



## 11: SQL Command





# What is SQL?

The language we use to get full access to the Rock database.





# What is a Database?

Relational Databases are a series of tables that link to each other.

Person

	A	B	C	D	E	
1	Id	FirstName	LastName	Email	ConnectionStatusValueId	
2	1	Ted	Decker	<a href="mailto:ted@rocksolidchurchdemo.com">ted@rocksolidchurchdemo.com</a>	26	
3	2	Cindy	Decker	<a href="mailto:cindy@gmail.com">cindy@gmail.com</a>	26	
4	3	Noah	Decker		27	
5	4	Alex	Decker		27	
6						
7						

PhoneNumber

	A	B	C	D	E	F	G	
1	Id	PersonId	Number	NumberTypeValueId	IsUnlisted	IsMessagingEnabled	CountryCode	
2	1	1	(623) 555-3322	56	1	0	1	
3	2	1	(623) 555-3321	57	1	1	1	
4	3	2	(623) 555-3341	56	1	0	1	
5	4	2	(623) 555-4532	57	1	1	1	
6								





# What is SQL?

Historically: SEQueL





# What is SQL?

Structured  
English  
Query  
Language







# What is SQL?

Structured  
Query  
Language



S-Q-L? Sequel? 🤖



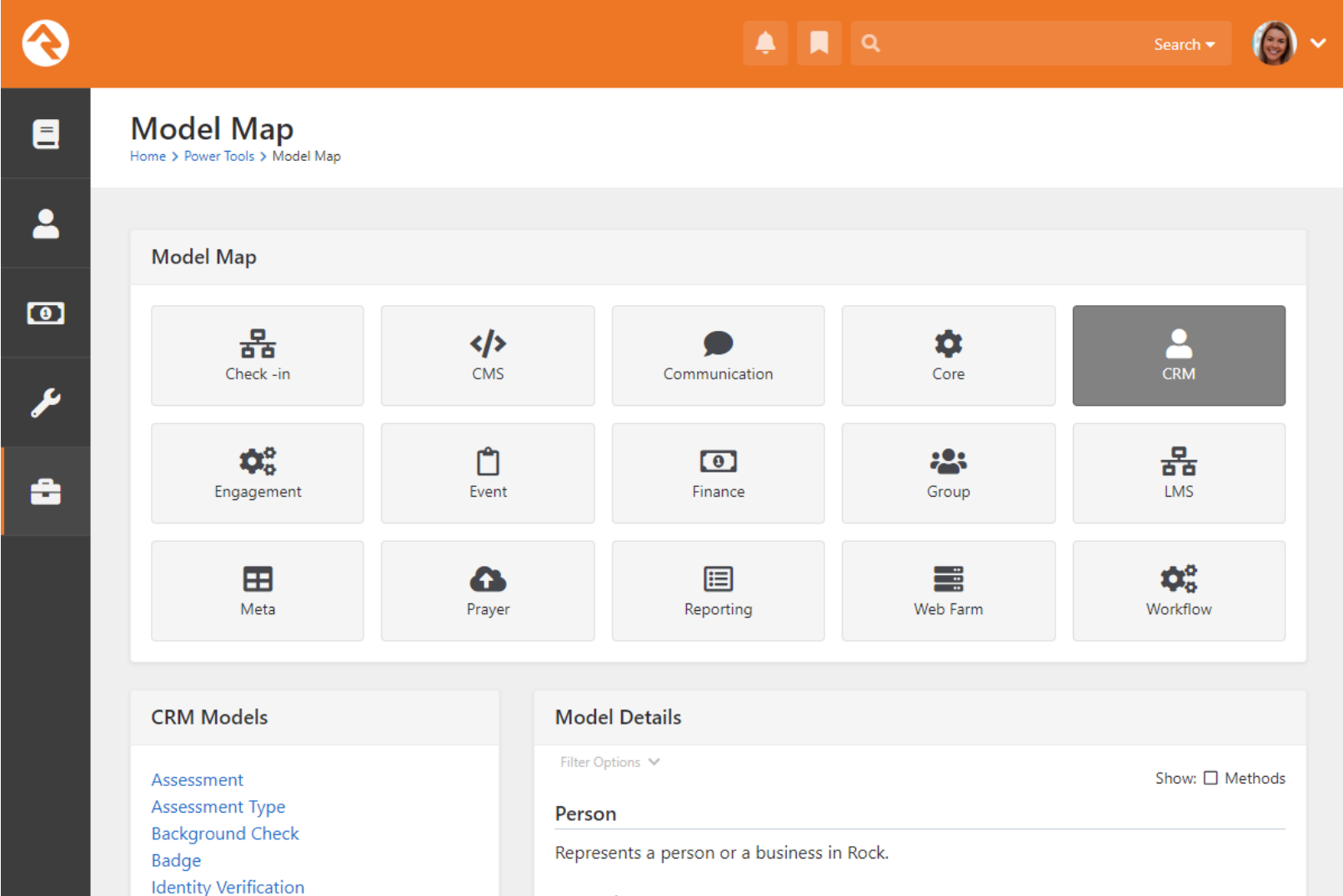
# What is SQL?

