

Databases & SQL



Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data

- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

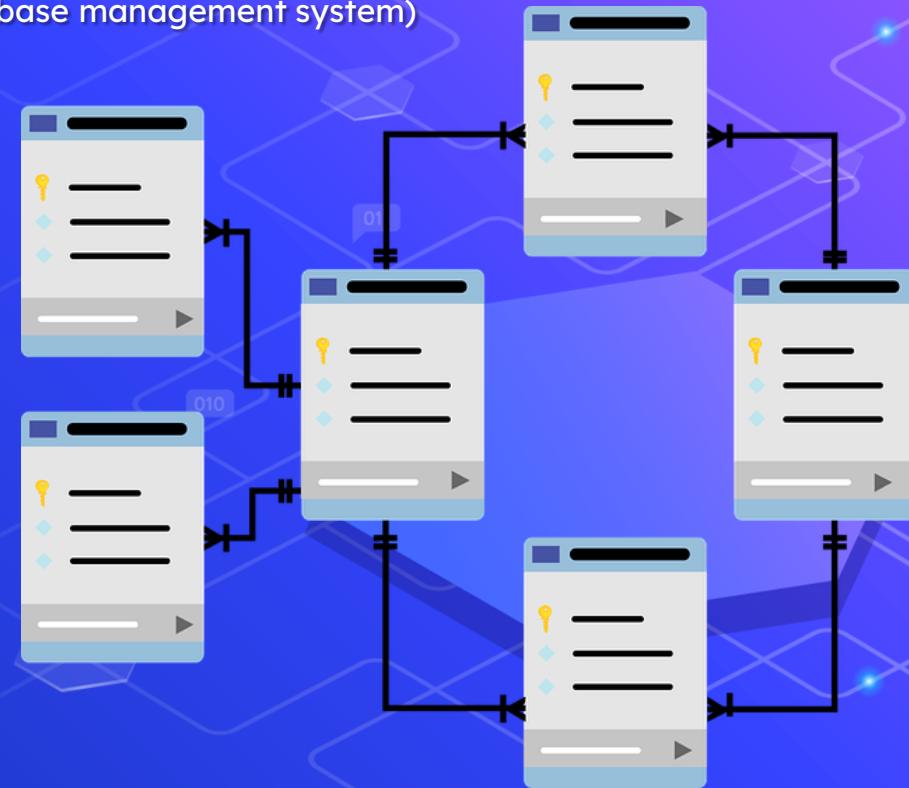
What is RDBMS. (relational database management system)

File systems made for storing and manipulating relational data.

Data organized in tabular form each table represents an entity like users, products, categories, etc...

Each table consists of columns that represent entity properties, each column can be any data type like numbers, text, etc...

SQL is an easy language to connect with database and query the data from it easily.



Agenda

- What is RDBMS.
- **Tables, Columns and Data types**
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Tables, Columns and Data types

Tables

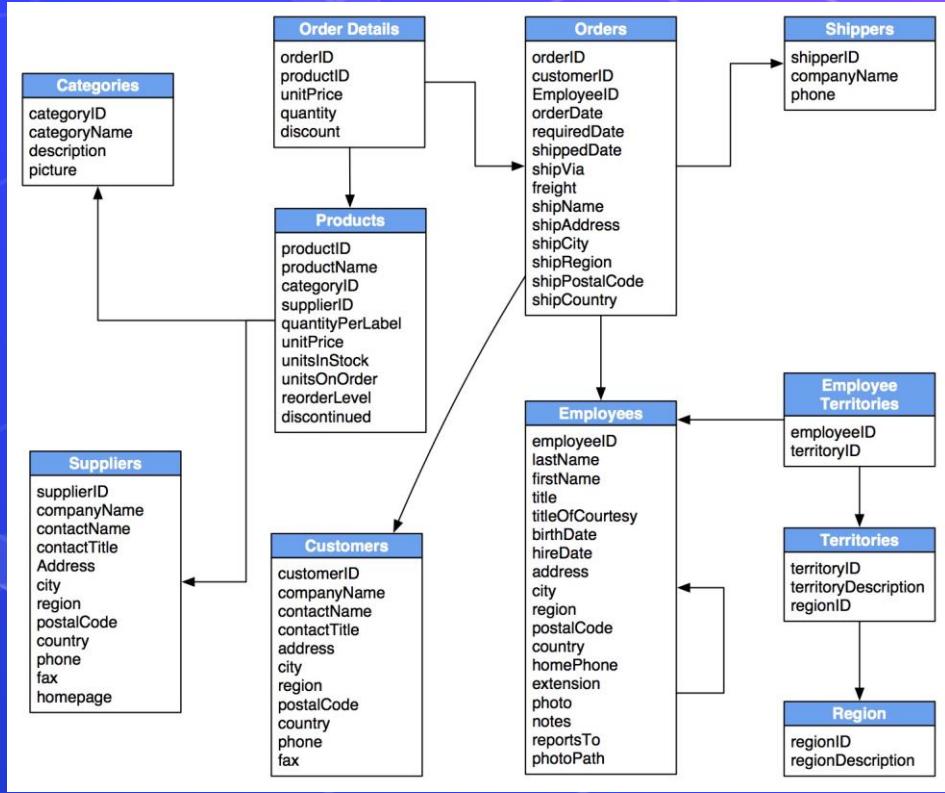
Users, posts, comments, products, ...

