Study Sessions SQL

Prepared by Andres Pascasio
GitHub Profile

Introduction to SQL: Essential Concepts and Commands

1. Overview

Objective:
Learn the foundational concepts and essential SQL commands needed to interact with relational databases, including creating databases and tables, querying data, and performing data manipulation tasks such as inserting, updating, and deleting records.

Audience:
This guide is designed for beginners who are new to SQL and relational databases, as well as intermediate learners looking to reinforce their understanding of SQL basics.

---

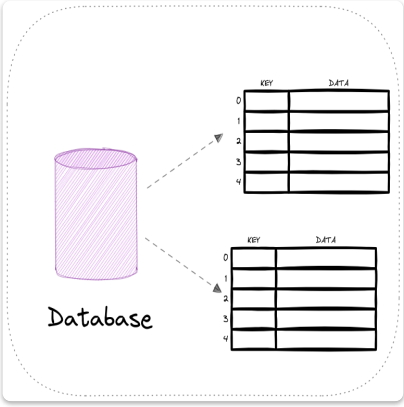
2. Command Overview

In this section, we will introduce and explain the essential SQL commands and concepts covered in this guide.

2.1. Understanding Databases

- **Purpose:** A database is a structured collection of data that is stored and managed electronically.
- **Explanation:** Databases are designed to store, retrieve, and manage data efficiently. They consist of one or more tables and can be used to handle a wide range of data

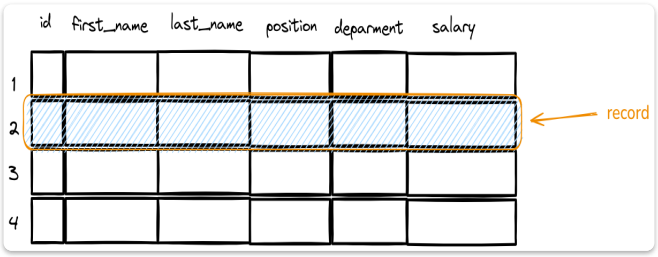
types and relationships.



2.2. Understanding Tables in a Database

- **Purpose:** Tables are the core structure in a relational database where data is stored. Each table consists of rows and columns.
- **Explanation:** A table organises data into rows (records) and columns (attributes). Each column has a specific data type, and each row represents a single entry in the table.

Table Structure and Example Data



id	first_name	last_name	department
1	John	Doe	Sales
2	Jane	Smith	IT
3	Emily	Johnson	IT
4	Michael	Brown	HR
5	Sarah	Davis	IT
6	David	Wilson	Finance

record

2.3. Creating Databases and Tables

- **Creating a Database:**

```
CREATE DATABASE my_database;
```

- **Creating a Table:**

```
CREATE TABLE employees (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    position VARCHAR(50),
    hire_date DATE
);
```

- **Explanation:** The `CREATE DATABASE` command initialises a new database, and the `CREATE TABLE` command defines a table with specific columns and their data types.

2.4. Basic SELECT Statement

- **Purpose:** Retrieve data from a table.
- **Usage:**

```
SELECT * FROM employees;
```

- **Explanation:** The `SELECT` statement fetches all columns for all rows in the `employees` table. The asterisk (*) indicates that all columns should be returned.

2.5. SELECT with WHERE Clause and Conditions

- **Purpose:** Filter records based on specific conditions.
- **Usage:**

```
SELECT * FROM employees WHERE position = 'Manager';
```

- **Additional Conditions:**

```
SELECT * FROM employees WHERE hire_date > '2023-01-01' AND position =
'Developer';
```

- **Explanation:** The `WHERE` clause allows you to specify conditions to filter the results. Multiple conditions can be combined using `AND` or `OR`.

2.6. SELECT with GROUP BY Clause

- **Purpose:** Group rows that have the same values in specified columns into summary rows, like "total" or "average."
- **Usage:**

```
SELECT position, COUNT(*)
FROM employees
GROUP BY position;
```

- **Explanation:** The `GROUP BY` clause is used in collaboration with aggregate functions (like `COUNT`, `SUM`, `AVG`, `MAX`, `MIN`) to group the result set by one or more columns. In the example above, the query groups the employees by their `position` and counts the number of employees in each position.
- **Using GROUP BY with HAVING:**

```
SELECT department, AVG(salary)
FROM employees
GROUP BY department
HAVING AVG(salary) > 60000;
```

- **Explanation:** The `HAVING` clause is similar to the `WHERE` clause, but it is used to filter groups instead of individual rows. The example filters departments to show only those where the average salary is greater than $60,000.