SELECT  
&  
FROM

- Keywords
- Minimum needed to write a SQL statement
- Reads like English:

**SELECT** the data I want **FROM** this table

## What Can I Use?



The screenshot displays the Model Map application interface. At the top is an orange header bar containing the Model Map logo, a notification bell, a bookmark icon, a search bar with a magnifying glass icon, and a user profile picture with a dropdown arrow. Below the header is a dark grey sidebar with five icons: a list, a person, a wallet, a wrench, and a briefcase. The main content area is titled "Model Map" with a breadcrumb trail "Home > Power Tools > Model Map". It features a grid of 15 model tiles, each with an icon and a label. The "CRM" tile is highlighted in dark grey. Below the grid are two panels: "CRM Models" on the left and "Model Details" on the right. The "CRM Models" panel lists five items: Assessment, Assessment Type, Background Check, Badge, and Identity Verification. The "Model Details" panel includes a "Filter Options" dropdown, a "Show: ☐ Methods" toggle, and a section for the "Person" model, which states "Represents a person or a business in Rock." and lists "Properties".

**Model Map**  
Home > Power Tools > Model Map

**Model Map**

Check-in	CMS	Communication	Core	<b>CRM</b>
Engagement	Event	Finance	Group	LMS
Meta	Prayer	Reporting	Web Farm	Workflow

**CRM Models**

- Assessment
- Assessment Type
- Background Check
- Badge
- Identity Verification

**Model Details**

Filter Options ▾

Show: ☐ Methods

**Person**

Represents a person or a business in Rock.

Properties









# Model Map

## What Can I Use?

Person Duplicate  
Person Previous Name  
Person Search Key  
Person Viewed  
Personal Device  
Phone Number  
User Login



### Key

- A required field.
-  A property on the database.
-  Not mapped to the database. These fields are computed and are only available in the object.
-  These fields are available where Lava is supported.
-  These methods or fields are obsolete and should not be used.



person, which is used by the enumeration detection and merge processes. This is a hard coded list of values defined in the code as an enumeration.

Show Values ▼

### AdditionalLavaFields



 Age 

Gets the Person's age.

 AgeBracket 

Gets or sets the age bracket. This is a hard coded list of values defined in the code as an enumeration.

Show Values ▼



 AgeClassification 

Gets or sets the age classification of the Person. Note: This is computed on save, so any manual changes to this will be ignored. This is a hard coded list of values defined in the code as an enumeration.


Show Values ▼

 AgePrecise 

Gets the Person's precise age (includes the fraction of the year).

 Aliases 

Gets or sets the [aliases](#) for this person.

