

Event Management System

Database Management & Frontend Development Demo
Supabase + Next js

Prepared by
Mounika Seelam
On August 2025

Overview

- What is Supabase
- Database creation - Supabase backend for Users, Events, RSVPs
- What is Next js
- Connect supabase and next js
- Next.js frontend connected to Supabase

What is Supabase

- Open-source Backend as a Service (BaaS)
- Provides PostgreSQL database, authentication, and APIs
- Easy to connect with Next.js or other frontend frameworks
- Ideal for quick app development without building backend from scratch

Supabase overview

The screenshot shows the Supabase dashboard for a new project named 'supabase-demo-db'. The interface is dark-themed. At the top, the browser address bar shows the URL 'supabase.com/dashboard/project/flnjhdjgprkjokswttd'. Below the browser, the Supabase header includes the project name 'supabase-demo-db', a 'NANO' label, and statistics for Tables (0), Functions (0), and Replicas (0). A 'Project Status' indicator shows a green dot. The left sidebar contains navigation links: Project overview, Table Editor, SQL Editor, Database, Authentication, Storage, Edge Functions, Realtime, Advisors, Reports, Logs, API Docs, Integrations, and Project Settings. The main content area has a 'Welcome to your new project' message, stating that the project is deployed on its own instance. Below this, a section titled 'Get started by building out your database' provides instructions on using the Table Editor or SQL Editor. A table of sample data is displayed, showing a 'todos' table with columns 'id', 'task', and 'status'. The table contains 7 rows of data. At the bottom, there is a section titled 'Explore our other products' with a brief description of Supabase's capabilities.

supabase-demo-db NANO

Tables 0 Functions 0 Replicas 0 Project Status

Welcome to your new project

Your project has been deployed on its own instance, with its own API all set up and ready to use.

Get started by building out your database

Start building your app by creating tables and inserting data. Our Table Editor makes Postgres as easy to use as a spreadsheet, but there's also our SQL Editor if you need something more.

Table Editor SQL Editor About Database

id	task	status
1	Create a project	Complete
2	Read documentation	Complete
3	Build application	In progress
4	Connect Supabase	In progress
5	Deploy project	Not started
6	Get users	Not started
7	Upgrade to Pro	Not started

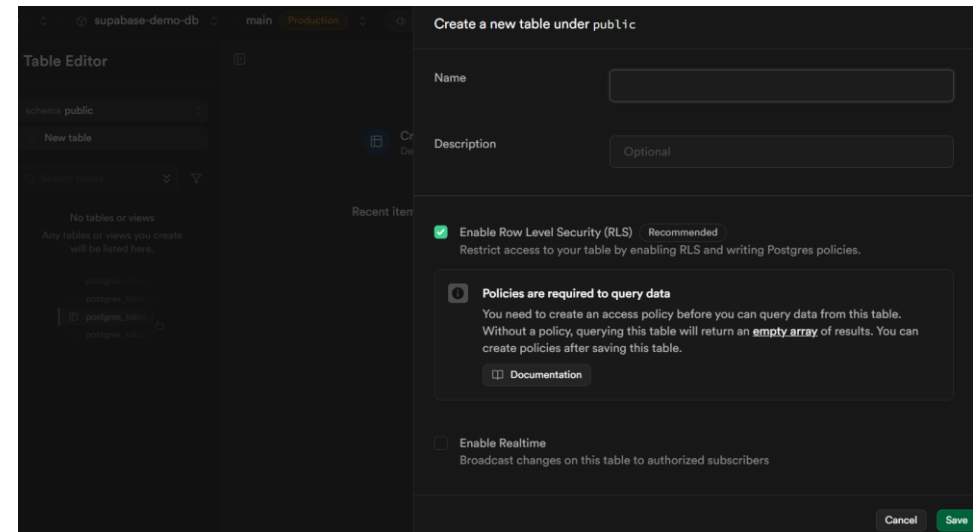
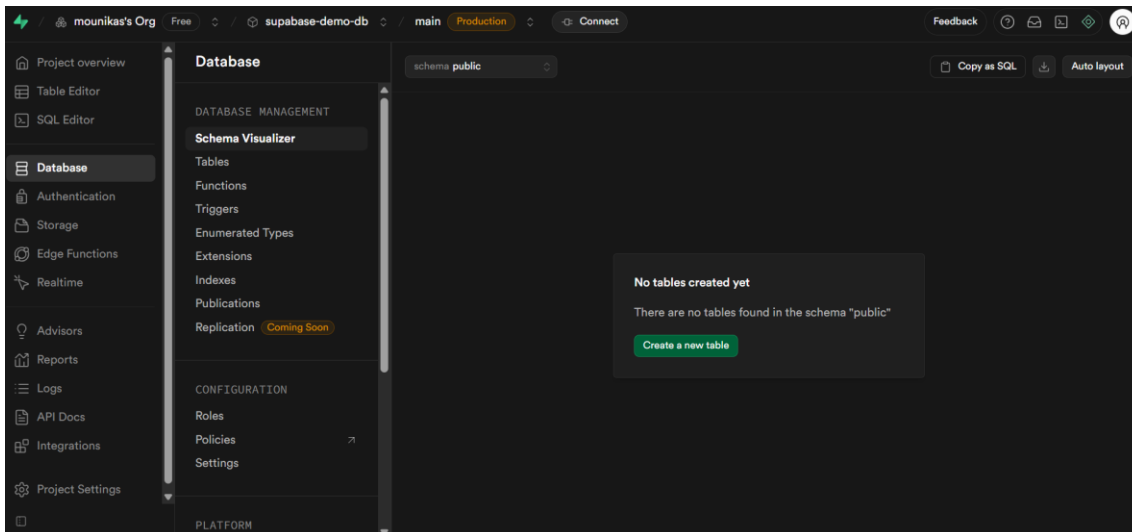
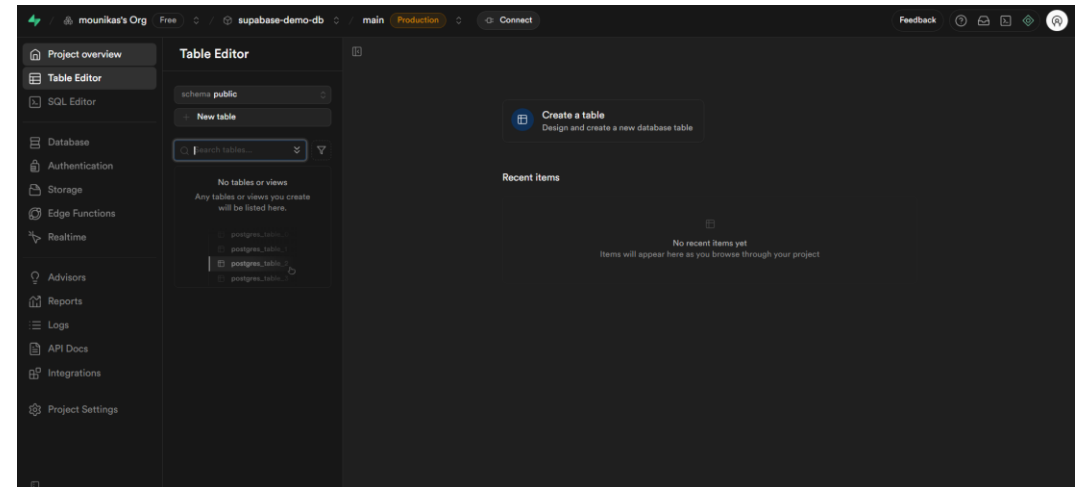
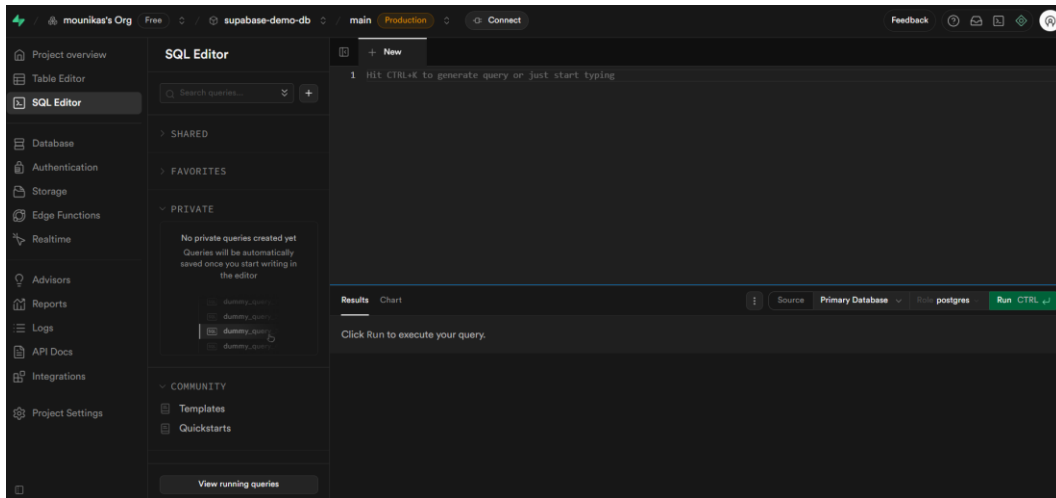
Explore our other products

