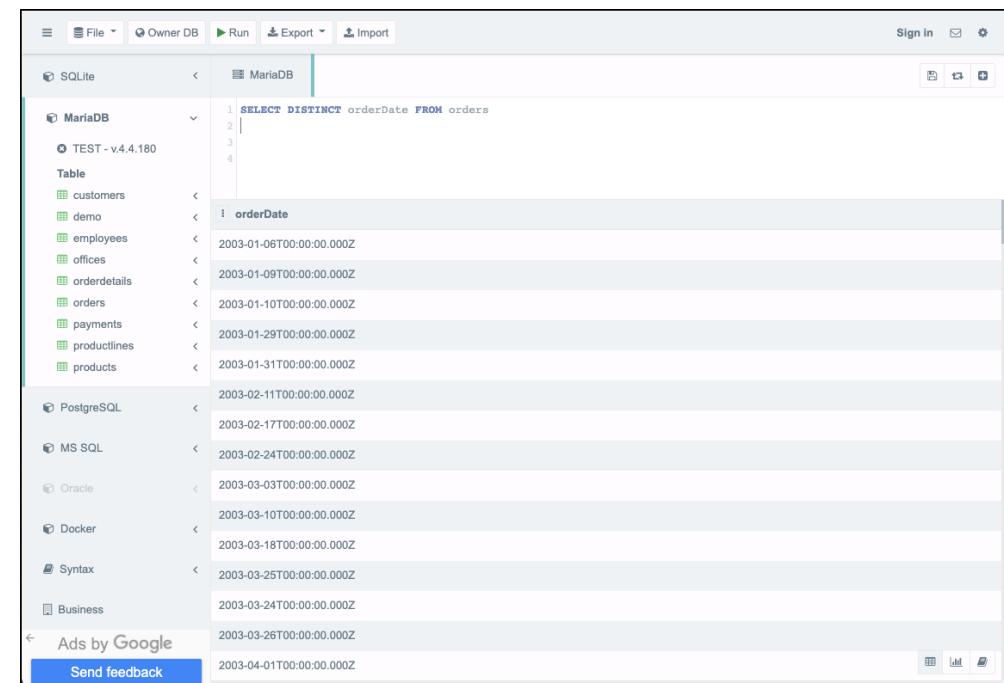
contactLastName	contactFirstName
Schmitt	Carine
King	Jean
Ferguson	Peter
Labrune	Janine
Bergulfsen	Jonas
Nelson	Susan
Piestrzenniewicz	Zbyszek
Keitel	Roland
Murphy	Julie
Lee	Kwai
Freyre	Diego
Berglund	Christina
Petersen	Jytte
Saveley	Mary
Natividad	Eric
- Bottom:** Ads by Google, Send feedback.

Selecting unique values from a column

- Select dates when there were orders made by *customers*
- Only show each date once (even if there is more than one order on the date)

Selecting unique values from a column

select distinct orderDate from
orders



The screenshot shows a SQL editor interface with a toolbar at the top and a code editor area. The code editor contains the following SQL query:

```
1 SELECT DISTINCT orderDate FROM orders
2 |
3 |
4 |
I orderDate
2003-01-06T00:00:00.000Z
2003-01-09T00:00:00.000Z
2003-01-10T00:00:00.000Z
2003-01-29T00:00:00.000Z
2003-01-31T00:00:00.000Z
2003-02-11T00:00:00.000Z
2003-02-17T00:00:00.000Z
2003-02-24T00:00:00.000Z
2003-03-03T00:00:00.000Z
2003-03-10T00:00:00.000Z
2003-03-18T00:00:00.000Z
2003-03-25T00:00:00.000Z
2003-03-24T00:00:00.000Z
2003-03-26T00:00:00.000Z
2003-04-01T00:00:00.000Z
```

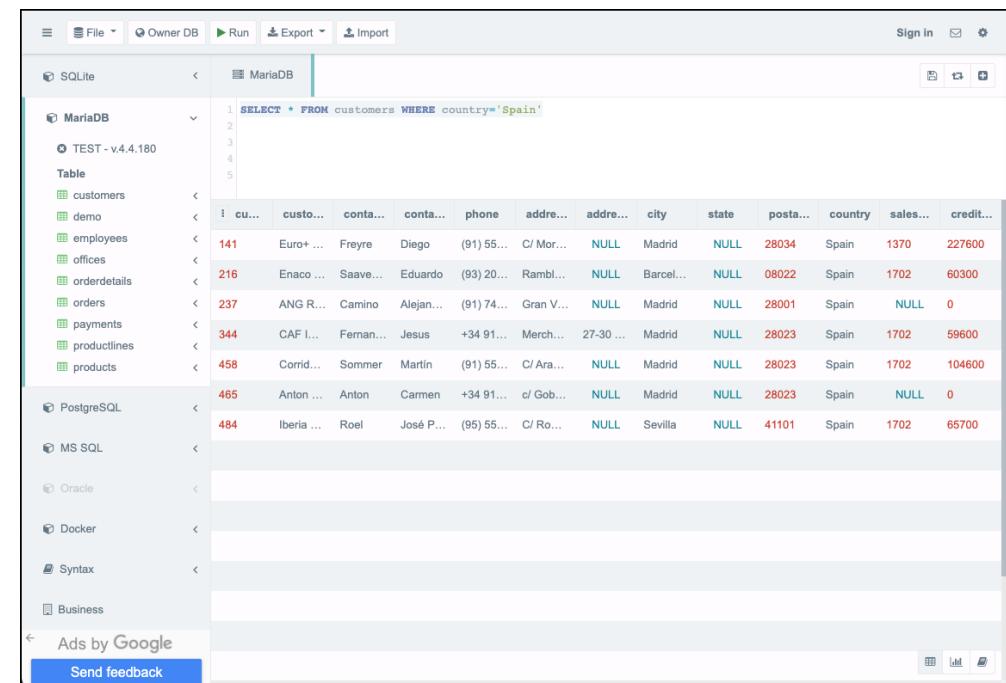
The results pane on the right side of the interface displays the output of the query, showing the distinct order dates.

Selecting rows (1/2)

Select all columns from
customers from Spain

Selecting rows (1/2)

select * from customers where country='Spain'



The screenshot shows a SQL editor interface with the following details:

- Database:** MariaDB
- Table:** customers
- Query:** SELECT * FROM customers WHERE country='Spain'
- Results:** The table displays 141 rows of customer data from Spain. The columns include: cu..., custo..., conta..., conta..., phone, addre..., addre..., city, state, posta..., country, sales..., credit... . Some values are highlighted in red.

