SQL Level 1

Part 1

Working with Databases using
Structured Query Language (SQL)

Benefits of Learning SQL

SQL (Structured Query Language) is Essential to Business

Uses of SQL

“Know your numbers” for any aspect of your business:

- Query/Retrieve data from a database and analyze it

Managing a database:

- Insert, update, & delete records (information) in a database
- Typically, only a Database Administrator (DBA) has permission to change the database so we won’t cover this.
  (Most people use SQL to query the info, as we’ll do in this class.)

SQL is Focused

- SQL has one main purpose: querying & working with data
- Therefore, SQL can be mastered faster than many other computer languages

SQL is Evergreen

- Originally developed in 1974
- Main features are largely unchanged
- By contrast, JavaScript has changed quite a lot over the years

Flavors of SQL

SQL is “mostly” the same everywhere, but there are minor differences.

Each type of database has its own format.

A database management system (DBMS)—like those listed below—each has its own “slightly” different flavor of SQL:

- PostgreSQL (often called Postgres)
- Microsoft SQL Server (often called SQL Server) uses Microsoft’s proprietary extension of SQL called Transact-SQL (T-SQL)
- MySQL, Oracle, etc.

What You'll Learn

Course Outline

This Class Will Cover

- PostgreSQL: Connecting to our database (on a server)
  SQL Server: Creating a database we give you (on your computer)
- PostgreSQL: Using DBeaver (an app)
  SQL Server: Using Microsoft SQL Server Management Studio (an app)
- Querying (retrieving) data from a database

Out of Scope for this Class

Database Management or Architecture

(Creating tables, inserting, updating, deleting data, etc.)

Performance Optimization

Theoretical Foundations

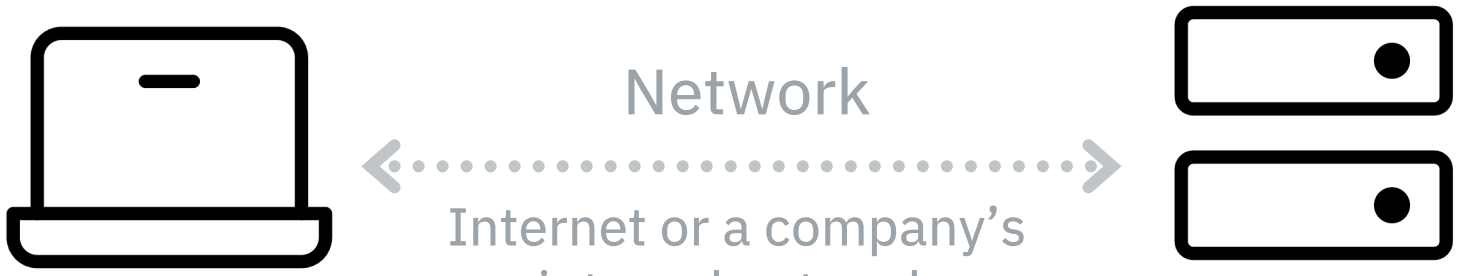
Connecting to (or Creating) Our Database

Exercise 1A in the Book

Connecting to a Database

Clients, Servers, & Databases Explained

High Level View



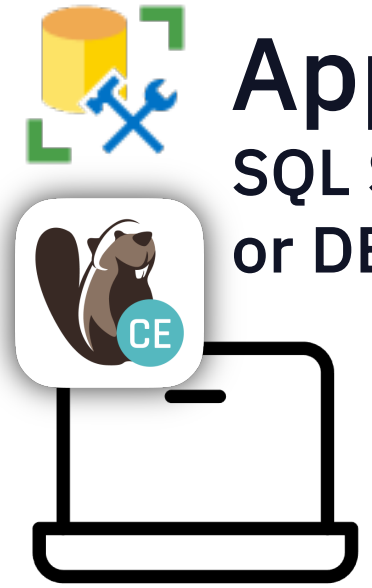
Client

Laptop/Desktop

Server/Host

Network connected computer

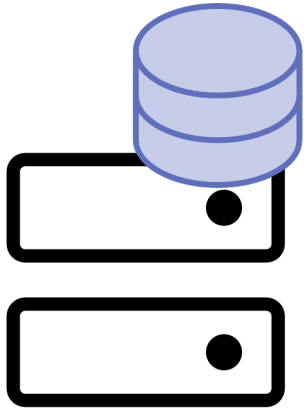
High Level View



Apps

SQL Server Management Studio
or DBeaver

Network



Databases

PostgreSQL
Microsoft SQL Server

Client

Laptop/Desktop

Server/Host

Network connected computer

Microsoft SQL Server

- In class you're working with a database on your computer (which was created by our SQL script).
- Outside of class, you'll likely connect to a server using information provided by the company you work with.