2.7. Inserting Data

- **Purpose:** Add new records to a table.
- **Usage:**

```
INSERT INTO employees (id, first_name, last_name, position, hire_date)
VALUES (1, 'John', 'Doe', 'Developer', '2024-05-01');
```

- **Explanation:** The `INSERT INTO` statement adds a new row to the `employees` table with specified values for each column.

2.8. Updating Data

- **Purpose:** Modify existing records in a table.
- **Usage:**

```
UPDATE employees
SET position = 'Developer'
WHERE id = 1;
```

- **Explanation:** The `UPDATE` statement changes existing records based on a condition specified in the `WHERE` clause.

2.9. Deleting Data

- **Purpose:** Remove records from a table.

- **Usage:**

```
DELETE FROM employees WHERE id = 1;
```

- **Explanation:** The `DELETE` statement removes rows from the `employees` table based on the condition provided in the `WHERE` clause.

---

3. Q&A and Wrap-up

3.1 Q&A

- **Questions:**
  If you have any questions or need clarification on any of the topics covered in this guide, feel free to ask! Whether you're unsure about a particular SQL command or need help with an exercise, we're here to support your learning journey.
  - **GitHub Discussions:**
    You can use the GitHub Discussions feature in this repository to ask questions, engage with other learners, and share knowledge.

3.2 Further Learning

- **Recommended Practice:**
  - **Hands-On Practice:**
    To reinforce the SQL concepts you've learned, practice by working with real databases. Use the provided scripts to create and populate tables, and experiment with various queries and commands. This hands-on experience is crucial for mastering SQL.
  - **Online SQL Platforms:**
    Consider using online SQL platforms like SQLFiddle or LeetCode to practice writing queries and solve SQL challenges.
- **Advanced Topics:**
  - **Database Optimisation:**
    Learn about indexing, query optimisation, and database normalisation to improve performance and efficiency in your SQL queries.
  - **Stored Procedures and Functions:**
    Explore how to create and use stored procedures and functions to encapsulate complex logic within the database.
  - **Database Design:**
    Delve into advanced database design principles, including designing relational

schemas and understanding different types of relationships (one-to-many, many-to-many).

- **Transaction Management:**
  Understand how to manage transactions, ensure data integrity, and implement proper error handling in SQL.

3.3 Feedback and Next Steps

- **Feedback:**
  Your feedback is invaluable to us. We encourage you to share your thoughts on this guide:
  - **Submitting Issues or Pull Requests on GitHub:**
    - Visit the GitHub repository for this guide.
    - Open an issue to report any errors, suggest improvements, or share your experiences.
    - Feel free to submit a pull request if you'd like to contribute by adding content or correcting any inaccuracies.
- **Next Steps:**
  - **Apply Your Knowledge:**
    Use what you've learned by working on real projects or datasets. Experiment with building queries, performing data analysis, or creating small database-driven applications.
  - **Explore More SQL Topics:**
    Consider diving deeper into specific SQL topics that interest you, such as data warehousing, big data processing with SQL, or integrating SQL with other programming languages.
  - **Contribute to the Community:**
    As you advance, consider contributing to open-source projects, participating in SQL forums, or even writing your own tutorials to help others learn.