Supabase provides all the backend features you need to build a product. You can use it completely, or just the features you need.

Supabase setup

- Go to supabase.com, start your project, and create account
- Create organization - Think of this like a folder where all your projects live
- Create project - database will be ready (Supabase-DB-Demo)
- Inside your project, you will see:
 - Table Editor (to see tables and data).
 - SQL Editor (to run SQL queries, like CREATE TABLE, INSERT, etc).
 - Relationships view (this shows ER diagrams – in database option).
- Here no need to download for database setup
- If you want advanced use (like connecting from Next.js app) then we need to install.
- As 'npm install @supabase/supabase-js', connect Supabase with their JavaScript client

SQL and table editor



Database Schema design

- Lets create 3 tables - with sample data (10 users, 5 events, 20 RSVPs)
 - **users** - holds people names and mails
 - Users – id (PK), name, email (unique constraint), created_at (timestamp, default now)
 - **events** - details of each - title, description, date, city, Also records who created it
 - Events – id (PK), title, description, date, city, created_by (FK → Users)
 - **rsvps** - Holds responses - did the person say Yes, No, or Maybe to attend
 - RSVPs – id (PK), user_id (FK → Users), event_id (FK → Events), status (Yes/No/Maybe)
- Users create Events, and respond (RSVP) to Events
- Constraints & Integrity:
 - PK & FK relationships
 - FK with ON DELETE CASCADE (if a user is deleted → their events and RSVPs are also deleted)
 - Unique constraints

Tables creation

The image displays two screenshots of the Supabase dashboard interface, illustrating the process of creating and viewing database tables.

Left Screenshot: SQL Editor

- Project:** mounikas's Org
- Database:** supabase-demo-db
- Environment:** main (Production)
- SQL Editor:** Contains SQL code for creating two tables:

```
1 -- Let's create tables for managing events in this database
2
3 -- tables are
4 -- Users Table
5 CREATE TABLE users (
6   id SERIAL PRIMARY KEY,
7   name VARCHAR(100) NOT NULL,
8   email VARCHAR(100) UNIQUE NOT NULL,
9   created_at TIMESTAMP DEFAULT NOW()
10 );
11
12 -- Events Table
13 CREATE TABLE events (
14   id SERIAL PRIMARY KEY,
15   title VARCHAR(200) NOT NULL,
16   description TEXT,
```
- Results:** Success. No rows returned.

Right Screenshot: Table Editor

- Project:** mounikas's Org
- Database:** supabase-demo-db
- Environment:** main (Production)
- Table:** public.users
- Columns:** id (int4), name (varchar), email (varchar), created_at (timestamp)
- Status:** This table is empty.
- Actions:** Import data from CSV, or drag and drop a CSV file here.