The Database

- A database is a collection of tables.
- We use a language called SQL (Structured Query Language) for all actions.
- A server can contain many databases.
  Each database can contain many tables.

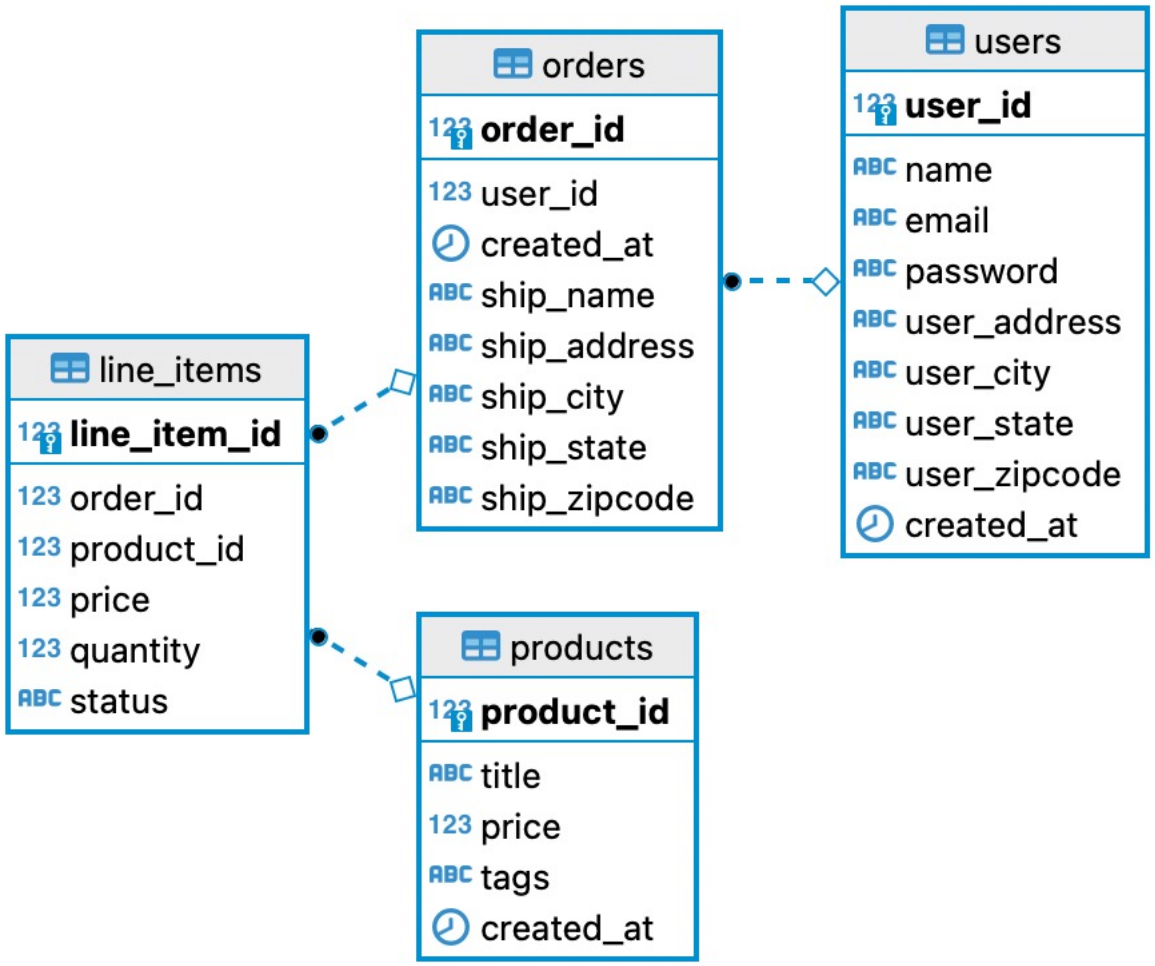
Tables in the Database

- A table is a collection of related information stored in rows & columns.
- Info for each item is a row.
- A table is also called a relation because it stores the relationship between data as columns & rows.

book_id	book_title	price	category
B001	The Long Journey	19.95	Fiction
B002	Manager's Manual	20.50	Nonfiction
B003	History of Greek Isles	33.95	Nonfiction

