**Mini Project: SQL Learning Platform**

A comprehensive Flutter application for SQL tutorials, interactive quizzes, and documentation with Supabase authentication.

**Table of Contents**

[1. Introduction](https://sabiqsbo.genspark.space/#introduction)

[2. Authentication Features](https://sabiqsbo.genspark.space/#auth-features)

[3. Platform Features](https://sabiqsbo.genspark.space/#features)

[4. Setup Instructions](https://sabiqsbo.genspark.space/#setup)

[5. UI Design](https://sabiqsbo.genspark.space/#ui-design)

[6. System Architecture](https://sabiqsbo.genspark.space/#architecture)

[7. Technical Details](https://sabiqsbo.genspark.space/#tech-details)

[8. Dependencies](https://sabiqsbo.genspark.space/#dependencies)

[9. Troubleshooting](https://sabiqsbo.genspark.space/#troubleshooting)

[10. Additional Resources](https://sabiqsbo.genspark.space/#resources)

**1. Introduction**

The SQL Learning Platform is a Flutter application designed to provide comprehensive SQL education through interactive tutorials, quizzes, and documentation. It leverages Supabase for backend services, including robust authentication options.

The platform aims to make SQL learning more engaging and accessible through gamified experiences, personalized learning paths, and AI-assisted features.

**2. Authentication Features**

**Email/Password Authentication**

Secure user registration and login using traditional email and password authentication with advanced security measures.

**Google Sign-In**

Streamlined login process using existing Google accounts for a faster and more convenient user onboarding experience.

**GitHub Sign-In**

Integration with GitHub authentication for developer-friendly login options, particularly appealing to the technical userbase.

**Password Reset**

Secure password recovery system with email verification to help users regain access to their accounts safely.

**User Profiles**

Customizable user profiles that track learning progress, achievements, and personalized learning paths across the platform.

**3. Platform Features**

**Interactive SQL Lessons**

Step-by-step tutorials covering SQL fundamentals with practical examples and interactive code execution.

**SQL Quiz**

Comprehensive quizzes on various SQL topics to test knowledge and reinforce learning through practical challenges.

**AI SQL Bot**

Intelligent assistant providing help with SQL queries and offering guidance on database concepts and best practices.

**Documentation**

Comprehensive SQL reference materials covering syntax, functions, and database management principles.

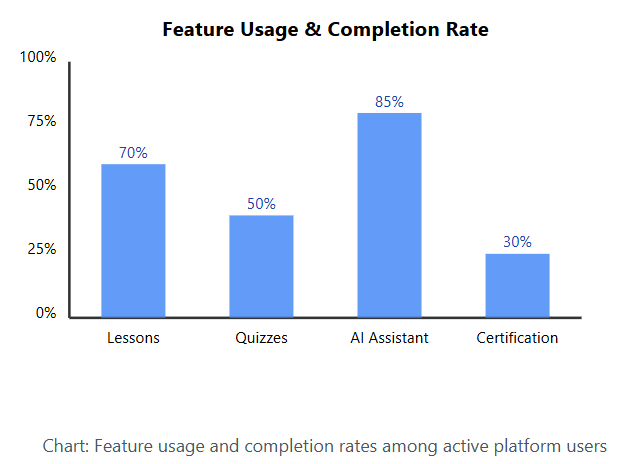
**AI Quiz Generator**

Create custom SQL quizzes with AI assistance tailored to specific learning objectives and skill levels.

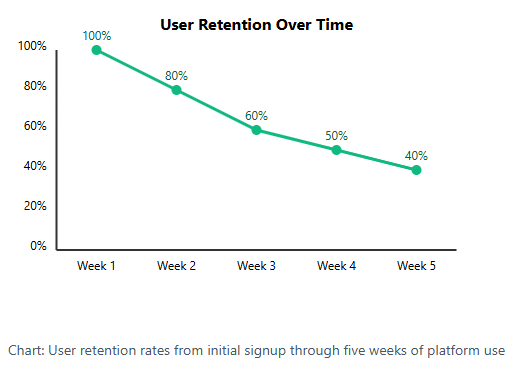
**Certification**

Earn certificates to showcase SQL skills and achievements, validating proficiency for academic or professional purposes.

**User Engagement Metrics**



**User Retention Analysis**



**4. Setup Instructions**

**Prerequisites**

* Flutter SDK (stable channel)
* A Supa base account - [Sign up here](https://supabase.com/)
* Google Developer account (for Google Sign-In) - [Sign up here](https://console.developers.google.com/)
* GitHub Developer account (for GitHub Sign-In) - [Register here](https://github.com/settings/developers)

**Step 1: Set up Supa base**

1. Create a new project on [Supabase](https://supabase.com/).
2. Once your project is created, go to the SQL Editor in the Supabase dashboard.
3. Copy the entire content of the supabase\_setup.sql file from this project.
4. Paste and run the SQL query in the SQL Editor. This will:
   * Create the required tables (profiles, user\_progress, user\_achievements)
   * Set up row-level security policies
   * Create functions and triggers for user management

**Step 2: Configure Authentication Providers**

**Email Provider**

1. In your Supabase dashboard, navigate to Authentication > Providers > Email.
2. Ensure the Email provider is enabled.
3. Configure settings for email confirmation if desired.

**Google Provider**

1. Navigate to Authentication > Providers > Google.
2. Enable the Google provider.
3. Follow the instructions to set up Google OAuth credentials.
4. Add your app's redirect URL (typically: io.supabase.sqlgame://login-callback/).
5. Copy your Google client ID and client secret to Supabase.

**GitHub Provider**

1. Navigate to Authentication > Providers > GitHub.
2. Enable the GitHub provider.
3. Register a new OAuth application on [GitHub](https://github.com/settings/developers).
4. Set the Authorization callback URL to your Supabase redirect URL.
5. Copy your GitHub client ID and client secret to Supabase.

**Step 3: Environment Setup**

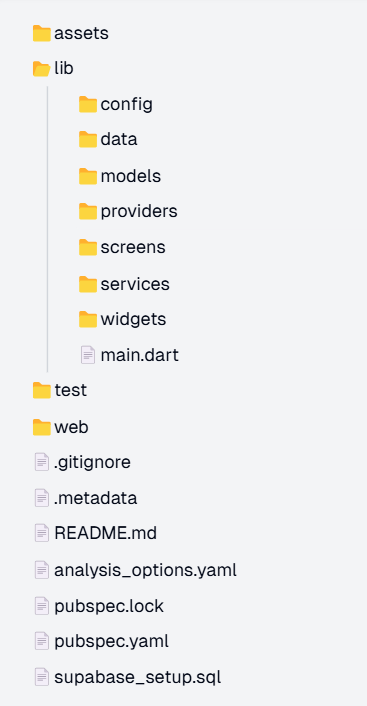
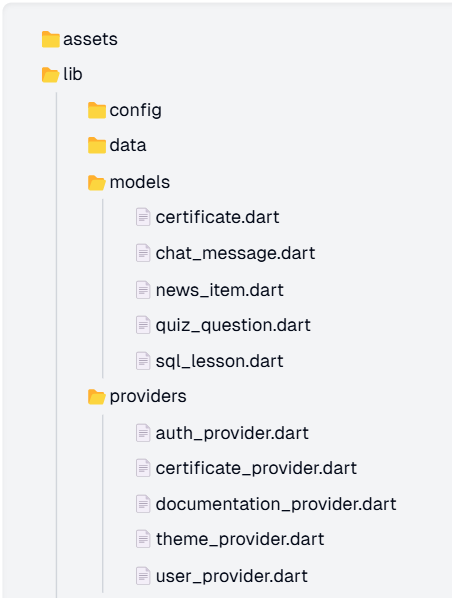
1. Create a .env file in the root of your project.
2. Add the following variables with your Supabase details:

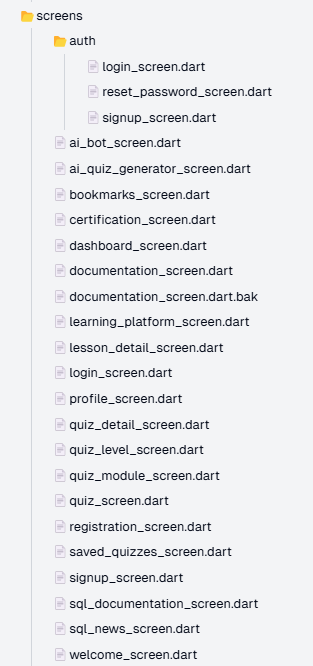
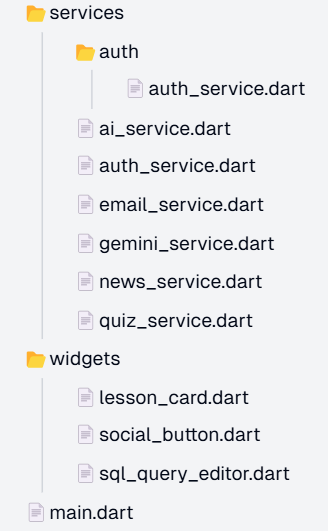
SUPABASE\_URL=https://your-project-ref.supabase.co SUPABASE\_ANON\_KEY=your-anon-key SUPABASE\_REDIRECT\_URL=io.supabase.sqlgame://login-callback/

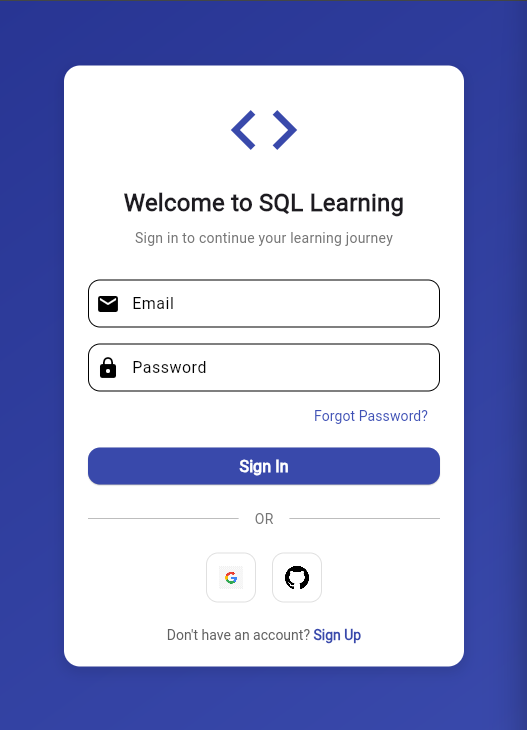
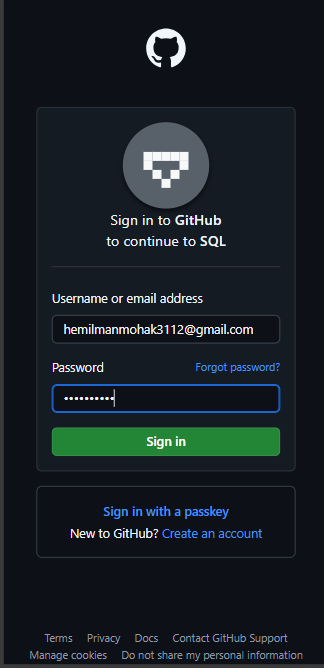
**Step 4: Add Social Icons**

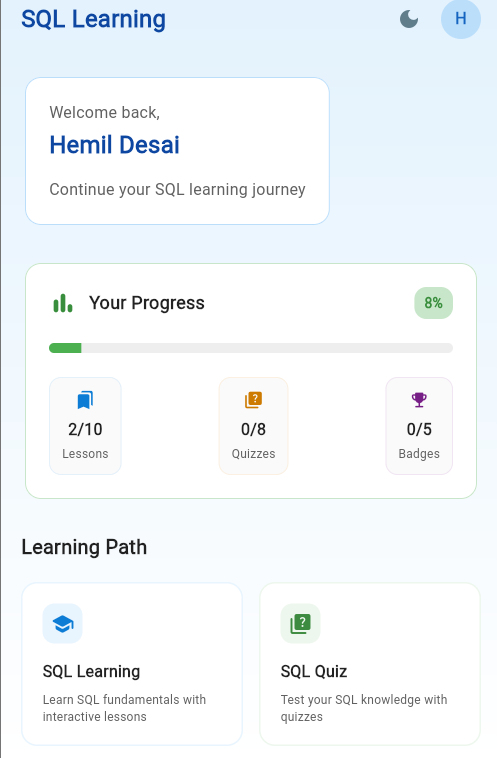
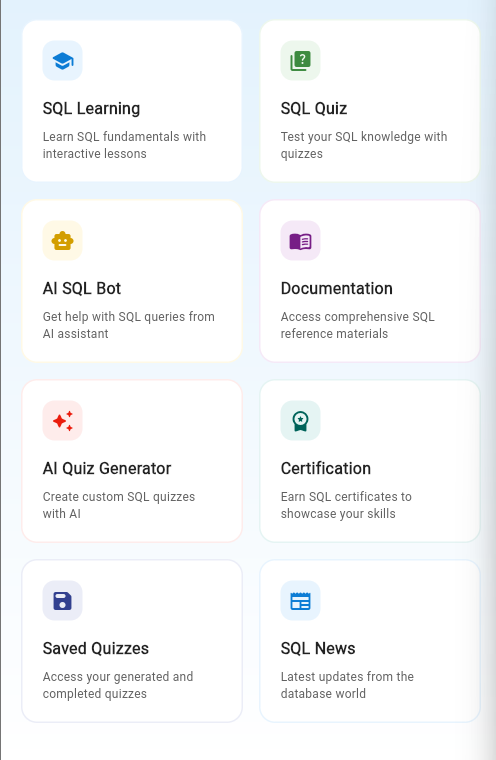
1. Ensure you have Google and GitHub icons in the assets/icons/ directory:
   * google\_icon.png
   * github\_icon.png

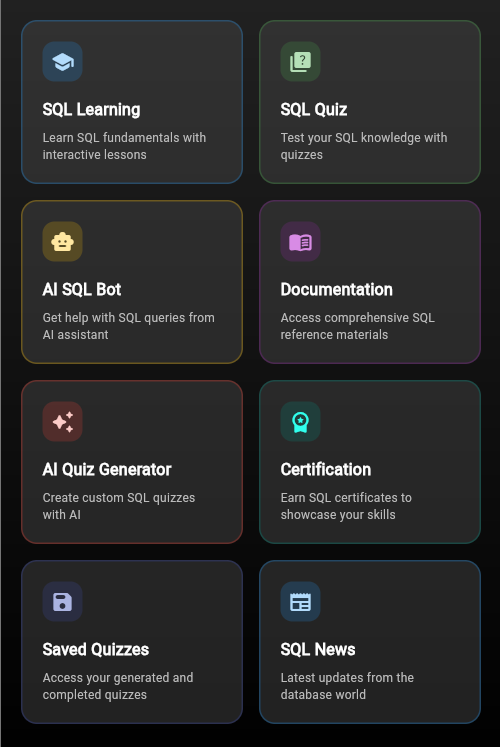
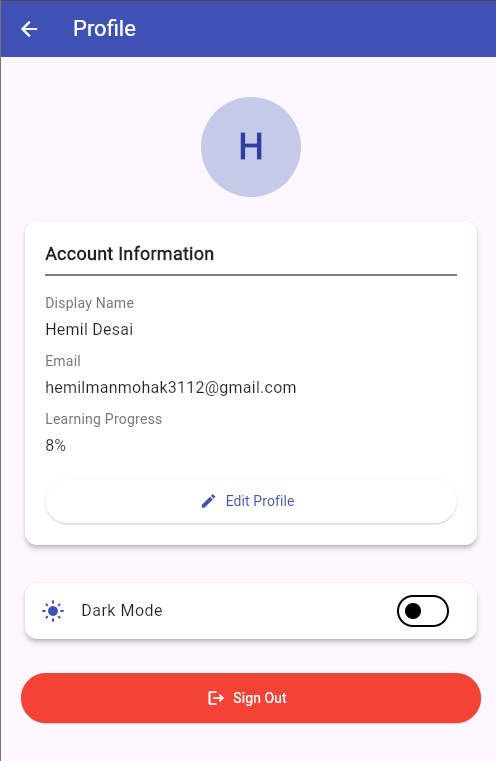
**5. UI Design**

