DB I Guided Project Solution

Guided project solution for **DB I** Module.

Starter code is here: DB I Guided Project.

Prerequisites

- SQLite Studio installed.
- This Query Tool Loaded in the browser.

Starter Code

There is no starter code for this project.

Introduce Module Challenge

Introduce the project for the afternoon. If they are done early, encourage them to study tomorrow's content and follow the tutorials on TK.

Introduce Relational Databases

Follow the content on TK to introduce `Relational Databases and SQL`.

Emphasize the difference between `Databases` and `Database Management Systems`.

Take a 5 min break

Use SELECT to Query Data

- Load this in the browser.
- show how to restore the database by clicking the button on the page.
- open the `Application` tab on chrome dev tools and show the `Web SQL` node. Every browser gets it's own copy of the data.
- introduce the `SELECT` statement.

```
select * from customers -- explain the most basic syntax for a select and what the
* means

-- next show how to cherry pick the columns we want on the results and that they
can be on any order
select City, CustomerName, ContactName from Customers

-- next show how to filter using the where clause
select City, CustomerName, ContactName
from Customers
where City = 'Berlin' -- this online tool is case sensitive when comparing
strings, that is normally not the case
```

```
-- introduce and and or operator
select City, CustomerName, ContactName
from Customers
where Country = 'France' and City = 'Paris'

-- change it to use or
where Country = 'France' and City = 'Paris'
```

wait for students to catch up, use a `yes/no` poll to let students tell you when they are done

You Do (estimated 3m to complete)

Ask student to write a query to get the list of products with category id of 2: how many records they get? Ask student to write a query to get the the name of category id of 2: what is the name?

A possible solution using sub-routes:

```
select * from products where CategoryId = 2

select name from Categories where CategoryId = 2
```

wait for students to catch up, use a `yes/no` poll to let students tell you when they are done

Use ORDER BY clause

```
select * from products order by price -- can be desc or asc, it is asc by default

-- what if we want it descending
select * from products order by price desc
```

You Do (estimated 3m to complete)

Ask student to write a query to get the list of products that cost more than 50 organized by price descending.

A possible solution:

```
select * from products
where price > 50        -- note that where must come before order by
order by price desc
```

wait for students to catch up, use a `yes/no` poll to let students tell you when they are done

Use Build-In functions

Some functions will be on most `RDBMS` like `count()`, `lenght()`, `sum()` and `max()`.

Most RDBMS will also provide specialized functions to help writing complex queries simple. Those vary from vendor to vendor. Students will learn the ones for the RDBMS they will be using on their projects.

Let's see the count() function in action.

```
select count(*) from OrderDetails -- that count(*) column on result is ugly, let's
add an alias
```

Use Column Aliases

```
select count(*) as Count from OrderDetails -- much better
-- using "as" is optional in most RDBMS, could be count(*) Count

-- another function: length
select CustomerName as Name, ContactName as Contact, length(CustomerName) as
NameLength from Customers
```

wait for students to catch up, use a yes/no poll to let students tell you when they are done

Use DISTINCT

```
select City from Customers -- returns 91 records, but we don't really have 91
unique cities

select distinct City from Customers -- returns 69 records
```

wait for students to catch up, use a yes/no poll to let students tell you when they are done

Use LIKE Operator

Load the example of LIKE on TK and introduce the % and _ wildcards.

```
select distinct City from Customers
where City like '%ar%'
order by City

select * from Employees where FirstName like 'J_n%' -- returns Janet
```

Use the NOT Operator

```
select distinct City from Customers
where City NOT like '%ar%'
```

```
order by City
```

Take a break if it's a good time

Use SQLite Studio to Create a Database

Demonstrate how to create a new database using `SQLite Studio`. Remind students that there is a short video that walks through it on TK.

wait for students to catch up, use a `yes/no` poll to let students tell you when they are done

Use SQLite Studio to add a Table to a Database

Demonstrate how to add a table to a database using `SQLite Studio`. Remind students that there is a short video that walks through it on TK.

Add a table to store students including an `id` and `name`.

Note that **SQLite will ignore max size for varchar (string) types**.

Take time to introduce data types. Note that SQLite is very flexible when it comes to types, but server database management systems offer many more data types.

wait for students to catch up, use a `yes/no` poll to let students tell you when they are done

Use SQLite Studio to Open a Database

Demonstrate how to open a database using `SQLite Studio`. Remind students that there is a short video that walks through it on TK. Open the database from the `api hubs` project.

Navigate through the tables and show the structure and data for the `messages` table.

wait for students to catch up, use a `yes/no` poll to let students tell you when they are done