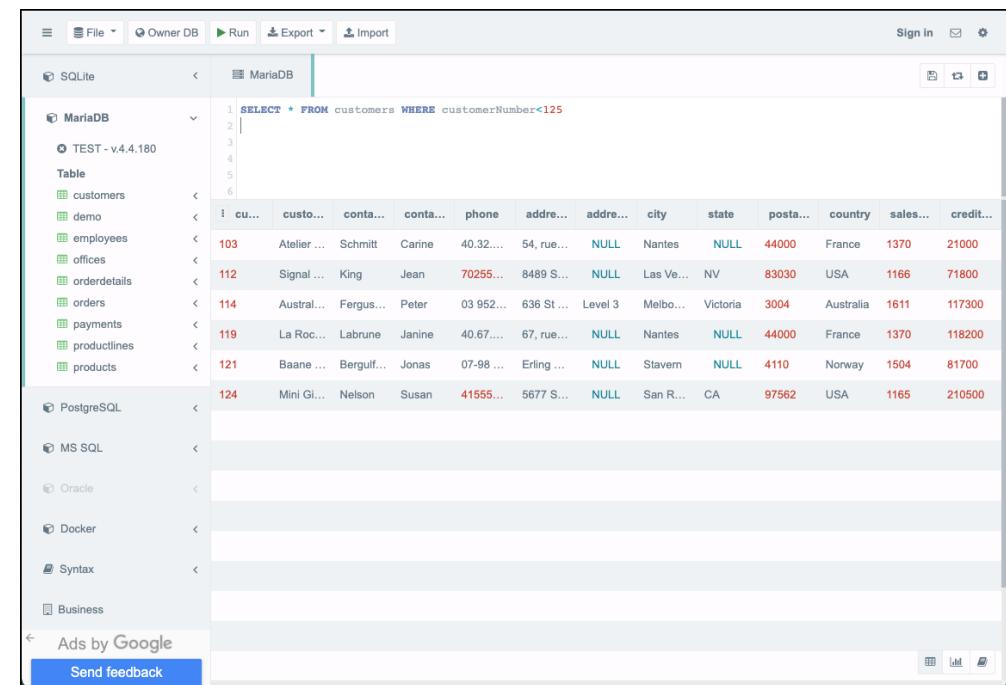
	cu...	custo...	conta...	conta...	phone	addre...	addre...	city	state	posta...	country	sales...	credit...
141	Euro+ ...	Freyre	Diego	(91) 55...	C/ Mor...	NULL	Madrid	NULL	28034	Spain	1370	227600	
216	Enaco ...	Saave...	Eduardo	(93) 20...	Rambl...	NULL	Barcel...	NULL	08022	Spain	1702	60300	
237	ANG R...	Camino	Alejan...	(91) 74...	Gran V...	NULL	Madrid	NULL	28001	Spain	NULL	0	
344	CAF I...	Fernan...	Jesus	+34 91...	Merch...	27-30 ...	Madrid	NULL	28023	Spain	1702	59600	
458	Corrid...	Sommer	Martin	(91) 55...	C/ Ara...	NULL	Madrid	NULL	28023	Spain	1702	104600	
465	Anton ...	Anton	Carmen	+34 91...	c/ Gob...	NULL	Madrid	NULL	28023	Spain	NULL	0	
484	Iberia ...	Roel	José P...	(95) 55...	C/ Ro...	NULL	Sevilla	NULL	41101	Spain	1702	65700	

Selecting rows (2/2)

Select all columns from
customers who have a
customerNumber smaller than
125

Selecting rows (2/2)

select * from customers where customerNumber<125



The screenshot shows a SQL editor interface with a sidebar containing database connections (SQLite, MariaDB, TEST - v4.4.180) and tables (customers, demo, employees, offices, orderdetails, orders, payments, productlines, products). The main area displays the results of the following SQL query:

```
1 SELECT * FROM customers WHERE customerNumber<125
```

The results table contains the following data:

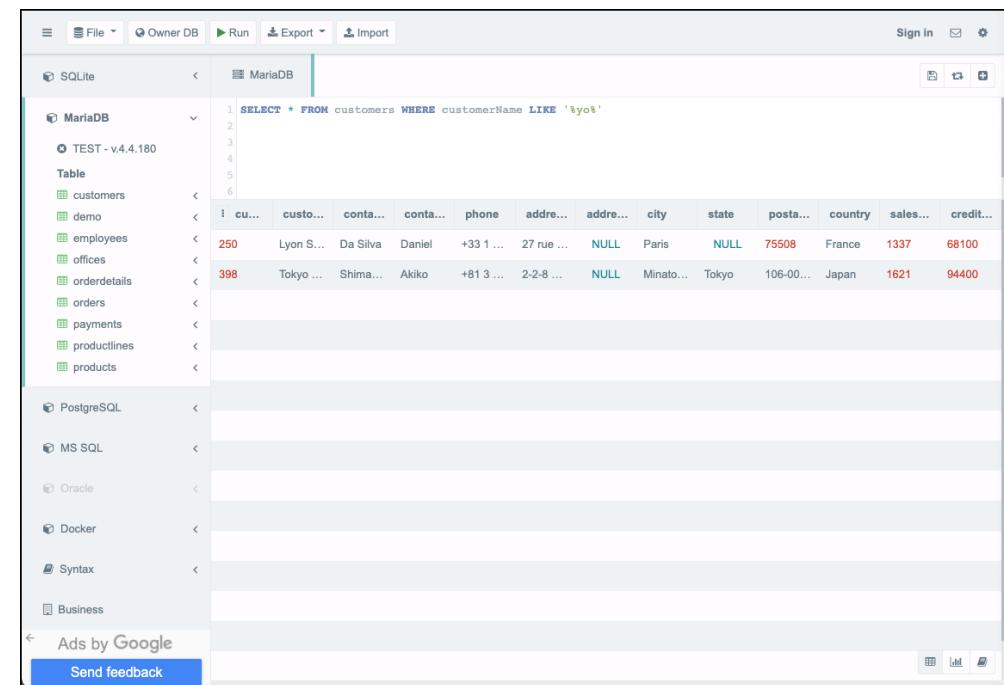
	customerNumber	customerName	contactName	contactTitle	phone	address	city	state	postalCode	country	salesRepEmployeeNumber	creditLimit
103	Atelier DC	Schmitt	Carine	40.32...	54, rue...	NULL	Nantes	NULL	44000	France	1370	21000
112	Signalochka	King	Jean	70255...	8489 S...	NULL	Las Ve...	NV	83030	USA	1166	71800
114	Australia Trading	Fergus...	Peter	03 952...	636 St...	Level 3	Melbo...	Victoria	3004	Australia	1611	117300
119	La Rocque	Labrune	Janine	40.67...	67, rue...	NULL	Nantes	NULL	44000	France	1370	118200
121	Baane & Sons	Bergulf...	Jonas	07-98 ...	Erling ...	NULL	Stavern	NULL	4110	Norway	1504	81700
124	Mini Giant	Nelson	Susan	41555...	5677 S...	NULL	San R...	CA	97562	USA	1165	210500