SQL Sample

The SQL Editor interface shows two SQL queries being executed. The first query inserts data into the 'users' table, and the second query inserts data into the 'events' table. The results section shows 'Success. No rows returned'.

```
41 -- Lets insert data - (10 users, 5 events, 20 RSVPs) ---
42 -- Insert Users data
43 INSERT INTO users (name, email) VALUES
44 ('Aarav Sharma', 'aarav@example.com'),
45 ('Ishita Reddy', 'ishita@example.com'),
46 ('Rohan Mehta', 'rohan@example.com'),
47 ('Sneha Kapoor', 'sneha@example.com'),
48 ('Vikram Das', 'vikram@example.com'),
49 ('Meera Nair', 'meera@example.com'),
50 ('Aditya Rao', 'aditya@example.com'),
51 ('Priya Iyer', 'priya@example.com'),
52 ('Karan Singh', 'karan@example.com'),
53 ('Neha Gupta', 'neha@example.com');
54
55 -- Insert events data
56 INSERT INTO events (title, description, date, city, created_by) VALUES
57 ('Tech Meetup', 'A meetup for tech enthusiasts.', '2025-09-05', 'Hyderabad', 1),
58 ('Startup Pitch', 'Startup founders pitch their ideas.', '2025-09-10', 'Bengaluru', 2),
59 ('AI Workshop', 'Hands-on AI learning.', '2025-09-15', 'Chennai', 3);
```

The Table Editor interface shows the 'public.rsmps' table. The table has columns: id, user_id, event_id, status, and created_at. The table is currently empty.

id	user_id	event_id	status	created_at
----	---------	----------	--------	------------

The Table Editor interface shows the 'public.users' table. The table has columns: id, name, email, and created_at. The table contains 10 rows of data.

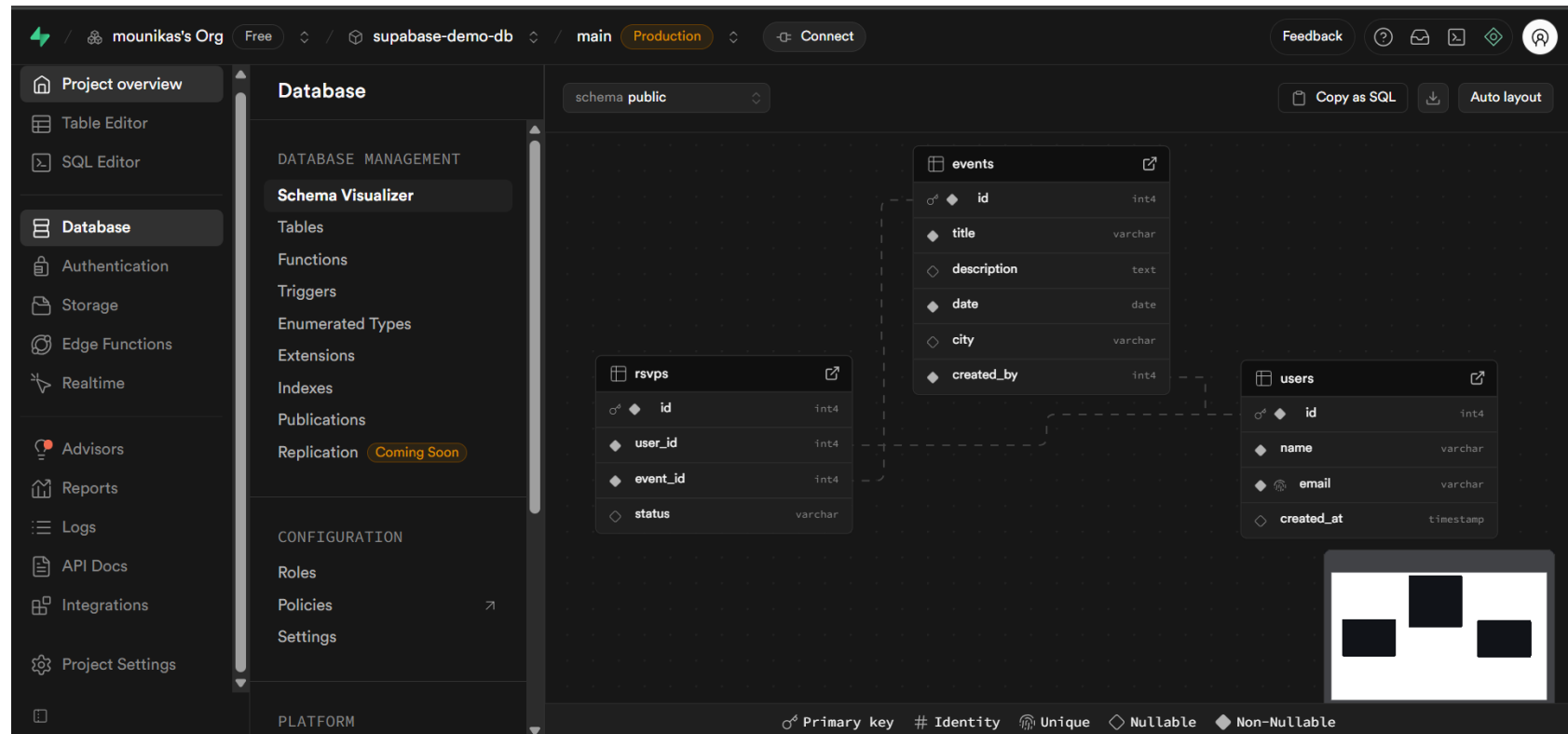
id	name	email	created_at
1	Aarav Sharma	aarav@example.com	2025-08-26 10:31:08.185438
2	Ishita Reddy	ishita@example.com	2025-08-26 10:31:08.185438
3	Rohan Mehta	rohan@example.com	2025-08-26 10:31:08.185438
4	Sneha Kapoor	sneha@example.com	2025-08-26 10:31:08.185438
5	Vikram Das	vikram@example.com	2025-08-26 10:31:08.185438
6	Meera Nair	meera@example.com	2025-08-26 10:31:08.185438
7	Aditya Rao	aditya@example.com	2025-08-26 10:31:08.185438
8	Priya Iyer	priya@example.com	2025-08-26 10:31:08.185438
9	Karan Singh	karan@example.com	2025-08-26 10:31:08.185438
10	Neha Gupta	neha@example.com	2025-08-26 10:31:08.185438

The Table Editor interface shows the 'public.events' table. The table has columns: id, title, description, date, and city. The table contains 5 rows of data.