 AllowsInteractiveBulkIndexing

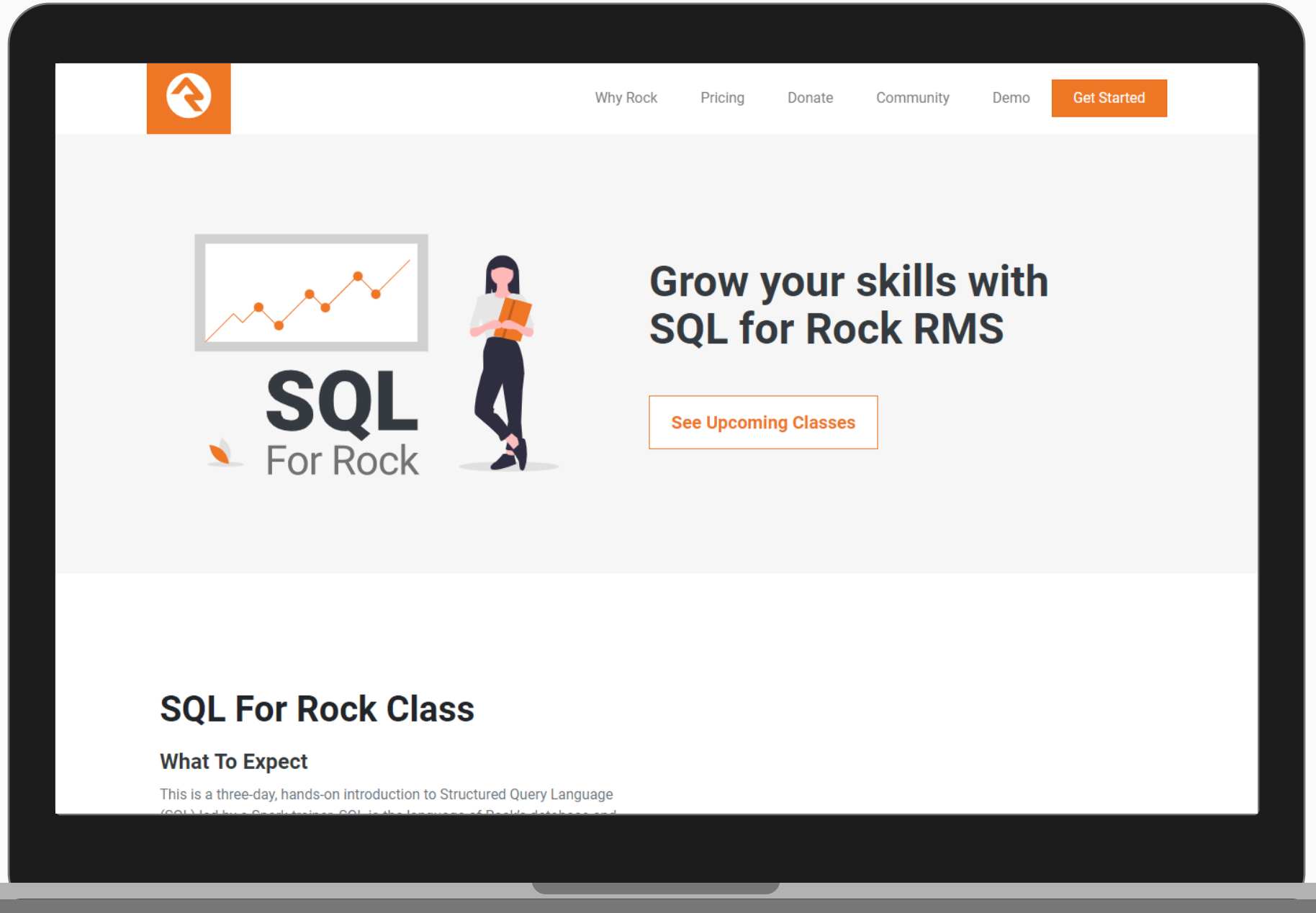
Gets a value indicating whether [allows interactive bulk indexing].

 AnniversaryDate 

Gets or sets the date of the Person's wedding anniversary. This property is nullable if the Person is not married or their anniversary date is not known.



