

Description



Drizzle ORM Full Course Tutorial For Beginners 🔥

157

Likes

5,382

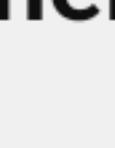
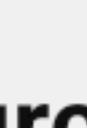
Views

Feb 10

2025

Welcome, Master Drizzle ORM in this in-depth tutorial! We'll compare Drizzle vs Prisma, explore SQL vs ORM, and perform full CRUD operations using Drizzle with MySQL. Learn how to set up Drizzle, configure migrations, and efficiently handle database operations like inserting, updating, and deleting data. If you're looking for a lightweight and powerful alternative to Prisma, this video is for you!

 Get All My YouTube Videos' Source Code for just ₹249! Grab Now - <https://thapatechnical...>

 Watch the complete Node.JS Playlist here :  • Node JS Tutorial for B...

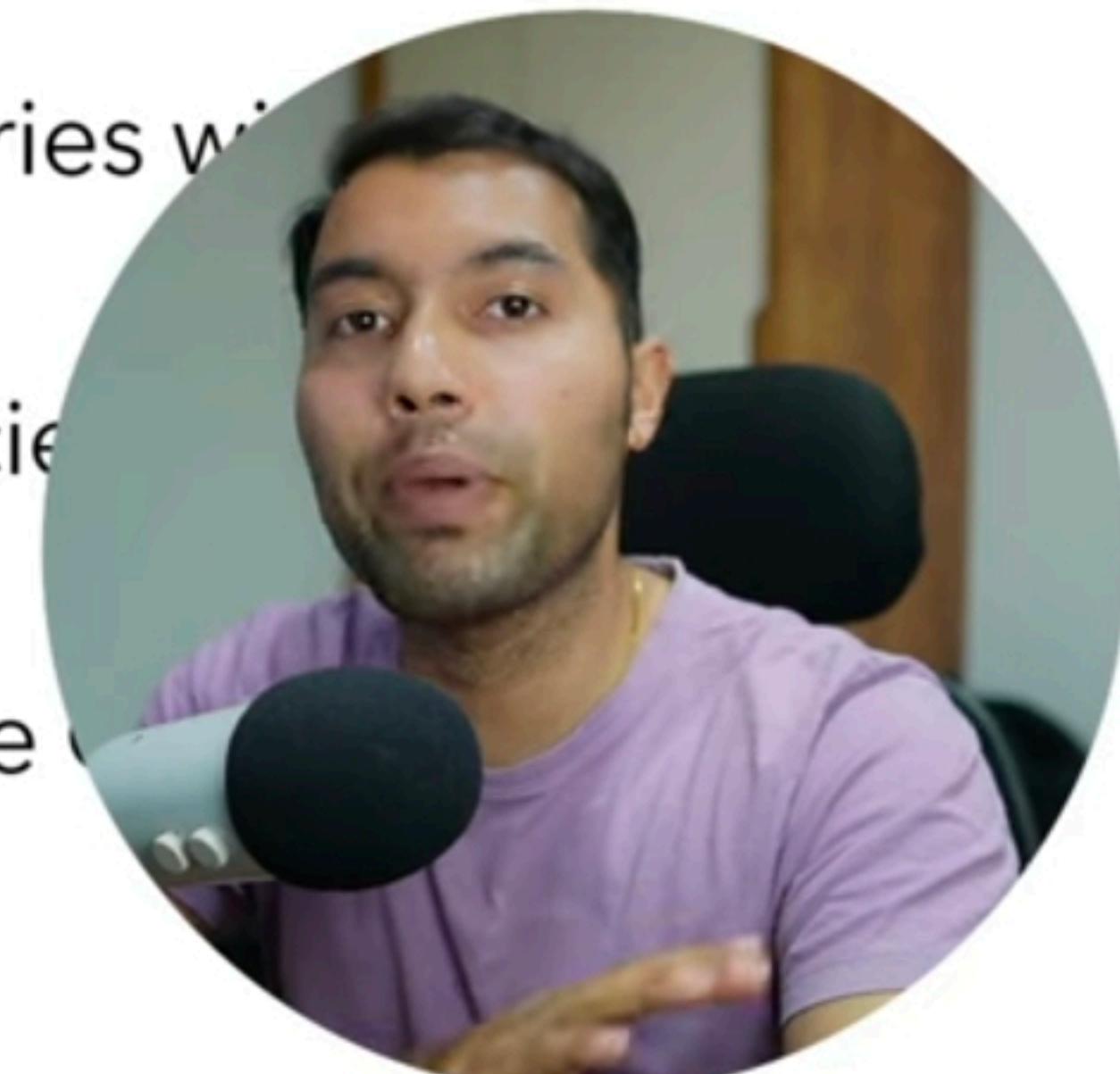
 Source Code & Docs For Free : <https://www.thapatechnical...>

 Boost Your Skills with these Pre-Requisite Videos:

 Best HTML Course - <https://www.thapatechnical...>

Introduction to Drizzle ORM

- Drizzle ORM is a lightweight and type-safe Object Relational Mapping (ORM) library for SQL databases.
- It focuses on providing a simple and developer-friendly API while maintaining high performance.
- Drizzle ORM supports both TypeScript and JavaScript, offering full type safety for queries.
- Drizzle ORM provides built-in query building and executes queries without relying on raw SQL strings.
- Designed for developers who want an ORM that's minimal, efficient, and easy to use.
- Even though Prisma is quite popular, we are going to use Drizzle in this course.



Why Drizzle ORM over Prisma?

- Drizzle ORM is lightweight and doesn't rely on a Rust engine like Prisma, making it faster and more straightforward to use.
- Unlike Prisma, Drizzle ORM always executes a single query per request unless you're using its optional query API, which we recommend you avoid.
- Drizzle ORM is closer to raw SQL, so if you're familiar with SQL, you can easily translate SQL queries into Drizzle queries.
- All configurations in Drizzle ORM are done in JavaScript or TypeScript, unlike Prisma, which uses its own domain-specific language (DSL).
- Drizzle ORM is designed for simplicity and minimal overhead, allowing you to write cleaner and more direct SQL-based code.
- Drizzle ORM's approach to queries and configurations makes it ideal for developers who prefer a simple, SQL-native experience over complex abstraction layers.



Comparison of SQL, Drizzle, and Prisma

Comparison Between SQL, Prisma, and Drizzle

```
// SQL
SELECT * FROM users;
```

```
// Prisma
const users = await prisma.user.findMany();
```

```
// Drizzle
const users = await db.select().from('users');
```

```
// Drizzle with Query API - Please don't use
const users = await db.query.users.findMany()
```





```
F:\classes\drizzle_mysql
```

```
› } npm init -y
```



```
F:\classes\drizzle_mysql is 📦 v1.0.0 via 🏛 v23.2.0
> npm i drizzle-orm mysql2
(node:4924) ExperimentalWarning: CommonJS module C:\Users\Thapa\AppData\Roaming\npm\node_modules\npm\node_modules\debug\src\node.js is loading ES Module C:\Users\Thapa\AppData\Roaming\npm\node_modules\npm\node_modules\supports-color\index.js using require().
Support for loading ES Module in require() is an experimental feature and might change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
```

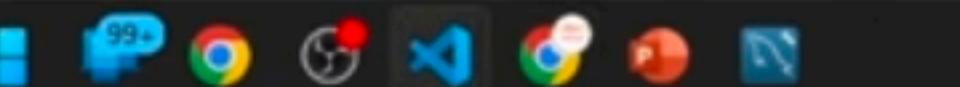


F:\classes\drizzle_mysql is ⚡ v1.0.0 via ⚡ v23.2.0

```
o > npm i -D drizzle-kit
```

5.3.2

33°C
Mostly sunny





```
  DATABASE_URL=mysql://root:Qwerty@123@localhost:3306/drizzle_db
```

Toggle auto-cloaking

+.env

1

PEN EDITORS 1 unsaved

package.json

● `git init .env`

RIZZLE MYSQL

node modules

14

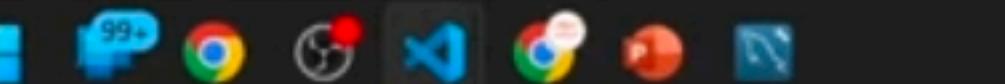
package-lock.json

package-lock.json

package.json

⊗ 0 △ 0 5.3.2

33°C
Mostly sunny



MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Query 1 short_links users users short_links short_links users short_links _prisma_migrations shortlink user_table

1. create database drizzle_db;

SCHEMAS

- authdb
 - Tables
 - Views
 - Stored Procedures
 - Functions
- dri_db
 - Tables
 - _drizzle_migrations
 - user_table
 - Views
 - Stored Procedures
 - Functions
- mydbprisma
- mysql_db
- mysql_users
- prismysql
- sys
- url_prisma_db
- url_short_auth_done
- url_shortener
- url_shortener_mysql

