🗄️ 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.