🚀 pg-schemata Quick Start Cheatsheet

1. Connect to the Database

```
const pgp = require('pg-promise')();
const db = pgp({ /* connection config */ });
```

✅ One global connection pool for all tenants.

2. Define a Model Schema

```
const userSchema = {
  schema: 'public',
  table: 'users',
  columns: [
    { name: 'id', type: 'uuid', default: 'gen_random_uuid()', notNull:
true, immutable: true },
    { name: 'tenant_id', type: 'uuid', notNull: true, immutable: true },
    { name: 'email', type: 'text', notNull: true },
    { name: 'password', type: 'text', notNull: true },
    { name: 'created_at', type: 'timestamp', default: 'now()', immutable:
true }
  ],
  constraints: {
    primaryKey: ['id'],
    unique: [['tenant_id', 'email']],
    indexes: [{ columns: ['email'] }]
  }
};
```

3. Create a Model

```
const BaseModel = require('pg-schemata').BaseModel;

class User extends BaseModel {
  constructor(db) {
    super(db, userSchema);
  }
}
```

✅ Inherit from `BaseModel` for automatic CRUD and schema switching.

4. Switch Tenant Schema Dynamically

```
const userModel = new User(db);

// Set schema at runtime based on user's organization
userModel.setSchema('org_abc');
// or chain it:
const user = await userModel.withSchema('org_abc').findById('uuid');
```

5. Use Basic CRUD Read Methods

```
await userModel.findById('uuid');
await userModel.findAll({ limit: 20 });
await userModel.findOneBy('email', 'user@example.com');
await userModel.findManyBy('tenant_id', 'tenant-uuid');
await userModel.findWithCursor('last-id', 20);
```

6. Auto-Create Schema and Tables for New Tenants

```
await db.none('CREATE SCHEMA IF NOT EXISTS "org_abc"');

const createTableSQL = createTableSQL(userSchema);
await db.none(createTableSQL.replace('public', 'org_abc'));
```

✅ Create schema + create tables dynamically for each new tenant.

7. Important Best Practices

- Always include `tenant_id` in every table
- Always use UUIDs for `id` and `tenant_id`
- Always use ColumnSets for insert/update
- Mark `immutable: true` on critical fields (`id`, `tenant_id`, `created_at`)
- Optionally create database triggers to enforce immutability

🧠 Quick Reference

Action	Method
Change tenant schema	`.setSchema('schemaName')`

Action	Method
Chain schema + query	`.withSchema('schemaName').findById(id)`
Insert safely	`pgp.helpers.insert(data, insertColumnSet)`
Update safely	`pgp.helpers.update(data, updateColumnSet)`
Cursor-based pagination	`.findWithCursor(afterId, limit)`
Create schema if needed	`CREATE SCHEMA IF NOT EXISTS "schemaName";`

✅ You're now fully armed to build multi-tenant SaaS apps with **pg-schemata**!