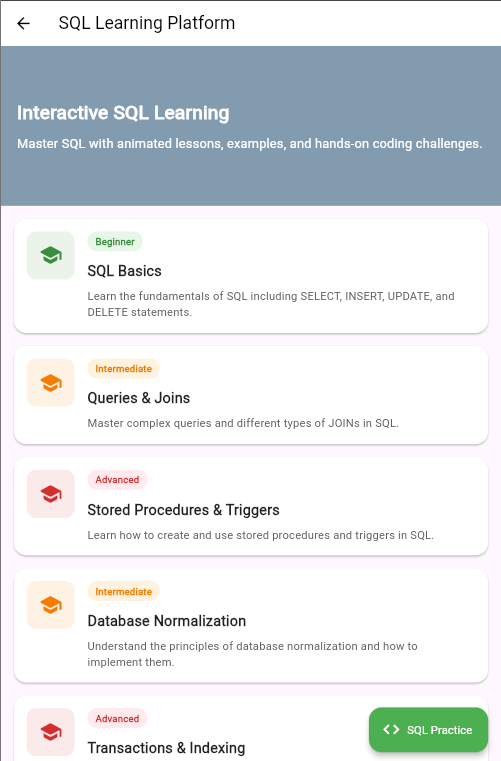
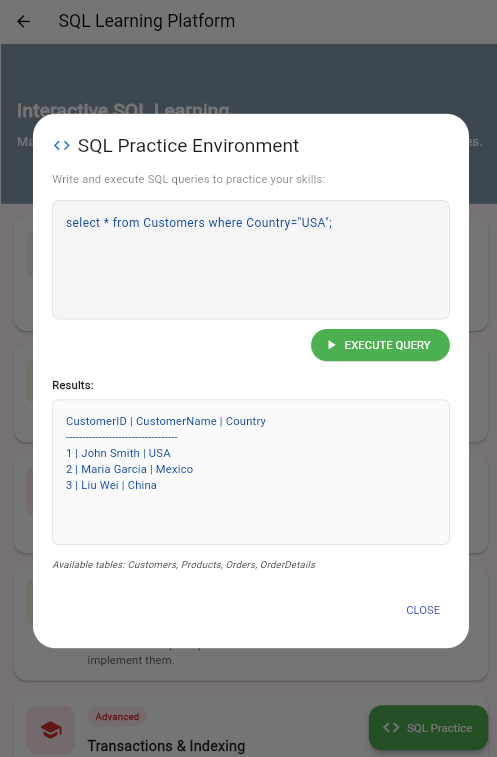
** **

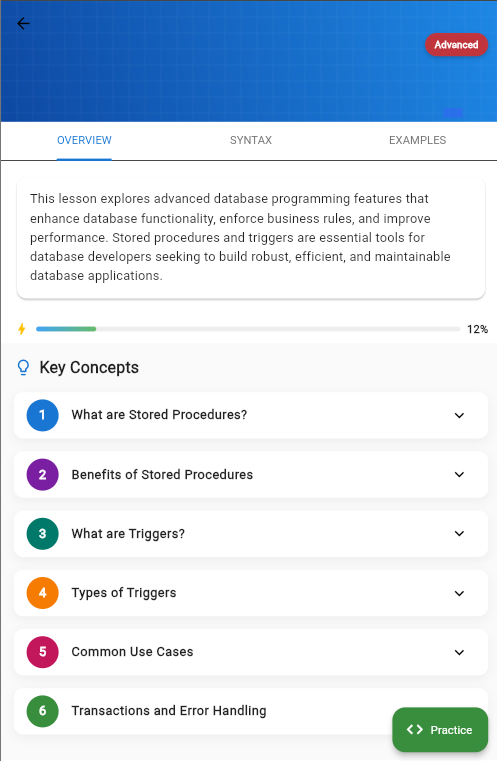
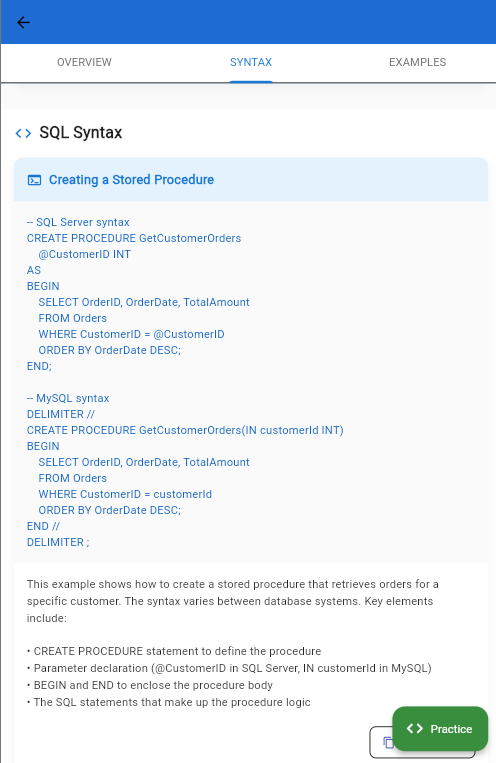
** **

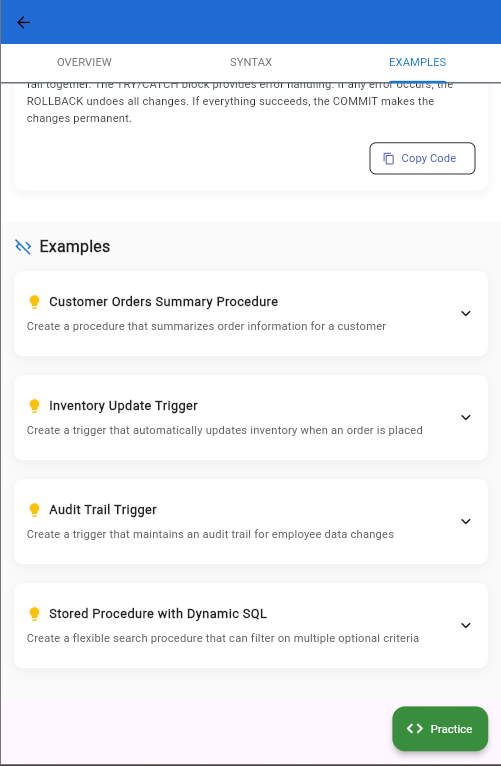
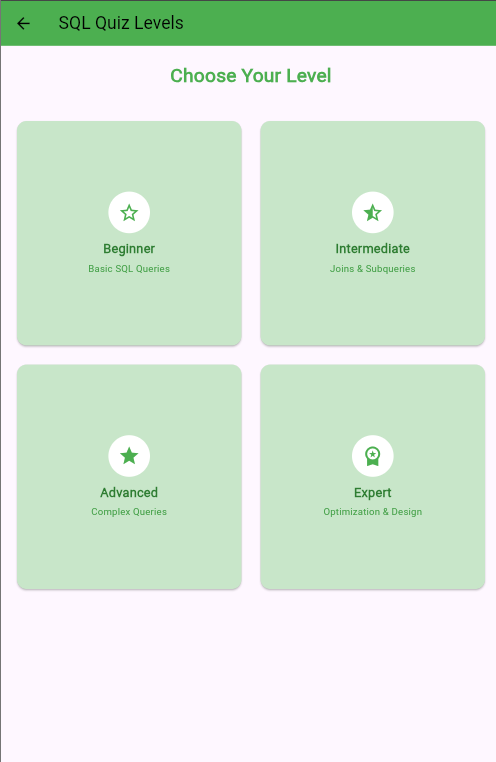
** **