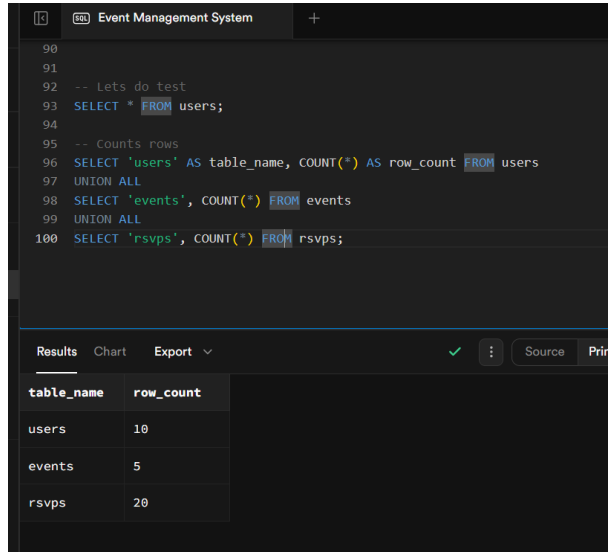
id	title	description	date	city
1	Tech Meetup	A meetup for tech enthusiasts.	2025-09-05	Hyderabad
2	Startup Pitch	Startup founders pitch their ideas.	2025-09-10	Bengaluru
3	AI Workshop	Hands-on AI learning.	2025-09-15	Chennai
4	Music Festival	Live music and fun.	2025-09-20	Hyderabad
5	Hackathon	24-hour coding challenge.	2025-09-25	Bengaluru

ER Diagram

- Supabase auto-generates ER diagrams
- Shows the relationships among tables

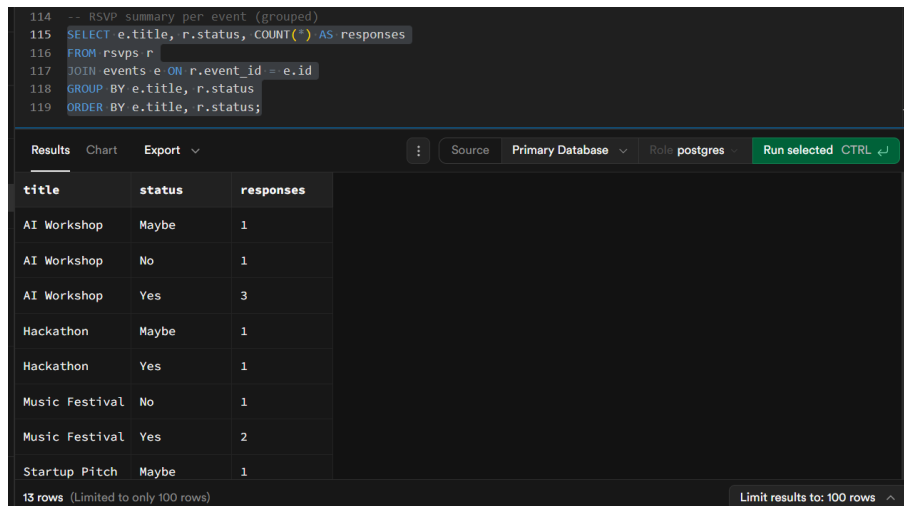


Let extract data



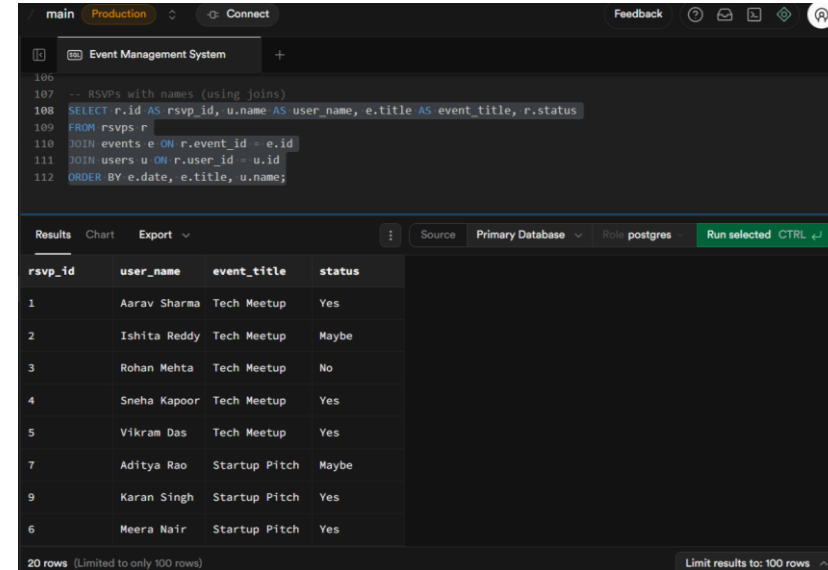
```
90
91
92 -- Lets do test
93 SELECT * FROM users;
94
95 -- Counts rows
96 SELECT 'users' AS table_name, COUNT(*) AS row_count FROM users
97 UNION ALL
98 SELECT 'events', COUNT(*) FROM events
99 UNION ALL
100 SELECT 'rsvps', COUNT(*) FROM rsvps;
```