Administration Schemas

Information

Schema: dri_db

Output

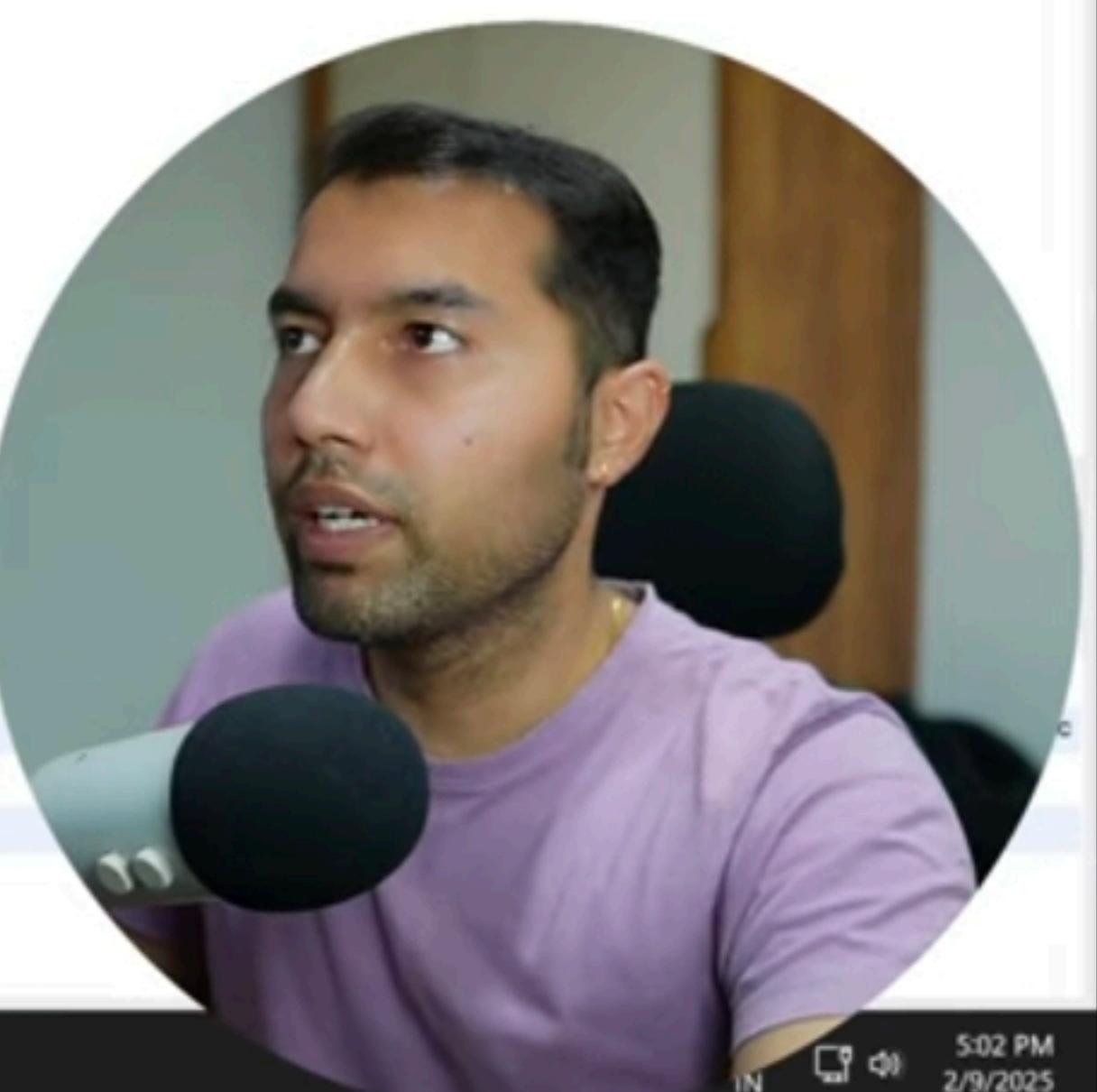
Action Output

#	Time	Action	Message
1	16:02:17	create database drizzle_db	1 row(s) affected
2	16:17:08	SELECT * FROM dri_db.user_table LIMIT 0, 1000	0 row(s) returned
3	17:01:46	SELECT * FROM drizzle_db LIMIT 0, 1000	Error Code: 1146. Table 'authdb drizzle_db' doesn't exist
4	17:01:58	create database drizzle_db	1 row(s) affected

Object Info Session

33°C 9 Mostly sunny

5:02 PM 2/9/2025



https://orm.drizzle.team/docs/get-started/mysql-new

drizzle Documentation Search Ask AI //

MEET DRIZZLE

Get started Why Drizzle? Guides Tutorials Latest releases Gotchas

FUNDAMENTALS

Schema Database connection Query data Migrations

CONNECT

PostgreSQL MySQL SQLite SingleStore Neon

TIPS

If you don't have a MySQL database yet and want to create one for testing, you can use our guide on how to set up MySQL in Docker. The MySQL in Docker guide is available [here](#). Go set it up, generate a database URL (explained in the guide), and come back for the next steps

v1.0 75%

Our goodies!

EDGE DB

upstash

Step 3 - Connect Drizzle ORM to the database

Create a `index.ts` file in the `src/db` directory and initialize the connection:

mysql2 mysql2 with config your mysql2 driver

```
import 'dotenv/config';
import { drizzle } from "drizzle-orm/mysql2";

const db = drizzle(process.env.DATABASE_URL);
```

IMPORTANT

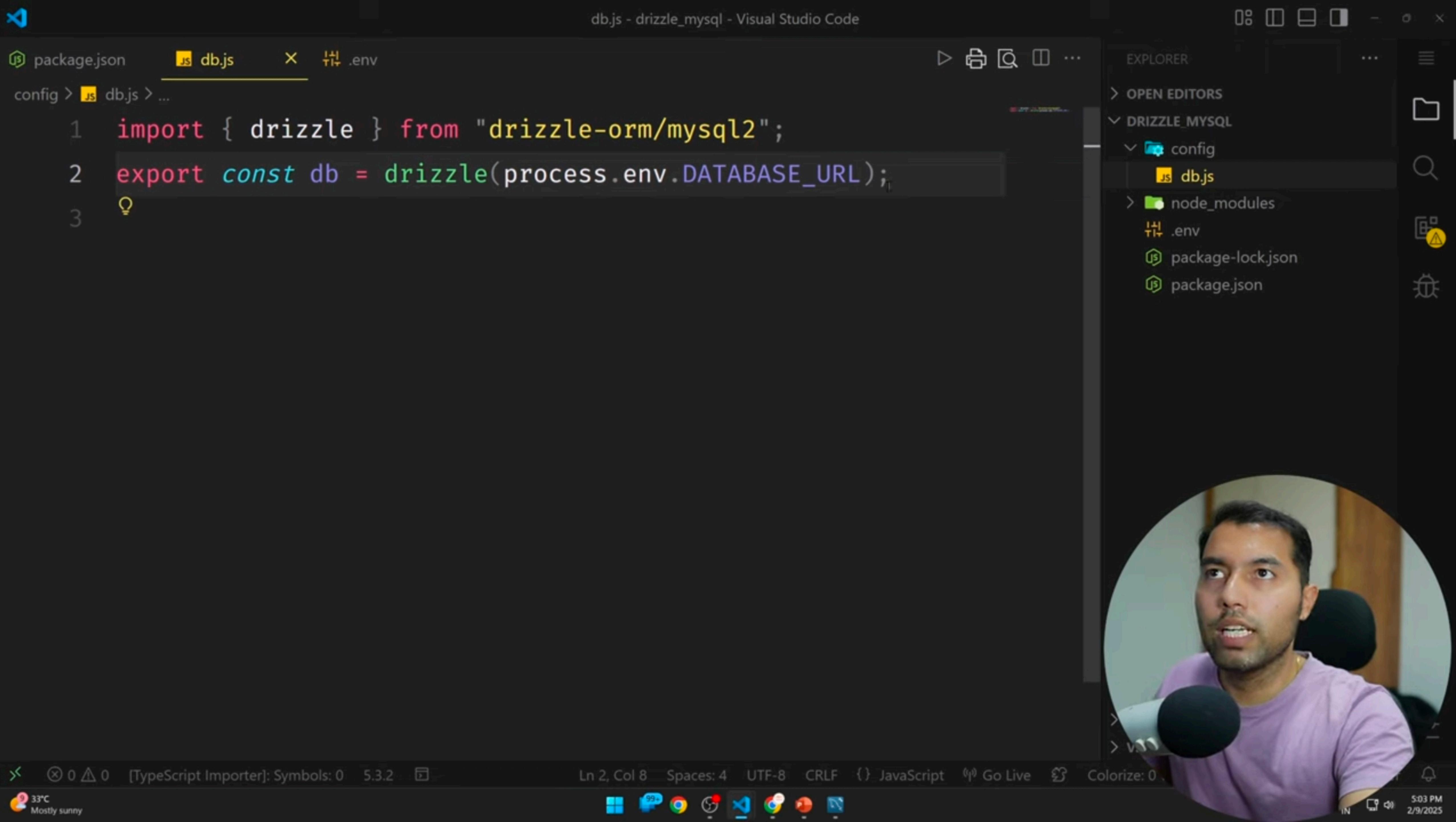
For the built in `migrate` function with DDL migrations we and drivers strongly encourage you to use single client connection.