Columns

Users table have columns like age, gender, username, email, is_online,...

Data Types

Age is Numeric, username is TEXT
Gender is CHAR, is_online is BOOL



Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

1) Analyze the system and divide it to its main entities (tables).

for example we make an ecommerce system so the main entities tables are Categories, Products, Orders, Customers, etc...

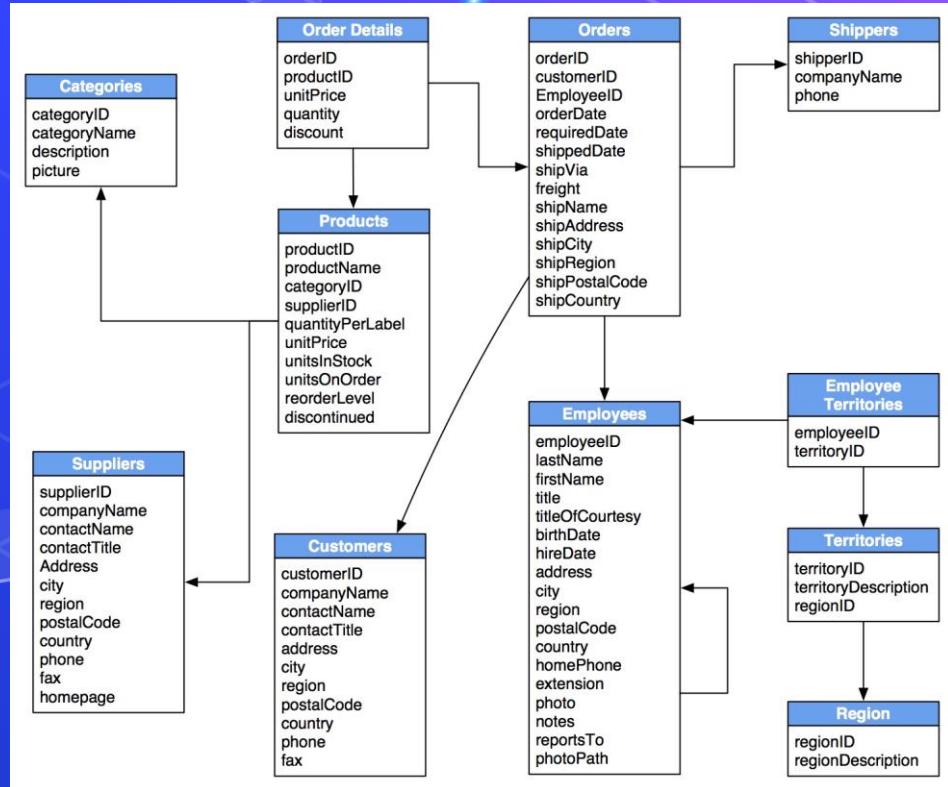
2) For each table define what its properties (columns) are and what is their data types.

for example customers table have name, city ,phone and products table have price, name, etc., the most important thing don't forget to add the unique auto increment **id** column (Primary Key) because it's the row identifier.

3) Define tables relationships and based on the relationship define the foreign keys.

for example one-to-many relationship between the products and categories tables so we define the **CategoryID** column (Foreign Key) in the products table, another relation is many-to-many relation between orders and products tables so we make Order Details table that contain the foreign keys

How to design a database.



Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- **One-To-Many & Many-To-Many Relationships.**
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data

- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

One-To-Many & Many-To-Many Relationships

001

Products --->

CategoryID is a foreign key that point to the CategoryID primary key in categories table

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97
10	Ikura	4	8	12 - 200 ml jars	31

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
3	Confections	Desserts, candies, and sweet breads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish

<--- Categories

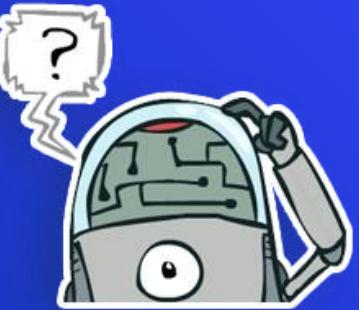
CategoryID is a primary key

010

One-To-Many & Many-To-Many Relationships

001

Products --->



How can we
represent the
relationship ?!

Orders --->

ProductID	ProductName	SupplierID	CategoryID	Unit
1	Chais	1	1	10 box
2	Chang	1	1	24 - 12 oz
3	Aniseed Syrup	1	2	12 - 55
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz
5	Chef Anton's Gumbo Mix	2	2	36 box
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 kg
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz
9	Mishi Kobe Niku	4	6	18 - 500 g
10	Ikura	4	8	12 - 200 g

OrderID	CustomerID	EmployeeID	OrderDate
10248	90	5	1996-07-04
10249	81	6	1996-07-05
10250	34	4	1996-07-08
10251	84	3	1996-07-08
10252	76	4	1996-07-09
10253	34	3	1996-07-10
10254	14	5	1996-07-11
10255	68	9	1996-07-12

Products

One-To-Many & Many-To-Many Relationships

Order Details

OrderDetailID	OrderID	ProductID	Quantity
1	10248	1	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40
6	10250	41	10
7	10250	1	35
8	10250	65	15
9	10251	1	6
10	10251	57	15

We make a new table that have both foreign keys

ProductID	ProductName
1	Chais
2	Chang
3	Aniseed Syrup
4	Chef Anton's Cajun Seasoning
5	Chef Anton's Gumbo Mix
6	Grandma's Boysenberry Spread
7	Uncle Bob's Organic Dried Pears
8	Northwoods Cranberry Sauce
9	Mishi Kobe Niku
10	Ikura

Orders

OrderID	CustomerID
10248	90
10249	81
10250	34
10251	84
10252	76
10253	34
10254	14

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data

- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Examples: FB, Youtube, Souq.com



Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

SQL

SQL or Structured Query Language used to communicate with the database by executing commands for creating tables and columns and also reading, filtering, ordering, inserting, updating, deleting data and more.



You can try SQL on bunch of examples from this link

<https://www.w3schools.com/sql/default.asp>

<https://www.sqlitetutorial.net>

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

CRUD

CRUD

Create

Read

Update

Delete



Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- **Selecting data**
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Selecting data

- Select all columns from a table
- Select specific columns from a table



```
1 SELECT * FROM Customers
```



010

011

001

100

Selecting data

- Select all columns from a table
- Select specific columns from a table

```
1 SELECT CustomerName, City FROM Customers
```

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Filtering data

- hexagon WHERE Clause
- hexagon Filtering with Text vs Numeric Fields
- hexagon Comparison operators ($>$, $<$, \geq , \leq , $=$, \neq)
- hexagon BETWEEN
- hexagon IN
- hexagon LIKE
- hexagon Logic operators (AND, OR, NOT)
- hexagon IS NULL, IS NOT NULL



```
1 SELECT * FROM Customers WHERE {STATEMENT}
```

Filtering data

- WHERE Clause
- Filtering with Text vs Numeric Fields
- Comparison operators ($>$, $<$, \geq , \leq , $=$, \neq)
- BETWEEN
- IN
- LIKE
- Logic operators (AND, OR, NOT)
- IS NULL, IS NOT NULL



```
1 -- With Text Field --
2 SELECT * FROM Customers WHERE Country='Mexico'
3
4
5 -- With Numeric Field --
6 SELECT * FROM Products WHERE Price=30
7
```

Filtering data

- hexagon WHERE Clause
- hexagon Filtering with Text vs Numeric Fields
- hexagon Comparison operators (=, !=, >, <, >=, <=)
- hexagon BETWEEN
- hexagon IN
- hexagon LIKE
- hexagon Logic operators (AND, OR, NOT)
- hexagon IS NULL, IS NOT NULL

```
1 SELECT * FROM Customers WHERE Country = 'Mexico'  
2  
3 SELECT * FROM Customers WHERE Age != 25  
4  
5 SELECT * FROM Products WHERE Price > 30  
6  
7 SELECT * FROM Books WHERE Rate >= 3  
8  
9 SELECT * FROM Products WHERE Price < 30  
10  
11 SELECT * FROM Books WHERE Rate <= 3
```