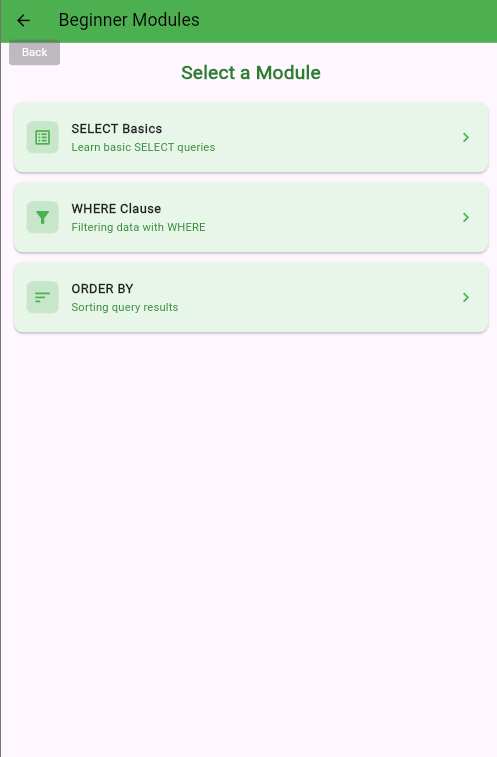
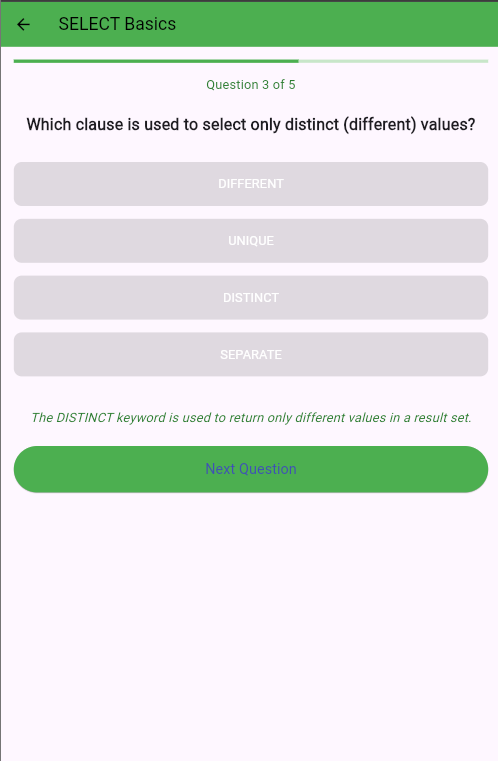
** **

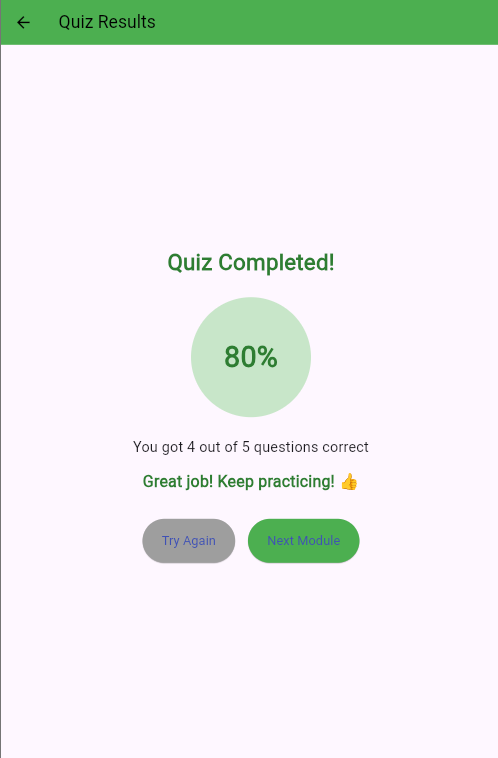
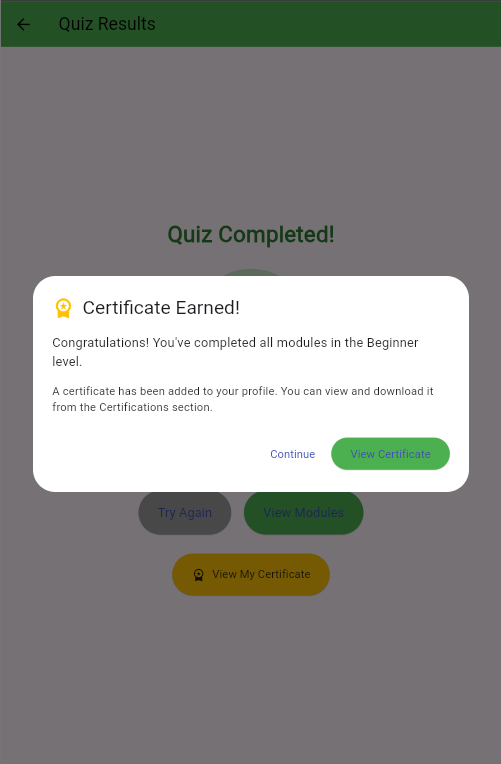
** **

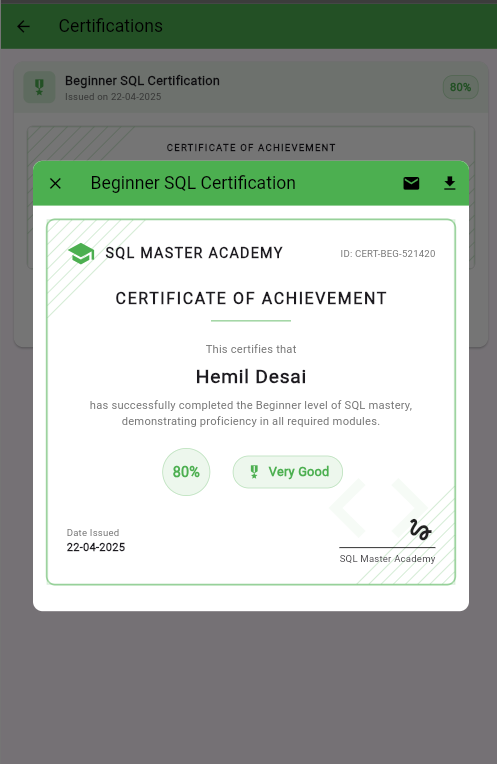
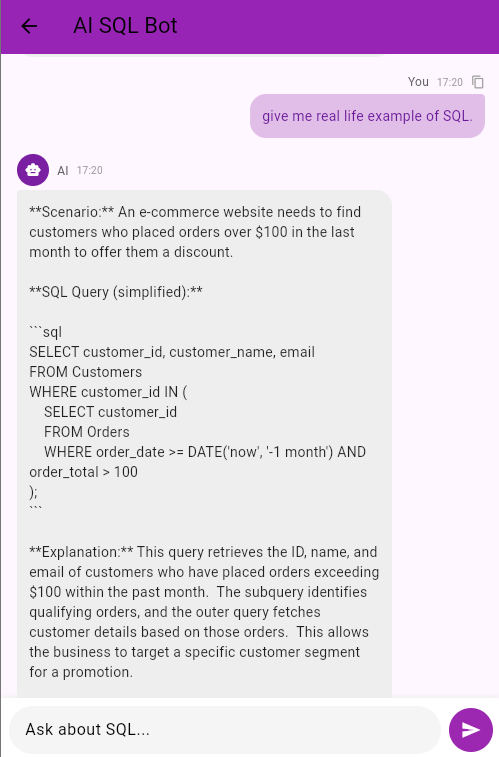
** **

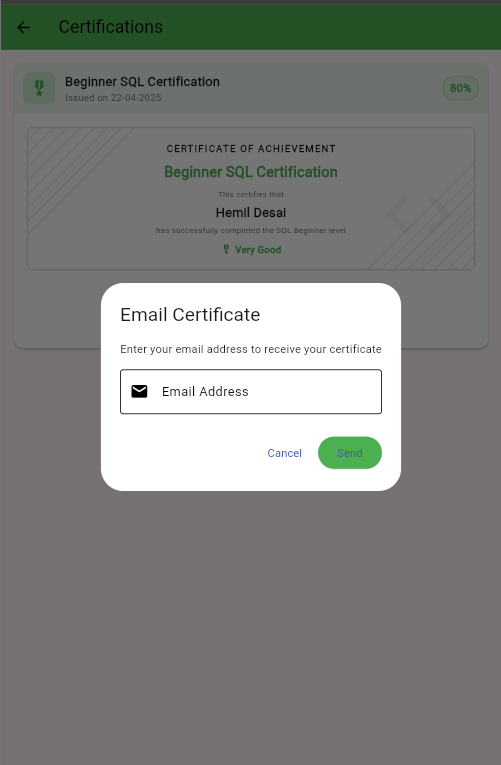
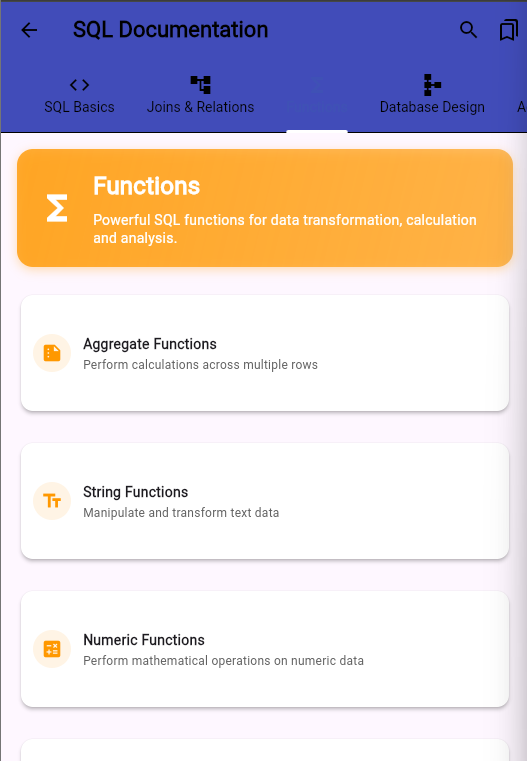
** **