For querying purposes feel free to use either `client` or `pool` based on your business demands.

Step 4 - Create a table

Create a `schema.ts` file in the `src/db` directory and declare your table:





package.json db.js .env

config > db.js > ...

```
1 import { drizzle } from "drizzle-orm/mysql2";
2 export const db = drizzle(process.env.DATABASE_URL);
3
```

EXPLORER

OPEN EDITORS

DRIZZLE_MYSQL

- config
- db.js

node_modules

.env

package-lock.json

package.json

Ln 2, Col 8 Spaces: 4 UTF-8 CRLF {} JavaScript ⓘ Go Live ⓘ Colorize: 0

33°C Mostly sunny

5:03 PM 2/9/2025



https://orm.drizzle.team/docs/get-started/mysql-new

drizzle Documentation Search CTRL K Ask AI 25k+ All Bookmarks

MEET DRIZZLE

Get started Why Drizzle? Guides Tutorials Latest releases Gotchas

FUNDAMENTALS

Schema Database connection Query data Migrations

CONNECT

PostgreSQL MySQL SQLite SingleStore Neon

TABLE

For querying purposes feel free to use either `client` or `pool` based on your business demands.

Step 4 - Create a table

Create a `schema.ts` file in the `src/db` directory and declare your table:

```
src/db/schema.ts
```

```
import { int, mysqlTable, serial, varchar } from 'drizzle-orm/mysql-core';

export const usersTable = mysqlTable('users_table', {
  id: serial().primaryKey(),
  name: varchar({ length: 255 }).notNull(),
  age: int().notNull(),
  email: varchar({ length: 255 }).notNull().unique(),
});
```



Our goodies! 75% DGE DB upstash

33°C Mostly sunny 5:04 PM 2/9/2025

https://orm.drizzle.team/docs/get-started/mysql-new

drizzle Documentation Search Ask AI 25k+ 75% Our goodies! EDGE DB upstash

MEET DRIZZLE

Get started Why Drizzle? Guides Tutorials Latest releases Gotchas

FUNDAMENTALS

Schema Database connection Query data Migrations

CONNECT

PostgreSQL MySQL SQLite SingleStore Neon

CREATE CONNECTION

For querying purposes feel free to use either `client` or `pool` based on your business demands.

Step 4 - Create a table

Create a `schema.ts` file in the `src/db` directory and declare your table:

```
src/db/schema.ts
import { int, mysqlTable, serial, varchar } from 'drizzle-orm/mysql-core';

export const usersTable = mysqlTable('users_table', {
  id: serial().primaryKey(),
  name: varchar({ length: 255 }).notNull(),
  age: int().notNull(),
  email: varchar({ length: 255 }).notNull().unique(),
});
```

Step 5 - Setup Drizzle config file

Drizzle config - a configuration file that is used by **Drizzle Kit** and contains all the information about your database connection, migration folder and schema files.

Create a `drizzle.config.ts` file in the root of your project and add the following content:

```
drizzle.config.ts
```



33°C Mostly sunny 5:04 PM 2/9/2025

• schema.js - drizzle_mysql - Visual Studio Code

package.json db.js schema.js .env

drizzle > schema.js > ...

```
1 import { int, mysqlTable, serial, varchar } from
  'drizzle-orm/mysql-core';
2 export const usersTable = mysqlTable('users_table', {
3   id: serial().primaryKey(),
4   name: varchar({ length: 255 }).notNull(),
5   age: int().notNull(),
6   email: varchar({ length: 255 }).notNull().unique(),
7 })|
```

EXPLORER

OPEN EDITORS 1 unsaved

DRIZZLE_MYSQL

- config
- db.js
- drizzle
- schema.js

node_modules

.env

package-lock.json

package.json



Thapa Technical

33°C 0 0 [TypeScript Importer]: Symbols: 0 5.3.2

Ln 7, Col 4 Spaces: 4 CRLF {} JavaScript ⓘ Go Live ⚡ Colorize: 0

33°C Mostly sunny

5:04 PM 2/9/2025

33°C Mostly sunny

5:06 PM 2/9/2025

Step 5 - Setup Drizzle config file

Drizzle config - a configuration file that is used by [Drizzle Kit](#) and contains all the information about your database connection, migration folder and schema files.

Create a `drizzle.config.ts` file in the root of your project and add the following content:

```
drizzle.config.ts
import 'dotenv/config';
import { defineConfig } from 'drizzle-kit';

export default defineConfig({
  out: './drizzle',
  schema: './src/db/schema.ts',
  dialect: 'mysql',
  dbCredentials: {
    url: process.env.DATABASE_URL!,
  },
});
```

Step 6 - Applying changes to the database

You can directly apply changes to your database using the `drizzle-kit push` command. This is a convenient method for quickly testing new schema designs or modifications in a local development environment, allowing for rapid iterations without the need to manage migration files:



drizzle Documentation Search Ask AI v1.0 25k+ Our goodies! EDGE DB upstash Thapa Technical

package.json

db.js

schema.js

drizzle.config.js X

.env

...

EXPLORER

...

```
1 import { defineConfig } from "drizzle-kit";
2
3 export default defineConfig({
4   out: "./drizzle",
5   schema: "./drizzle/schema.js",
6   dialect: "mysql",
7   dbCredentials: {
8     url: process.env.DATABASE_URL,
9   },
10 });
11
```



package.json 2 X db.js schema.js drizzle.config.js .env

EXPLORER

OPEN EDITORS

DRIZZLE_MYSQL

- config
- db.js
- drizzle
- schema.js
- node_modules
- .env
- drizzle.config.js
- package-lock.json
- package.json 2

version: "1.0.0",
main: "index.js",
Debug
scripts: {
 "dev": "node --env-file=.env --watch app.js ",
 "start": "node --env-file=.env app.js ",
 "db:generate": "drizzle-kit generate",
 "db:migrate": "drizzle-kit migrate",
 "db:studio": "drizzle-kit studio"
},
[],
keywords: [],
author: "",
license: "ISC",
description: "",
dependencies: .

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

[TypeScript Importer]: Symbols: 0 5.3.2

Ln 11, Col 5 Spaces: 2 UTF-8 LF {} Json ⓘ Go Live Colorize: 0

Top Stories BJP's Milkipur vi...

5:08 PM 2/9/2025



package.json - drizzle_mysql - Visual Studio Code

package.json X db.js schema.js drizzle.config.js .env

EXPLORER

OPEN EDITORS

DRIZZLE_MYSQL

- config
- db.js
- drizzle
- schema.js
- node_modules
- .env
- drizzle.config.js
- package-lock.json
- package.json

▶ Debug

```
version: "1.0.0",
main: "index.js",
"scripts": {
  "dev": "node --env-file=.env --watch app.js",
  "start": "node --env-file=.env app.js",
  "db:generate": "drizzle-kit generate",
  "db:migrate": "drizzle-kit migrate",
  "db:studio": "drizzle-kit studio"
},
"keywords": [],
"author": "",
"license": "ISC",
"description": "",
"dependencies": {
  "drizzle": "0.0.2"
}
```