# SQL for Rock Class





# SQL

```
1 SELECT
2   [Id]
3   , [NickName]
4   , [LastName]
5 FROM
6   [Person]
7 WHERE
8   [LastName] = 'Decker'
```

Id	NickName	LastName
5	Ted	Decker
6	Cindy	Decker
7	Noah	Decker
8	Alex	Decker





LAVA



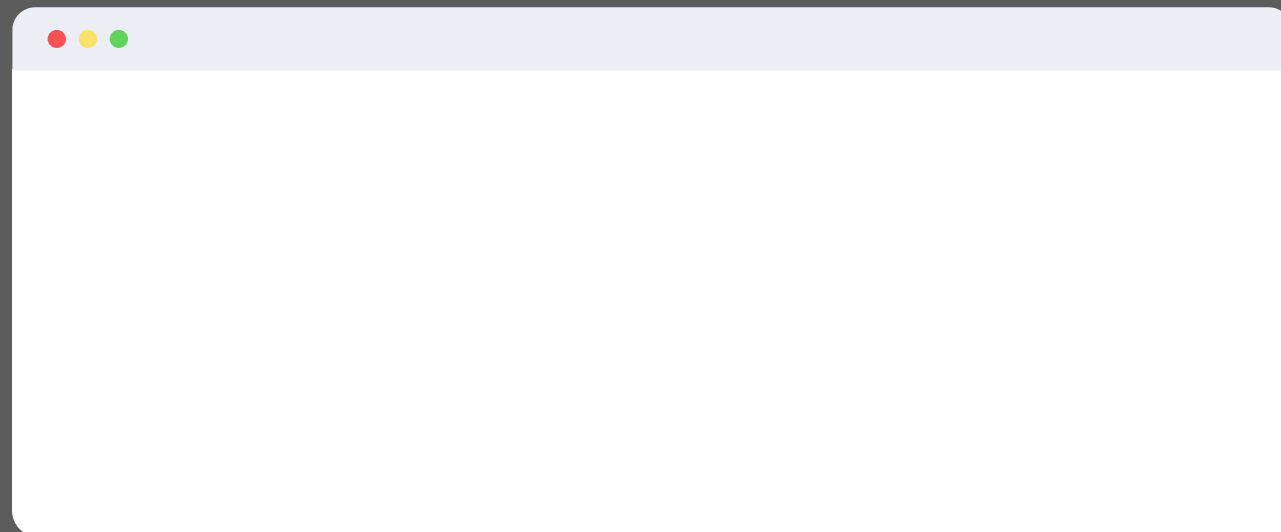
# SQL Command

```
1  {% sql %}  
2      SELECT  
3          [Id]  
4          , [NickName]  
5          , [LastName]  
6      FROM  
7          [Person]  
8      WHERE  
9          [LastName] = 'Decker'  
10 {% endsql %}
```



## Remember:

If there aren't any `{{ merge fields }}`,  
you're probably not going to  
see anything on the page.





# SQL Command

```
1  {% sql %}  
2  SELECT  
3      [Id]  
4      , [NickName]  
5      , [LastName]  
6  FROM  
7      [Person]  
8  WHERE  
9      [LastName] = 'Decker'  
10 {% endsql %}  
11  
12 <pre>{{ results | ToJSON }}</pre>
```

```
[  
  {  
    "Id": 5,  
    "NickName": "Ted",  
    "LastName": "Decker"  
  },  
  {  
    "Id": 6,
```





# SQL Command

```
1  {% sql %}
2  SELECT
3      [Id]
4      , [NickName]
5      , [LastName]
6  FROM
7      [Person]
8  WHERE
9      [LastName] = 'Decker'
10 {% endsql %}
11
12 <ul>
13     {% for row in results %}
14         <li>{{ row.LastName }}, {{ row.NickName }}</li>
15     {% endfor %}
16 </ul>
```

- Decker, Ted
- Decker, Cindy
- Decker, Noah
- Decker, Alex



# SQL Command

```
1  {% sql %}  
2  SELECT  
3      [Id] AS [PersonId]  
4      , [NickName] + ' ' + [LastName] AS [Name]  
5  FROM  
6      [Person]  
7  WHERE  
8      [LastName] = 'Decker'  
9  {% endsql %}  
10  
11 <pre>{{ results | ToJSON }}</pre>
```

```
[  
  {  
    "PersonId": 5,  
    "Name": "Ted Decker"  
  },  
  {  
    "PersonId": 6,  
    "Name": "Cindy Decker"  
  }  
]
```



# SQL Command

```
1  {% sql %}  
2  SELECT  
3      [Id] AS [PersonId]  
4      , [NickName] + ' ' + [LastName] AS [Name]  
5  FROM  
6      [Person]  
7  WHERE  
8      [LastName] = 'Decker'  
9  {% endsql %}  
10  
11 <ul>  
12     {% for row in results %}  
13         <li>{{ row.Name }}</li>  
14     {% endfor %}  
15 </ul>
```