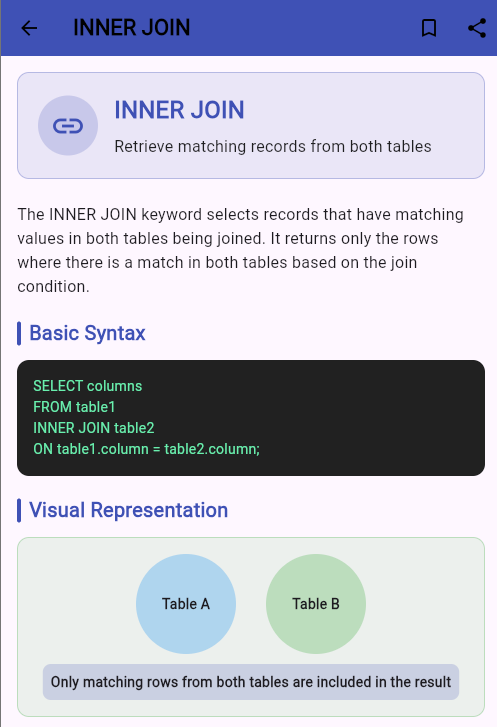
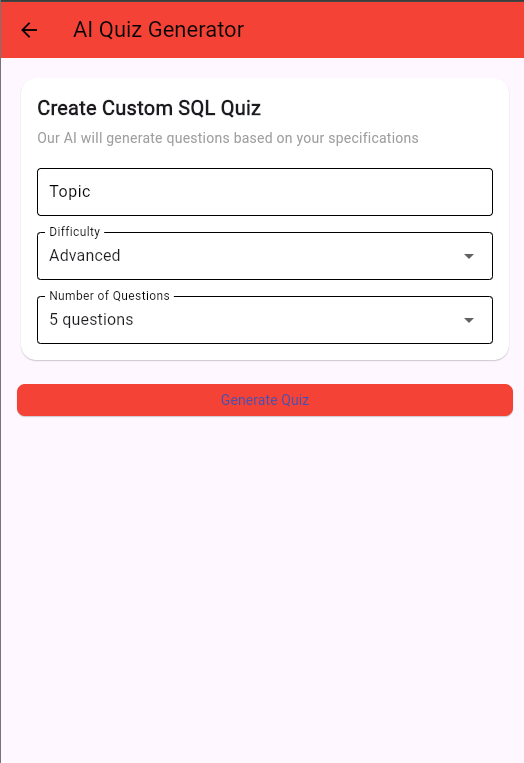
** **

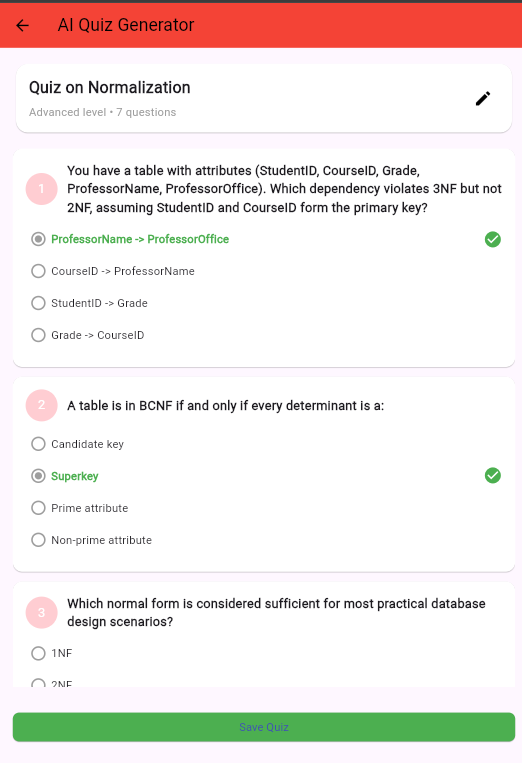
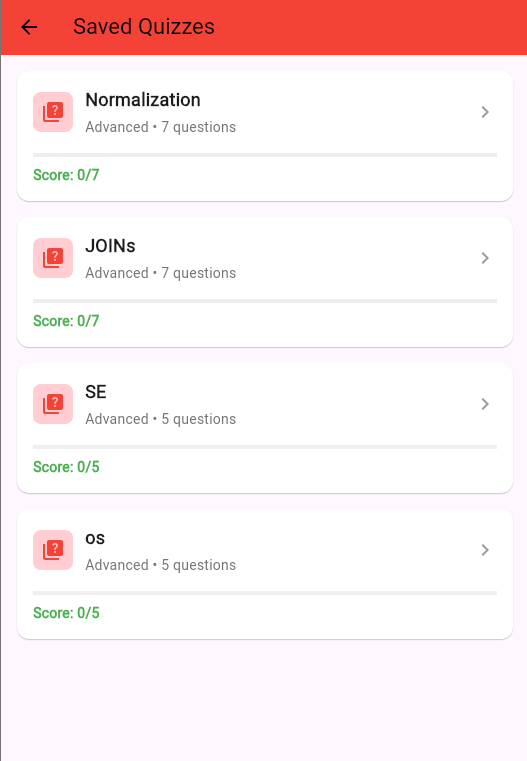
** **