Selecting rows with “Like”

Select all columns from *customers* where the customerName includes the letters “yo” in any part of the name

Selecting rows with “Like”

select * from customers where
customerName like '%yo%'



The screenshot shows a SQL editor interface with the following details:

- Toolbar:** File, Owner DB, Run, Export, Import.
- Database:** MariaDB (selected), TEST - v.4.4.180.
- Table:** customers.
- Query:** `SELECT * FROM customers WHERE customerName LIKE '%yo%'`
- Results:** Two rows are displayed:
 - Customer ID: 250, Name: Lyon S..., Contact: Da Silva, Phone: +33 1 ..., Address: 27 rue ..., City: Paris, State: NULL, Postcode: 75508, Country: France, Sales: 1337, Credit: 68100.
 - Customer ID: 398, Name: Tokyo ..., Contact: Shima..., Phone: +81 3 ..., Address: 2-2-8 ..., City: Minato..., State: NULL, Postcode: 106-00..., Country: Japan, Sales: 1621, Credit: 94400.
- Servers:** SQLite, PostgreSQL, MS SQL, Oracle, Docker, Syntax.
- Business:** Ads by Google.
- Bottom:** Send feedback button.

Selecting rows with logic operators (1/4)

Inputs		Outputs			
A	B	AND	NOT AND	OR	NOT OR
0	0	0	1	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	0	1	0

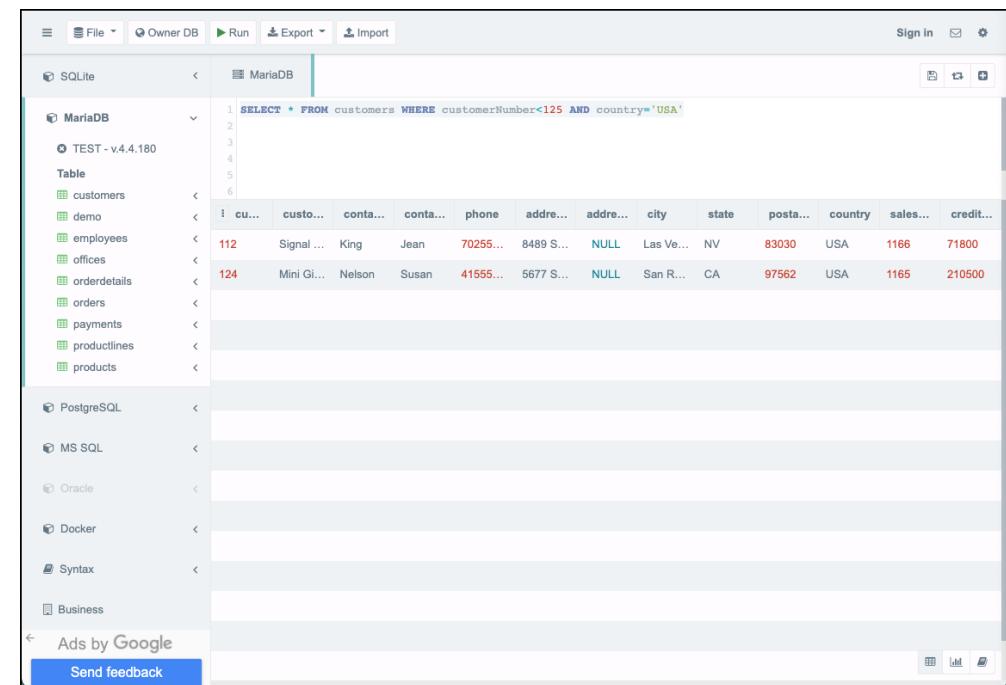
Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
!=	Not equal to

Selecting rows with logic operators (2/4)

Select all columns from
customers from the “USA” with a
customerNumber smaller than
125

Selecting rows with logic operators (2/4)

select * from customers where
customerNumber<125 and
country='USA'



The screenshot shows a SQL editor interface with a sidebar containing database connections and tables. The main area displays a query and its results.

```
SELECT * FROM customers WHERE customerNumber<125 AND country='USA'
```

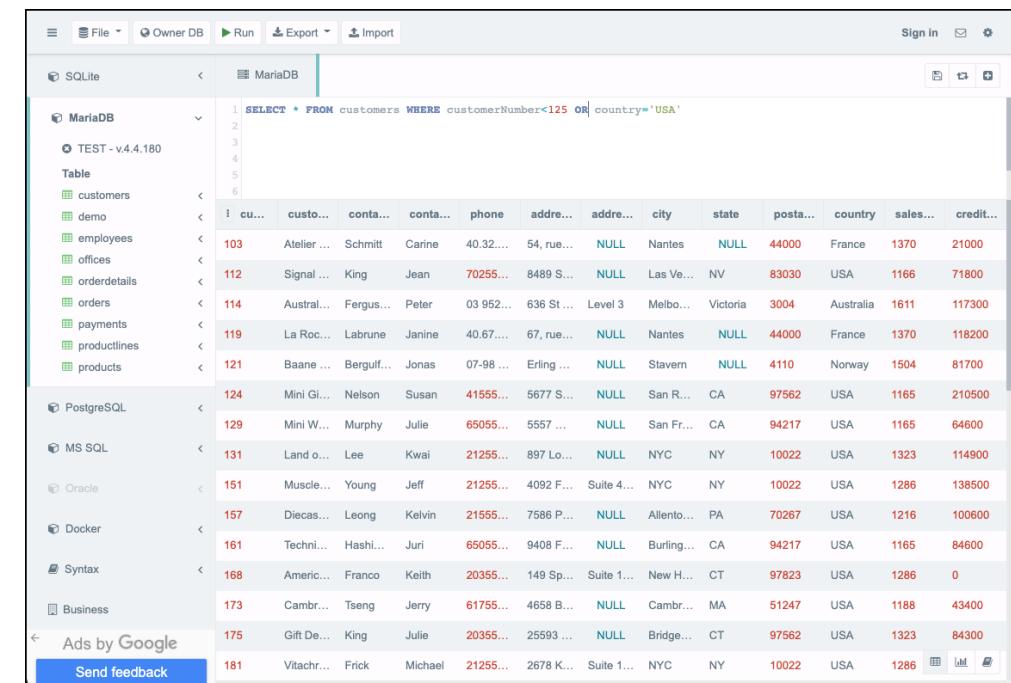
	customerNumber	customerName	contactName	phone	address	city	state	postalCode	country	salesRepEmployeeNumber	creditLimit	
112	Signal W...	King	Jean	7025550087	8489 S...	NULL	Las Ve...	NV	83030	USA	1166	71800
124	Mini G...	Nelson	Susan	4155550087	5677 S...	NULL	San R...	CA	97562	USA	1165	210500