table_name	row_count
users	10
events	5
rsvps	20



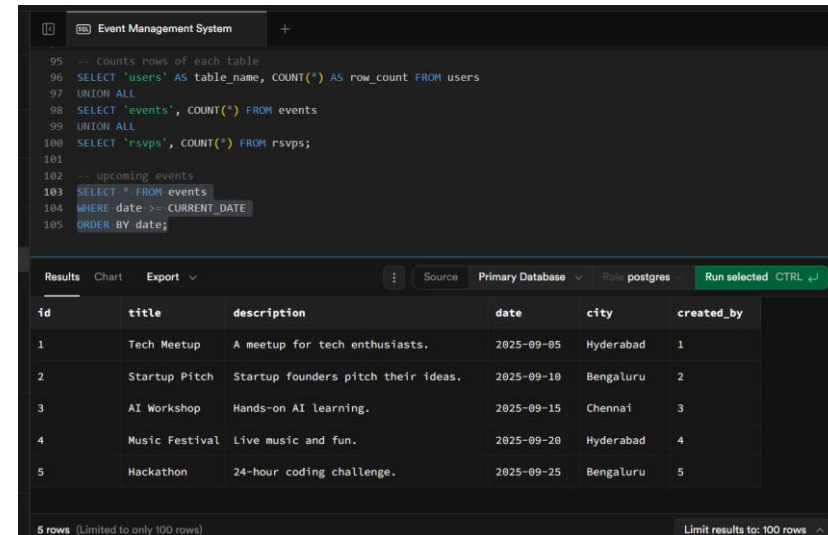
```
114 -- RSVP summary per event (grouped)
115 SELECT e.title, r.status, COUNT(*) AS responses
116 FROM rsvps r
117 JOIN events e ON r.event_id = e.id
118 GROUP BY e.title, r.status
119 ORDER BY e.title, r.status;
```

title	status	responses
AI Workshop	Maybe	1
AI Workshop	No	1
AI Workshop	Yes	3
Hackathon	Maybe	1
Hackathon	Yes	1
Music Festival	No	1
Music Festival	Yes	2
Startup Pitch	Maybe	1



```
106
107 -- RSVPs with names (using joins)
108 SELECT r.id AS rsvp_id, u.name AS user_name, e.title AS event_title, r.status
109 FROM rsvps r
110 JOIN events e ON r.event_id = e.id
111 JOIN users u ON r.user_id = u.id
112 ORDER BY e.date, e.title, u.name;
```

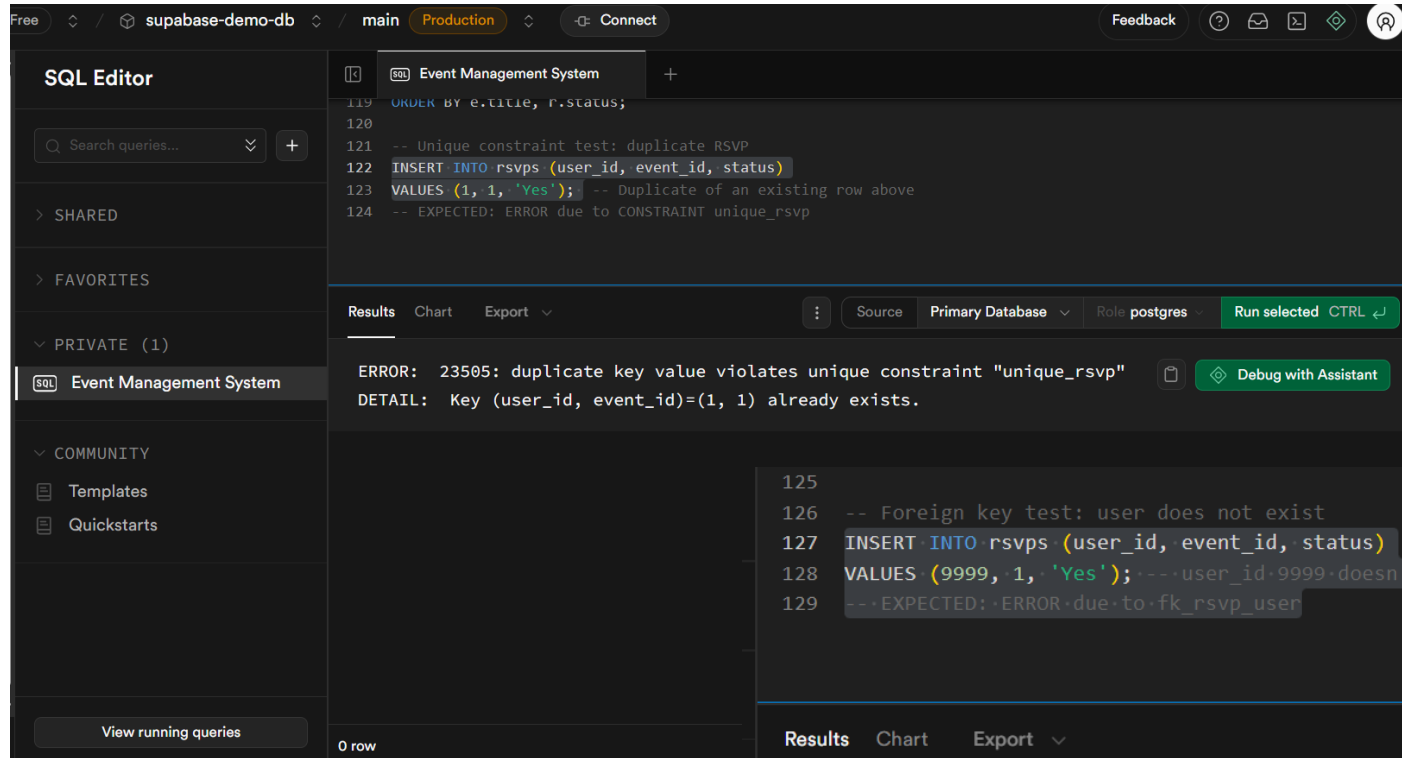
rsvp_id	user_name	event_title	status
1	Aarav Sharma	Tech Meetup	Yes
2	Ishita Reddy	Tech Meetup	Maybe
3	Rohan Mehta	Tech Meetup	No
4	Sneha Kapoor	Tech Meetup	Yes
5	Vikram Das	Tech Meetup	Yes
7	Aditya Rao	Startup Pitch	Maybe
9	Karan Singh	Startup Pitch	Yes
6	Meera Nair	Startup Pitch	Yes



```
95 -- Counts rows of each table
96 SELECT 'users' AS table_name, COUNT(*) AS row_count FROM users
97 UNION ALL
98 SELECT 'events', COUNT(*) FROM events
99 UNION ALL
100 SELECT 'rsvps', COUNT(*) FROM rsvps;
101
102 -- upcoming events
103 SELECT * FROM events
104 WHERE date >= CURRENT_DATE
105 ORDER BY date;
```

id	title	description	date	city	created_by
1	Tech Meetup	A meetup for tech enthusiasts.	2025-09-05	Hyderabad	1
2	Startup Pitch	Startup founders pitch their ideas.	2025-09-10	Bengaluru	2
3	AI Workshop	Hands-on AI learning.	2025-09-15	Chennai	3
4	Music Festival	Live music and fun.	2025-09-20	Hyderabad	4
5	Hackathon	24-hour coding challenge.	2025-09-25	Bengaluru	5