Filtering data

- hexagon WHERE Clause
- hexagon Filtering with Text vs Numeric Fields
- hexagon Comparison operators (=, !=, >, <, >=, <=)
- hexagon BETWEEN
- hexagon IN
- hexagon LIKE
- hexagon Logic operators (AND, OR, NOT)
- hexagon IS NULL, IS NOT NULL

```
1 SELECT * FROM Products  
2 WHERE Price BETWEEN 10 AND 20  
3  
4  
5 SELECT * FROM Products  
6 WHERE Price NOT BETWEEN 10 AND 20
```

Filtering data

- hexagon WHERE Clause
- hexagon Filtering with Text vs Numeric Fields
- hexagon Comparison operators (=, !=, >, <, >=, <=)
- hexagon BETWEEN
- hexagon IN
- hexagon LIKE
- hexagon Logic operators (AND, OR, NOT)
- hexagon IS NULL, IS NOT NULL

```
1 SELECT * FROM Customers  
2 WHERE Country IN ('Germany', 'France', 'UK')  
3  
4  
5 SELECT * FROM Products  
6 WHERE Price NOT IN (10, 30, 50)
```

Filtering data

- WHERE Clause
- Filtering with Text vs Numeric Fields
- Comparison operators (=, !=, >, <, >=, <=)
- BETWEEN
- IN
- LIKE
- Logic operators (AND, OR, NOT)
- IS NULL, IS NOT NULL

```
1 -- Select rows that start with 'a'  
2 SELECT * FROM Customers WHERE CustomerName LIKE 'a%'  
3  
4  
5 -- Select rows that start with any letter except 'a'  
6 SELECT * FROM Customers WHERE CustomerName NOT LIKE 'a%'  
7  
8  
9 -- Select rows that end with 'a'  
10 SELECT * FROM Customers WHERE CustomerName LIKE '%a'  
11  
12  
13 -- Select rows that have 'or' in any position  
14 SELECT * FROM Customers WHERE CustomerName LIKE '%or%'  
15  
16  
17 -- Select rows that start with 'a' and end with 'o'  
18 SELECT * FROM Customers WHERE CustomerName LIKE 'a%o'
```

Filtering data

- hexagon WHERE Clause
- hexagon Filtering with Text vs Numeric Fields
- hexagon Comparison operators (=, !=, >, <, >=, <=)
- hexagon BETWEEN
- hexagon IN
- hexagon LIKE
- hexagon Logic operators (AND, OR, NOT)
- hexagon IS NULL, IS NOT NULL

```
1 SELECT * FROM Customers
2 WHERE Country='Germany' AND City='Berlin'
3
4
5 SELECT * FROM Customers
6 WHERE City='Berlin' OR City='München'
7
8
9 SELECT * FROM Products
10 WHERE Price NOT BETWEEN 10 AND 30
11
12
13 -- combine multi logic operator in one statement
14 SELECT * FROM Customers
15 WHERE Country='Germany' AND (City='Berlin' OR City='München')
16
```

Filtering data

- hexagon WHERE Clause
- hexagon Filtering with Text vs Numeric Fields
- hexagon Comparison operators (=, !=, >, <, >=, <=)
- hexagon BETWEEN
- hexagon IN
- hexagon LIKE
- hexagon Logic operators (AND, OR, NOT)
- hexagon IS NULL, IS NOT NULL

```
1 SELECT CustomerName, ContactName, Address  
2 FROM Customers  
3 WHERE Address IS NULL  
4  
5  
6 SELECT CustomerName, ContactName, Address  
7 FROM Customers  
8 WHERE Address IS NOT NULL
```

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Ordering data

- Order rows by single column ascending
- Order rows by single column descending
- Filter then Order
- Order rows by several columns



```
1 SELECT * FROM Customers ORDER BY Country ASC
2
3 -- by default ORDER BY is in ascending order
4 SELECT * FROM Customers ORDER BY Country
5
```

Ordering data

- Order rows by single column ascending
- Order rows by single column descending
- Filter then Order
- Order rows by several columns



```
1 SELECT * FROM Customers ORDER BY Country DESC
```

Ordering data

- Order rows by single column ascending
- Order rows by single column descending
- Filter then Order
- Order rows by several columns



```
1 SELECT * FROM Products  
2 WHERE Price BETWEEN 10 and 30  
3 ORDER BY ProductName DESC
```