Selecting rows with logic operators (3/4)

Select all columns from *customers* from the “USA” or with a *customerNumber* smaller than 125

Selecting rows with logic operators (3/4)

select * from customers where
customerNumber<125 or
country='USA'



The screenshot shows a database interface with a sidebar containing database connections (SQLite, MariaDB, TEST - v.4.4.180) and tables (customers, demo, employees, offices, orderdetails, orders, payments, productlines, products). The main area displays the results of a SQL query:

```
1 SELECT * FROM customers WHERE customerNumber<125 OR country='USA'
```

The results table shows 18 rows of customer data. The columns are: cu..., custo..., conta..., conta..., phone, addre..., addre..., city, state, posta..., country, sales..., credit... . The data includes various customers from different countries like France, USA, Australia, and Norway.

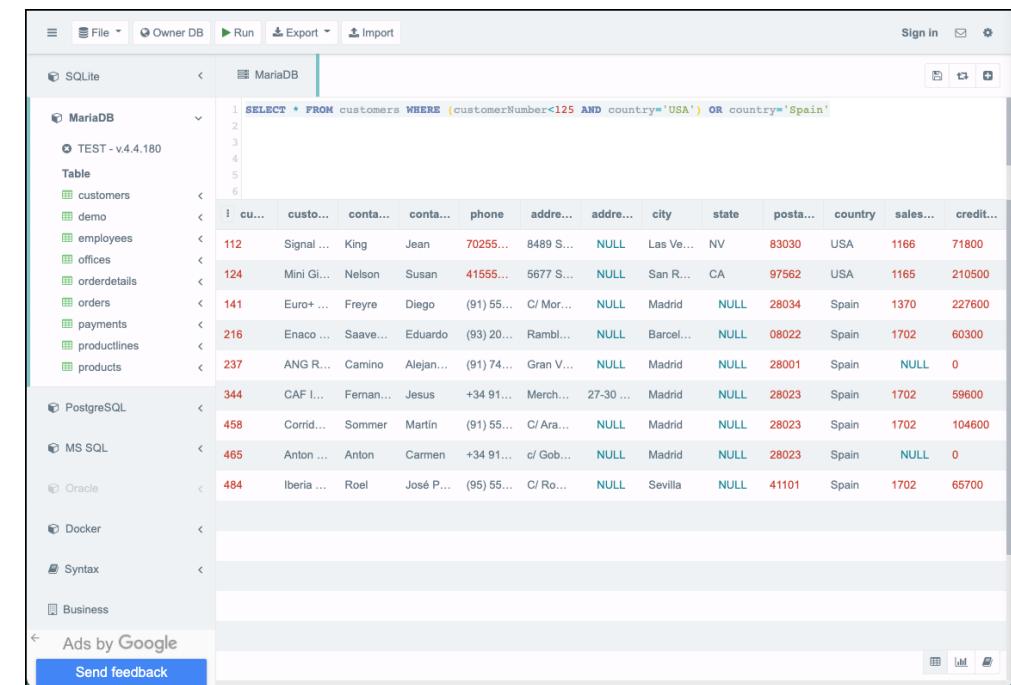
cu...	custo...	conta...	conta...	phone	addre...	addre...	city	state	posta...	country	sales...	credit...
103	Atelier ...	Schmitt	Carine	40.32...	54, rue...	NULL	Nantes	NULL	44000	France	1370	21000
112	Signal ...	King	Jean	70255...	8489 S...	NULL	Las Ve...	NV	83030	USA	1166	71800
114	Austral...	Fergus...	Peter	03 952...	636 St...	Level 3	Melbo...	Victoria	3004	Australia	1611	117300
119	La Roc...	Labrune	Janine	40.67...	67, rue...	NULL	Nantes	NULL	44000	France	1370	118200
121	Baane ...	Bergulf...	Jonas	07-98 ...	Erling ...	NULL	Stavern	NULL	4110	Norway	1504	81700
124	Mini Gi...	Nelson	Susan	41555...	5677 S...	NULL	San R...	CA	97562	USA	1165	210500
129	Mini W...	Murphy	Julie	65055...	5557 ...	NULL	San Fr...	CA	94217	USA	1165	64600
131	Land o...	Lee	Kwai	21255...	897 Lo...	NULL	NYC	NY	10022	USA	1323	114900
151	Muscle...	Young	Jeff	21255...	4092 F...	Suite 4...	NYC	NY	10022	USA	1286	138500
157	Diecas...	Leong	Kelvin	21555...	7586 P...	NULL	Allento...	PA	70267	USA	1216	100600
161	Techni...	Hashi...	Juri	65055...	9408 F...	NULL	Burling...	CA	94217	USA	1165	84600
168	Americ...	Franco	Keith	20355...	149 Sp...	Suite 1...	New H...	CT	97823	USA	1286	0
173	Cambr...	Tseng	Jerry	61755...	4658 B...	NULL	Cambr...	MA	51247	USA	1188	43400
175	Gift De...	King	Julie	20355...	25593 ...	NULL	Bridge...	CT	97562	USA	1323	84300
181	Vitachr...	Frick	Michael	21255...	2678 K...	Suite 1...	NYC	NY	10022	USA	1286	0

Selecting rows with logic operators (4/4)

Select all columns from *customers* from the “USA” with a *customerNumber* smaller than 125, or customers from Spain, independently of their number