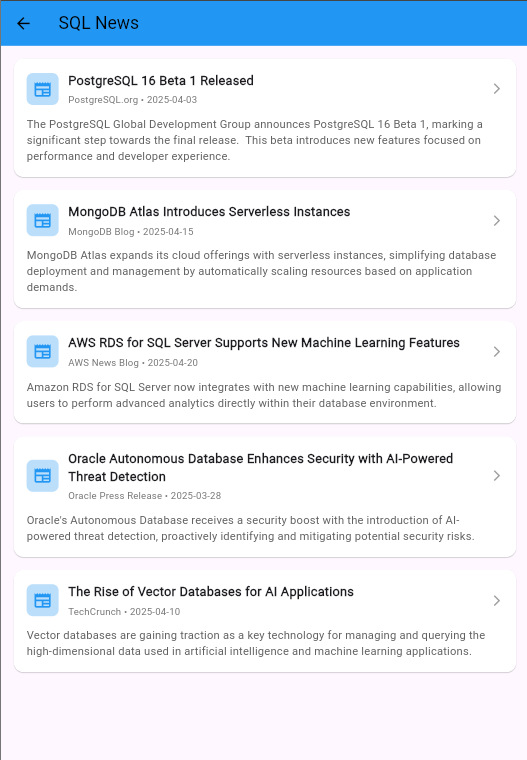
** **

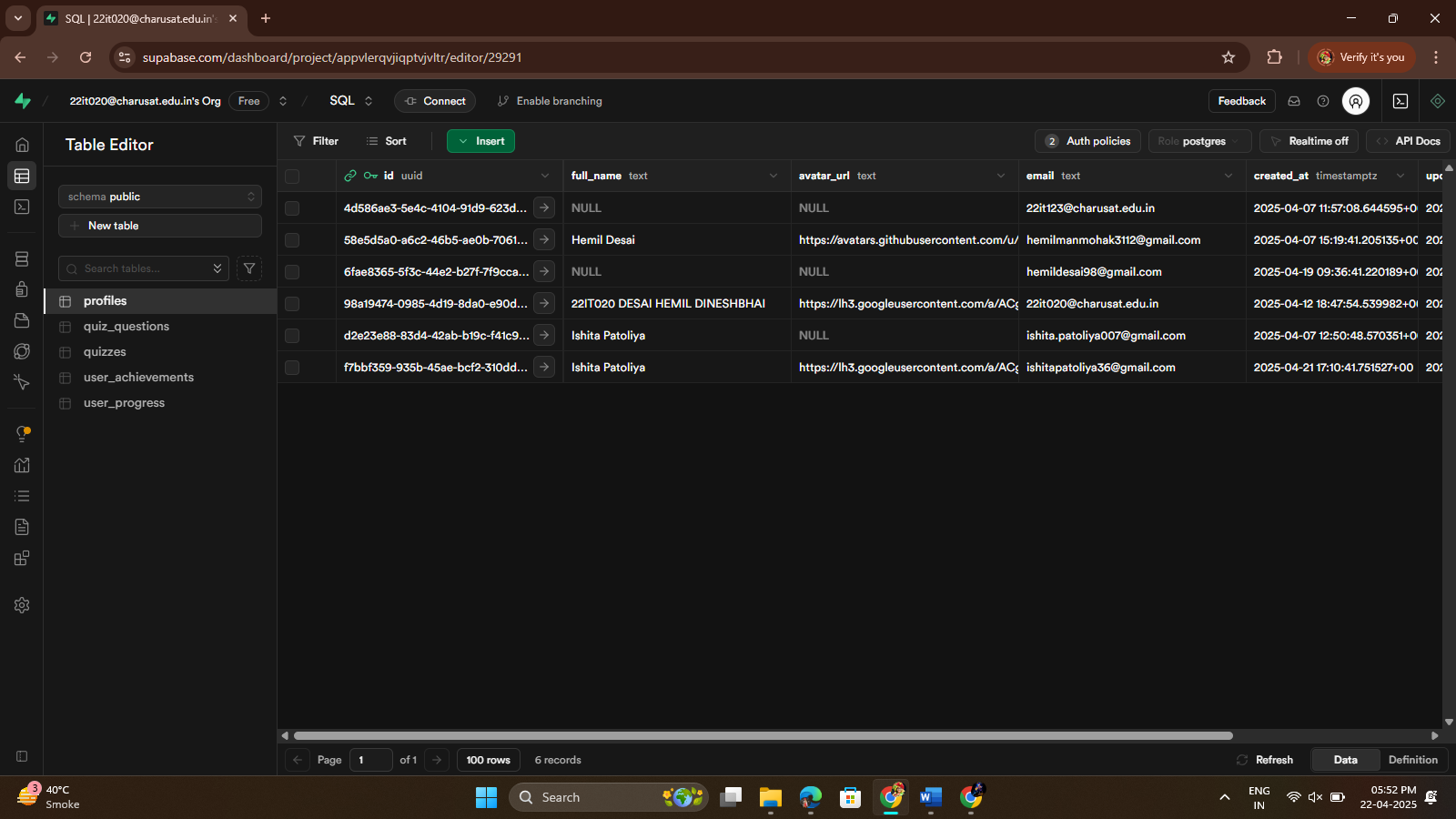
** **

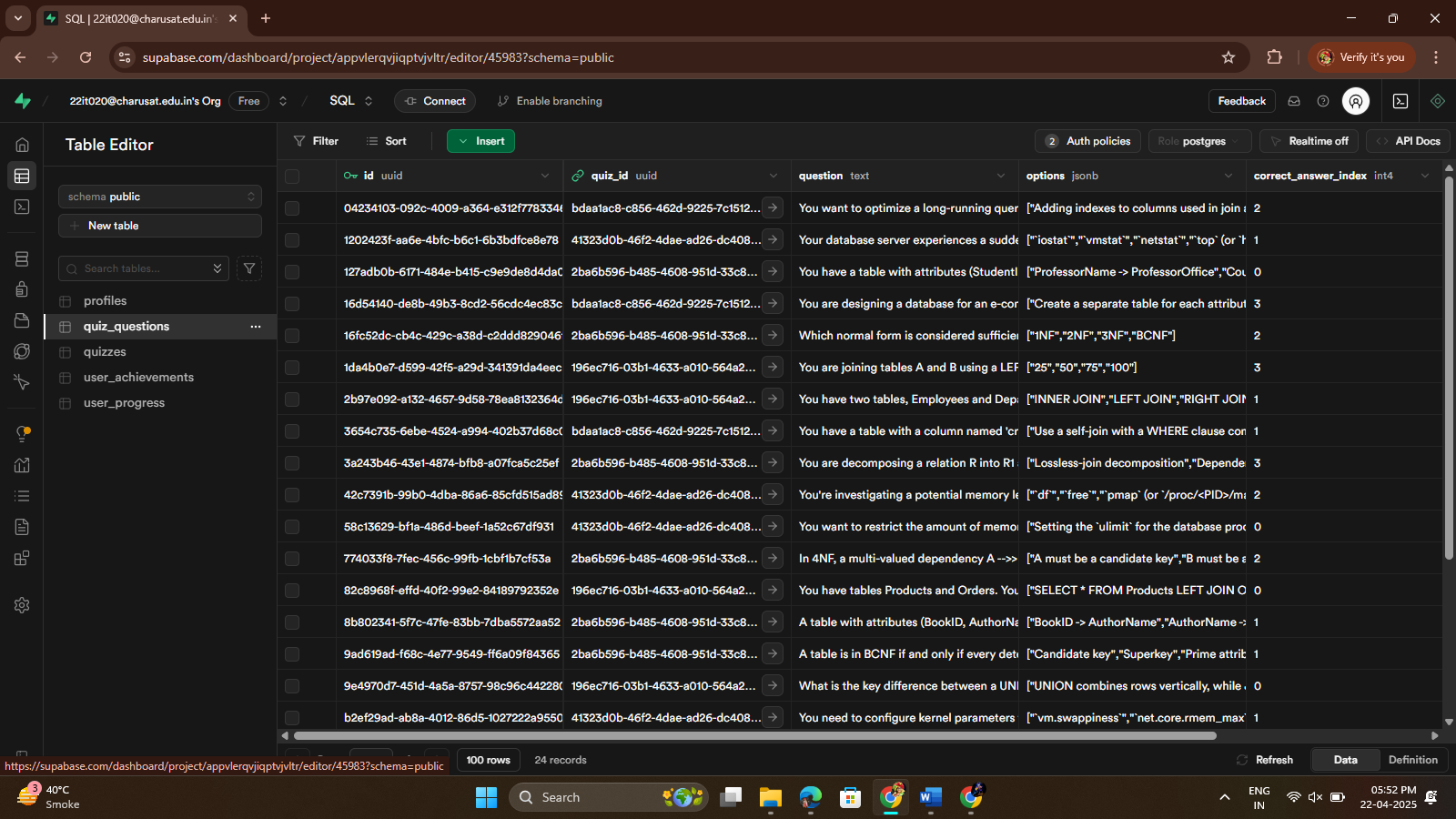
** **

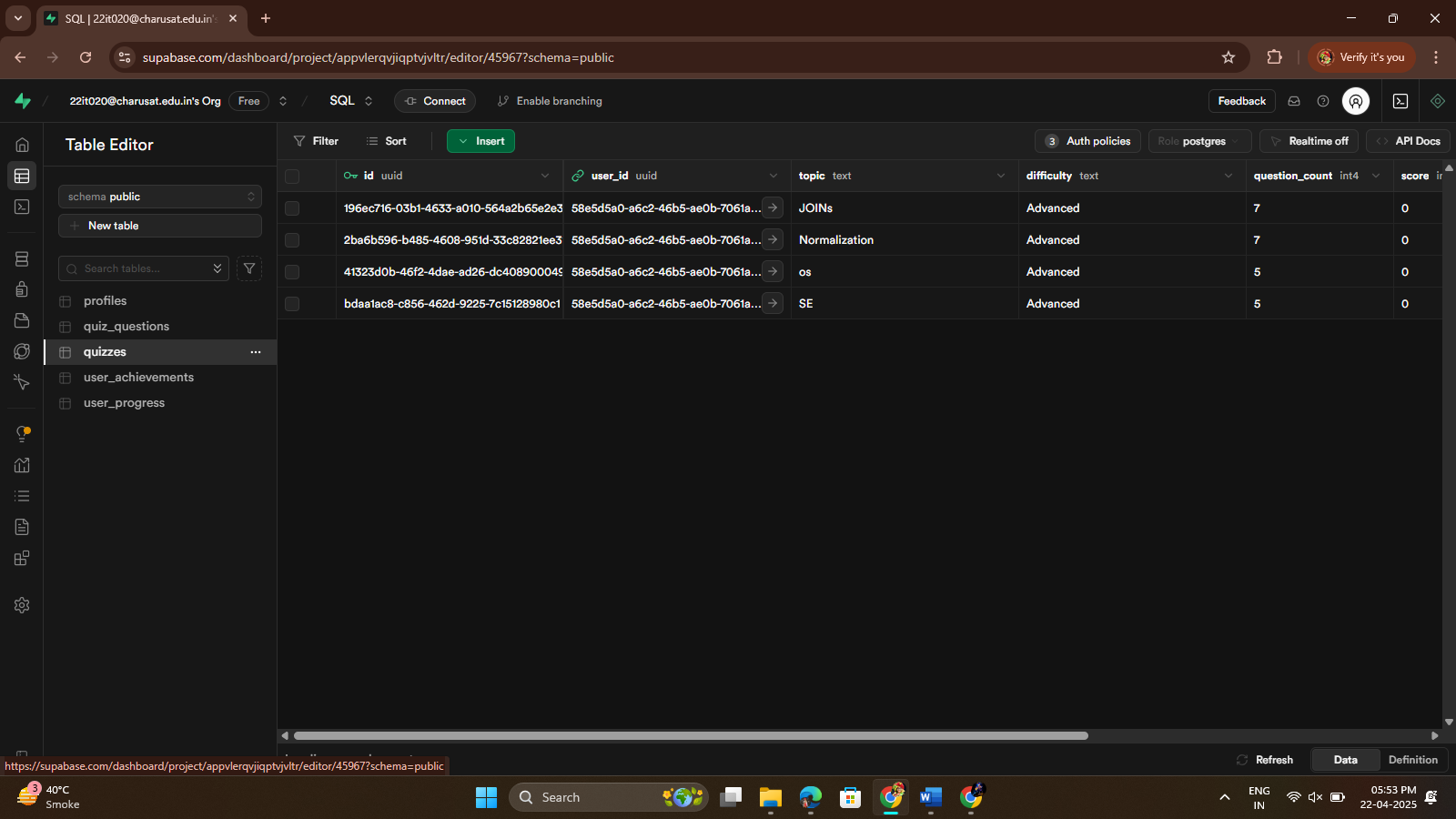
** **

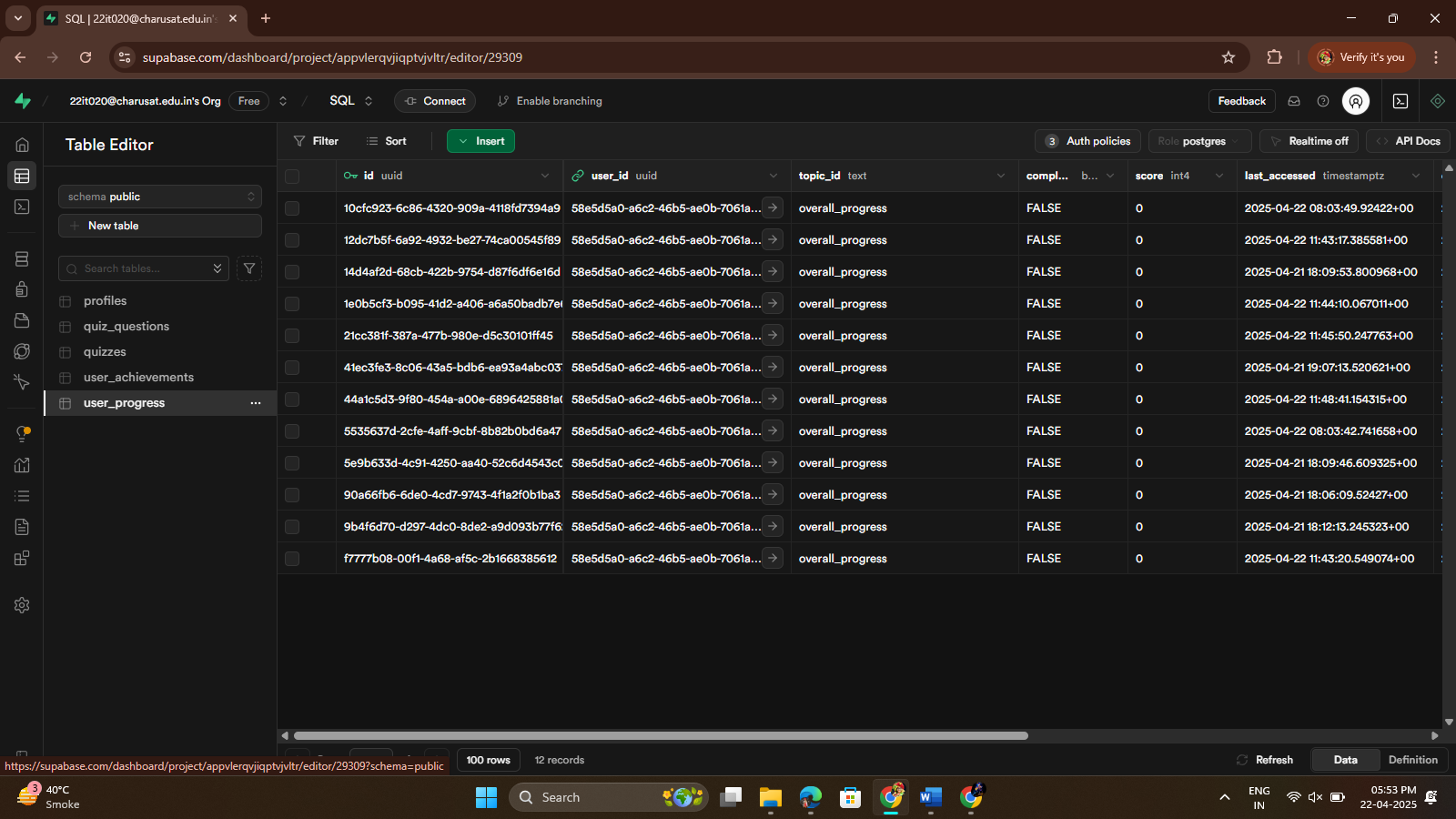
** **

****









**Design Highlights**

**Clean Interface**

Minimalist design with proper