Ordering data

- Order rows by single column ascending
- Order rows by single column descending
- Filter then Order
- Order rows by several columns**

```
1 /*
2 Order first with Country ASC
3 and when there are many rows that have same countries
4 order them by CustomerName ASC
5 */
6 SELECT * FROM Customers
7 ORDER BY Country, CustomerName
8
9
10 /*
11 Order first with Country ASC
12 and when there are many rows that have same countries
13 order them by CustomerName DESC
14 */
15 SELECT * FROM Customers
16 ORDER BY Country ASC, CustomerName DESC
17
18
19 /*
20 Order first with Country DESC
21 and when there are many rows that have same countries
22 order them by CustomerName DESC
23 */
24 SELECT * FROM Customers
25 ORDER BY Country DESC, CustomerName DESC
```

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data

- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Limiting data

- Hexagon icon: Limit resulted rows to a specific number from first row
- Hexagon icon: Limit resulted rows to a specific number from a specific offset row

```
1 -- return only the first 3 rows  
2 SELECT * FROM Customers LIMIT 3
```

Limiting data

- Limit resulted rows to a specific number from first row
- Limit resulted rows to a specific number from a specific offset row

```
1 -- return only 3 rows after the second row  
2 SELECT * FROM Customers LIMIT 2, 3  
3  
4 -- or you can use OFFSET keyword  
5 SELECT * FROM Customers LIMIT 3 OFFSET 2
```



Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data

- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Aggregate Functions

-  MIN
-  MAX
-  COUNT
-  AVG
-  SUM

```
1 SELECT MIN(Price) AS SmallestPrice FROM Products
```

Aggregate Functions

-  MIN
-  MAX
-  COUNT
-  AVG
-  SUM



```
1 SELECT MAX(Price) AS LargestPrice FROM Products
```

Aggregate Functions

-  MIN
-  MAX
-  COUNT
-  AVG
-  SUM

```
1 SELECT COUNT(ProductID) AS ProductsCount FROM Products
```

Aggregate Functions

-  MIN
-  MAX
-  COUNT
-  AVG
-  SUM

```
1 SELECT AVG(PRICE) AS AveragePrice FROM Products
```

Aggregate Functions

- MIN
- MAX
- COUNT
- AVG
- SUM

```
1 SELECT SUM(Quantity) AS QtySum FROM OrderDetails
```

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data

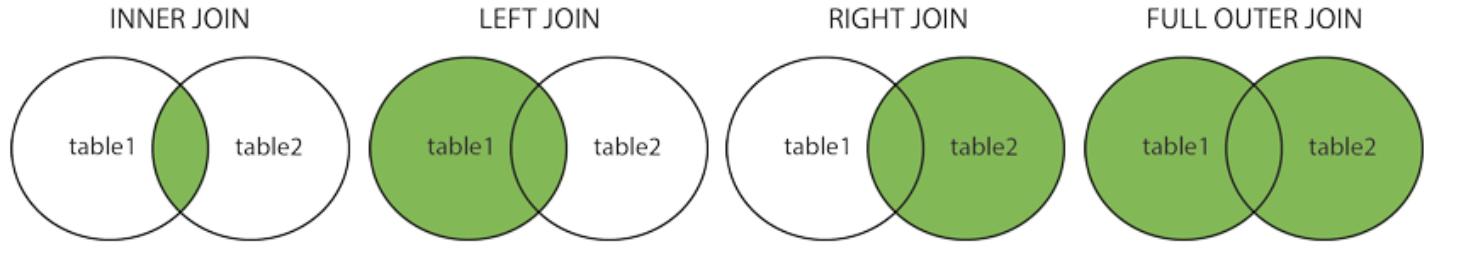
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Joining tables

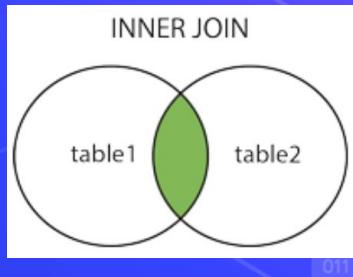
Different Types of SQL JOINS

Here are the different types of the JOINs in SQL:

- **(INNER) JOIN:** Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN:** Return all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN:** Return all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN:** Return all records when there is a match in either left or right table



Joining tables (INNER)