Selecting rows with logic operators (4/4)

select * from customers where
 (customerNumber<125 and
 country='USA') or
 country='Spain'



```

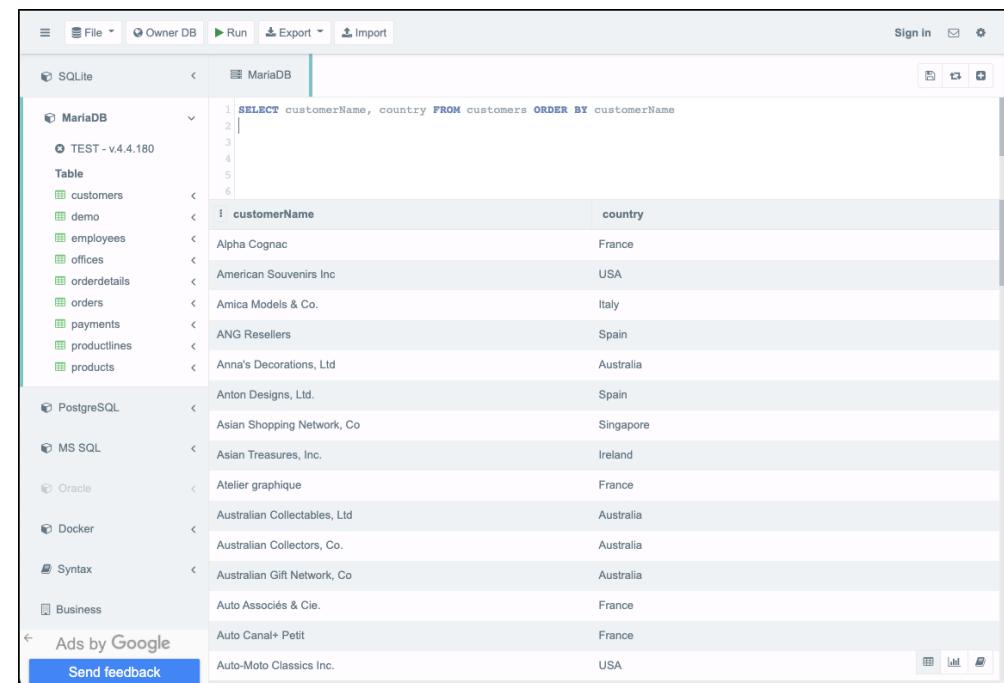
SELECT * FROM customers WHERE (customerNumber<125 AND country='USA') OR country='Spain'
  112  Signal... King   Jean   70255... 8489 S... NULL  Las Ve... NV   83030 USA   1166  71800
  124  Mini Gi... Nelson Susan  41555... 5677 S... NULL  San R... CA   97562 USA   1165  210500
  141  Euro+ ... Freyre Diego  (91) 55... C/Mor... NULL  Madrid  NULL  28034 Spain  1370  227600
  216  Enaco ... Saave... Eduardo (93) 20... Rambl... NULL  Barcel... NULL  08022 Spain  1702  60300
  237  ANG R... Camino Alejan... (91) 74... Gran V... NULL  Madrid  NULL  28001 Spain  NULL   0
  344  CAF I... Ferman Jesus +34 91... Merch... 27-30 ... Madrid  NULL  28023 Spain  1702  59600
  458  Corrid... Sommer Martin (91) 55... C/Ara... NULL  Madrid  NULL  28023 Spain  1702  104600
  465  Anton ... Anton Carmen +34 91... c/Gob... NULL  Madrid  NULL  28023 Spain  NULL   0
  484  Iberia ... Roel   José P... (95) 55... C/Ro... NULL  Sevilla NULL  41101 Spain  1702  65700
  
```

Sorting results (1/2)

Select *customerName* and *country* from *customers* in ascend order of the *customerName*

Sorting results (1/2)

select customerName, country
 from customers order by
 customerName



```
SELECT customerName, country FROM customers ORDER BY customerName
```

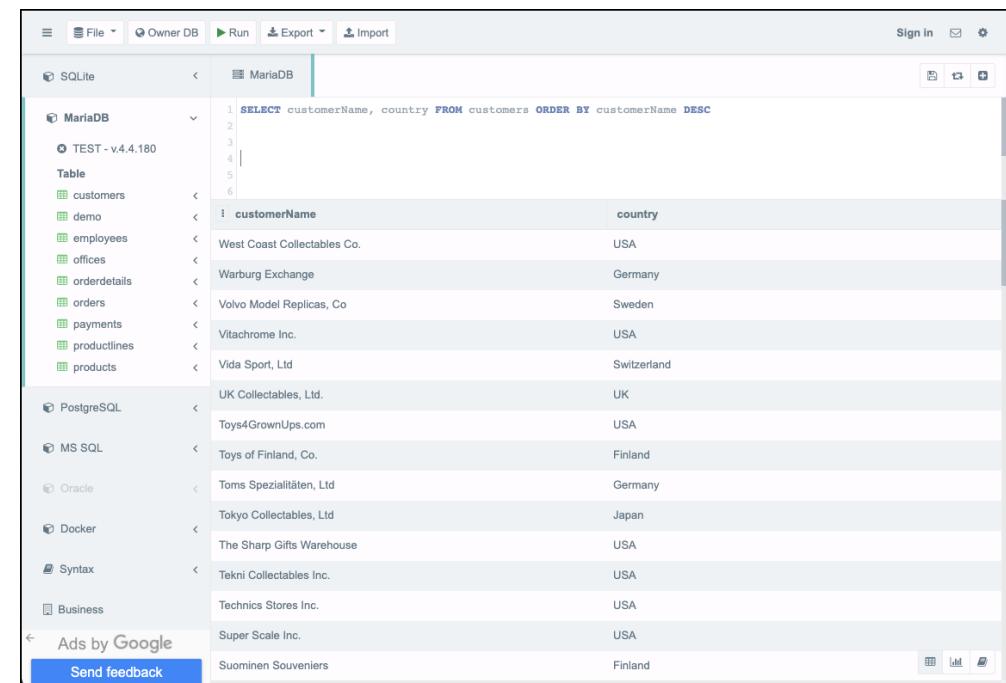
customerName	country
Alpha Cognac	France
American Souvenirs Inc	USA
Amica Models & Co.	Italy
ANG Resellers	Spain
Anna's Decorations, Ltd	Australia
Anton Designs, Ltd.	Spain
Asian Shopping Network, Co	Singapore
Asian Treasures, Inc.	Ireland
Atelier graphique	France
Australian Collectables, Ltd	Australia
Australian Collectors, Co.	Australia
Australian Gift Network, Co	Australia
Auto Associés & Cie.	France
Auto Canal+ Petit	France
Auto-Moto Classics Inc.	USA

Sorting results (2/2)

Select *customerName* and *country* from *customers* in descend order of the *customerName*

Sorting results (2/2)

```
select customerName, country
from customers order by
customerName desc
```



The screenshot shows a database interface with a sidebar containing databases like SQLite, MariaDB, TEST - v4.4.180, PostgreSQL, MS SQL, Oracle, Docker, Syntax, and Business. The main area displays the results of the following SQL query:

```
SELECT customerName, country FROM customers ORDER BY customerName DESC
```