- Ted Decker
- Cindy Decker
- Noah Decker
- Alex Decker





# SQL Command

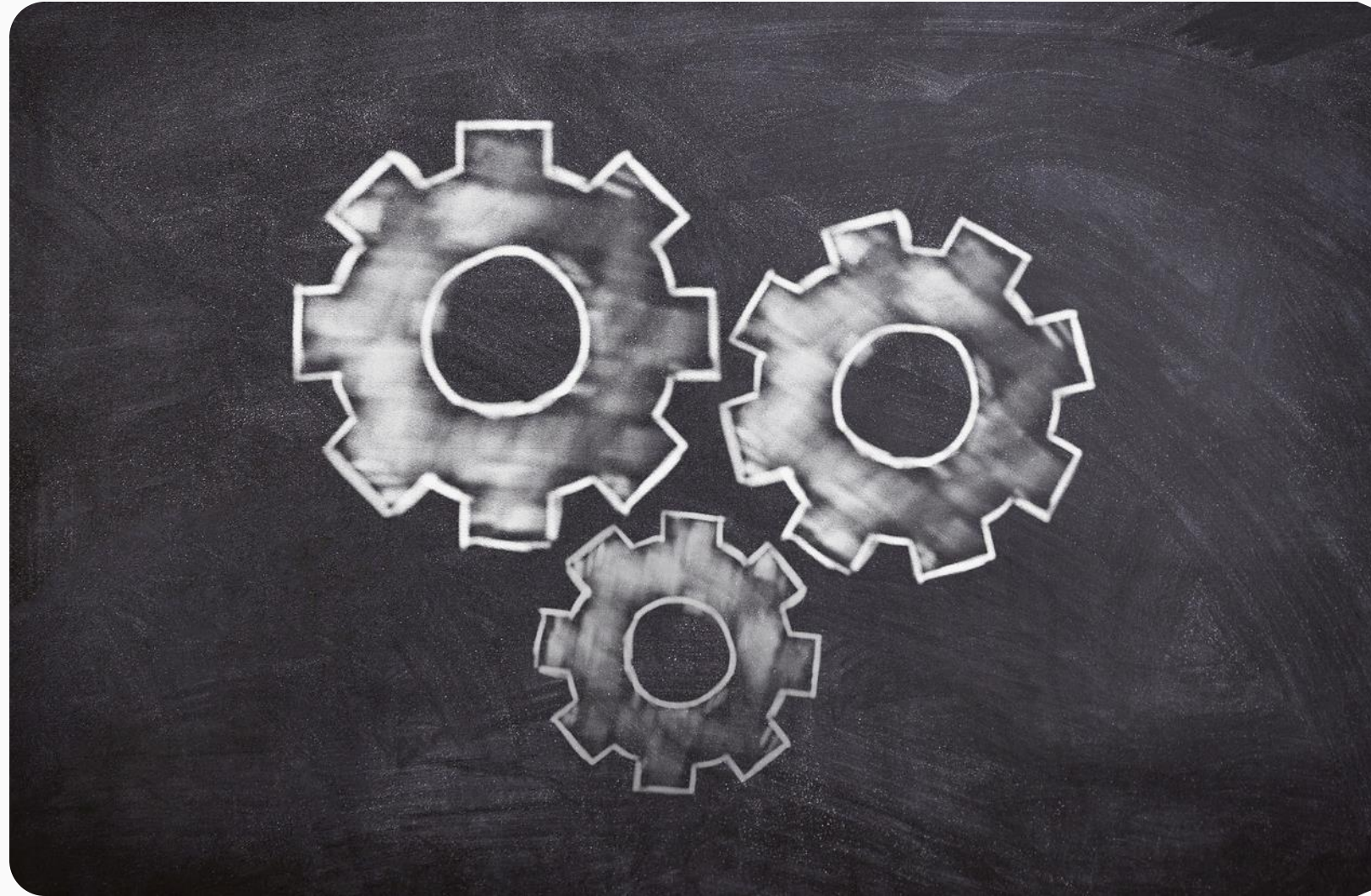
```
1  {% sql return: 'Deckers' %}  
2  SELECT  
3      [Id] AS [PersonId]  
4      , [NickName] + ' ' + [LastName] AS [Name]  
5  FROM  
6      [Person]  
7  WHERE  
8      [LastName] = 'Decker'  
9  {% endsql %}  
10  
11 <ul>  
12     {% for person in Deckers %}  
13         <li>{{ person.Name }}</li>  
14     {% endfor %}  
15 </ul>
```

- Ted Decker
- Cindy Decker
- Noah Decker
- Alex Decker



# Getting Dynamic

We frequently don't get to  
hardcode everything.





# SQL Command

```
1  {% sql return:'people' %}  
2  SELECT  
3      [Id] AS [PersonId]  
4      , [NickName] + ' ' + [LastName] AS [Name]  
5  FROM  
6      [Person]  
7  WHERE  
8      [LastName] = '{{ 'Global' | PageParameter:'Family' }}'  
9  {% endsql %}  
10  
11 <ul>  
12     {% for person in people %}  
13         <li>{{ person.Name }}</li>  
14     {% endfor %}  
15 </ul>
```



## Warning!

This can be dangerous.  
Let's look at why.