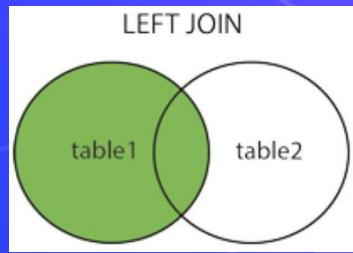


The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns. If there are records in the "Orders" table that do not have matches in "Customers", these orders will not be shown!



```
1 -- This SQL statement selects all orders with customer information
2 SELECT Orders.OrderID, Customers.CustomerName
3 FROM Orders INNER JOIN Customers
4 ON Orders.CustomerID = Customers.CustomerID
```

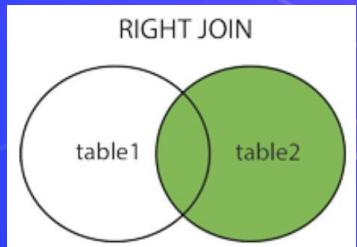
Joining tables (LEFT)



The LEFT JOIN keyword returns all records from the left table (Customers), and the matched records from the right table (Orders). The result is NULL from the right side, if there is no match.

```
1 -- This SQL statement select all customers  
2 -- and any orders they might have  
3 SELECT Customers.CustomerName, Orders.OrderID  
4 FROM Customers LEFT JOIN Orders  
5 ON Customers.CustomerID = Orders.CustomerID  
6 ORDER BY Customers.CustomerName
```

Joining tables (RIGHT)

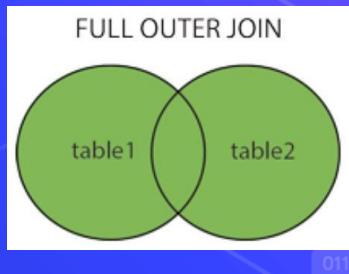


The RIGHT JOIN keyword returns all records from the right table (Employees), and the matched records from the left table (Orders). The result is NULL from the left side, when there is no match.



```
1 -- This SQL statement will return all employees
2 -- and any orders they might have placed
3 SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
4 FROM Orders RIGHT JOIN Employees
5 ON Orders.EmployeeID = Employees.EmployeeID
6 ORDER BY Orders.OrderID
```

Joining tables (FULL)



The FULL OUTER JOIN keyword returns all matching records from both tables whether the other table matches or not.

So, if there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well with NULL

```
1 SELECT Customers.CustomerName, Orders.OrderID  
2 FROM Customers FULL OUTER JOIN Orders  
3 ON Customers.CustomerID=Orders.CustomerID  
4 ORDER BY Customers.CustomerName;
```

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data

- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Grouping data

- GROUP BY
- HAVING

```
1 SELECT COUNT(CustomerID) AS CustomersCount, City  
2 FROM Customers WHERE Country = 'France'  
3 GROUP BY City  
4 ORDER BY CustomersCount DESC
```

Grouping data

- GROUP BY
- HAVING

```
1 SELECT COUNT(CustomerID) AS CustomersCount, City  
2 FROM Customers WHERE Country = 'France'  
3 GROUP BY City  
4 HAVING CustomersCount > 4  
5 ORDER BY CustomersCount DESC, City ASC
```

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Subqueries

```
1 SELECT * FROM Products  
2 WHERE SupplierID =  
3 (SELECT SupplierID FROM Suppliers WHERE SupplierName='Exotic Liquid')
```

```
1 SELECT * FROM Orders  
2 WHERE EmployeeID IN  
3 (SELECT EmployeeID FROM Employees WHERE Country='Canada' )
```

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Inserting new data

```
1 INSERT INTO Customers  
2 (CustomerName, ContactName, Address, City, PostalCode, Country)  
3 VALUES  
4 ('Ahmed', 'Mr. Ahmed', 'Maadi 21', 'Cairo', '4006', 'Egypt')
```

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Updating data

```
1 /*
2 The following SQL statement updates the first customer
3 (CustomerID = 1) with a new contact person and a new city.
4 */
5 UPDATE Customers
6 SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
7 WHERE CustomerID = 1
8
9
10 /*
11 The following SQL statement will update the contactname
12 to "Juan" for all records where country is "Mexico"
13 */
14 UPDATE Customers
15 SET ContactName='Juan'
16 WHERE Country='Mexico'
```

Updating data

```
1 UPDATE Products SET Stock = Stock - 2 WHERE ProductID = 200  
2  
3 UPDATE Products SET Stock = Stock + 2 WHERE ProductID = 200
```

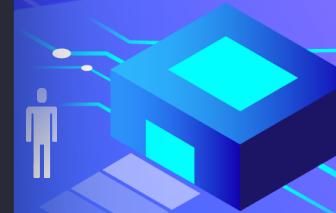
Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data

- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Deleting data

```
1 /*  
2 The following SQL statement deletes the customer  
3 "Alfreds Futterkiste" from the "Customers" table  
4 */  
5 DELETE FROM Customers  
6 WHERE CustomerName='Alfreds Futterkiste'  
7  
8  
9 /*  
10 The following SQL statement deletes all rows  
11 in the "Customers" table, without deleting the table  
12 */  
13 DELETE FROM Customers
```



001

100

Check Also

- Create / Drop / Backup Database
- Create / Drop / Alter Table
- Constraints (Not Null, Unique, Primary and Foreign Key)
- Indexes
- Database normalization
- Transactions
- Triggers & Stored Procedures
- Views
- SQL Injection
- NoSQL
- Replication
- Shards

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

Python and SQLite

There is a lot of RDBMS out there

- SQLite
- MySQL
- PostgreSQL
- ...

We will use SQLite because it's simple and easy to work with.

Python and SQLite

 Read

 Create

 Update

 Delete

```
1 import sqlite3
2
3 connection = sqlite3.connect("weather.db")
4 cursor = connection.cursor()
5
6 sql = "SELECT * FROM users WHERE age > ? AND username LIKE ?"
7 data = [20, '%hmed']
8
9 cursor.execute(sql, data)
10 rows = cursor.fetchall()
11 connection.close()
12
13 print(rows)
14
15 """
16 [
17     (3, 'eman', 'Giza', 20),
18     (12, 'Eslam', 'Cairo', 20),
19     (15, 'Ahmed', 'Cairo', 22)
20 ]
21 """
```

Python and SQLite

- hexagon icon Read
- hexagon icon Create
- hexagon icon Update
- hexagon icon Delete

```
 1 import sqlite3  
2  
3 connection = sqlite3.connect("weather.db")  
4 cursor = connection.cursor()  
5  
6 sql = "INSERT INTO users (name,city,age) VALUES (?, ?, ?)"  
7 data = ['Sara', 'Cairo', '35']  
8  
9 cursor.execute(sql, data)  
10 connection.commit()  
11 connection.close()
```

Python and SQLite

- Read
- Create
- Update
- Delete

```
1 import sqlite3
2
3 connection = sqlite3.connect("weather.db")
4 cursor = connection.cursor()
5
6 sql = "UPDATE users SET name=?, age=? WHERE id=?"
7 data = [ 'Mohammed', 20, 12]
8
9 cursor.execute(sql, data)
10 connection.commit()
11 connection.close()
```

Python and SQLite

- Read
- Create
- Update
- Delete

```
1 import sqlite3
2
3 connection = sqlite3.connect("weather.db")
4 cursor = connection.cursor()
5
6 sql = "DELETE FROM users WHERE age >= ?"
7 data = [30]
8
9 cursor.execute(sql, data)
10 connection.commit()
11 connection.close()
```

Agenda

- What is RDBMS.
- Tables, Columns and Data types
- How to design a database.
- One-To-Many & Many-To-Many Relationships.
- Examples: FB, Youtube, Souq.com
- SQL
- CRUD
- Selecting data
- Filtering data
- Ordering data
- Limiting data
- Aggregate Functions
- Joining tables
- Grouping data
- Subqueries
- Inserting new data
- Updating data
- Deleting data
- Python and SQLite
- DB Browser for SQLite