Lets test constraints



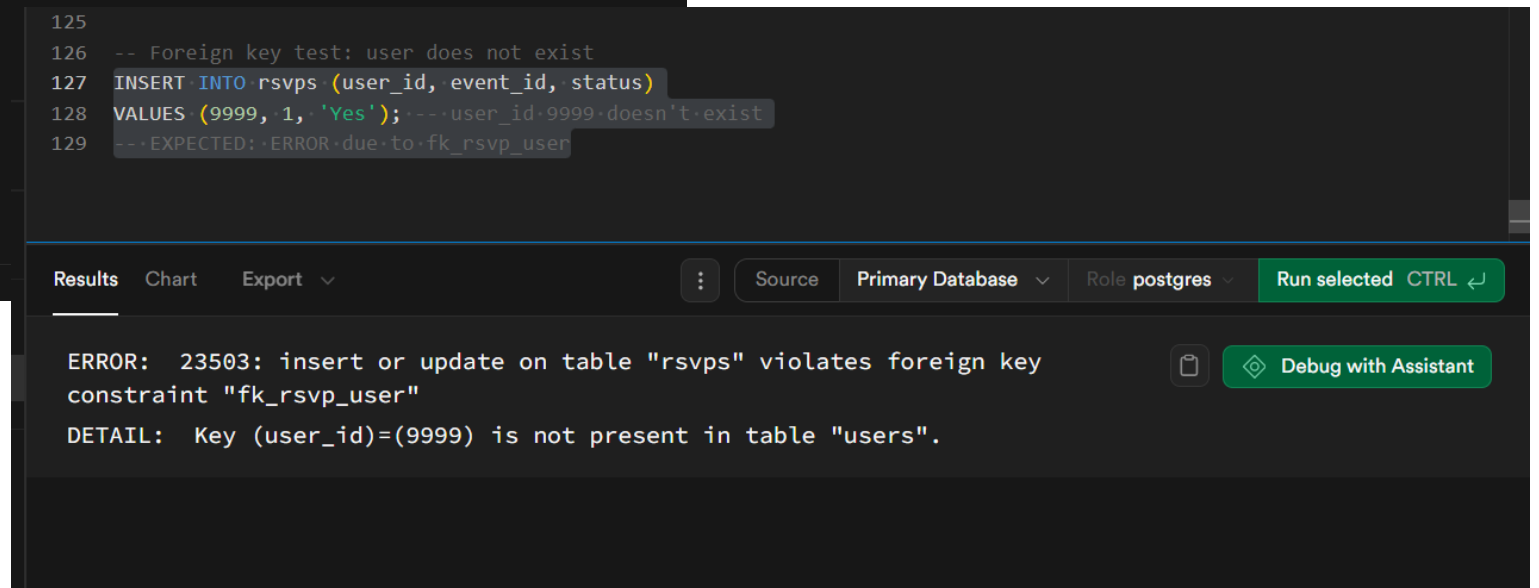
The screenshot shows the Supabase SQL Editor interface. The top bar indicates the connection to 'supabase-demo-db' and the 'main' database in 'Production' mode. The left sidebar shows the 'SQL Editor' with a search bar and a list of queries under 'PRIVATE (1)'. The main editor area shows a query titled 'Event Management System' with the following SQL code:

```
119 ORDER BY e.title, r.status;  
120  
121 -- Unique constraint test: duplicate RSVP  
122 INSERT INTO rsmps (user_id, event_id, status)  
123 VALUES (1, 1, 'Yes'); -- Duplicate of an existing row above  
124 -- EXPECTED: ERROR due to CONSTRAINT unique_rsvp
```

Below the query, the 'Results' tab is active, displaying the following error message:

```
ERROR: 23505: duplicate key value violates unique constraint "unique_rsvp"  
DETAIL: Key (user_id, event_id)=(1, 1) already exists.
```

Buttons for 'Source', 'Primary Database', 'Role postgres', 'Run selected CTRL ↵', and 'Debug with Assistant' are visible.



The screenshot shows the Supabase SQL Editor interface with a query titled 'Event Management System' containing the following SQL code:

```
125  
126 -- Foreign key test: user does not exist  
127 INSERT INTO rsmps (user_id, event_id, status)  
128 VALUES (9999, 1, 'Yes'); -- user_id 9999 doesn't exist  
129 -- EXPECTED: ERROR due to fk_rsvp_user
```

Below the query, the 'Results' tab is active, displaying the following error message:

```
ERROR: 23503: insert or update on table "rsmps" violates foreign key  
constraint "fk_rsvp_user"  
DETAIL: Key (user_id)=(9999) is not present in table "users".
```

Buttons for 'Source', 'Primary Database', 'Role postgres', 'Run selected CTRL ↵', and 'Debug with Assistant' are visible.