17 [TypeScript Importer]: Symbols: 0 5.3.2

Ln 6, Col 1 (94 selected) Spaces: 2 UTF-8 LF {} Json Go Live Colorize: 0

Top Stories 99+ 5:08 PM

BJP's Milkpur vi... 2/9/2025

package.json X db.js schema.js drizzle.config.js .env

package.json > {} scripts > abc db:generate

3 "version": "1.0.0",
4 "main": "index.js",

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

F:\classes\drizzle_mysql is 🏭 v1.0.0 via 🏭 v23.2.0

o > npm run db:generate

EXPLORER

OPEN EDITORS

DRIZZLE_MYSQL

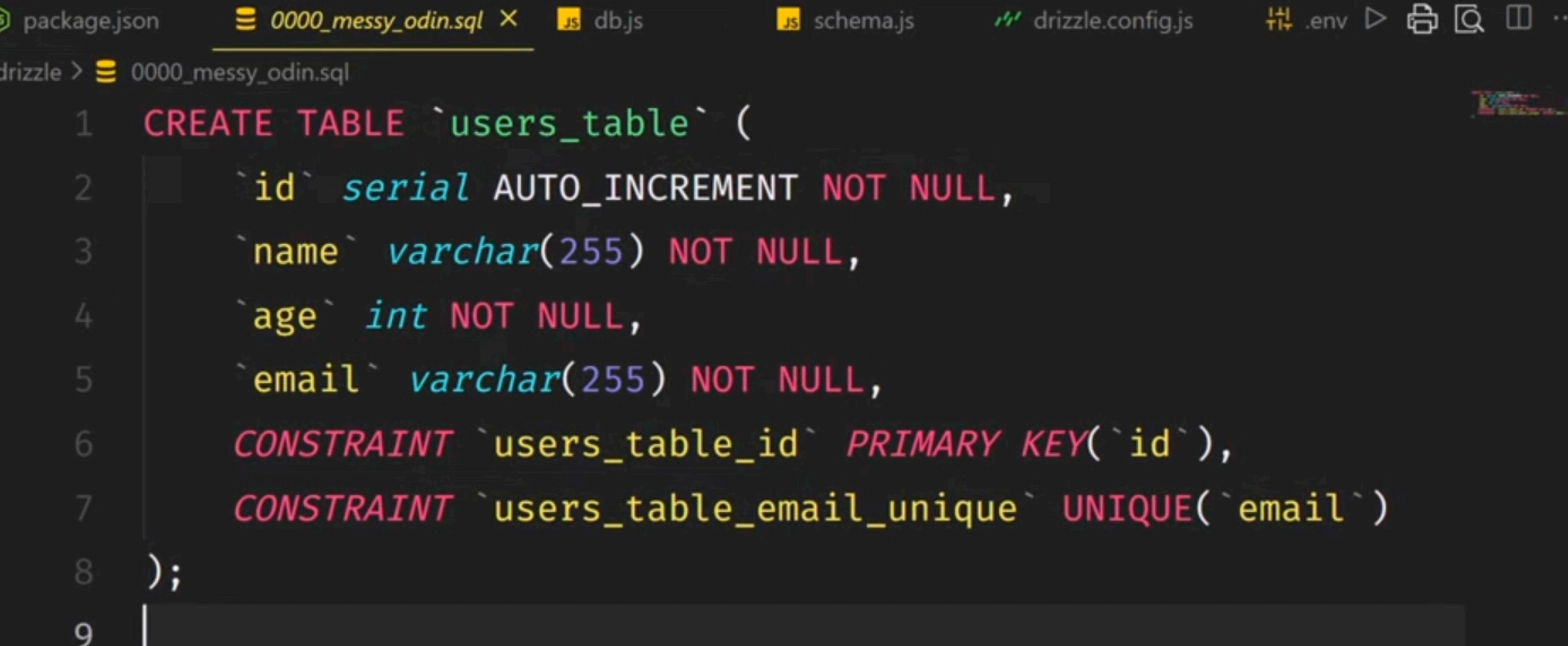
- config
- db.js

- drizzle
- schema.js

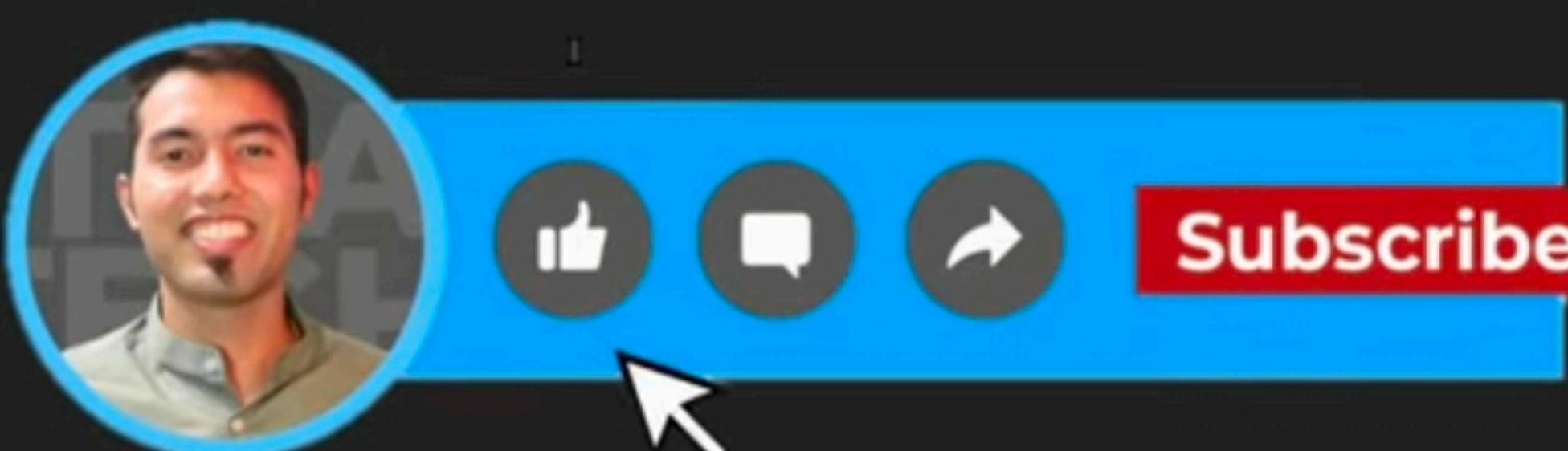
node_modules

- .env
- drizzle.config.js
- package-lock.json
- package.json





```
1 CREATE TABLE `users_table` (
2     `id` serial AUTO_INCREMENT NOT NULL,
3     `name` varchar(255) NOT NULL,
4     `age` int NOT NULL,
5     `email` varchar(255) NOT NULL,
6     CONSTRAINT `users_table_id` PRIMARY KEY(`id`),
7     CONSTRAINT `users_table_email_unique` UNIQUE(`email`)
8 );
9
```



```
 3     "version": "1.0.0",  
 4     "main": "index.js",
```

F:\classes\drizzle_mysql is 📦 v1.0.0 via ⬢ v23.2.0

o } npm run db:migrate



package.json - drizzle_mysql - Visual Studio Code

package.json X db.js schema.js drizzle.config.js .env

package.json > {} scripts > abc db:studio

3 "version": "1.0.0",

4 "main": "index.js",

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

F:\classes\drizzle_mysql is 🏭 v1.0.0 via 🏭 v23.2.0

› npm run db:studio

EXPLORER

OPEN EDITORS

DRIZZLE_MYSQL

- config
- db.js
- drizzle
- meta
 - 0000_messy_odin.sql
 - schema.js
- node_modules
- .env
- drizzle.config.js
- package-lock.json
- package.json



Thapa Technical

IND - ENG Live

0 0 [TypeScript Importer]: Symbols: 0 5.3.2

Ln 10, Col 38 Spaces: 2 UTF-8 LF {} Json Go Live Colorize: 0

5:10 PM 2/9/2025

JS app.js X

JS app.js

1 |

OPEN EDITORS

DRIZZLE_MYSQL

- config
- db.js
- drizzle
- meta
- 0000_messy_odin.sql
- schema.js
- node_modules
- .env
- app.js
- drizzle.config.js
- package-lock.json
- package.json

Air: Moderate Now

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JavaScript ⌂ Go Live ⌂ Colorize: 0

5:11 PM 2/9/2025



Thapa Technical

• app.js - drizzle_mysql - Visual Studio Code

JS app.js

JS app.js > main

```
1 import { db } from "./config/db.js";
2 import { userTable } from "./drizzle/schema.js";
3
4 const main = async () => {
5
6 }
7
8 main().catch((error) => {
9   console.log(error);
10 });
11
```

Air: Moderate Now

Ln 5, Col 5 Spaces: 4 UTF-8 CRLF {} JavaScript ⓘ Go Live ⚙ Colorize: 0

5:13 PM 2/9/2025



JS app.js X

JS app.js > main > insertUser

```
4  const main = async () => {
5      // 1 insert
6      const insertUser = await db
7          .insert(userTable)
8          .values({ name: "vinod", age: "31", email:
9              "test@thapa.com" });
10
11     main().catch((error) => {
12         console.log(error);
13     });