---

4. Resources

https://sqliteonline.com/

```
CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    position VARCHAR(50),
```

```
    department VARCHAR(50),
    salary DECIMAL(10, 2),
    hire_date DATE
);


INSERT INTO employees (first_name, last_name, position, department, salary,
hire_date)
VALUES
('John', 'Doe', 'Manager', 'Sales', 75000, '2022-05-15'),
('Jane', 'Smith', 'Developer', 'IT', 65000, '2023-02-10'),
('Emily', 'Johnson', 'Developer', 'IT', 70000, '2023-03-22'),
('Michael', 'Brown', 'Manager', 'HR', 80000, '2021-07-30'),
('Sarah', 'Davis', 'Developer', 'IT', 72000, '2022-11-12'),
('David', 'Wilson', 'Analyst', 'Finance', 68000, '2023-01-05'),
('Laura', 'Taylor', 'Manager', 'Sales', 77000, '2023-07-15'),
('Robert', 'Miller', 'Developer', 'IT', 73000, '2023-01-25'),
('Linda', 'Anderson', 'Analyst', 'Finance', 69000, '2022-06-18'),
('James', 'Thomas', 'HR Specialist', 'HR', 54000, '2021-09-10');
```

5.Scenario

Scenario: Online Store

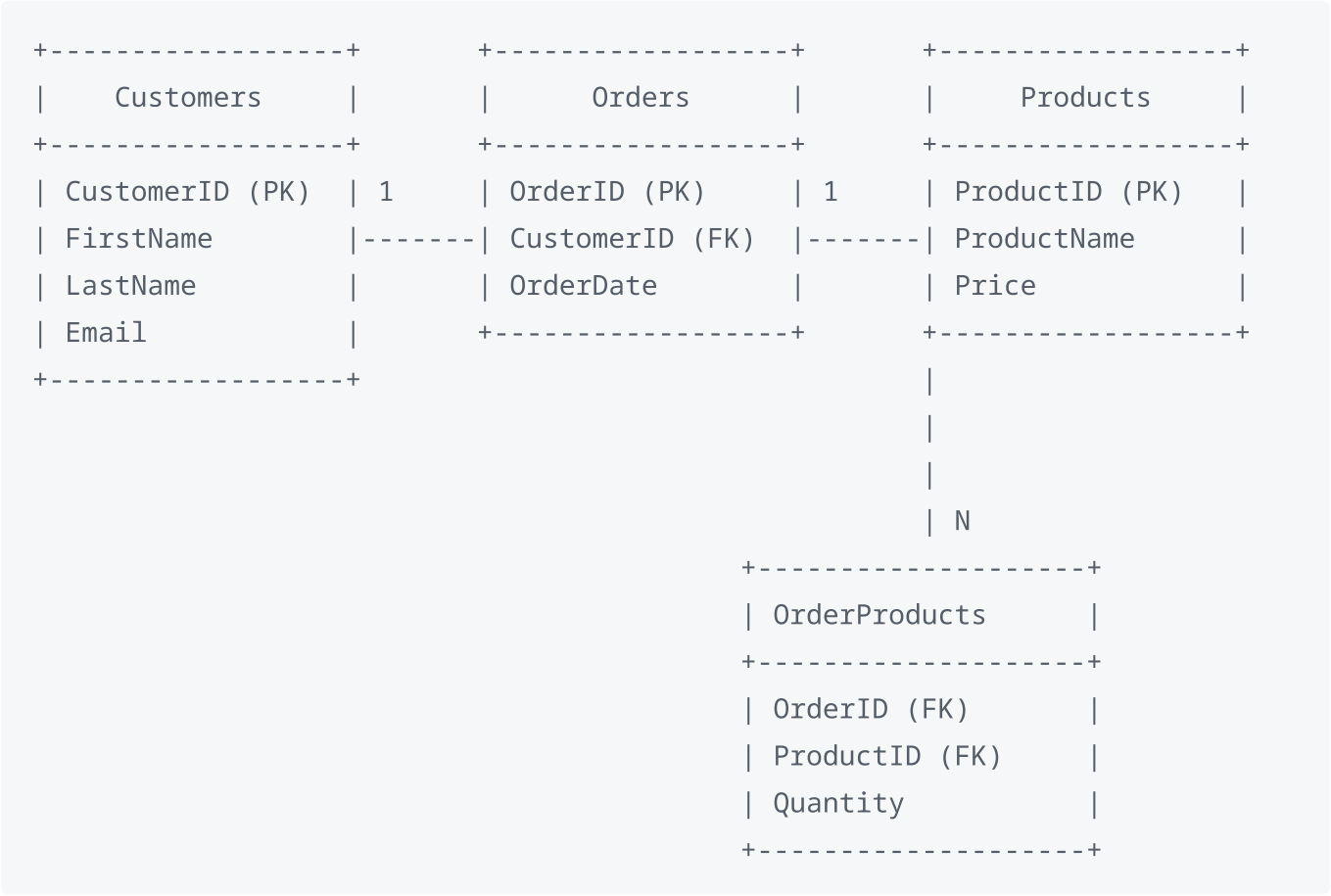
1. Tables:

- **Customers**: Stores customer information.
- **Products**: Stores product information.
- **Orders**: Stores order details.
- **OrderProducts**: Manages the many-to-many relationship between orders and products.

2. Relationships:

- Each order is placed by one customer (foreign key in Orders table).
- Each order can contain multiple products (many-to-many relationship, managed through the `OrderProducts` table).
- Each product can appear in multiple orders.

Diagram:



SQL Scripts

1. Create the Database:

```sql
CREATE DATABASE OnlineStore;
USE OnlineStore;
```

2. Create Tables:

```sql
-- Create Customers Table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100)
);

-- Create Products Table
CREATE TABLE Products (
    ProductID INT PRIMARY KEY AUTO_INCREMENT,
    ProductName VARCHAR(100),
```

```
    Price DECIMAL(10, 2)
);

-- Create Orders Table
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY AUTO_INCREMENT,
    CustomerID INT,
    OrderDate DATE,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

-- Create OrderProducts Table
CREATE TABLE OrderProducts (
    OrderID INT,
    ProductID INT,
    Quantity INT,
    PRIMARY KEY (OrderID, ProductID),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

3. Insert Sample Data:

```
-- Insert Customers
INSERT INTO Customers (FirstName, LastName, Email) VALUES
('John', 'Doe', 'john.doe@example.com'),
('Jane', 'Smith', 'jane.smith@example.com'),
('Emily', 'Jones', 'emily.jones@example.com'),
('Michael', 'Brown', 'michael.brown@example.com'),
('Linda', 'Davis', 'linda.davis@example.com'),
('Robert', 'Wilson', 'robert.wilson@example.com'),
('Mary', 'Taylor', 'mary.taylor@example.com'),
('James', 'Anderson', 'james.anderson@example.com'),
('Patricia', 'Thomas', 'patricia.thomas@example.com'),
('William', 'Moore', 'william.moore@example.com'),
('Elizabeth', 'Jackson', 'elizabeth.jackson@example.com'),
('David', 'White', 'david.white@example.com'),
('Jennifer', 'Martin', 'jennifer.martin@example.com'),
('Richard', 'Lee', 'richard.lee@example.com'),
('Susan', 'Harris', 'susan.harris@example.com'),
('Joseph', 'Clark', 'joseph.clark@example.com'),
('Jessica', 'Lewis', 'jessica.lewis@example.com'),
('Charles', 'Walker', 'charles.walker@example.com'),
('Sarah', 'Hall', 'sarah.hall@example.com'),
('Thomas', 'Young', 'thomas.young@example.com'),
```

```
('Karen', 'Allen', 'karen.allen@example.com'),
('Daniel', 'King', 'daniel.king@example.com'),
('Nancy', 'Wright', 'nancy.wright@example.com'),
('Matthew', 'Scott', 'matthew.scott@example.com'),
('Sandra', 'Green', 'sandra.green@example.com'),
('Mark', 'Adams', 'mark.adams@example.com'),
('Lisa', 'Baker', 'lisa.baker@example.com'),
('Paul', 'Gonzalez', 'paul.gonzalez@example.com'),
('Mary', 'Carter', 'mary.carter@example.com'),
('Steven', 'Mitchell', 'steven.mitchell@example.com'),
('Laura', 'Perez', 'laura.perez@example.com'),
('Brian', 'Roberts', 'brian.roberts@example.com'),
('Helen', 'Phillips', 'helen.phillips@example.com'),
('Kevin', 'Evans', 'kevin.evans@example.com'),
('Barbara', 'Campbell', 'barbara.campbell@example.com'),
('George', 'Parker', 'george.parker@example.com'),
('Deborah', 'Collins', 'deborah.collins@example.com'),
('Andrew', 'Stewart', 'andrew.stewart@example.com'),
('Melissa', 'Sanchez', 'melissa.sanchez@example.com'),
('Joshua', 'Morris', 'joshua.morris@example.com'),
('Carol', 'Rogers', 'carol.rogers@example.com'),
('Ryan', 'Reed', 'ryan.reed@example.com'),
('Angela', 'Cook', 'angela.cook@example.com'),
('Nicholas', 'Morgan', 'nicholas.morgan@example.com'),
('Debra', 'Bell', 'debra.bell@example.com'),
('Daniel', 'Bailey', 'daniel.bailey@example.com'),
('Rebecca', 'Cooper', 'rebecca.cooper@example.com'),
('Edward', 'Richardson', 'edward.richardson@example.com'),
('Sharon', 'Cox', 'sharon.cox@example.com'),
('Patrick', 'Howard', 'patrick.howard@example.com'),
('Jessica', 'Ward', 'jessica.ward@example.com'),
('Henry', 'Watson', 'henry.watson@example.com'),
('Emma', 'Brooks', 'emma.brooks@example.com');