Lets add/delete temp user

```
137 -- CASCADE DELETE DEMO (safe, no errors)
138 -- This shows ON DELETE CASCADE working across users -> events -> rsvps
139 WITH new_user AS (
140   INSERT INTO users (name, email)
141   VALUES ('Temp User', 'tempuser@example.com')
142   RETURNING id
143 ),
144 new_event AS (
145   INSERT INTO events (title, description, date, city, created_by)
146   SELECT 'Temp Event', 'Cascade test event', '2025-12-31', 'TestCity', id
147   FROM new_user
148   RETURNING id, created_by
149 )
150 INSERT INTO rsvps (user_id, event_id, status)
151 SELECT ne.created_by, ne.id, 'Yes' FROM new_event ne;
152
```

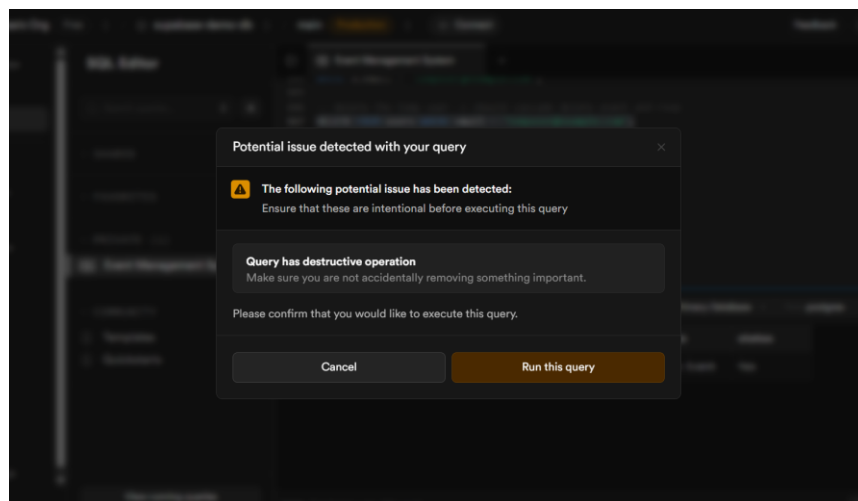
Results Chart Export Source Primary Database Role postgres Run selected CTRL ↵

Success. No rows returned

```
153 -- Show the temp rows exist
154 SELECT
155   u.id AS temp_user_id,
156   e.id AS temp_event_id,
157   r.id AS temp_rsvp_id,
158   u.email,
159   e.title,
160   r.status
161 FROM users u
162 JOIN events e ON e.created_by = u.id AND e.title = 'Temp Event'
163 JOIN rsvps r ON r.user_id = u.id AND r.event_id = e.id
164 WHERE u.email = 'tempuser@example.com';
```

Results Chart Export Source Primary Database Role postgres Run selected CTRL ↵

temp_user_id	temp_event_id	temp_rsvp_id	email	title	status
11	6	23	tempuser@example.com	Temp Event	Yes



```
166 -- Delete the temp user -> should cascade delete event and rsvp
167 DELETE FROM users WHERE email = 'tempuser@example.com';
168
169 -- Lets check (should return 0 rows)
170 SELECT *
171 FROM rsvps r
172 JOIN events e ON r.event_id = e.id
173 WHERE e.title = 'Temp Event';
```

Results Chart Export Source Primary Database Role postgres Run selected CTRL ↵

Success. No rows returned

0 row (Limited to only 100 rows) Limit results to: 100 rows

What is Next.js

- Next.js is a React framework for building server-side rendered (SSR) and statically generated web apps.
- Features:
 - File-based routing
 - Server-side rendering (SSR) & static site generation (SSG)
 - API routes for backend logic
 - Optimized performance & SEO
- It supports fast development and easy integration with Supabase.

Next.js Setup

- Installed Node.js and Next.js
- Created Next.js app:
 - `npx create-next-app@latest event-management-app`
- Created project folder: event-management-app
- Open project in VS Code:
 - Run locally:
`cd event-management-app`
`npm run dev`
 - Local link: <http://localhost:3000>
- Installed `@supabase/supabase-js` for DB connection
- Project structure: app/, lib/, public/, .env.local

Connecting Supabase

- Install Supabase JS SDK: `npm install @supabase/supabase-js`
- Create `.env.local` with keys:
 - `NEXT_PUBLIC_SUPABASE_URL=https://your-project-id.supabase.co`
 - `NEXT_PUBLIC_SUPABASE_ANON_KEY=your-anon-key`
- Create `lib/supabaseClient.js` to connect
- Now Supabase is connected to your app.

Pages & Features

- Pages Created:
 - /events → List all upcoming events
 - /rsvps → Users can RSVP (Yes/No/Maybe)
 - /users → List all users
- Let see event page
 - Path: app/events/page.js
 - Fetch events from Supabase and display:

```
const { data: events } = await supabase.from("events").select("*");
```
 - Displays **event title, description, city, date**
- Features:
 - Tailwind CSS for:
 - Clear UI, color contrast
 - Table formatting, alerts for add/delete operations
 - Navbar for navigation
 - Fully CRUD functional
 - Created **Navbar** for navigation: app/components/navbar.js