```





JS app.js X

JS app.js > main

4 const main = async () => {

PROBLEMS 1

OUTPUT

DEBUG CONSOLE

TERMINAL

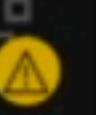
PORTS

POSTMAN CONSOLE

+ ⌂ ⌂ ⌂ X

esbuild

pwsh



F:\classes\drizzle_mysql is 🏭 v1.0.0 via ⚙ v23.2.0

o > npm run start

|

I



✖ 0 ⚠ 0 ⓘ 1 [TypeScript Importer]: Symbols: 0 5.3.2

Ln 8, Col 68 Spaces: 4 CRLF {} JavaScript ⓘ Go Live ⚡ Colorize: 0

33°C
Mostly sunny5:15 PM
2/9/2025

• app.js - drizzle_mysql - Visual Studio Code

JS app.js

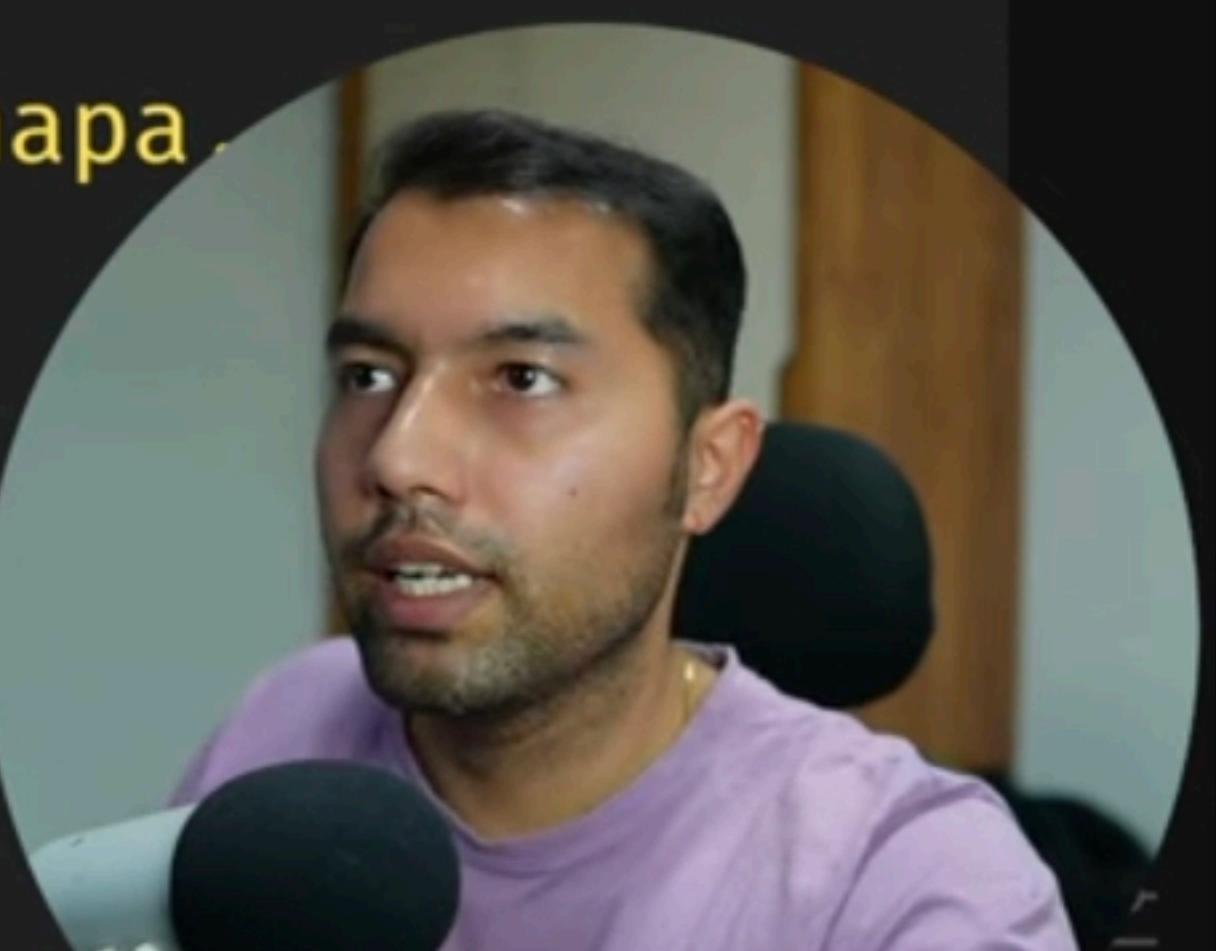
JS app.js > main > insertUser

```
1 import { db } from "./config/db.js";
2 import { usersTable } from "./drizzle/schema.js";
3
4 const main = async () => {
5   // 1 insert
6   const insertUser = await db
7     .insert(usersTable)
8     .values({ name: "vinod", age: "31", email: "test@thapa.
9       com" });
10  console.log(insertUser);
11
```

33°C 5:15 PM
Mostly sunny

Ln 6, Col 30 Spaces: 4 UTF-8 CRLF {} JavaScript ⓘ Go Live ⓘ Colorize: 0 ⓘ

File Explorer Search Issues



JS app.js X

JS app.js > main > insertUser

```
1 import { db } from "./config/db.js";
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

F:\classes\drizzle_mysql is 🏙 v1.0.0 via 🏙 v23.2.0

o > npm run start

```
> drizzle_mysql@1.0.0 start
> node --env-file=.env app.js
```

(node:3136) [MODULE_TYPELESS_PACKAGE_JSON] Warning: Module type of file:///F:/classes/drizzle_mysql/app.js is not specified and it doesn't parse as CommonJS.

Reparsing as ES module because module syntax was detected. This incurs a performance overhead.

To eliminate this warning, add "type": "module" to F:\classes\drizzle_mysql\package.json.

(Use `node --trace-warnings ...` to show where the warning was created)

✖ 0 ⚠ 0 ⓘ 1 [TypeScript Importer]: Symbols: 0 5.3.2 🔍 Ln 6, Col 30 Spaces: 4 CRLF {} JavaScript ⓘ Go Live 🔍 Colorize: 0 🔍

33°C Mostly sunny

5:15 PM 2/9/2025



https://local.drizzle.studio

All Bookmarks

SQL runner

Drizzle runner

Search...

1 row • 320ms

50 0 +

	id	name	age	email
	1	vinod	31	test@thapa.com

_drizzle_migrations 1

users_table 0

33°C Mostly sunny

Drizzle Studio

33°C Mostly sunny

5:15 PM 2/9/2025



JS app.js

```
 7  //     .insert(usersTable)
 8  //     .values({ name: "vinod", age: "31", email: "test@thapa.
com" });
 9
10 const insertUser = await db.insert(usersTable).values([
11   { name: "vinod", age: "31", email: "test@thapa.com" },
12   { name: "vinod", age: "31", email: "test@thapa.com" },
13   { name: "vinod", age: "31", email: "test@thapa.com" },
14 ]);
15
16 console.log(insertUser);
17 }
18
19 main().catch((error) => {
```

✖ 0 ⚠ 0 ⓘ 4 [TypeScript Importer]: Symbols: 0 5.3.2 🔍 Ln 11, Col 19 (5 selected) Spaces: 4 CRLF {} JavaScript ⓘ Go Live ⚡ Colorize: 0

33°C
Mostly sunny



app.js - drizzle_mysql - Visual Studio Code

```
JS app.js  X
JS app.js > ...
17
18 // 2 Read
19 const users = await db.select().from(usersTable);
20 console.log(users);
21 };
22 |
23 main().catch((error) => {
24   console.log(error);
25 });
26
```

33°C 5:18 PM
Mostly sunny

Ln 22, Col 1 Spaces: 4 UTF-8 CRLF JavaScript Go Live Colorize: 0

Thapa Technical

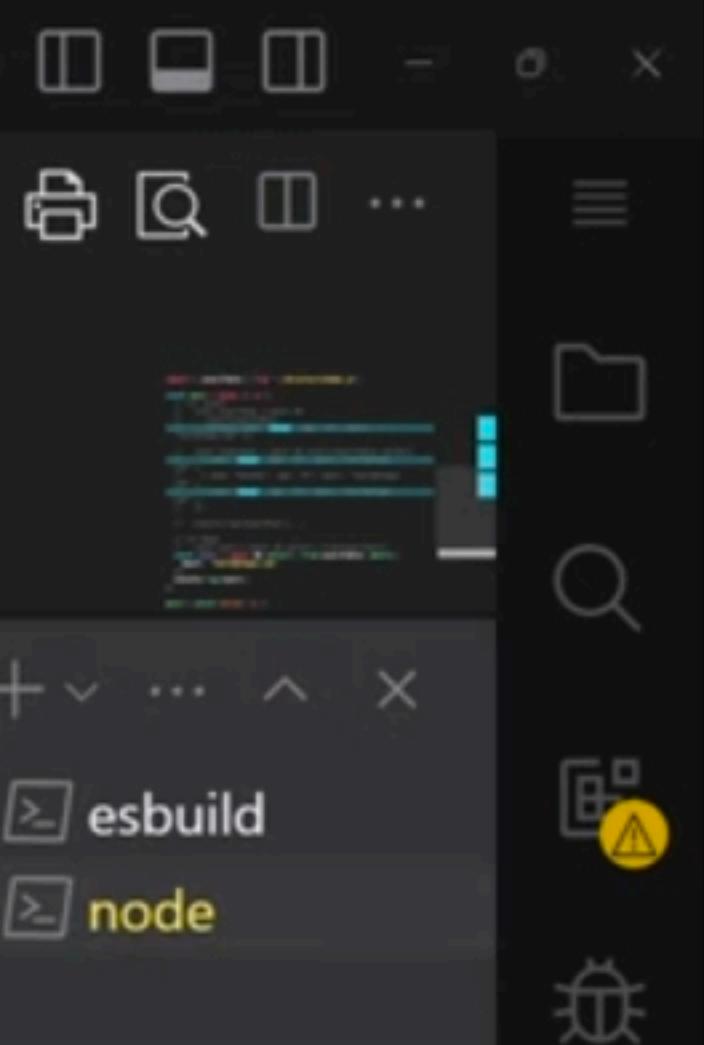


JS app.js X

JS app.js > main

