



Data Handling

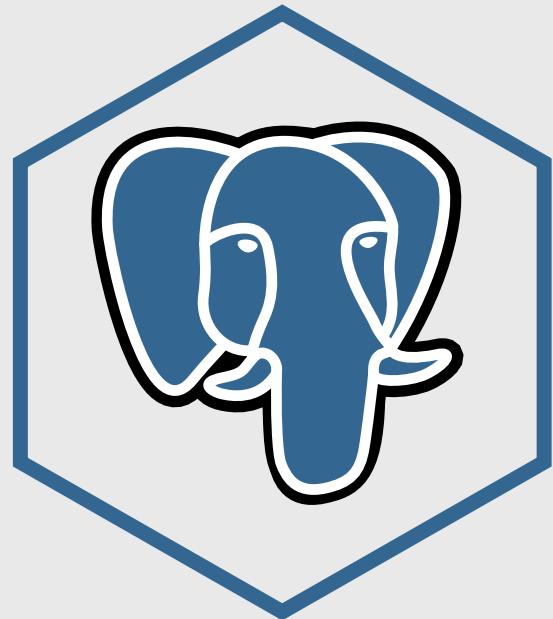
Data Handling

1. Storing data
2. Fetching data

Data Handling

1. Storing data
2. Fetching data

Data is stored in any PostgreSQL database



PostgreSQL

We use [Supabase](#) as a free, open-source option



Supabase

Store data in [Supabase](#)

Steps to connect a database via Supabase:

1. Create a Supabase account
2. Create a Supabase project
3. Copy your credentials

Full details on the [Storing Data docs page](#)

Creating a project

Create a new project
Your project will have its own dedicated instance and full Postgres database.
An API will be set up so you can easily interact with your new database.

Organization

Project name Project name

Database Password Type in a strong password
This is the password to your postgres database, so it must be strong and hard to guess. [Generate a password](#)

Region West US (North California)
Select the region closest to your users for the best performance.

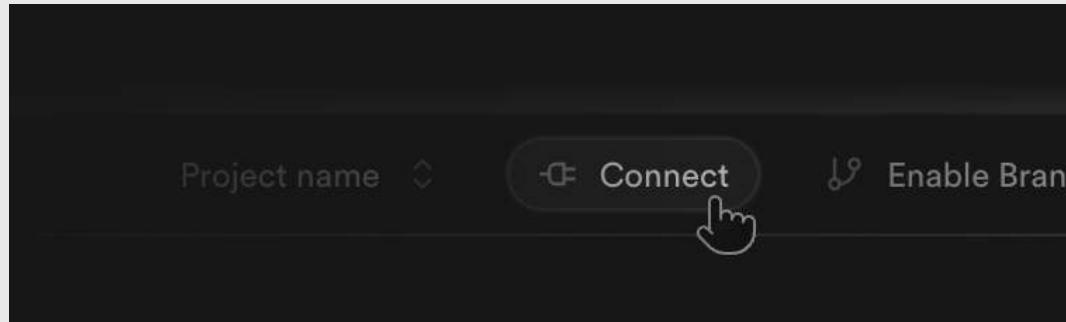
SECURITY OPTIONS >

[Cancel](#) You can rename your project later [Create new project](#)

- Choose a project name (this is your "database")
- Each database can have multiple tables
- Choose a strong password

Getting your Supabase credentials

Click the "connect" button in your project



Find the “Transaction pooler” section

A screenshot of the Supabase configuration interface. On the left, there's a sidebar with 'Project name' and a dropdown arrow. The main area has a title 'Transaction pooler' followed by a descriptive text: 'Ideal for stateless applications like serverless functions where each interaction with Postgres is brief and isolated.' Below this, there's a code block showing a PostgreSQL connection string: 'postgresql://postgres.axzkymswaxcdkbus...@'. Underneath the code block, there's a note 'Does not support PREPARE statements'. A 'View parameters' button is shown with a dropdown arrow. Below the parameters, there's a list of connection details: 'host: aws-0-us-east-1.pooler.supabase.com', 'port: 6543', 'database: postgres', 'user: postgres.axzkymswaxcdkbus...', and 'pool_mode: transaction'. At the bottom, there's a note 'For security reasons, your database password is never shown.'

Transaction pooler

Ideal for stateless applications like serverless functions where each interaction with Postgres is brief and isolated.

```
postgresql://postgres.axzkymswaxcdkbus...@
```

Does not support PREPARE statements

View parameters

host: aws-0-us-east-1.pooler.supabase.com

port: 6543

database: postgres

user: postgres.axzkymswaxcdkbus...

pool_mode: transaction

For security reasons, your database password is never shown.

Store your database credentials

In your R console, run:

```
surveydown::sd_db_config()
```

Credentials are stored in a `.env` file in your root project folder.

```
> sd_db_config()
```

— Database Configuration Setup —

Press Enter to keep current value shown in brackets

Host [aws-0-us-east-1.pooler.supabase.com]:

Port [6543]:

Database name [postgres]:

User [postgres.axzkymswaxcdkbushwxb]:

Password [****]:

Table name [mytable1]:

GSS encryption mode [disable]:

✓ Database configuration updated

— Current database configuration: —

SD_HOST=aws-0-us-east-1.pooler.supabase.com

SD_PORT=6543

SD_DBNAME=postgres

SD_USER=postgres.axzkymswaxcdkbushwxb

SD_TABLE=mytable1

SD_PASSWORD=****

SD_GSSENCMODE=disable

app.R

```
library(surveydown)

# Connects to database
db <- sd_db_connect()

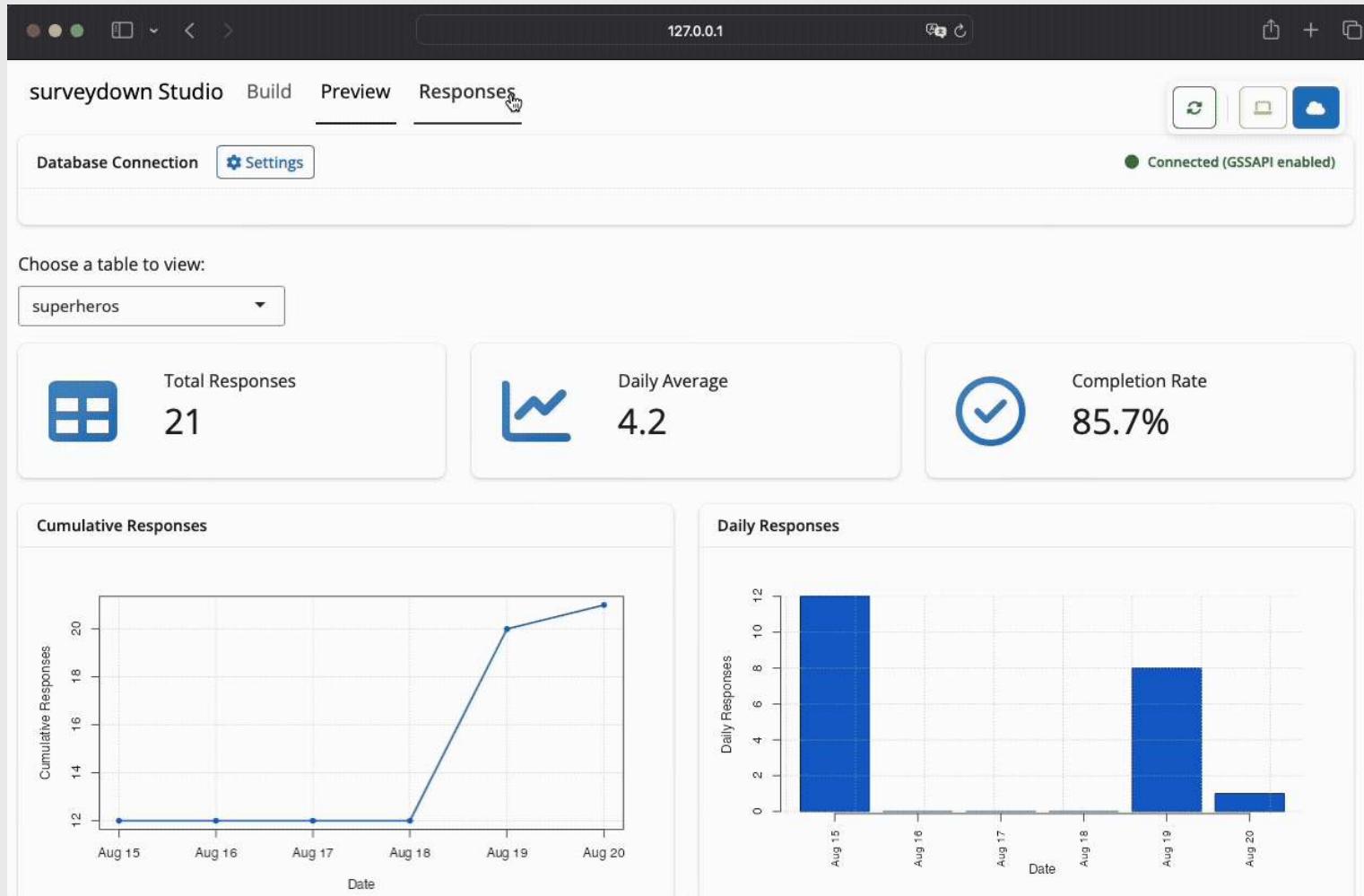
# Main UI
ui <- sd_ui()

server <- function(input, output, session) {
  # Main server
  sd_server(db)
}

shiny::shinyApp(
  ui = ui,
  server = server
)
```

The `sd_db_connect()` function uses the `.env` file to make the database connection.

Use `sdstudio::launch()` to locally view data



10:00

Your turn

- Create a Supabase account and database.
- Run `surveydown::sd_db_config()` in your console to store your Supabase credentials.
- Run your survey locally, answer questions to generate data.
- View your response data with `sdstudio::launch()`

Data Handling

1. Storing data
2. Fetching data

Static Data Fetching

Once your database is properly set up, you can fetch the data using:

```
db <- sd_db_connect()  
data <- sd_get_data(db)
```

Or simply:

```
data <- sd_get_data(sd_db_connect())
```

Reactive Data Fetching

You can also reactively fetch the data live inside the survey

In `app.R`:

```
db <- sd_db_connect()

server <- function(input, output, session) {
  data <- sd_get_data(db, refresh_interval = 5)
  sd_server()
}
```

Reactive Data Fetching

Use the reactive data to create some output

In `app.R`:

```
server <- function(input, output, session) {  
  data <- sd_get_data(db, refresh_interval = 5)  
  
  output$my_plot <- renderPlot({  
    my_data <- data()  
  
    # insert code here to make a plot  
  })  
  
  sd_server()  
}
```

In `survey.qmd`:

```
```{r}  
plotOutput("my_plot")
```
```

10:00

Your turn

- Use `sd_get_data(db)` to read in a copy of your survey response data.
- Edit your `app.R` file to reactively access your survey data.
- Use your data to make a plot about your data.
- Display your plot in your `survey.qmd` file with `plotOutput("my_plot")`