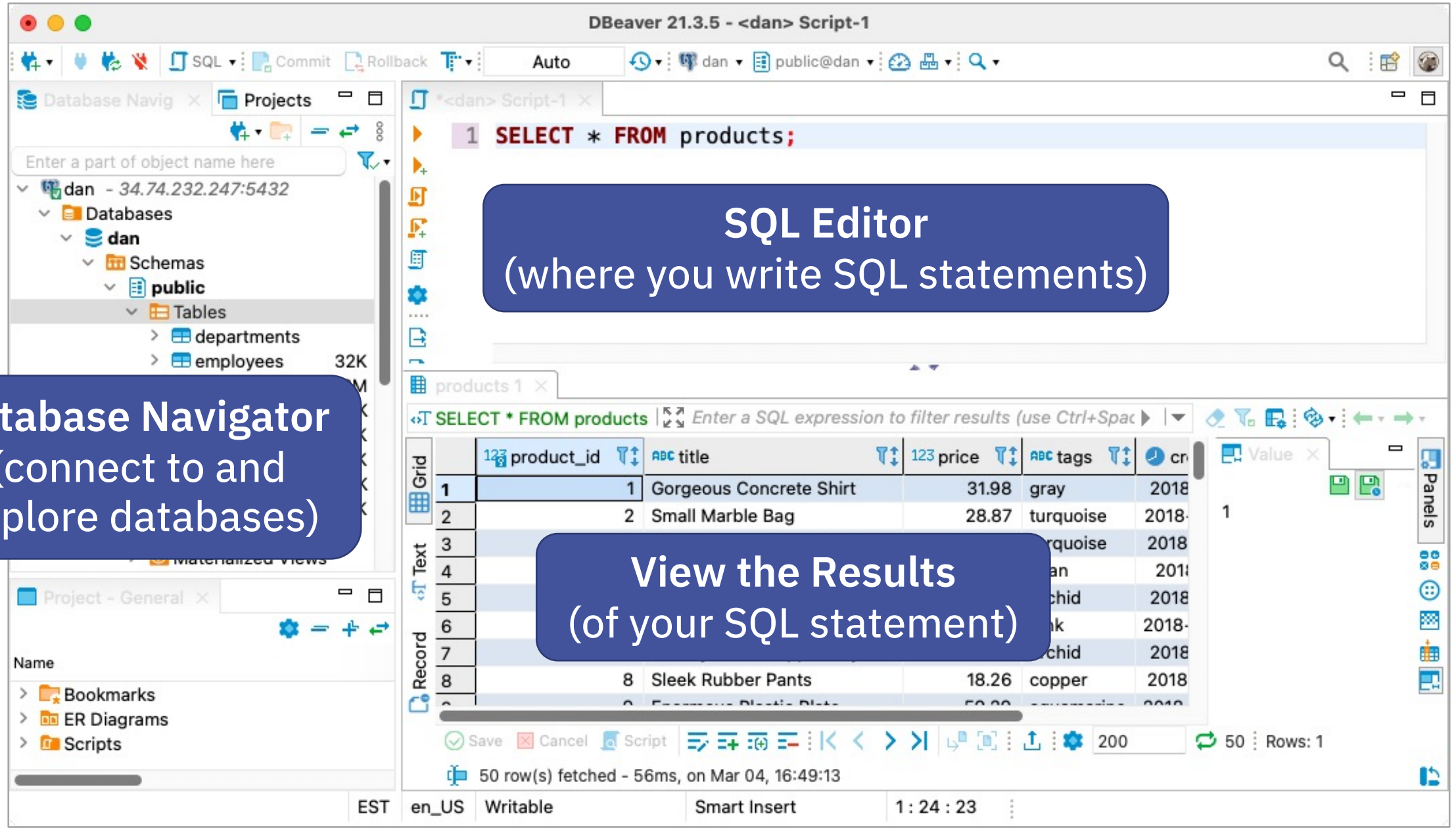
Schema

- A database schema is a set of tables.
- The schema defines how data is organized, the relations among tables.