<https://church.com/people?Family=Decker>

- **Ted Decker**
- **Cindy Decker**
- **Noah Decker**
- **Alex Decker**





# SQL Command

```
1 {{ 'Global' | PageParameter:'Family' }}
```

<https://church.com/people?Family=Decker>

**Decker**



# SQL Command

WHERE

```
[LastName] = '{{ 'Global' | PageParameter:'Family' }}
```



<https://church.com/people?Family=Decker>

WHERE

```
[LastName] = 'Decker'
```



# SQL Command

```
1  {% sql return:'Deckers' %}  
2  SELECT  
3      [Id] AS [PersonId]  
4      , [NickName] + ' ' + [LastName] AS [Name]  
5  FROM  
6      [Person]  
7  WHERE  
8      [LastName] = 'Decker'  
9  {% endsql %}  
10  
11 <ul>  
12     {% for person in Deckers %}  
13         <li>{{ person.Name }}</li>  
14     {% endfor %}  
15 </ul>
```



## Semi-Trusted:

Anything you've put into block settings should be able to be trusted to not change.



## Untrusted:

Anything provided by the browser must be considered untrusted.



<https://church.com/people?Family=Decker>

- **URL**
- **Cookies**
- **Script**
- **(etc)**



# SQL Injection

```
https://church.com/people?Family=' OR [Id] IN (SELECT pa.[PersonId] FROM  
[FinancialTransaction] ft INNER JOIN [PersonAlias] pa ON pa.[Id] = ft.[AuthorizedPersonAliasId]);--
```

```
WHERE
```

```
[LastName] = '{{ 'Global' | PageParameter:'Family' }}'
```



# SQL Injection

```
https://church.com/people?Family=' OR [Id] IN (SELECT pa.[PersonId] FROM  
[FinancialTransaction] ft INNER JOIN [PersonAlias] pa ON pa.[Id] = ft.[AuthorizedPersonAliasId]);--
```

```
WHERE  
  [LastName] = '{{ 'Global' | PageParameter:'Family' }}'
```

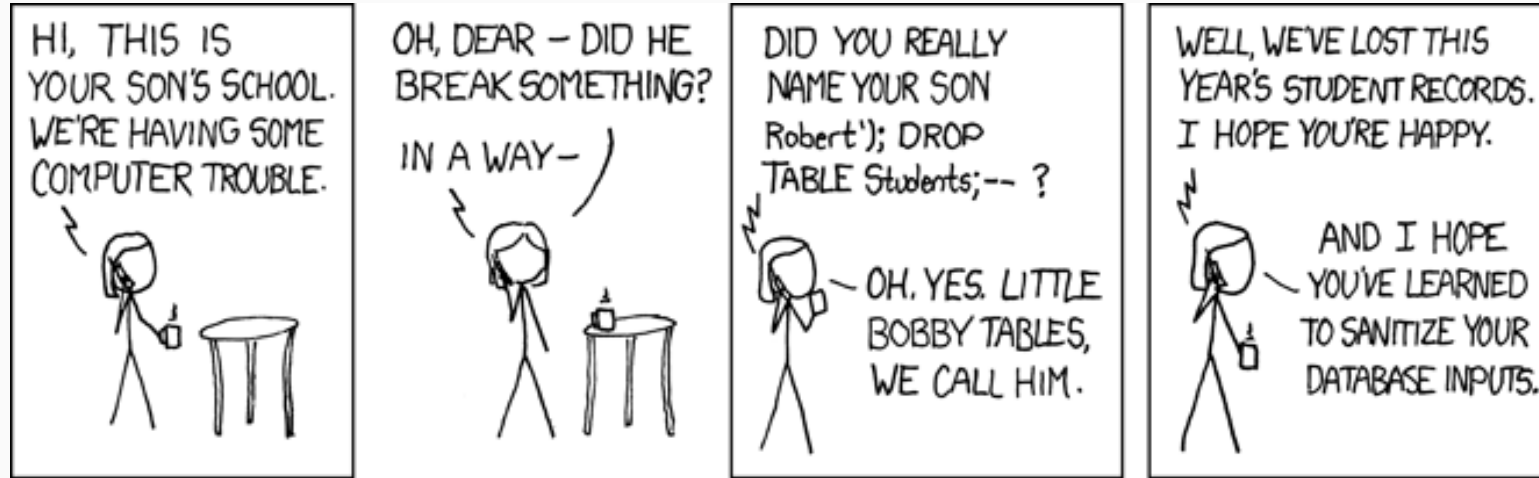
```
WHERE  
  [LastName] = '  
OR [Id] IN (  
  SELECT pa.[PersonId]  
  FROM [FinancialTransaction] ft  
  INNER JOIN [PersonAlias] pa ON pa.[Id] = ft.[AuthorizedPersonAliasId]  
);  
--'
```