The results are:

customerName	country
West Coast Collectables Co.	USA
Warburg Exchange	Germany
Volvo Model Replicas, Co	Sweden
Vitachrome Inc.	USA
Vida Sport, Ltd	Switzerland
UK Collectables, Ltd.	UK
Toys4GrownUps.com	USA
Toys of Finland, Co.	Finland
Toms Spezialitäten, Ltd	Germany
Tokyo Collectables, Ltd	Japan
The Sharp Gifts Warehouse	USA
Tekni Collectables Inc.	USA
Technics Stores Inc.	USA
Super Scale Inc.	USA
Suominen Souveniers	Finland

Join selections (1/2)

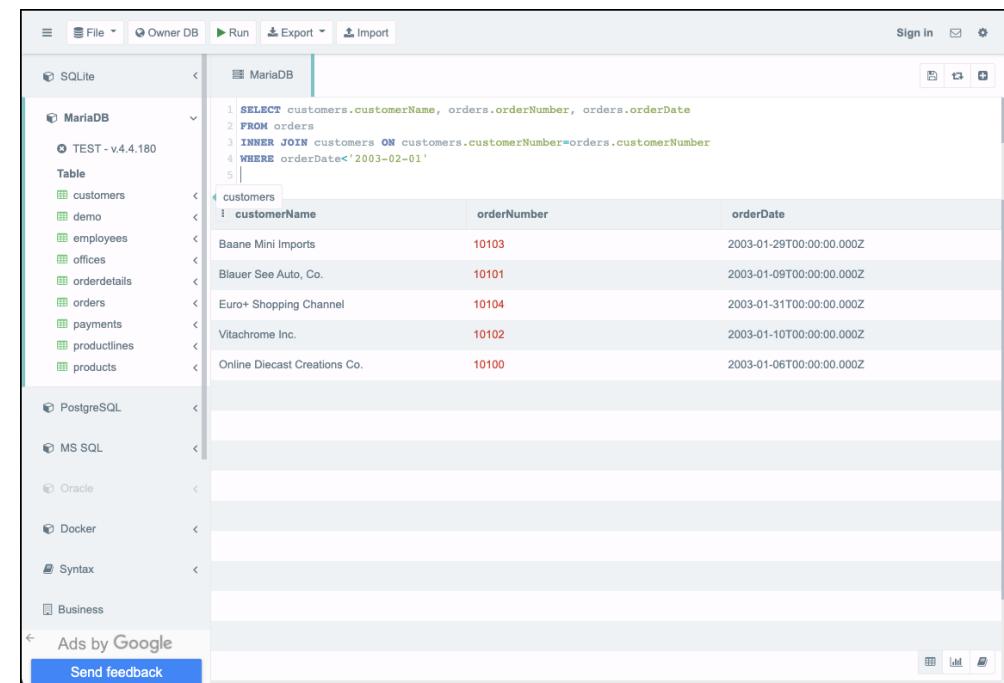
- Find all customers who made orders before 2003-02-01
- Return customer name, order number, and order date

Join selections (1/2)

```

select
  customers.customerName,
  orders.orderNumber,
  orders.orderDate
from orders
inner join customers on
  customers.customerNumber=orders.customerNumber
where orderDate<'2003-02-01'

```



The screenshot shows a database interface with a sidebar containing databases like SQLite, MariaDB, TEST - v4.4.180, PostgreSQL, MS SQL, Oracle, Docker, Syntax, and Business. The main area displays a table named 'customers' with columns 'customerName', 'orderNumber', and 'orderDate'. The table contains five rows of data:

customerName	orderNumber	orderDate
Baane Mini Imports	10103	2003-01-29T00:00:00.000Z
Blauer See Auto, Co.	10101	2003-01-09T00:00:00.000Z
Euro+ Shopping Channel	10104	2003-01-31T00:00:00.000Z
Vitachrome Inc.	10102	2003-01-10T00:00:00.000Z
Online Diecast Creations Co.	10100	2003-01-06T00:00:00.000Z

The query in the editor is:

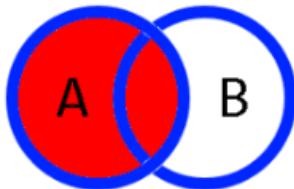
```

1 SELECT customers.customerName, orders.orderNumber, orders.orderDate
2 FROM orders
3 INNER JOIN customers ON customers.customerNumber=orders.customerNumber
4 WHERE orderDate<'2003-02-01'
5

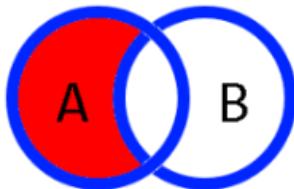
```

Join selections (2/2)

LEFT OUTER JOIN

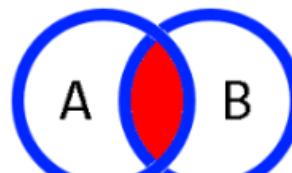


```
SELECT *\nFROM TableA a\nLEFT JOIN TableB b\nON a.KEY = b.KEY
```



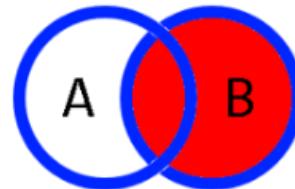
```
SELECT *\nFROM TableA a\nLEFT JOIN TableB b\nON a.KEY = b.KEY\nWHERE b.KEY IS NULL
```

INNER JOIN

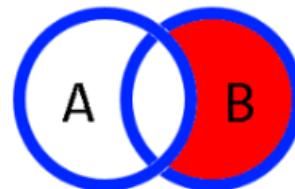


```
SELECT *\nFROM TableA a\nINNER JOIN TableB b\nON a.KEY = b.KEY
```

RIGHT OUTER JOIN



```
SELECT *\nFROM TableA a\nRIGHT JOIN TableB b\nON a.KEY = b.KEY
```



```
SELECT *\nFROM TableA a\nRIGHT JOIN TableB b\nON a.KEY = b.KEY\nWHERE a.KEY IS NULL  
OR b.KEY IS NULL
```

