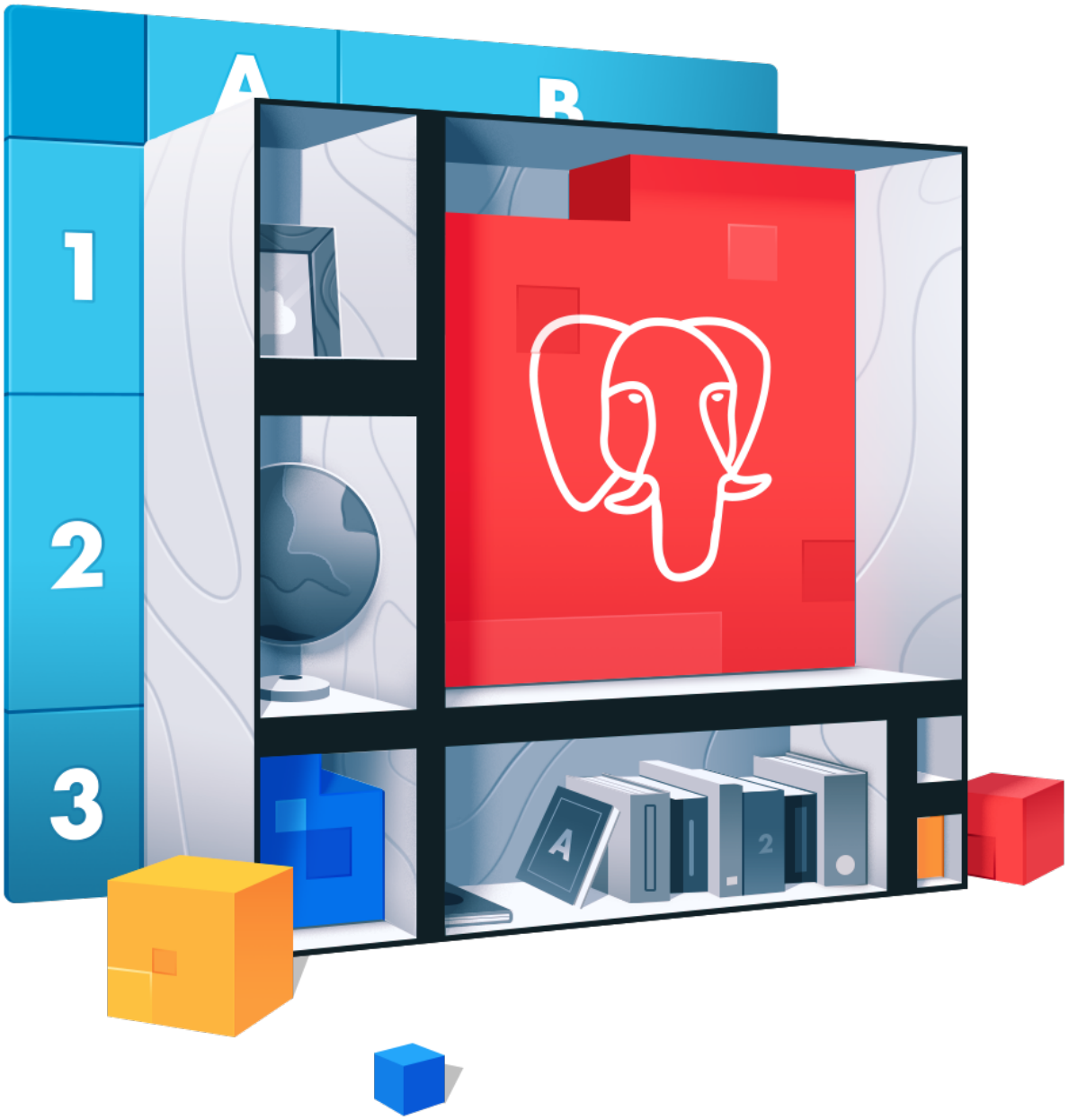
egghead.io

SQL Fundamentals

Cheat Sheet by Tyler Clark



egghead.io

SQL Fundamentals

Cheat Sheet by Tyler Clark

Full Course

egghead.io/courses/sql-fundamentals

Tyler Clark

egghead.io/instructors/tyler-clark
twitter.com/iamtylerwclark

SQL FUNDAMENTALS

Getting Started

Open the Postgres Console

```
psql database_name
```

Use Commands in the Console

Hit the `enter` key to go the next line.

Finish commands with a `;`.

Add Data to a Table

```
insert into ${Name} (${first_column_name}
${next_column_name}) values
('first_val', 'next_val');
```

EXAMPLE

```
insert into Users (create_date,
user_handle, last_name, first_name)
values ('2018-06-06',
'a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11',
'clark', 'tyler');
```

If you are inserting data for every column in your table, you can just pass values, but it's best practice to use the column names as seen above.

```
insert into Users values ('2018-06-06',
'a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11',
'clark', 'tyler');
```

The order of the data you provide after values needs to match the column order.

Create a Table

`create`

Replace `${Name}` with a Title Case name, and `column_name` with a snake_case name. `type` corresponds to a data type.

```
create table ${TableName} (
  ${column_name} type,
  ${next_column_name} type
);
```

EXAMPLE

```
create table Users (
  create_date date,
  user_handle uuid,
  last_name text,
  first_nametext
);
```

Helper Functions

`now();` The current date.