- Ted Decker
- Trish Lowe
- Bill Marble
- Frank Dexter





# SQL Injection



<https://xkcd.com/327/>



## Bottom Line:

You must not trust values you didn't provide yourself.  
Verify or Sanitize them before using them.





# SanitizeSql Filter

```
1  {% sql return:'people' %}  
2  SELECT  
3    [Id] AS [PersonId]  
4    , [NickName] + ' ' + [LastName] AS [Name]  
5  FROM  
6    [Person]  
7  WHERE  
8    [LastName] = '{{ 'Global' | PageParameter:'Family' | SanitizeSql }}'  
9  {% endsql %}  
10  
11 <ul>  
12   {% for person in people %}  
13     <li>{{ person.Name }}</li>  
14   {% endfor %}  
15 </ul>
```

<https://church.com/people?Family=Decker>

- **Ted Decker**
- **Cindy Decker**
- **Noah Decker**
- **Alex Decker**



# SanitizeSql Filter

```
1  {% sql return:'people' %}  
2  SELECT  
3    [Id] AS [PersonId]  
4    , [NickName] + ' ' + [LastName] AS [Name]  
5  FROM  
6    [Person]  
7  WHERE  
8    [LastName] = '{{ 'Global' | PageParameter:'Family' | SanitizeSql }}'  
9  {% endsql %}  
10  
11 <ul>  
12   {% for person in people %}  
13     <li>{{ person.Name }}</li>  
14   {% endfor %}  
15 </ul>
```

https://church.com/people?Family=' OR [Id] IN (SELECT pa.[Id]



# SanitizeSql Filter

```
1 {{ "Robert'); DROP TABLE [Students];--" | SanitizeSql }}
```

**Robert"); DROP TABLE [Students];--**





## Even Better: Parameters





# SQL Command Parameters

```
1  {% assign nameParameter = 'Global' | PageParameter:'Family' %}  
2  
3  {% sql return:'people' family:'{{ nameParameter }}' %}  
4    SELECT  
5      [Id] AS [PersonId]  
6      , [NickName] + ' ' + [LastName] AS [Name]  
7    FROM  
8      [Person]  
9    WHERE  
10     [LastName] = @family  
11  {% endsql %}  
12  
13  <ul>  
14    {% for person in people %}  
15      <li>{{ person.Name }}</li>  
16    {% endfor %}  
17  </ul>
```

<https://church.com/people?Family=Decker>

- **Ted Decker**
- **Cindy Decker**
- **Noah Decker**
- **Alex Decker**



# SQL Command Parameters

```
1 {% assign nameParameter = 'Global' | PageParameter:'Family' %}  
2  
3 {% sql return:'people' family:'{{ nameParameter }}' %}  
4     SELECT  
5         [Id] AS [PersonId]  
6         , [NickName] + ' ' + [LastName] AS [Name]  
7     FROM  
8         [Person]  
9     WHERE  
10         [LastName] = @family  
11 {% endsql %}  
12  
13 <ul>  
14     {% for person in people %}  
15         <li>{{ person.Name }}</li>  
16     {% endfor %}  
17 </ul>
```

https://church.com/people?Family=' OR [Id] IN (SELECT pa.[Id]



## Sanitizing:

By nature, sanitized nefarious inputs in a SELECT statement should find zero matches.





# SQL Does More Than Read Data

It's the language that creates, edits, and deletes as well.