source: imgur.com

Exercise: aggregate functions

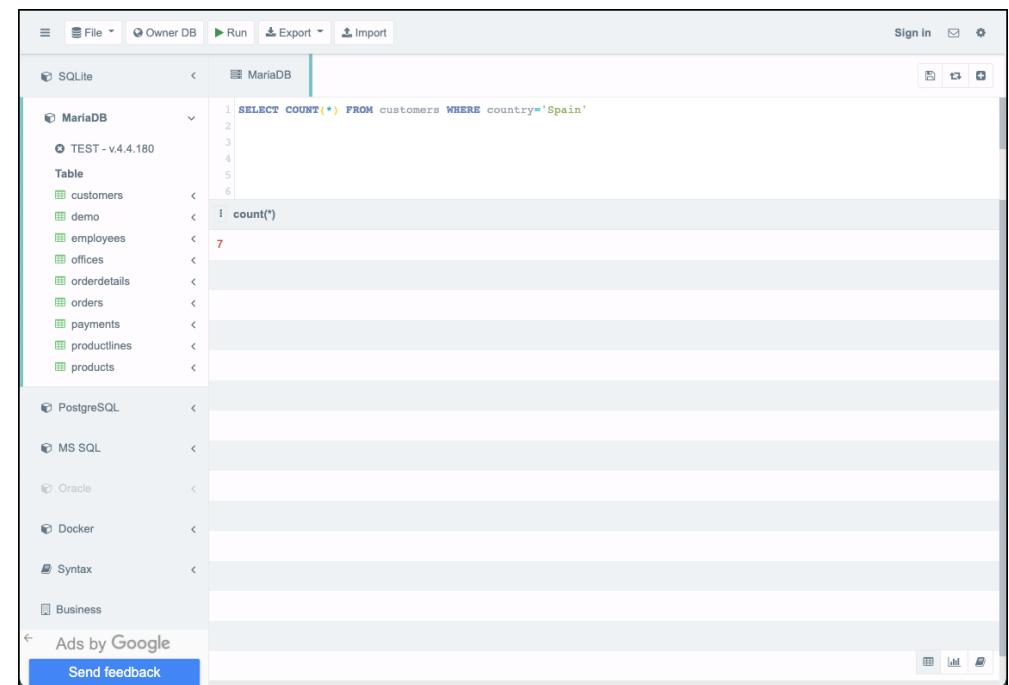
Databases and SQL

COUNT (1/2)

Count the number of customers
from Spain

COUNT (1/2)

select count(*) from customers
where country='Spain'



The screenshot shows a SQL editor interface with a sidebar containing database connections and tables. The main area displays a query in the MariaDB connection:

```
1 SELECT COUNT(*) FROM customers WHERE country='Spain'
2
3
4
5
6
7 count(*)
```

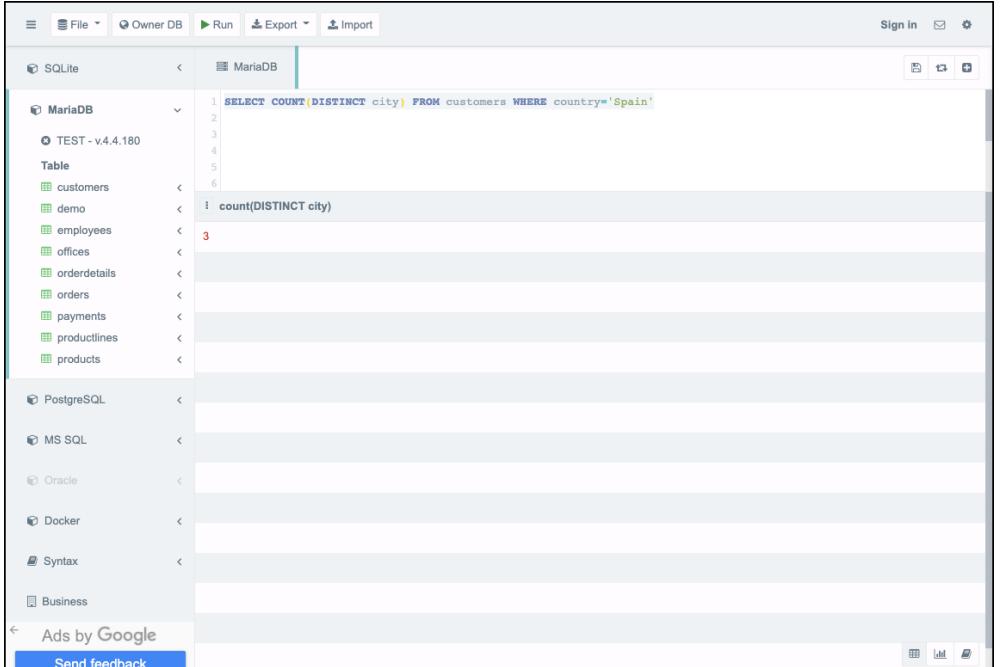
The result of the query, '7', is displayed in red text below the cursor.

COUNT (2/2)

Count how many cities in Spain
are customers from

COUNT (2/2)

select count(DISTINCT city)
from customers where
country='Spain'



The screenshot shows a SQL editor interface with a sidebar containing database connections and tables. The main area displays a query being run against a MariaDB connection.

```
SELECT COUNT(DISTINCT city) FROM customers WHERE country='Spain'
```

The result of the query is shown in red text:

```
3
```

The sidebar includes connections to SQLite, TEST - v.4.4.180, PostgreSQL, MS SQL, Oracle, Docker, Syntax, and Business. There is also an "Ads by Google" section at the bottom.

SUM, MAX, MIN, and AVG



What is the total value of the orders made?

SUM, MAX, MIN, and AVG

```
select
sum(od.priceEach*od.quantityOrdered) as totalPrice
from orderdetails od
```

Check the maximum, minimum and average, by replacing the “sum” by the “max”, “min” and “avg”

```
SELECT SUM(od.priceEach*od.quantityOrdered) AS totalPrice
FROM orderdetails od
```

totalPrice
9604190.61

Aggregation with subquery

- Products which the buy price are higher than the average buy price
- Return the product code, product name, and buy price, ordered by buy price (descent)

Aggregation with subquery

```
select productCode,
       productName, buyPrice
  from products
 where buyprice>(
 select avg(buyprice)
   from products)
 order by buyprice desc
```

The screenshot shows a SQL editor interface with a code editor at the top and a results table below. The code editor contains the following SQL query:

```
1 SELECT productCode, productName, buyPrice
2 FROM products
3 WHERE buyprice>(
4 SELECT AVG(buyprice)
5   FROM products)
6 ORDER BY buyprice DESC
```

The results table displays 15 rows of data from the 'products' table, ordered by buyprice in descending order. The columns are 'productCode', 'productName', and 'buyPrice'. The data includes various car models and their purchase prices.