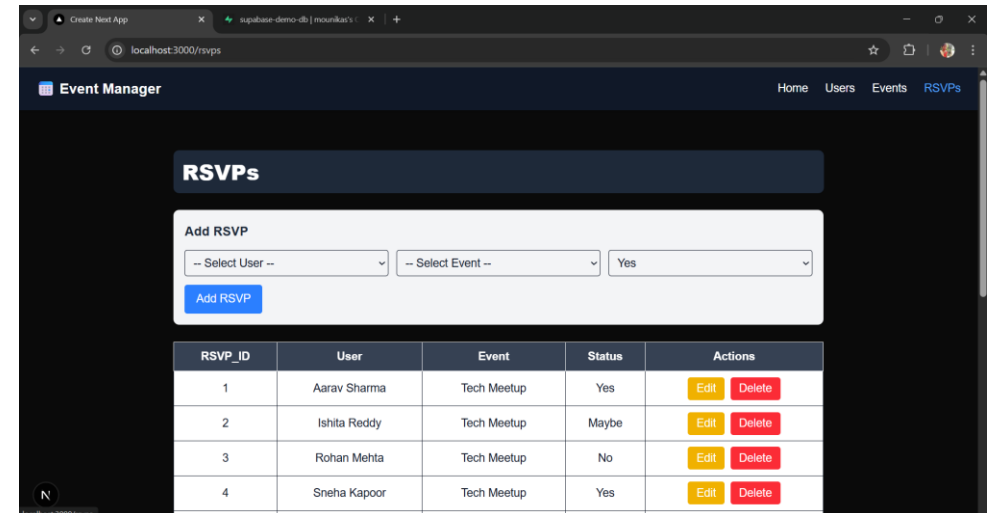
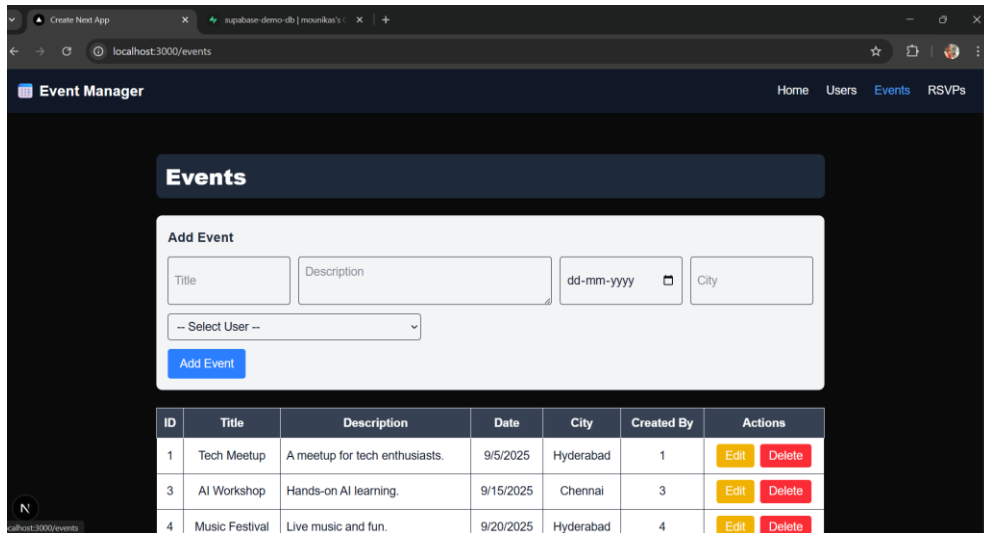
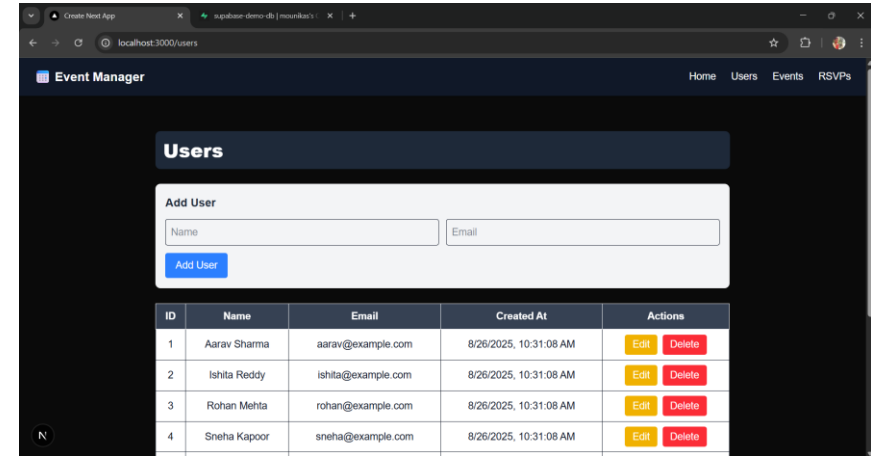
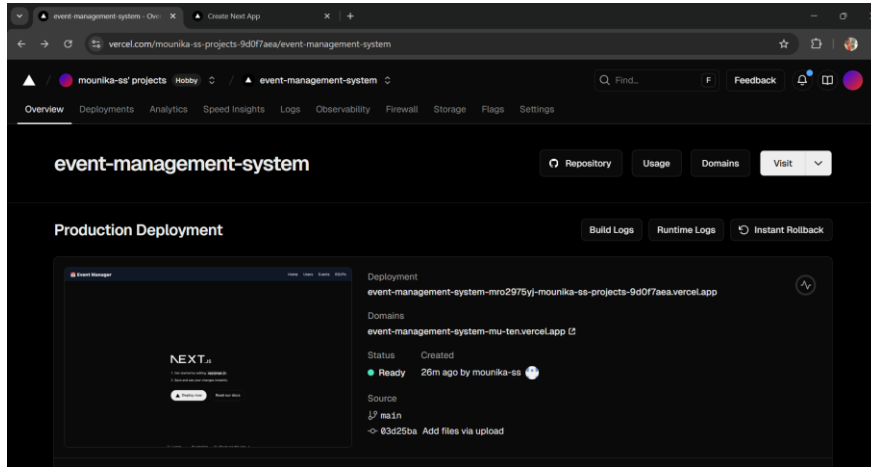
GitHub Upload

- Repository: event-management-system
- Folders uploaded:
 - app/ (Next.js source code), lib/ (Supabase connection), public/ (Images & static assets)
 - .gitignore, package.json, package-lock.json, README.md
- Excluded: node_modules/, .next/, .env.local
- README included with instructions and project overview

Vercel Deployment

- Connect GitHub repo to Vercel
- Leave defaults:
 - Build: npm run build
 - Output: .next
 - Install: npm install
- Add environment variables from .env.local,
 - that is KEY (names) and VALUE (project url, API keys)
- Deploy -> live/Production URL:
 - <https://event-management-system-mu-ten.vercel.app/>
- Live demo works with CRUD operations and Supabase backend

Vercel dashboard and frontend pages



Summary / Takeaways

- Learned database design & relationships
- Connected Supabase backend with Next.js frontend
- Implemented CRUD operations
- Deployed live project on Vercel
- Fully documented in GitHub repo & README

Conclusion

- Successfully created a Supabase database with Users, Events, and RSVPs tables.
- Developed a Next.js app to list events and manage RSVPs.
- Integrated Supabase backend using `@supabase/supabase-js`.
- Project deployed on Vercel with live URL shared.
- GitHub repo contains code, SQL dump, and screenshots for reference.

Thank you

Mounika Seelam

Database Management & Frontend Development Enthusiast