DB Browser for SQLite

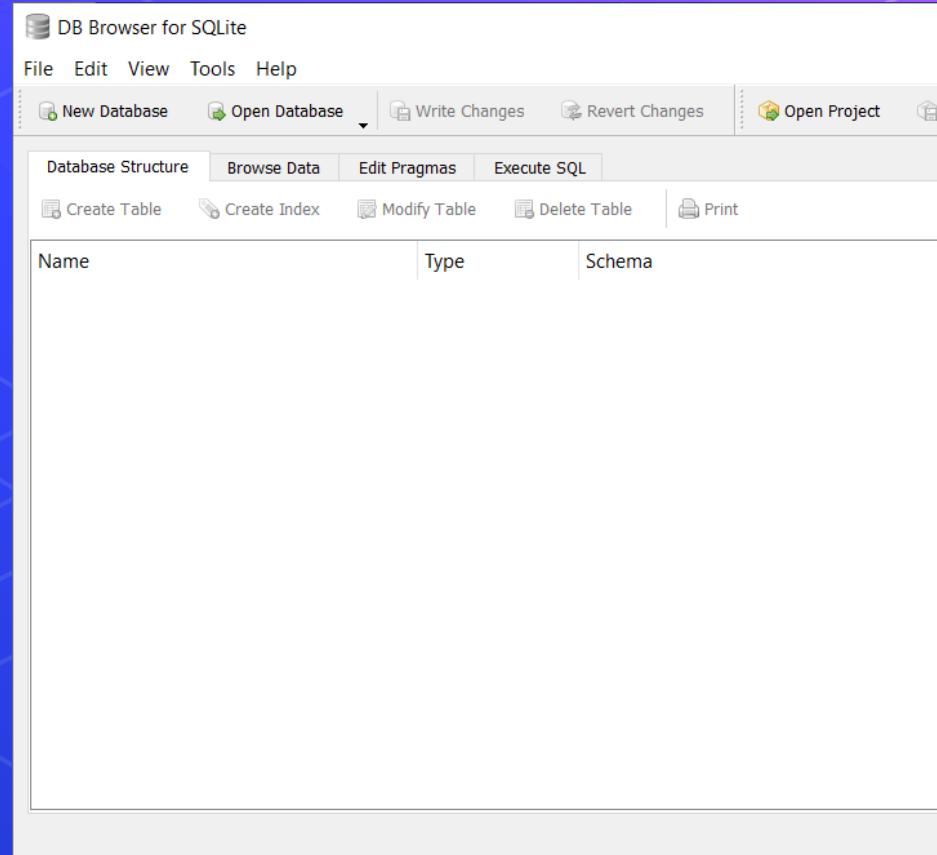
GUI to work with SQLite

Download

<https://sqlitebrowser.org>

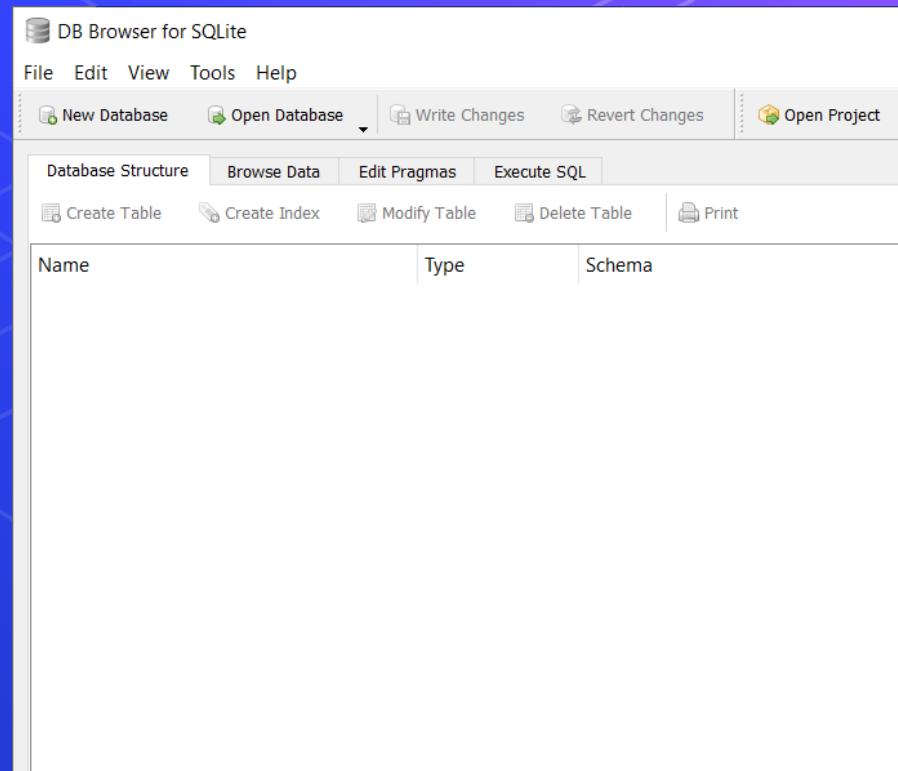
Tutorial

<https://www.youtube.com/playlist?list=PLU70qqWW4frGdwNh4czgTrCrHvPhyt2aI>



DB Browser for SQLite

- Open Database
- Create Database and tables
- Create columns
- Create Foreign Keys
- Modify tables
- CRUD with GUI
- Execute SQL
- Import from SQL file
- Export table to CSV



Questions ?!



Thanks!

>_ Live long and prosper