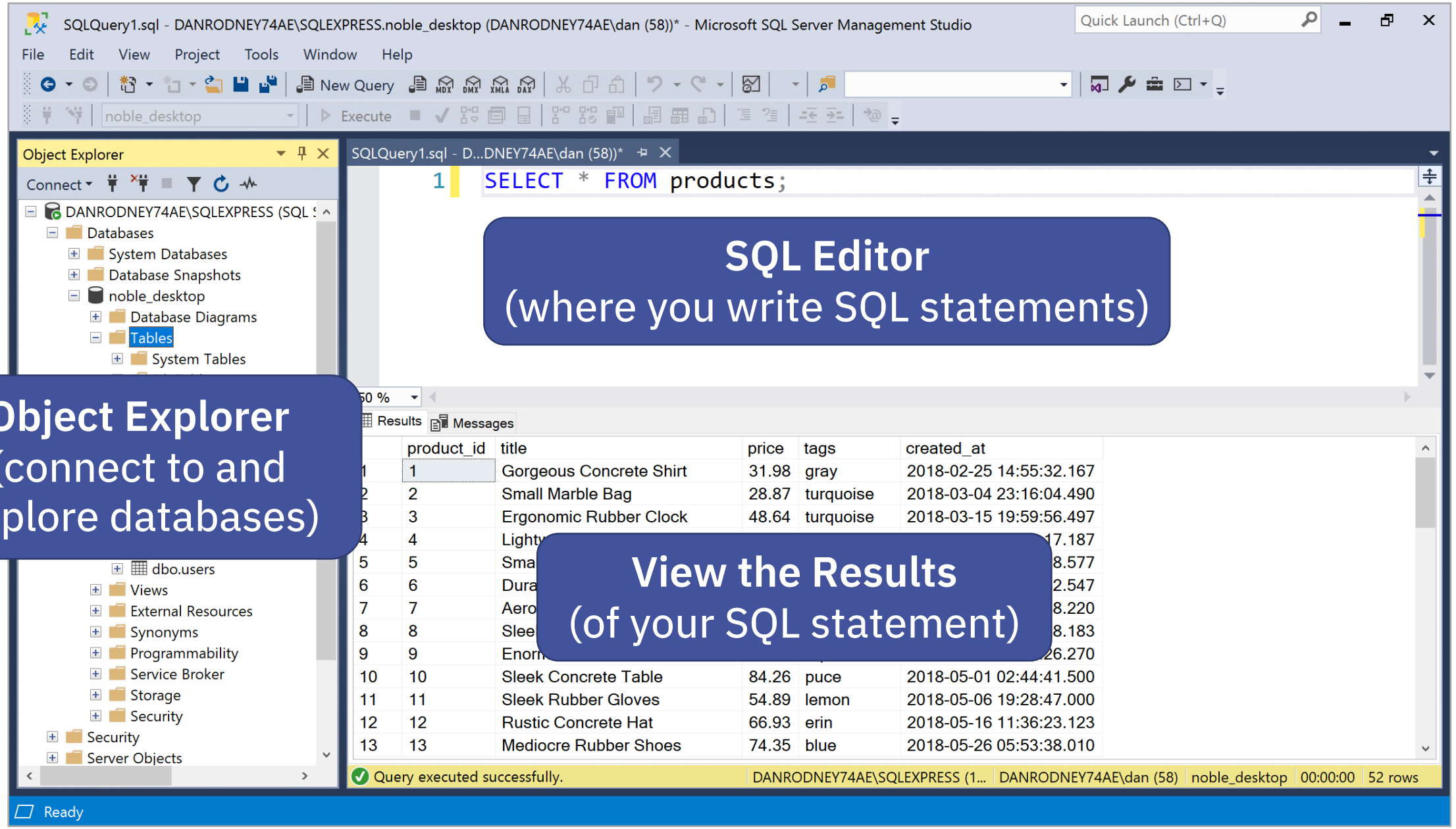
DBeaver (PostgreSQL)

A free, open-source SQL software (app) & database administration tool



Microsoft SQL Server Management Studio

Microsoft's SQL software (app) & database administration tool



Statements & Queries

Let’s Start Learning SQL Code!

SQL Statements & Queries

A **statement** is any text recognized as a valid SQL command.

- Examples: SELECT, INSERT, or DELETE

A **query** is a statement that requests information and returns a record set (which could be empty)

- Uses the SELECT statement (which you'll learn next).
- When a query is executed, data is fetched & results are displayed.

The SELECT Statement

We use SQL's SELECT statement to retrieve info from a table in the database.

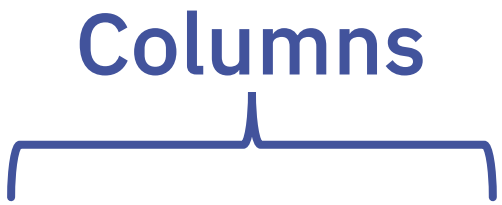
Keywords can be written in UPPERCASE or lowercase, but most documentation uses UPPERCASE (so we will too).

