

BrazilRetail-BI: Brazilian E-Commerce Business Intelligence System

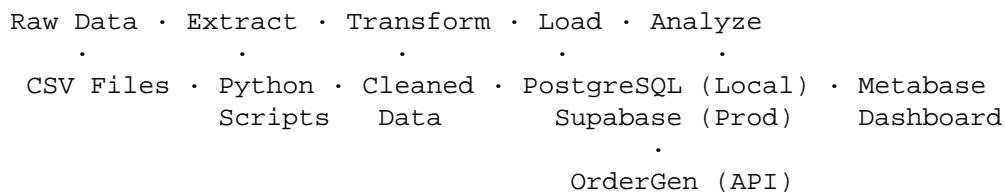
A complete ETL pipeline and business intelligence system for Brazilian e-commerce data analysis.

Status: **FULLY OPERATIONAL** - ETL pipeline successfully executed on November 17, 2025 **Data Loaded:** 32,340 products, 3,095 sellers, and all related datasets

Overview

This system processes the Brazilian E-Commerce Public Dataset by Olist, providing a clean, transformed, and analyzed dataset for business intelligence and data science applications.

Architecture



Key Features

- **Dual ETL Environments:** Local (PostgreSQL) and Production (Supabase)
- **Incremental Updates:** Production ETL supports incremental data loading
- **Order Generator:** Synthetic data generation trained on real data patterns
- **API Service:** FastAPI-based backend for triggering ETL and OrderGen tasks
- **Dockerized:** Production backend containerized for easy deployment
- **Complete ETL Pipeline:** Extract, transform, and load 8 datasets
- **Data Quality:** Robust cleaning, type conversion, and validation
- **Multilingual Support:** Portuguese to English category translation
- **Modular Design:** Separate concerns for maintainability
- **Error Handling:** Comprehensive logging and failure recovery
- **Idempotent Operations:** Safe re-runs with full reload option
- **Production Ready:** Successfully tested with real data loads

Quick Start

1. Environment Setup

```
# Install dependencies
pip install -r requirements.txt
# Configure database
# Edit .env file with your PostgreSQL/Supabase credentials
```

2. Data Acquisition

```
# Download datasets (see docs/dataset_setup.md)
kaggle datasets download -d olistbr/brazilian-ecommerce
```

```
unzip brazilian-e-commerce.zip  
mv *.csv data/
```

3. Run with Docker (Recommended for Prod)

```
# Build and run the backend container  
.run_docker.sh  
# Trigger Order Generation via API  
curl -X POST http://localhost:8000/orders/generate -H  
"Content-Type: application/json" -d '{"count": 10}'
```

4. Local Development

```
# Create database schema  
python db_schema/create_schema.py  
# Run Local ETL pipeline  
python -m etl_local.main
```

5. Analysis

Connect Metabase or your preferred BI tool to the PostgreSQL/Supabase database for analysis.

ETL Pipeline Details

Extract Phase

- Robust CSV reading with automatic encoding detection
- Handles UTF-8, Latin-1, CP1252, and ISO-8859-1 encodings
- Validates file existence and basic structure

Transform Phase

- **Data Type Conversion:** Ensures proper types for all columns
- **Text Cleaning:** City names title-cased, consistent formatting
- **State Mapping:** Brazilian states mapped to full names with initials preserved
- **Category Translation:** Product categories translated from Portuguese to English
- **Datetime Handling:** Proper timezone removal and format standardization
- **Data Validation:** Removes rows with excessive missing values, eliminates duplicates

Load Phase

- Bulk insertion using SQLAlchemy for performance
- Transaction management with rollback on errors
- Schema validation before loading
- Full reload capability with table truncation

Dataset Schema

Core Entities


```

.   ... __init__.py    # Package initialization
.   ... main.py        # ETL orchestration
.   ... extract.py    # Data extraction
.   ... transform/     # Transformation modules
.   ...   ... __init__.py
.   ...   ... customers.py
.   ...   ... orders.py
.   ...   ... products.py
.   ...   ... [other transforms]
.   ... load.py        # Load coordination
.   ... utils.py       # Logging utilities
... db_schema/         # Database layer (Python package)
.   ... __init__.py    # Package initialization
.   ... create_schema.py # SQLAlchemy table definitions
.   ... dbmanip.py     # Bulk data operations
... dashboard/         # Metabase dashboard configurations (planned)
... docs/              # Documentation
.   ... system_overview.md
.   ... etl_setup.md
.   ... dataset_setup.md
.   ... database_setup.md
... .env                # Environment configuration
... .gitignore          # Git ignore rules
... requirements.txt    # Python dependencies
... README.md          # Project documentation

```

Performance Characteristics

- **Extract:** ~30 seconds for all datasets
- **Transform:** ~2-3 minutes with category translation
- **Load:** ~1-2 minutes bulk insertion (tested with 32,340 products + 3,095 sellers)
- **Total:** ~4-6 minutes end-to-end (verified November 17, 2025)
- **Memory Usage:** Efficient pandas processing with chunked operations
- **Scalability:** Handles full dataset reloads safely

Error Handling

- **Schema Validation:** Checks database readiness before loading
- **Transaction Rollback:** Failed loads don't corrupt data
- **Encoding Detection:** Automatic fallback for problematic files
- **Logging:** Comprehensive success/error reporting
- **Import Resolution:** Proper Python package structure for reliable execution

Future Enhancements

- · **ETL Pipeline:** Complete and operational
- · **Data Loading:** Successfully tested with real data
- · **Incremental Loading:** Change detection for new data
- · **Metabase Dashboard:** Dashboard creation and configuration
- .. **Cloud Deployment:** Docker/Kubernetes containerization
- · **Monitoring:** Pipeline health and performance metrics

- **API Integration**: Real-time data ingestion capabilities

System Status

· Completed Features

- Full ETL pipeline with 8 dataset processing
- PostgreSQL database schema with proper relationships
- Data quality transformations (cleaning, translation, validation)
- Error handling and transaction management
- Comprehensive logging and monitoring
- Modular Python package structure
- Successful production data load (32,340+ records)

· In Progress / Planned

- Metabase dashboard development
- Docker containerization
- Automated testing suite
- Performance monitoring
- Incremental loading capabilities

Contributing

- 1 Follow the modular architecture with proper Python packages
- 2 Add tests for new transformations (testing framework planned)
- 3 Update documentation for schema changes
- 4 Ensure backward compatibility
- 5 Test ETL pipeline execution before committing changes

Testing & Validation

· Production Testing Completed

- **Date:** November 17, 2025
- **Command:** `python -m etl.main`
- **Result:** SUCCESS - All datasets loaded successfully
- **Records:** 32,340 products, 3,095 sellers, plus all related data
- **Duration:** ~4-6 minutes end-to-end
- **Import Issues:** Resolved through proper package structure

· Recommended Testing

```
# Test ETL execution
python -m etl.main
# Test full reload capability
python -m etl.main --full-reload
# Validate database contents
# Connect to PostgreSQL and verify table counts
```

License

This project uses the Brazilian E-Commerce Public Dataset. Check Kaggle for licensing terms.