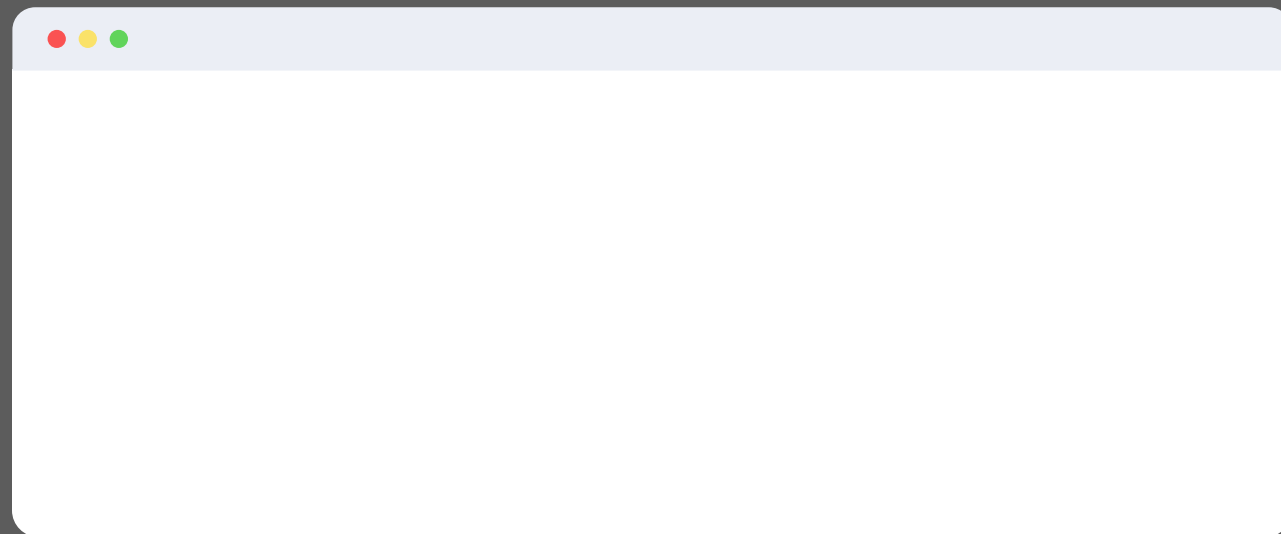
# SQL Command

```
1 {% sql statement:'command' %}  
2     UPDATE  
3     [Person]  
4     SET  
5     [NickName] = 'Teddy'  
6     WHERE  
7     [Id] = 5  
8 {% endsql %}
```



## Statement parameter:

This tells Lava that the query is not selecting data. With this, `{{ results }}` will have the count of changed rows.





# SQL Command

```
1  {% sql statement:'command' %}  
2  UPDATE  
3    [Person]  
4  SET  
5    [NickName] = 'Teddy'  
6  WHERE  
7    [Id] = 5  
8  {% endsql %}  
9  
10 {{ 'record' | ToQuantity:results }} {{ 'was' | PluralizeForQuantity:results }} modified.
```

**1 record was modified.**



# Concerned?



### HTML Content CMS / Id: 375

Basic Settings

Advanced Settings

Name

HTML Content

Enabled Lava Commands

☐ All

☐ Adaptive Message

☐ Cache

☐ Calendar Events

☐ Event Scheduled Instance

☐ Execute

☐ Interaction Content Channel Item Write

☐ Interaction Intent Write

☐ Interaction Write

☐ Observe

☐ Rock Entity

☐ Rock Entity Delete

☐ Rock Entity Modify

☐ Search

☐ Sql

☐ Web Request

☐ Workflow Activate



# Concerned?

Don't be: you're in control!

**HTML Content** CMS / Id: 375 ×

**Basic Settings** Advanced Settings

**Name** \*

HTML Content

**Enabled Lava Commands** i

<input type="checkbox"/> All	<input type="checkbox"/> Interaction Content Channel Item Write	<input type="checkbox"/> Rock Entity Modify
<input type="checkbox"/> Adaptive Message	<input type="checkbox"/> Interaction Intent Write	<input type="checkbox"/> Search
<input type="checkbox"/> Cache	<input type="checkbox"/> Interaction Write	<input type="checkbox"/> <b>Sql</b>
<input type="checkbox"/> Calendar Events	<input type="checkbox"/> Observe	<input type="checkbox"/> Web Request
<input type="checkbox"/> Event Scheduled Instance	<input type="checkbox"/> Rock Entity	<input type="checkbox"/> Workflow Activate
<input type="checkbox"/> Execute	<input type="checkbox"/> Rock Entity Delete	