

# Database Setup Guide

This guide covers database configuration and schema setup for the BrazilRetail-BI project.

## Prerequisites

- PostgreSQL 12+
- Database user with creation privileges

## Step 1: Create Database

Connect to PostgreSQL and create the database:

```
CREATE DATABASE brazilretail_bi;
```

## Step 2: Create Database User (Optional)

Create a dedicated user for the application:

```
CREATE USER brazilretail_user WITH PASSWORD 'your_password';  
GRANT ALL PRIVILEGES ON DATABASE brazilretail_bi TO  
brazilretail_user;
```

## Step 3: Configure Environment

Update the .env file in the project root:

```
# Local PostgreSQL  
  
DATABASE_URL=postgresql://username:password@localhost:5432/brazilretail_bi  
# Production Supabase  
SUPABASE_URL=https://your-project.supabase.co  
SUPERKEY=your-service-role-key
```

Replace with your actual credentials.

## Step 4: Create Schema

Run the schema creation script (works for both Local and Supabase if configured):

```
python db_schema/create_schema.py
```

This creates all tables:

- customers
- geolocation
- orders
- order\_items
- order\_payments
- order\_reviews
- products
- sellers

## Step 5: Setup Constraints and Indexes

After creating the schema, run the constraints script to add Foreign Keys and Indexes. This is crucial for performance and data integrity.

```
python db_schema/setup_constraints.py
```

## Database Schema Overview

### Core Tables

#### customers

- customer\_id (PK)
- customer\_unique\_id
- customer\_zip\_code\_prefix
- customer\_city
- customer\_state
- customer\_state\_initials

#### orders

- order\_id (PK)
- customer\_id (FK)
- order\_status
- order\_purchase\_timestamp
- order\_approved\_at
- order\_delivered\_carrier\_date
- order\_delivered\_customer\_date
- order\_estimated\_delivery\_date

#### order\_items

- order\_item\_id (PK)
- order\_id (FK)
- product\_id (FK)
- seller\_id (FK)
- shipping\_limit\_date
- price
- freight\_value

#### products

- product\_id (PK)
- product\_category\_name
- product\_category\_name\_english
- product\_name\_lenght
- product\_description\_lenght
- product\_photos\_qty
- product\_weight\_g
- product\_length\_cm
- product\_height\_cm
- product\_width\_cm

## Supporting Tables

### **sellers**

- seller\_id (PK)
- seller\_zip\_code\_prefix
- seller\_city
- seller\_state
- seller\_state\_initials

### **order\_payments**

- order\_id (PK)
- payment\_sequential (PK)
- payment\_type
- payment\_installments
- payment\_value

### **order\_reviews**

- review\_id (PK)
- order\_id (FK)
- review\_score
- review\_comment\_title
- review\_comment\_message
- review\_creation\_date
- review\_answer\_timestamp

### **geolocation**

- geolocation\_zip\_code\_prefix (PK)
- geolocation\_lat
- geolocation\_lng
- geolocation\_city
- geolocation\_state
- geolocation\_state\_initials

## Data Loading

After schema creation, run the ETL pipeline to load data:

```
python etl/main.py
```

The pipeline will extract, transform, and load all datasets into the database.

## Verification

Connect to the database and verify data loading:

```
SELECT COUNT(*) FROM customers;
SELECT COUNT(*) FROM orders;
-- Check other tables
```

## Backup and Maintenance

Regular backups are recommended:

```
pg_dump brazilretail_bi > brazilretail_backup.sql
```

## Troubleshooting

- **Connection errors:** Verify DATABASE\_URL format and credentials
- **Permission denied:** Ensure user has proper database privileges
- **Schema creation fails:** Check if tables already exist (drop them first if needed)
- **Data loading errors:** Check CSV files are present and ETL logs for details