-- Insert Products
INSERT INTO Products (ProductName, Price) VALUES
('Laptop', 999.99),
('Mouse', 25.50),
('Keyboard', 45.75),
('Monitor', 199.99),
('Printer', 89.99),
('Headphones', 120.00),
('Webcam', 55.00),
('Desk Chair', 180.00),
('External Hard Drive', 120.00),
('USB Flash Drive', 15.00),
```

```
('Smartphone', 699.99),
('Tablet', 329.99),
('Smartwatch', 249.99),
('Speakers', 89.99),
('Router', 79.99),
('Modem', 69.99),
('Mouse Pad', 10.00),
('Docking Station', 120.00),
('Graphics Card', 300.00),
('CPU Cooler', 70.00),
('Memory RAM', 150.00);

-- Insert Orders
INSERT INTO Orders (CustomerID, OrderDate) VALUES
(1, '2024-08-01'),
(2, '2024-08-02'),
(3, '2024-08-03'),
(4, '2024-08-04'),
(5, '2024-08-05'),
(6, '2024-08-06'),
(7, '2024-08-07'),
(8, '2024-08-08'),
(9, '2024-08-09'),
(10, '2024-08-10'),
(11, '2024-08-11'),
(12, '2024-08-12'),
(13, '2024-08-13'),
(14, '2024-08-14'),
(15, '2024-08-15'),
(16, '2024-08-16'),
(17, '2024-08-17'),
(18, '2024-08-18'),
(19, '2024-08-19'),
(20, '2024-08-20'),
(21, '2024-08-21'),
(22, '2024-08-22'),
(23, '2024-08-23'),
(24, '2024-08-24'),
(25, '2024-08-25'),
(26, '2024-08-26'),
(27, '2024-08-27'),
(28, '2024-08-28'),
(29, '2024-08-29'),
(30, '2024-08-30'),
(31, '2024-08-31'),
(32, '2024-09-01'),
```

```
(33, '2024-09-02'),
(34, '2024-09-03'),
(35, '2024-09-04'),
(36, '2024-09-05'),
(37, '2024-09-06'),
(38, '2024-09-07'),
(39, '2024-09-08'),
(40, '2024-09-09'),
(41, '2024-09-10'),
(42,

 '2024-09-11'),
(43, '2024-09-12'),
(44, '2024-09-13'),
(45, '2024-09-14'),
(46, '2024-09-15'),
(47, '2024-09-16'),
(48, '2024-09-17'),
(49, '2024-09-18'),
(50, '2024-09-19'),
(51, '2024-09-20'),
(52, '2024-09-21'),
(53, '2024-09-22'),
(54, '2024-09-23'),
(55, '2024-09-24'),
(56, '2024-09-25'),
(57, '2024-09-26'),
(58, '2024-09-27'),
(59, '2024-09-28'),
(60, '2024-09-29'),
(61, '2024-09-30'),
(62, '2024-10-01'),
(63, '2024-10-02'),
(64, '2024-10-03'),
(65, '2024-10-04'),
(66, '2024-10-05'),
(67, '2024-10-06'),
(68, '2024-10-07'),
(69, '2024-10-08'),
(70, '2024-10-09'),
(71, '2024-10-10'),
(72, '2024-10-11'),
(73, '2024-10-12'),
(74, '2024-10-13'),
(75, '2024-10-14'),
(76, '2024-10-15'),
```