spacing and visual hierarchy for optimal readability and focus.

**Card-Based Layout**

Organized content in cards with consistent styling for a cohesive visual experience.

**Visual Progress Tracking**

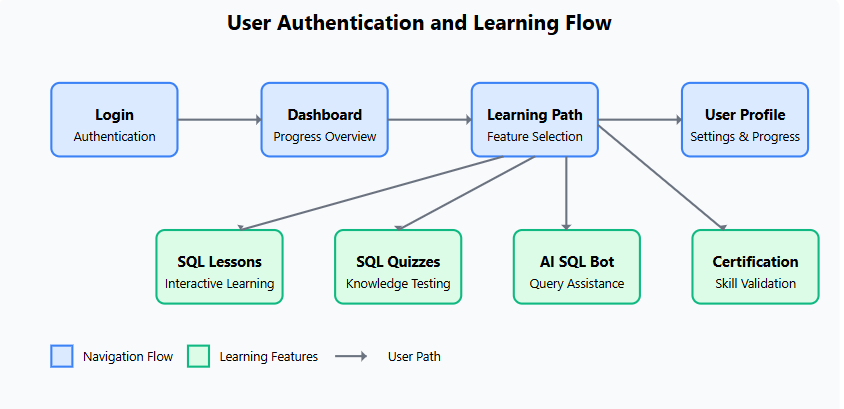
Clear progress indicators and statistics help users track their learning journey.

