PostgreSQL Notes for Laravel Projects

1. Basic Setup

- Install PostgreSQL and ensure the service is running.

- In Laravel's .env, set DB_CONNECTION=pgsql and configure DB_HOST, DB_PORT (usually 5432), DB_DATABASE, DB_USERNAME, DB_PASSWORD.

2. Key Differences vs MySQL

- JSON fields use jsonb for better indexing.

- UUIDs are commonly used with default uuid_generate_v4().

- Timestamps default precision differs; migration adjustments may be needed.

3. Migrations and Schema

- Use $table->jsonb('column') for JSON.

- For UUID primary keys: $table->uuid('id')->primary();

- Enum types should be replaced with check constraints or text with validation.

4. Importing a MySQL Dump into PostgreSQL

Steps:

1. Clean the MySQL dump:

- Remove MySQL-specific syntax (ENGINE, CHARSET, COLLATE, etc.).

- Replace backticks (`) with double quotes (") or remove them.

2. Convert data types:

- INT → INTEGER

- TINYINT(1) → BOOLEAN

- DATETIME → TIMESTAMP

3. Use a tool to automate conversion:

- pgloader mysql://user:pass@host/db postgresql://user:pass@host/db

4. Import:

- With pgloader: it handles schema + data import automatically.

- Manual import via psql requires fully converted SQL.

5. Working with PostgreSQL in Eloquent

- Use $casts for jsonb fields.

- Use ->selectRaw, ->whereRaw carefully due to quoting differences.

- PostgreSQL uses ILIKE for case-insensitive matching.

6. Helpful Tips

- Enable extensions in migrations: DB::statement('CREATE EXTENSION IF NOT EXISTS "uuid-ossp";');

- Use indexing on jsonb with: DB::statement('CREATE INDEX ... USING gin(column);');

- Be mindful of reserved keywords (more strict than MySQL).

This cheat sheet covers essential concepts to start using PostgreSQL effectively in Laravel.