```
17
18 // 2 Read
19 // const users = await db.select().from(usersTable);
20 const users = await db.select().from(usersTable).where({
21   email: "test2@thapa.com",
22 });
23 console.log(users);
24 };
25
26 main().catch((error) => {
27   console.log(error);
28 }
```





app.js

app.js > main

17

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

F:\classes\drizzle_mysql is 🏙 v1.0.0 via 🏙 v23.2.0

› npm run start

> drizzle_mysql@1.0.0 start

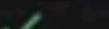
> node --env-file=.env app.js

(node:16904) [MODULE_TYPELESS_PACKAGE_JSON] Warning: Module type of file:///F:/classes/drizzle_mysql/app not specified and it doesn't parse as CommonJS.

Reparsing as ES module because module syntax was detected. This incurs a performance overhead.

To eliminate this warning, add "type": "module" to F:\classes\drizzle_mysql\package.json.

(Use `node --trace-warnings ...` to show where the warning was created)



0

△ 0

① 3

[TypeScript Importer]: Symbols: 0 5.3.2

Ln 22, Col 6

Spaces: 4

CRLF

{ } JavaScript

{} Go Live



Colorize: 0

5:19 PM
2/9/2025

```
20 // console.log(users);
21 //? 3 Update Query
22 const updatedUser = await db
23   .update(usersTable)
24   .set({ name: "bahadur thapa" })
25   .where({ email: "test2@thapa.com" });
26
27 console.log(updatedUser);
28 };
29
30 main().catch((error) => {
31   console.log(error);
```



JS app.js X

JS app.js > main ↗ 0 // console.log(users);

21 //? 3 Update Overview

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

F:\classes\drizzle_mysql is 🏙 v1.0.0 via 🏙 v23.2.0

o > npm run start

> drizzle_mysql@1.0.0 start

> node --env-file=.env app.js

(node:24064) [MODULE_TYPELESS_PACKAGE_JSON] Warning: Module type of file:///F:/classes/drizzle_mysql/app not specified and it doesn't parse as CommonJS.

Reparsing as ES module because module syntax was detected. This incurs a performance overhead.

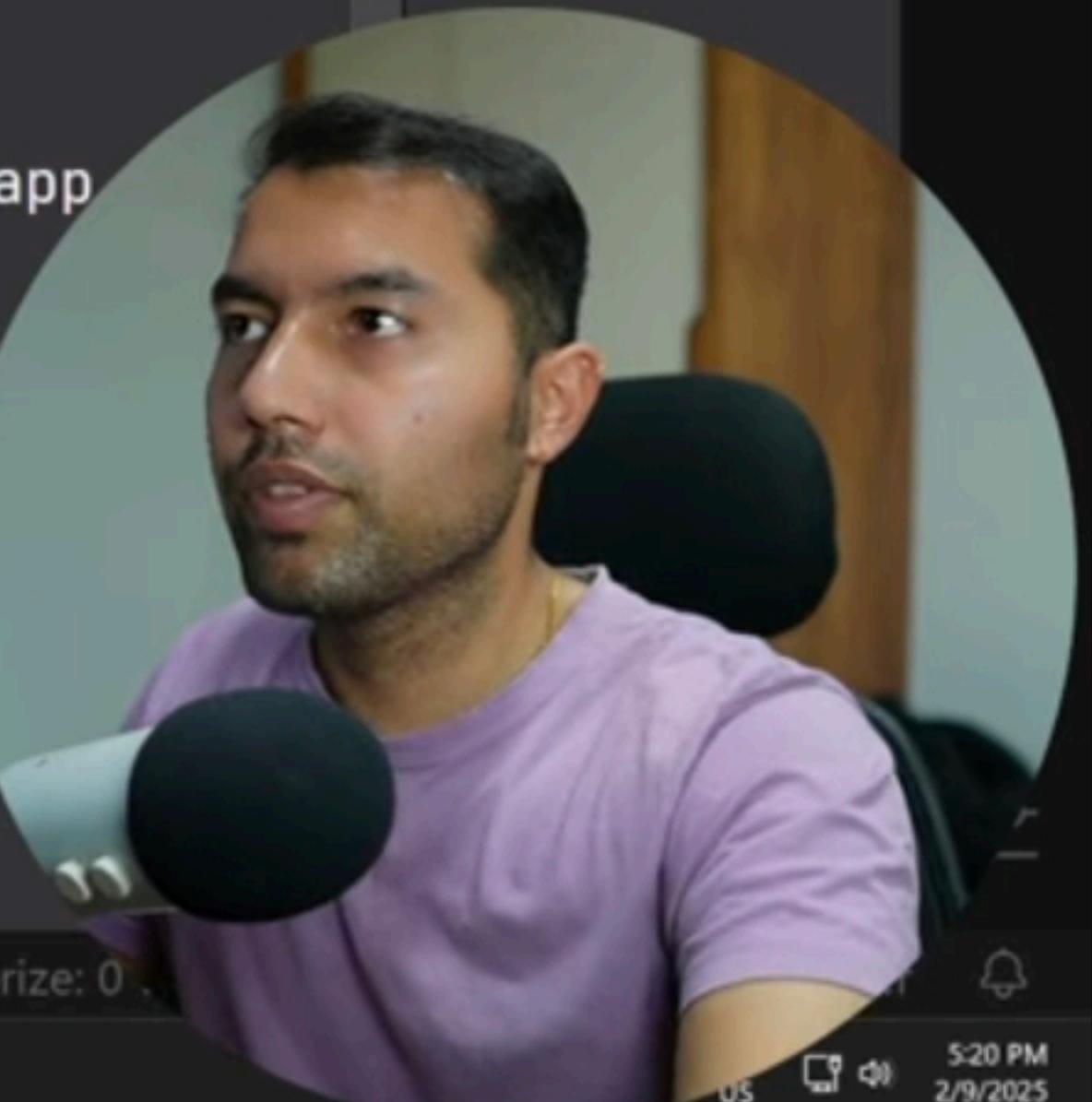
To eliminate this warning, add "type": "module" to F:\classes\drizzle_mysql\package.json.

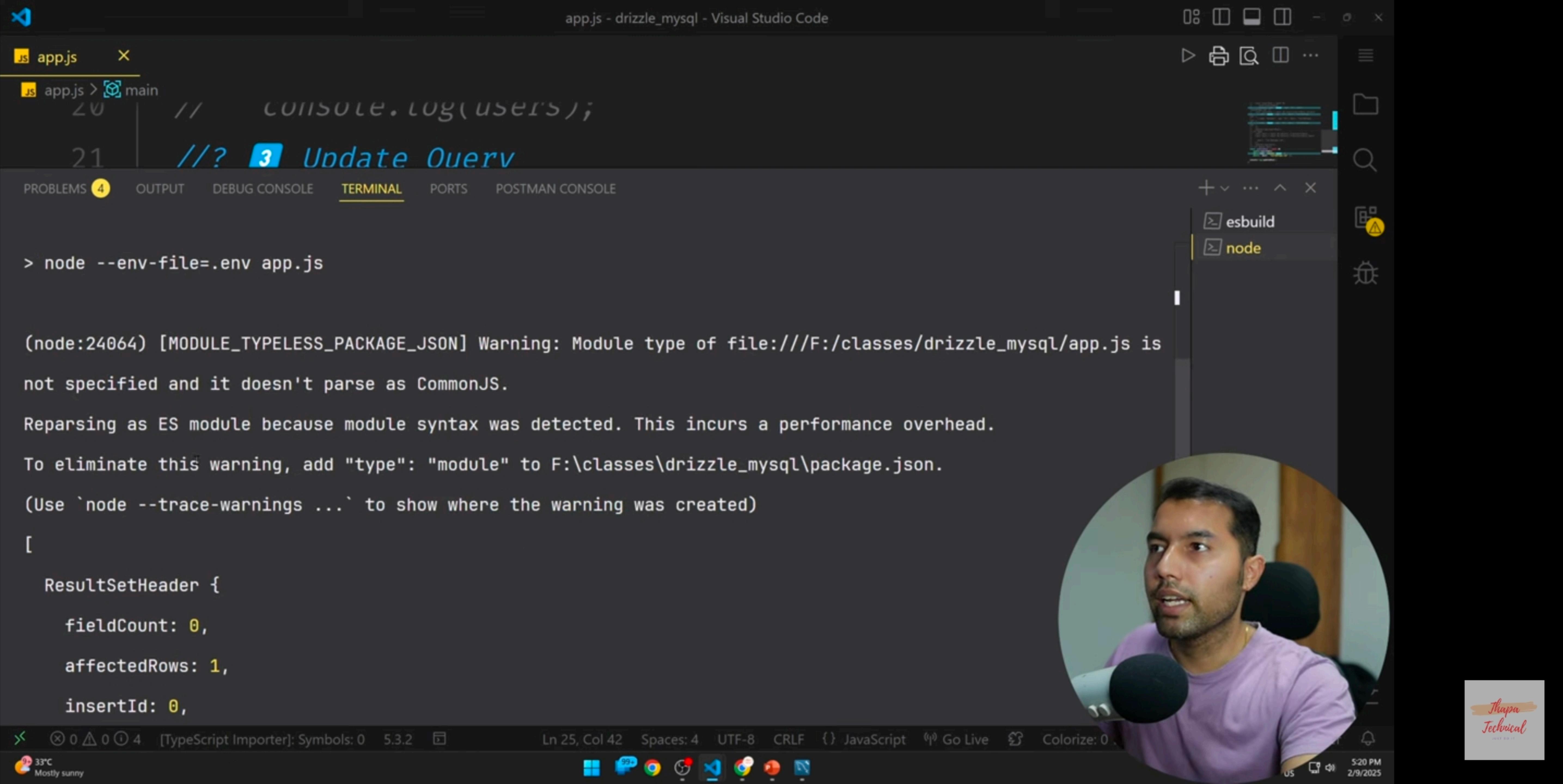
(Use `node --trace-warnings ...` to show where the warning was created)

✖ 0 ⚠ 0 ⓘ 4 [TypeScript Importer]: Symbols: 0 5.3.2 ↗ Ln 25, Col 42 Spaces: 4 CRLF {} JavaScript ⓘ Go Live ⚡ Colorize: 0

33°C Mostly sunny

5:20 PM 2/9/2025





package.json - drizzle_mysql - Visual Studio Code

EXPLORER

OPEN EDITORS

DRIZZLE_MYSQL

- config
- db.js
- drizzle
- meta
- 0000_messy_odin.sql
- schema.js

node_modules

- .env
- app.js
- drizzle.config.js
- package-lock.json
- package.json

Debug