**Modern Color Scheme**

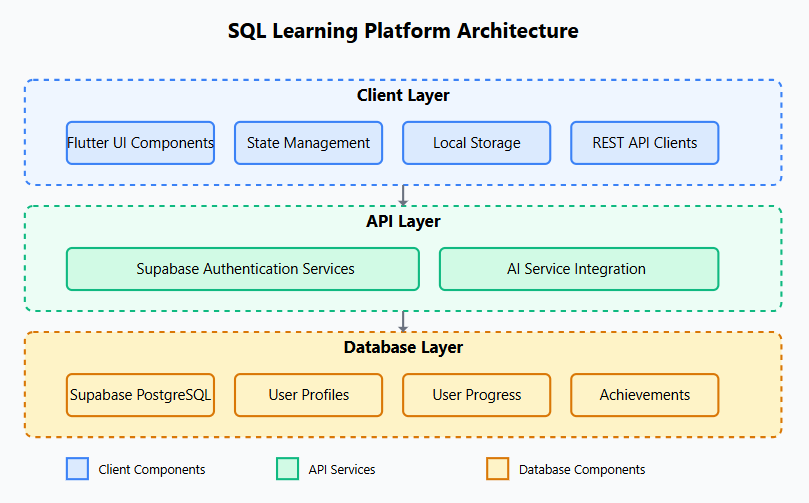
Pleasant color palette with light/dark mode support for comfort in different environments.  
**Descriptive Cards**

Feature cards with intuitive icons and clear descriptions help users understand available functionality at a glance.

**User Flow Diagram**



**6. System Architecture**



**Client Layer**

* Flutter UI Components
* Provider State Management
* SharedPreferences for local storage
* SQLite for offline data
* Supabase API clients

**API Layer**

* Supabase Authentication
* Row-Level Security Policies
* AI Service Integrations
* REST API Endpoints
* Webhooks for notifications

**Database Layer**

* PostgreSQL Database
* User Profiles Schema
* Progress Tracking Tables
* Achievement System
* SQL Quiz Repository

**Data Flow**

**Authentication Flow**

1. User initiates authentication via email/password or social providers
2. Authentication request is sent to Supabase Auth service
3. Supabase validates credentials and issues JWT token
4. JWT token is stored securely on the device
5. App configures API clients with authentication token
6. On subsequent API calls, token is included for authorization

**Learning Flow**

1. User selects learning content from dashboard
2. App retrieves content and user progress from Supabase
3. Content is cached locally for offline access
4. User interacts with lessons, quizzes, or AI features
5. Progress updates are sent to the server and stored locally
6. Achievements and certifications are issued based on milestone completion

**7. Technical Details**

**Platform Specifications**

**Cross-Platform Support**

Built with Flutter for seamless deployment across iOS, Android, and web platforms from a single codebase, ensuring consistent user experience across devices.

**State Management**

Implemented using Provider for efficient reactive state updates and enhanced maintainability, with clear separation between UI and business logic.

**Local Storage**

Combination of SharedPreferences for settings and user preferences, with SQLite for structured local database to support offline functionality.

**Authentication & Security**

Secure authentication via Supabase with JWT tokens, social login options, and row-level security policies to ensure data privacy and access control.

**AI Integration**

Integration with AI services for chatbot functionality, quiz generation, and personalized learning recommendations based on user performance.

**PDF Generation**

Certificates and learning materials generated as PDFs using the PDF library, supporting custom designs and digital credentials for completed courses.  
 **Database Schema**

| Table | Description | Key Fields |
| --- | --- | --- |
| profiles | Stores user profile information including preferences and settings | id (PK, UUID) updated\_at (timestamp) username (text) avatar\_url (text) preferences (JSONB) |
| user\_progress | Tracks user progress through lessons, quizzes, and learning paths | id (PK, UUID) user\_id (FK, UUID) content\_type (text) content\_id (UUID) progress (integer) completed\_at (timestamp) |
| user\_achievements | Records badges, certificates, and other achievements earned by users | id (PK, UUID) user\_id (FK, UUID) achievement\_id (UUID) achieved\_at (timestamp) metadata (JSONB) |
| sql\_lessons | Contains lesson content, examples, and explanations for SQL concepts | id (PK, UUID) title (text) content (JSONB) difficulty (text) order (integer) |
| sql\_quizzes | Stores quiz questions, answers, and explanations for assessment | id (PK, UUID) title (text) questions (JSONB) difficulty (text) lesson\_id (FK, UUID) |

**Security Considerations**

**Row-Level Security (RLS):** Supabase PostgreSQL policies ensure users can only access their own data, with per-table access control policies.

**Authentication Tokens:** JWT tokens are securely stored with appropriate expiry and refresh mechanisms.

**Data Encryption:** Sensitive data is encrypted both in transit (HTTPS) and at rest.

**Input Validation:** All user inputs are validated on both client and server sides to prevent injection attacks.

**OAuth Security:** Strict redirect URL validation for social authentication to prevent authorization code interception.

**8. Dependencies**

| Package | Version | Purpose |
| --- | --- | --- |
| flutter | SDK | Core SDK for building the application |
| cupertino\_icons | ^1.0.5 | iOS-style icons for Cupertino widgets |
| lottie | ^2.3.2 | Animations and interactive visual elements |
| http | ^1.1.0 | HTTP requests to APIs and services |
| sqflite | ^2.2.8+4 | SQLite database for local storage |
| path | ^1.8.3 | File and directory path manipulation |
| provider | ^6.0.5 | State management for the application |
| shared\_preferences | ^2.1.1 | Persistent key-value storage for app settings |
| flutter\_svg | ^2.0.5 | SVG rendering for scalable graphics |
| path\_provider | ^2.0.15 | Finding common paths on the device file system |
| url\_launcher | ^6.1.11 | Opening URLs in browser and handling deep links |
| pdf | ^3.10.1 | PDF generation for certificates and documentation |
| flutter\_quill | ^7.2.0 | Rich text editor for notes and documentation |
| supabase\_flutter | latest | Supabase client for Flutter integration |

**9. Troubleshooting**

**Social Login Issues**

**If you encounter problems with social logins, check the following:**

* Verify that redirect URLs match exactly between Supabase and OAuth provider settings
* Check that client IDs and secrets are correctly entered in the Supabase dashboard
* Ensure your OAuth app has the correct permissions and scopes enabled
* Verify the app package name matches the OAuth configuration

**Google Authentication on Android**

**For issues with Google authentication on Android:**

* Ensure you have configured the Google Services file (google-services.json) in your Android project
* Verify SHA-1 and SHA-256 certificate fingerprints are registered in the Google Developer Console
* Check that the OAuth 2.0 Client ID is correctly set up for Android
* Confirm the package name in the Android manifest matches the Google Developer Console configuration

**Email Authentication Problems**

**If email authentication isn't working properly:**

* Check Supabase logs to ensure emails are being sent correctly
* Verify that email templates are configured properly in the Supabase dashboard
* Check that email confirmation settings match your application's requirements
* Look for email deliverability issues - emails might be going to spam folders
* Ensure your Supabase project has email sending capabilities enabled

**Database Setup Issues**

**If the database tables weren't created properly:**

* Manually run each part of the SQL script in the SQL Editor
* Check for error messages in the SQL Editor console
* Verify that you have the right permissions to create tables in your Supabase project
* Ensure the RLS policies are correctly configured for each table
* Check that table relationships and foreign keys are properly defined

**Common Error Codes and Solutions**

| Error Code | Description | Solution |
| --- | --- | --- |
| 401 | Unauthorized | Check JWT token validity and refresh token if needed |
| 403 | Forbidden | Verify RLS policies allow the operation for the user |
| 422 | Validation Error | Check input data format and required fields |
| 500 | Server Error | Check Supabase logs for details on the server-side error |

**10. Additional Resources**

**Documentation**

**Supabase Documentation** - Official docs for Supabase authentication and database services

**Flutter Documentation** - Comprehensive guides for Flutter development

**Supabase Flutter SDK** - Integration guide for Supabase with Flutter

**Development Resources**

**Google Sign-In for Flutter** - Package for implementing Google authentication

**Flutter GitHub Auth** - Package for GitHub authentication in Flutter

**Supabase Flutter** - Official Supabase package for Flutter integration

Useful Tools

**Supabase Dashboard** - Manage your Supabase project, database, and authentication settings

**Google Cloud Console** - Configure OAuth credentials for Google Sign-In

**GitHub Developer Settings** - Configure OAuth apps for GitHub authentication

**Flutter DevTools** - Performance and debugging tools for Flutter applications