productCode	productName	buyPrice
S10_4962	1962 LanciaA Delta 16V	103.42
S18_2238	1998 Chrysler Plymouth Prowler	101.51
S10_1949	1952 Alpine Renault 1300	98.58
S24_3856	1956 Porsche 356A Coupe	98.3
S12_1108	2001 Ferrari Enzo	95.59
S12_1099	1968 Ford Mustang	95.34
S18_1984	1995 Honda Civic	93.89
S18_4027	1970 Triumph Spitfire	91.92
S10_4698	2003 Harley-Davidson Eagle Drag Bike	91.02
S12_3148	1969 Corvair Monza	89.14
S18_1749	1917 Grand Touring Sedan	86.7
S10_4757	1972 Alfa Romeo GTA	85.68
S18_4600	1940s Ford truck	84.76
S18_1129	1993 Mazda RX-7	83.51
S12_3891	1969 Ford Falcon	83.05

GROUP BY_(1/2)

- Get the total ordered, date of first order, and the date of last order, per customer
- Return customer number, customer name, total ordered, date of first order, and the date of the last order

GROUP BY (1/2)

```
select c.customerNumber,
c.customerName,
sum(od.priceEach*od.quantityOrdered)
as totalOrdered,
min(o.orderDate) as dateFirstOrder,
max(o.orderDate) as dataLastOrder
from orderdetails od
inner join orders o on
o.orderNumber=od.orderNumber
inner join customers c on
c.customerNumber=o.customerNumber
group by c.customerNumber
order by c.customerName
```

	customerNumber	customerName	totalOrdered	dateFirstOrder	dataLastOrder
1	242	Alpha Cognac	60483.36	2003-07-04T00:00:00.000Z	2005-03-28T00:00:00.000Z
2	249	Amica Models & Co.	82223.23	2004-08-17T00:00:00.000Z	2004-09-09T00:00:00.000Z
3	276	Anna's Decorations, Ltd	137034.22	2003-09-11T00:00:00.000Z	2005-03-09T00:00:00.000Z
4	103	Atelier graphique	22314.36	2003-05-20T00:00:00.000Z	2004-11-25T00:00:00.000Z
5	471	Australian Collectables, Ltd	55866.02	2003-11-21T00:00:00.000Z	2005-05-09T00:00:00.000Z
6	114	Australian Collectors, Co.	180585.07	2003-04-29T00:00:00.000Z	2004-11-29T00:00:00.000Z
7	333	Australian Gift Network, Co	55190.16	2003-09-25T00:00:00.000Z	2005-02-02T00:00:00.000Z
8	256	Auto Associes & Cie.	58876.41	2004-02-02T00:00:00.000Z	2004-10-11T00:00:00.000Z
9	406	Auto Canal+ Pelit	86436.97	2004-01-15T00:00:00.000Z	2005-04-07T00:00:00.000Z
10	198	Auto-Moto Classics Inc.	21554.26	2003-06-16T00:00:00.000Z	2004-12-03T00:00:00.000Z
11	187	AV Stores, Co.	148410.09	2003-03-18T00:00:00.000Z	2004-11-17T00:00:00.000Z
12	121	Baane Mini Imports	104224.79	2003-01-29T00:00:00.000Z	2004-11-05T00:00:00.000Z
13	415	Bavarian Collectables Im...	31310.09	2004-09-15T00:00:00.000Z	2004-09-15T00:00:00.000Z

GROUP BY_(2/2)

- Get the number of orders, total ordered, average per order, and average per order line, per customer
- Return customer number, customer name, total ordered, average per order, and average per order line

GROUP BY (2/2)

```

select c.customerNumber,
c.customerName, count(DISTINCT
o.orderNumber) as orders,
sum(od.priceEach*od.quantityOrdered)
as totalOrdered,
sum(od.priceEach*od.quantityOrdered)/c
ount(DISTINCT o.orderNumber) as
avgPerOrder,
avg(od.priceEach*od.quantityOrdered) as
avgPerOrderLine
from orders o
inner join orderdetails od on
od.orderNumber=o.orderNumber
inner join customers c on
c.customerNumber=o.customerNumber
group by c.customerNumber
order by c.customerName

```

customerNumber	customerName	orders	totalOrdered	avgPerOrder	avgPerOrderLine
242	Alpha Cognac	3	60483.36	20161.12	3024.168
249	Amica Models & Co.	2	82223.23	41111.615	3162.431923
276	Anna's Decorations,...	4	137034.22	34258.555	2979.004783
103	Atelier graphique	3	22314.36	7438.12	3187.765714
471	Australian Collectabl...	3	55866.02	18622.006667	2428.957391
114	Australian Collector...	5	180585.07	36117.014	3283.364909
333	Australian Gift Netw...	3	55190.16	18396.72	3679.344
256	Auto Associés & Cie.	2	58876.41	29438.205	3270.911667
406	Auto Canal+ Petit	3	86436.97	28812.323333	3201.369259
198	Auto-Moto Classics ...	3	21554.26	7184.753333	2694.2825
187	AV Stores, Co.	3	148410.09	49470.03	2910.001765
121	Baane Mini Imports	4	104224.79	26056.1975	3257.024688
415	Bavarian Collectabl...	1	31310.09	31310.09	2236.435

Combining LEFT JOIN and aggregation

- Get the number of orders, number of products, number of distinct products, and total quantities, per customer and year (even for customers without no orders)
- Return customer name, year, number of orders, number of products, number of distinct products, total quantity

Combining LEFT JOIN and aggregation

```

select customers.customerName as Name,
       ifnull(year(orders.orderDate),0) as Year,
       count(orders.orderNumber) as Orders,
       count(orderdetails.productCode) as ProductsOrdered,
       count(DISTINCT orderdetails.productCode) as DistinctProductsOrdered,
       ifnull(SUM(orderdetails.quantityOrdered),0) as QtyOrdered
  from customers
 left join orders on
    orders.customerNumber=customers.customerNumber
 left join orderdetails on
    orderdetails.orderNumber=orders.orderNumber
 group by customers.customerName,
          year(orders.orderDate)

```

	Name	Year	Orders	ProductsOrdered	DistinctProductsO... QtyOrdered
Alpha Cognac	2003	15	15	15	515
Alpha Cognac	2005	5	5	5	172
American Souvenirs ...	0	0	0	0	0
Amica Models & Co.	2004	26	26	26	843
ANG Resellers	0	0	0	0	0
Anna's Decorations, ...	2003	27	27	27	874
Anna's Decorations, ...	2005	19	19	15	595
Anton Designs, Ltd.	0	0	0	0	0
Asian Shopping Net...	0	0	0	0	0
Asian Treasures, Inc.	0	0	0	0	0
Atelier graphique	2003	4	4	4	156
Atelier graphique	2004	3	3	3	114
Australian Collectabl...	2003	16	16	16	447

Questions?

Data Science for Marketing

© 2021-2022 Nuno António (Rev. 2022-08-10)

Acreditações e Certificações



Instituto Superior de Estatística e Gestão da Informação
Universidade Nova de Lisboa