```
(77, '2024-10-16'),
(78, '2024-10-17'),
(79, '2024-10-18'),
(80, '2024-10-19'),
(81, '2024-10-20'),
(82, '2024-10-21'),
(83, '2024-10-22'),
(84, '2024-10-23'),
(85, '2024-10-24'),
(86, '2024-10-25'),
(87, '2024-10-26'),
(88, '2024-10-27'),
(89, '2024-10-28'),
(90, '2024-10-29'),
(91, '2024-10-30'),
(92, '2024-10-31'),
(93, '2024-11-01'),
(94, '2024-11-02'),
(95, '2024-11-03'),
(96, '2024-11-04'),
(97, '2024-11-05'),
(98, '2024-11-06'),
(99, '2024-11-07'),
(100, '2024-11-08');

-- Insert OrderProducts
INSERT INTO OrderProducts (OrderID, ProductID, Quantity) VALUES
(1, 1, 1),
(1, 2, 2),
(2, 3, 1),
(3, 4, 1),
(4, 5, 2),
(5, 6, 1),
(6, 7, 1),
(7, 8, 1),
(8, 9, 2),
(9, 10, 3),
(10, 1, 2),
(11, 2, 1),
(12, 3, 1),
(13, 4, 1),
(14, 5, 1),
(15, 6, 2),
(16, 7, 1),
(17, 8, 1),
(18, 9, 1),
```

```
(19, 10, 1),
(20, 1, 1),
(21, 2, 1),
(22, 3, 1),
(23, 4, 1),
(24, 5, 2),
(25, 6, 1),
(26, 7, 1),
(27, 8, 1),
(28, 9, 2),
(29, 10, 1),
(30, 1, 2),
(31, 2, 2),
(32, 3, 1),
(33, 4, 1),
(34, 5, 1),
(35, 6, 1),
(36, 7, 1),
(37, 8, 2),
(38, 9, 1),
(39, 10, 3),
(40, 1, 1),
(41, 2, 1),
(42, 3, 2),
(43, 4, 1),
(44, 5, 1),
(45, 6, 1),
(46, 7, 1),
(47, 8, 1),
(48, 9, 2),
(49, 10, 2),
(50, 1, 1),
(51, 2, 1),
(52, 3, 1),
(53, 4, 2),
(54, 5, 1),
(55, 6, 2),
(56, 7, 1),
(57, 8, 1),
(58, 9, 1),
(59, 10, 1),
(60, 1, 2),
(61, 2, 1),
(62, 3, 1),
(63, 4, 2),
(64, 5, 1),
```

```
(65, 6, 1),
(66, 7, 1),
(67, 8, 1),
(68, 9, 2),
(69, 10, 2),
(70, 1, 1),
(71, 2, 1),
(72, 3, 1),
(73, 4, 1),
(74, 5, 2),
(75, 6, 1),
(76, 7, 1),
(77, 8, 1),
(78, 9, 1),
(79, 10, 3),
(80, 1, 1),
(81, 2, 1),
(82, 3, 2),
(83, 4, 1),
(84, 5, 2),
(85, 6, 1),
(86, 7, 1),
(87, 8, 1),
(88, 9, 2),
(89, 10, 1),
(90, 1, 1),
(91, 2, 1),
(92, 3, 2),
(93, 4, 1),
(94, 5, 1),
(95, 6, 2),
(96, 7, 1),
(97, 8, 1),
(98, 9, 1),
(99, 10, 1),
(100, 1, 1);
```

Explanation:

1. **Customers Table:** Includes 50 customer records with unique emails.
2. **Products Table:** Contains 20 product records with varied prices.
3. **Orders Table:** Includes 100 order records, each associated with a customer.
4. **OrderProducts Table:** Manages the relationship between orders and products with varied quantities.

Prepared by Andres Pascasio
GitHub Profile