Getting Started & Your First Query

Exercise 1B in the Book

Specifying the Columns You Want

Which columns do you want included in the query results?




```
SELECT title, price FROM products;
```

LIMIT or TOP

In PostgreSQL:

```
SELECT * FROM products LIMIT 10;
```

In Microsoft SQL Server:

```
SELECT TOP(10) * FROM products;
```

PostgreSQL: LIMIT is always the last clause (it goes at the end).
SQL Server: TOP always goes after SELECT.
Here you can see a difference between 2 flavors of SQL.

Ordering the Results

There is no official SQL sort order. PostgreSQL usually does it based on when records were last updated, but do not rely on that.

```
SELECT * FROM products;
```

To guarantee a specific sort order, specify a column to order the query results:

```
SELECT * FROM products ORDER BY price;
```

ORDER BY

The default order is **ascending** (small–large, A–Z, old–new):

```
SELECT * FROM products ORDER BY price;
```

To use descending order (large–small, Z–A, new–old):

```
SELECT * FROM products ORDER BY price DESC;
```

Processing Order

SQL evaluates clauses in the SELECT statement in the following order: FROM, SELECT, ORDER BY


```
SELECT * FROM products ORDER BY price;
```

DISTINCT

We use DISTINCT to return only unique values (do not show duplicates):

```
SELECT DISTINCT price FROM products;
```

Coding Exercise

PostgreSQL: Exercise 1C in the Book

SQL Server: Open the file “1.0 SELECT statements.sql” (in the SQL Level 1 folder)

Data Types

Working with Characters & Numbers

Data Types

- Each column is assigned a data type.
- Determines how data is stored, and what kinds of queries you can run.

Numeric Data Types

Here are some of the ways numbers can be stored in a database:

Name	Description	Range
smallint	small-range integer	-32768 to +32767
int (integer)	typical choice for integer	-2147483648 to +2147483647
numeric	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
serial (Postgres)	autoincrementing integer	1 to 2147483647

Text Data Types

Here are some of the ways text can be stored in a database:

Name	Description
character varying(n), varchar(n)	variable-length with limit
character(n), char(n)	fixed-length, blank padded
text	variable unlimited length

(n) would be replaced with your desired number of characters.

Strings versus Numbers

- A string is sequence of characters.
- A string is wrapped in single quotes, such as **'this is a string'**
- You do not wrap numbers in quotes. A number is not merely a set of characters, it has a numerical value so math can be performed.
- If you wrap a number in quotes like **'12345'** it would now be considered a string of characters.

When to Use Numbers vs Strings

- Numbers are used when math needs to be performed: such as calculating tax, totals, quantity, etc.
- When math is not required, we do not store the value as a number. Example: a zipcode never is used in a math equation, so it should be stored as text:
  - A zipcode may be 12345-1234 (the dash is not a number).
  - Numbers do not start with 0, but zipcodes can. Storing it as a number would lose the initial 0.

Filtering the Results

Selecting Only the Records That Match Certain Criteria

Filtering Results

Instead of returning all rows from a table, we can filter them.

Examples:

- Who spent more than $500?
- How many customers were from Florida?

The WHERE Clause

- To retrieve certain rows from a table, we add the **WHERE** clause in the **SELECT** statement.
- **WHERE** appears immediately after **FROM**

```
SELECT * FROM products WHERE price = 10;
```

Comparison Operators

- Comparison operators are symbols for comparing 2 values.
- They cannot be used with text or image data types.
- The output is:
  TRUE, FALSE, or UNKNOWN
  (an operator with 1 or 2 NULL expressions returns UNKNOWN)

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

Examples of Comparison Operators

```
SELECT * FROM products WHERE price < 10;

SELECT * FROM products WHERE price <= 10;
```

The OR Operator

The **OR** operator lets you include multiple conditions.

```
SELECT * FROM products
WHERE price = 9.99
OR price = 15.99;
```

The IN Operator

The **IN** operator is shorthand for multiple **OR** conditions. It lets you to specify multiple values in a **WHERE** clause.

```
SELECT * FROM products
WHERE price IN (9.99, 15.99);
```

- Checks a column value against the list of values.
- Values in the list must be separated by a comma.

Exercise

Open the file “1.1 Filtering WHERE, OR, IN.sql” (in SQL Level 1 folder)