Supabase Dev Workflow Summary (Final)

1. Project Context
You’re building a Casino App (App A) with a Supabase backend (DB A), plus a separate Admin App (App B). For development, there’s also a Dev DB (DB B). Both apps share the same DB for their respective environments.

2. Original Workflow
Plan: clone DB A → create DB B → develop App B → merge App B to main → DB B becomes live DB → repeat.
Problem: Supabase prevents cloning a DB that’s itself a clone.

3. Identified Challenges
- Supabase does not support nested DB cloning.
- CLI limitations (IPv6-only) blocked direct tooling.
- Manual copying of data risky once live user data exists.

4. Adopted Solution: Migration-Based Workflow
We now use schema migrations to manage DB changes.
Key steps:
1. Export live schema (DB A) with pg_dump (schema-only).
2. Create fresh Supabase project (DB B) and apply schema.
3. Connect Dev Apps to DB B and develop features using fake seed data.
4. After development, use diff tools (or manual compare) to generate a migration file.
5. Test migration on staging DB (clone of live).
6. Apply migration to live DB (DB A) once verified.

Benefits:
- Safe, repeatable process.
- Avoids limitations of Supabase clone.
- Preserves live user data integrity.

5. IPv4 Enablement
Supabase IPv4 was enabled for all DBs to allow:
- Direct Postgres access (psql, pg_dump, etc.)
- Schema diffs and migration creation outside CLI tunnels.

6. Staging Workflow (Finalized)
Before applying to live, we test migrations:
1. Restore live DB to a staging project (DB C).
2. Apply migration to DB C.
3. Test both apps against DB C.
4. If successful, apply migration to live DB A.

7. Current State (Post Migration)
■ Migration tested on staging DB.
■ Applied successfully to live DB (DB A).
■ Casino App verified working with live DB.
■ Dev branches still point to dev DB for now.

8. Future Dev Cycle
For new features:
1. Create new dev DB from live schema.
2. Point dev branches to it.
3. After development, create and test migration as above.
4. Apply to live DB and update apps.

5. Clean up old dev DB.

9. Data Considerations
- Pre-live: optional data copy to dev DB.
- Post-live: use fake/anonymized data in dev DBs.

This process ensures a robust, repeatable workflow for future development and deployment.