`distinct();` Matching data without duplicates.

Helper functions can be combined.

Count number of distinct last names in a Users database:

```
select count(distinct(last_name)) from Users;
```

SQL FUNDAMENTALS

Querying Data from Table

Get Column Data

```
select * from ${TableName};
```

EXAMPLE

```
select * from Users;
```

Get Specific Column Data

```
select ${column_name}, ${other_column}
from ${TableName};
```

EXAMPLE

```
select first_name, last_name from Users;
```

Use Built-In Functions to Act as Columns

Database systems include functions that can act as "virtual columns".

Retrieve the `first_name` column and add the current time:

```
select first_name, current_time, from Users;
```

Alias Column Names

`as`

Return data from a query with columns renamed.

```
select first_name as name from Users;
```

Match Anything from a List

`in`

```
select * from ${TableName} where
${column_name} in ('value1', 'value2');
```

EXAMPLE

```
select * from Users where
last_name in ('smith', 'jones');
```

Filter Query Results

Use the where `keyword` to add a condition:

```
select * from ${TableName} where
${column_name} = 'value';
```

Logical operators can be added to filter further:

`and` `or` `>` `>=` `<` `=<` `=` all work as expected.

To find null values, we use `is null` because an `= null` will return null itself (instead of true or false)

`between` is used as a short hand for:

```
condition 1 < data you want < condition 2
select * from Users where create_date
between '2018-05-01' and '2018-09-24';
```

Return Number of Rows in Table

`count`

```
select count(*) from Users;
```

SQL FUNDAMENTALS

Combining Tables (join)

Left Outer Join

Take all rows of the table referenced to the left of “left outer join” and match it up with the rows from the table to the right of “left outer join”.

```
select * from ${TableA} aliasToA left outer join
${TableB} aliasToB on ${a.column_name}
= ${b.column_name};
```

EXAMPLE

```
select * from Users u left outer join Purchases
p on u.user_handle = p.userhandle;
```

Results in data showing all users, with blank values for those without Purchase records.

Right Outer Join

Start with all of the rows in the table to the right of “right outer join” and then will add matching rows from the table to the left of “right outer join”

```
select * from ${TableA} aliasToA right outer join
${TableB} aliasToB on ${a.column_name}
= ${b.column_name};
```

EXAMPLE

```
select * from Users u right outer join Purchases
p on u.user_handle = p.userhandle;
```

Results in data showing only users who have Purchase records.

Full Outer Join

Joins two tables showing all rows, with null values for missing data.

EXAMPLE

```
select * from Users u full outer join Purchases
p on u.user_handle = p.userhandle;
```

Results in data showing only users who have Purchase records.

Inner Join

The opposite of “full outer join”, this will return only data that is matched across both tables.

EXAMPLE

```
select * from Users u inner join Purchases
p on u.user_handle = p.userhandle;
```

Results in data showing only users who have Purchase records.

! "ambiguous reference" error

When using select on a column shared between two joining tables, you’ll get an “ambiguous reference” error. This is why aliasing tables is important!

Cross Join

Takes the first table’s rows and assigns it each row of the joining table.

```
select * from Users u
cross join Purchases p;
```

SQL FUNDAMENTALS

Indexes and Subqueries

Organize a Table with `create index`

Indexes are used to retrieve data faster.

```
create index ${index_name} on
${TableName} (${column_to_index});
```

EXAMPLE

```
create index test1_user_handle
on Users (user_handle);
```

You can also index on multiple columns that you query together often. For example,

```
select * from Users where user_handle = val1 and
create_date between val2 and val3;
```

can be indexed with:

```
create index test1_user_handle_and_create_date_index
on Users (user_handle, create_date);
```

`d Users;` Displays all indexed on the Users table.

`drop index index_name;` Removes the `index_name` index.

Run a Query within another Query (subquery)

For example, to find the `create_date` and `first_name` of the User who was added first, we would write:

```
select create_date, first_name
from Users where create_date
= (select min(create_date) from Users);
```

This matches the filter from our inner query to the filter on our outer query.

SQL FUNDAMENTALS

Add Constraints to Data

Add a Primary Key

BEST PRACTICE

A primary key ensures that a given value is not null and is unique. It is a best practice to add a primary key when creating a table. When creating a table, use this syntax:

```
constraint PK_name primary key (${column_name})
```

EXAMPLE

```
create table Users (
  first_nametext, user_handle uuid,
  constraint PK_users primary key (user_handle
);
```

Connect data between tables with a Foreign Key

`reference`

If the same data exists in multiple tables, the `references` keyword will check that whatever data is being inserted into one table exists in the other as well.

Example to make sure only existing Users can make purchases:

```
create table Purchases (
  product_sku text,
  user_handle uuid references Users (user_handle),
);
```

Ensure Data is not null

`not null`

When creating a table, you can make sure data being added is `not null`.

```
create table Users (
  first_name text, user_handle uuid not null,
  constraint PK_users primary key (user_handle)
);
```

Ensure Data is unique

`unique`

Ensure data stored in a column is unique with the unique keyword:

```
create table Users (
  first_name text, user_handle uuid not null unique,
  constraint PK_users primary key (user_handle)
);
```