```
1  {
2      "name": "drizzle_mysql",
3      "version": "1.0.0",
4      "main": "index.js",
5      "type": "module",
6      "scripts": {
7          "dev": "node --env-file=.env --watch app.js",
8          "start": "node --env-file=.env app.js",
9          "db:generate": "drizzle-kit generate",
10         "db:migrate": "drizzle-kit migrate",
11         "db:studio": "drizzle-kit studio"
12     },
13     "keywords": []
14 }
```

33°C
Mostly sunny

Ln 5, Col 19 Spaces: 2 UTF-8 LF {} Json ⚡ Go Live ⚡ Colorize: 0

5:21 PM 2/9/2025



```

  23 //     .update(usersTable)
  24 //     .set({ name: "bahadur thapa" })
  25 //     .where({ email: "test2@thapa.com" });
  26
  27 const updatedUser = await db
  28     .update(usersTable)
  29     .set({ name: "bahadur thapa" })
  30     .where(eq email: "test2@thapa.com" });
  31
  32 console.log(`ExpoSQLiteDatabase: ${updatedUser}`);
  33 }
  34
  35 main().catch(error => {
    console.error(`SQLite error: ${error}`);
  });

```

const eq: BinaryOperator<unknown> = (a, b) => a === b;

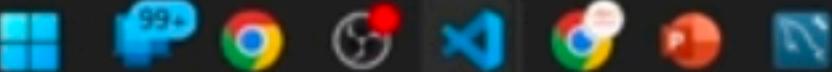
Test that two values are equal.

Remember that the SQL standard dictates that two NULL values are not equal, so if you want to test whether a value is null, you may want to use `isNull` instead.

Examples

// Select cars





```
JS app.js
JS app.js > main > updatedUser
28  const updatedUser = await db
29    .update(usersTable)
30    .set({ name: "bahadur thapa" })
31    .where(eq(email, "test2@thapa.com"));
32
33  console.log(updatedUser);
34 };
```

35

```
36 main().catch((error) => {
37   console.log(error);
38 });
39
```



JS app.js X

JS app.js > main > updatedUser

```
26 //     .where({ email: "test2@thapa.com" });
27
28 const updatedUser = await db
29     .update(usersTable)
30     .set({ name: "bahadur" })
31     .where(eq(usersTable.email, "test2@thapa.com"));
32
33 console.log(updatedUser);
34 };
35
36 main().catch((error) => {
37     console.log(error);
38 });

  1
```

LN 30, Col 31 Spaces: 4 UTF-8 CRLF {} JavaScript ⓘ Go Live ⚙ Colorize: 0

NH75F / Bund... 5.3.2 ⓘ 0 4 [TypeScript Importer]: Symbols: 0

5:23 PM 2/9/2025



https://orm.drizzle.team/docs/operators#eq

Gmail Maps one liner javascript... Channel analytics ... All Bookmarks

drizzle Documentation Search CTRL K Ask AI //

25k+ Bell Moon

ACCESS YOUR DATA

Query Select Insert Update Delete Filters

eq ne gt gte lt lte exists notExists isNull isNotNull inArray notInArray between

Filter and conditional operators

We natively support all dialect specific filter and conditional operators.

You can import all filter & conditional from `drizzle-orm`:

```
import { eq, ne, gt, gte, ... } from "drizzle-orm";
```

`eq`

✓ PostgreSQL ✓ MySQL ✓ SQLite ✓ SingleStore

Value equal to `n`

```
import { eq } from "drizzle-orm";  
  
db.select().from(table).where(eq(table.column, 5));  
  
SELECT * FROM table WHERE table.column = 5
```

```
import { eq } from "drizzle-orm";  
  
db.select().from(table).where(eq(table.column1, table.column2));
```

v1.0 75%

Our goodies!

EDGE DB

upstash



30 Seconds of Knowledge

IND - ENG Live

5:23 PM 2/9/2025

Thapa Technical

```
JS app.js  X
JS app.js > fx catch() callback
  26 //   const updatedUser = await db
  27 //     .update(usersTable)
  28 //       .set({ name: "bahadur" })
  29 //       .where(eq(usersTable.email, "test2@thapa.com"));
  30 //   console.log(updatedUser);
  31 //   // Delete Query
  32 //   await db.delete(usersTable).where({ email: "test2@thapa.com" });
  33
  34 };
  35
  36 main().catch((error) => [
  37   console.log(error);
  38 ]);
  39
```

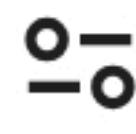
VS Code status bar: Air: Moderate Now, Symbols: 0, 5.3.2, Ln 36, Col 26, Spaces: 4, UTF-8, CRLF, {}, JavaScript, Go Live, Colorize: 0, 5:25 PM, 2/9/2025



```
26 //   .where({ email: "test2@thapa.com" }),  
27 //   const updatedUser = await db  
28 //     .update(usersTable)  
29 //       .set({ name: "bahadur" })  
30 //       .where(eq(usersTable.email, "test2@thapa.com"));  
31 //   console.log(updatedUser);  
32 //   4 Delete Query  
33 //   await db.delete(usersTable).where({ email: "test2@thapa.com" });  
34 await db.delete(usersTable).where(eq(usersTable.email, "test2@thapa.com"));  
35 };  
36  
37 main().catch((error) => {  
38   console.log(error);  
39 };
```







Your a Drizzle Orm database engineer and give me drizzle orm all interviews questions and answers for job interviews

Got it You want Drizzle ORM interview questions and answers for job interviews. Since you're already using Drizzle ORM in your project (from our earlier chats), I'll prepare a full interview prep set: from basics → intermediate → advanced → project-level use cases → tricky edge cases.

📌 Drizzle ORM Interview Questions & Answers

1. Basics of Drizzle ORM

Q1. What is Drizzle ORM?

A: Drizzle ORM is a TypeScript-first ORM for modern SQL databases. It provides a strongly typed query builder, schema migrations, and end-to-end type safety without relying on runtime reflection. It is designed for serverless, edge functions, and modern web apps where performance and correctness matter.

Q2. How does Drizzle ORM differ from Prisma or Sequelize?

A:

- Drizzle ORM: Lightweight, SQL-like syntax, no code generation, type-safe migrations, edge-compatible.
- Prisma: Schema-first, requires code generation, can feel heavier in serverless/edge environments.
- Sequelize: Older, less type-safe, more runtime overhead.

Drizzle is faster, more lightweight, and has better TypeScript type inference.

Q3. Which databases does Drizzle ORM support?

A:

- PostgreSQL
- MySQL
- SQLite
- Neon (Serverless Postgres)
- PlanetScale (Serverless MySQL)
- Turso (SQLite at the edge)
- Cloudflare D1, Vercel Postgres

Q4. How do you define a schema in Drizzle ORM?

A: Example for PostgreSQL:

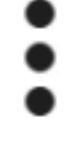
```
ts                                     ⌂ Copy ⌂ Edit

import { pgTable, serial, text, integer, timestamp } from "drizzle-orm/pg-"

export const users = pgTable("users", {
  id: serial("id").primaryKey(),
  name: text().notNull(),
  email: text().notNull().unique(),
  password: text().notNull(),
  created_at: timestamp().defaultNow(),
  updated_at: timestamp().updatedAt()
})
```

Drizzle ORM i...

chatgpt.com



Share

Q4. How do you define a schema in Drizzle ORM?

A: Example for PostgreSQL:

```
ts Copy Edit  
  
import { pgTable, serial, text, integer, timestamp } from "drizzle-orm/pg-  
  
export const users = pgTable("users", {  
  id: serial("id").primaryKey(),  
  name: text("name").notNull(),  
  age: integer("age"),  
  createdAt: timestamp("created_at").defaultNow()  
});
```

Q5. How do you connect Drizzle ORM to a database?

A:

```
ts Copy Edit  
  
import { drizzle } from "drizzle-orm/node-postgres";  
import { Pool } from "pg";  
  
const pool = new Pool({ connectionString: process.env.DATABASE_URL });  
export const db = drizzle(pool);
```

2. CRUD Operations

Q6. How do you insert data in Drizzle ORM?

A:

```
ts Copy Edit  
  
await db.insert(users).values({ name: "Alice", age: 25 });
```

Q7. How do you query data with conditions?

A:

```
ts Copy Edit  
  
const result = await db.select().from(users).where(eq(users.name, "Alice"))
```

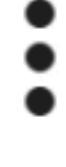
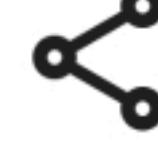
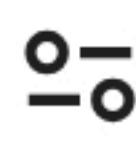
Q8. How do you update a record?

A:

```
ts Copy Edit  
  
await db.update(users).set({ age: 30 }).where(eq(users.name, "Alice"));
```

Q9. How do you delete a record?

A:



Q8. How do you update a record?

A:

```
ts Copy Edit  
await db.update(users).set({ age: 30 }).where(eq(users.name, "Alice"));
```

Q9. How do you delete a record?

A:

```
ts Copy Edit  
await db.delete(users).where(eq(users.id, 1));
```

Q10. How do you join two tables?

A:

```
ts Copy Edit  
const result = await db.select()  
  .from(users)  
  .innerJoin(posts, eq(users.id, posts.userId));
```

3. Migrations

Q11. How are migrations handled in Drizzle ORM?

A:

Drizzle uses drizzle-kit for migrations.

Steps:

1. Define schema in TypeScript.

2. Run:

```
bash Copy Edit  
npx drizzle-kit generate
```

→ Generates SQL migration files.

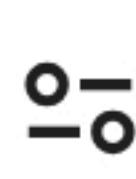
3. Apply migration:

```
bash Copy Edit  
npx drizzle-kit push
```

Q12. Can you explain the benefits of Drizzle ORM's migrations?

A:

- Type-safe migrations (linked to schema).
- Bidirectional (up/down) migrations.
- Works well in serverless/CI/CD pipelines.



Q12. Can you explain the benefits of Drizzle ORM's migrations?

A:

- Type-safe migrations (linked to schema).
- Bidirectional (up/down) migrations.
- Works well in serverless/CI/CD pipelines.
- No runtime schema parsing → very fast.

4. Advanced Usage

Q13. How do you use Drizzle ORM with Next.js?

A:

- Place DB connection in `lib/db.ts`.
- Use Drizzle with API routes or server actions.
- Example:

ts

Copy Edit

```
// lib/db.ts
import { drizzle } from "drizzle-orm/vercel-postgres";
import { sql } from "@vercel/postgres";

export const db = drizzle(sql);
```

Q14. How do you handle transactions in Drizzle ORM?

A:

ts

Copy Edit

```
await db.transaction(async (tx) => {
  await tx.insert(users).values({ name: "John" });
  await tx.insert(posts).values({ title: "Hello", userId: 1 });
});
```

Q15. How do you use relations in Drizzle ORM?

A:

Drizzle supports relations via `relations()` helper:

ts

Copy Edit

```
import { relations } from "drizzle-orm";

export const usersRelations = relations(users, ({ many }) => ({
  posts: many(posts),
}));
```

Q16. How do you do pagination in Drizzle ORM?

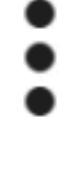
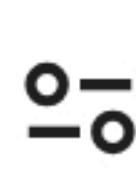
A:

ts

Copy Edit

```
const page = 2;
```

```
const limit = 10;
```



Q16. How do you do pagination in Drizzle ORM?

A:

```
ts Copy Edit  
  
const page = 2;  
const limit = 10;  
const offset = (page - 1) * limit;  
  
const results = await db.select()  
  .from(users)  
  .limit(limit)  
  .offset(offset);
```

Q17. How do you do raw SQL queries in Drizzle ORM?

A:

```
ts Copy Edit  
  
import { sql } from "drizzle-orm";  
  
const result = await db.execute(sql`SELECT * FROM users WHERE age > ${18}`);
```

Q18. How do you add indexes and constraints in Drizzle ORM?

A:

```
ts Copy Edit  
  
import { index, uniqueIndex } from "drizzle-orm/pg-core";  
  
export const users = pgTable("users", {  
  id: serial("id").primaryKey(),  
  email: text("email").notNull(),  
}, (table) => ({  
  emailIndex: uniqueIndex("email_idx").on(table.email),  
}));
```

5. Integration & Edge Cases

Q19. How do you use Drizzle ORM with Neon/PlanetScale?

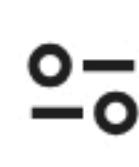
A:

- For Neon (Postgres):

```
ts Copy Edit  
  
import { drizzle } from "drizzle-orm/neon-http";  
import { neon } from "@neondatabase/serverless";  
  
const sql = neon(process.env.DATABASE_URL!);  
export const db = drizzle(sql);
```

- For PlanetScale (MySQL):

```
ts Copy Edit  
  
import { drizzle } from "drizzle-orm/planetscale-serverless";
```



Share

```
const sql = neon(process.env.DATABASE_URL!);
```

```
export const db = drizzle(sql);
```

[Copy](#) [Edit](#)

- For PlanetScale (MySQL):

ts

[Copy](#) [Edit](#)

```
import { drizzle } from "drizzle-orm/planetscale-serverless";
import { connect } from "@planetscale/database";
```

```
const connection = connect({ url: process.env.DATABASE_URL! });
export const db = drizzle(connection);
```

Q20. What are the advantages of Drizzle ORM for serverless/edge apps?

A:

- Zero runtime overhead.
- SQL-like syntax → no learning curve.
- Fully typed queries → less runtime errors.
- Compatible with serverless DBs (Neon, PlanetScale, Turso, Cloudflare D1).
- Small bundle size.

Q21. How do you debug queries in Drizzle ORM?

A:

Enable logging:

ts

[Copy](#) [Edit](#)

```
const db = drizzle(pool, { logger: true });
```

Q22. How do you handle enums in Drizzle ORM?

A:

ts

[Copy](#) [Edit](#)

```
import { pgEnum } from "drizzle-orm/pg-core";
```

```
export const userRole = pgEnum("role", ["admin", "user", "guest"]);
```

```
export const users = pgTable("users", {
  id: serial("id").primaryKey(),
  role: userRole("role").default("user")
});
```

6. Project / Real-World

Q23. Suppose you're building a SaaS app with Drizzle ORM. How would you structure your database?

A (Example Approach):

- `users` (auth + profile)
- `organizations` (multi-tenancy)
- `projects` (belongs to orgs)
- `subscriptions` (billing)

```
export const userRole = pgEnum("role", ["admin", "user", "guest"]);

export const users = pgTable("users", {
  id: serial("id").primaryKey(),
  role: userRole("role").default("user")
});
```

6. Project / Real-World

Q23. Suppose you're building a SaaS app with Drizzle ORM. How would you structure your database?

A (Example Approach):

- `users` (auth + profile)
- `organizations` (multi-tenancy)
- `projects` (belongs to orgs)
- `subscriptions` (billing)

- Use Drizzle relations for type-safe joins.
- Write migrations with `drizzle-kit` for CI/CD.
- Use serverless Postgres (Neon/Vercel) with Drizzle.

Q24. Have you faced any limitations with Drizzle ORM?

A:

- No built-in migrations UI (CLI only).
- Fewer advanced abstractions compared to Prisma.
- Some features (complex relations) are still evolving.

Q25. Why would you choose Drizzle ORM over Prisma in production?

A:

- Performance (no code generation, smaller bundle).
- Edge compatibility.
- Familiar SQL syntax.
- Safer migrations.
- TypeScript-first design.

⚡ That's a complete Drizzle ORM interview prep – from basic → advanced → real-world scenarios.

👉 Do you want me to also prepare system design + Drizzle ORM case study questions (like: "Design a multi-tenant SaaS with Drizzle ORM")? This often comes up in senior-level interviews.

< 2/2 > 🗑️ 🙌 🗺️ 🔍 ⬆️ ⬇️

Ask anything

