# 🗄️ Wheels & Wins – Pam 2.0 Schema Playbook

This playbook defines all **Supabase schema changes** required for Pam 2.0. Apply changes **only after completing the relevant build phase**. Keep migrations small, incremental, and documented.

## 📌 General Rules

1. Always test schema changes in **staging first**.
2. Keep existing schema intact; only **add** new tables/columns.
3. Apply changes via **SQL migration scripts**, checked into repo.
4. Use **Row Level Security (RLS)** from the start.
5. One feature = one migration file.

## 📌 Phase 2 – Conversational Engine

**Table: pam\_messages**

CREATE TABLE IF NOT EXISTS pam\_messages (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,  
 session\_id UUID,  
 role TEXT CHECK (role IN ('user','pam')),  
 content TEXT,  
 metadata JSONB,  
 created\_at TIMESTAMPTZ DEFAULT NOW()  
);  
  
ALTER TABLE pam\_messages ENABLE ROW LEVEL SECURITY;  
CREATE POLICY "Users can read own messages" ON pam\_messages  
 FOR SELECT USING (auth.uid() = user\_id);  
CREATE POLICY "Users can insert own messages" ON pam\_messages  
 FOR INSERT WITH CHECK (auth.uid() = user\_id);

## 📌 Phase 3 – Context Manager

**Table: pam\_sessions**

CREATE TABLE IF NOT EXISTS pam\_sessions (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,  
 context JSONB,  
 updated\_at TIMESTAMPTZ DEFAULT NOW()  
);  
  
ALTER TABLE pam\_sessions ENABLE ROW LEVEL SECURITY;  
CREATE POLICY "Users can read own sessions" ON pam\_sessions  
 FOR SELECT USING (auth.uid() = user\_id);  
CREATE POLICY "Users can insert own sessions" ON pam\_sessions  
 FOR INSERT WITH CHECK (auth.uid() = user\_id);  
CREATE POLICY "Users can update own sessions" ON pam\_sessions  
 FOR UPDATE USING (auth.uid() = user\_id);

## 📌 Phase 4 – Passive Trip Logger (Wheels)

**Table: trips**

CREATE TABLE IF NOT EXISTS trips (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,  
 start TIMESTAMPTZ,  
 end TIMESTAMPTZ,  
 route JSONB,  
 stops JSONB,  
 created\_at TIMESTAMPTZ DEFAULT NOW()  
);  
  
ALTER TABLE trips ENABLE ROW LEVEL SECURITY;  
CREATE POLICY "Users can read own trips" ON trips  
 FOR SELECT USING (auth.uid() = user\_id);  
CREATE POLICY "Users can insert own trips" ON trips  
 FOR INSERT WITH CHECK (auth.uid() = user\_id);

## 📌 Phase 5 – Savings Tracker (Wins)

**Table: pam\_savings**

CREATE TABLE IF NOT EXISTS pam\_savings (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,  
 month DATE NOT NULL,  
 total\_saved NUMERIC NOT NULL DEFAULT 0,  
 free\_month BOOLEAN DEFAULT FALSE,  
 created\_at TIMESTAMPTZ DEFAULT NOW()  
);  
  
ALTER TABLE pam\_savings ENABLE ROW LEVEL SECURITY;  
CREATE POLICY "Users can read own savings" ON pam\_savings  
 FOR SELECT USING (auth.uid() = user\_id);  
CREATE POLICY "Users can insert own savings" ON pam\_savings  
 FOR INSERT WITH CHECK (auth.uid() = user\_id);  
CREATE POLICY "Users can update own savings" ON pam\_savings  
 FOR UPDATE USING (auth.uid() = user\_id);

## 📌 Phase 6 – Safety Layer

**Table: safety\_events**

CREATE TABLE IF NOT EXISTS safety\_events (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,  
 event\_type TEXT,  
 details JSONB,  
 created\_at TIMESTAMPTZ DEFAULT NOW()  
);  
  
ALTER TABLE safety\_events ENABLE ROW LEVEL SECURITY;  
CREATE POLICY "Users can read own safety events" ON safety\_events  
 FOR SELECT USING (auth.uid() = user\_id);  
CREATE POLICY "Users can insert own safety events" ON safety\_events  
 FOR INSERT WITH CHECK (auth.uid() = user\_id);

## 📌 Phase 7 – Testing Tables (Optional)

**Table: pam\_test\_data** (for staging only)

CREATE TABLE IF NOT EXISTS pam\_test\_data (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 user\_id UUID,  
 test\_name TEXT,  
 payload JSONB,  
 created\_at TIMESTAMPTZ DEFAULT NOW()  
);

# ✅ End State

* Messages, sessions, trips, savings, and safety events tracked.
* All tables secure with RLS.
* Schema changes incremental, easy to extend.
* Future modules can add their own tables without clutter.