User Guide

Group 1 - BID3000 Date: October 2025

Table of Contents

---

1. Introduction

This guide accompanies group 1 home exam,it provides process explanations, and setup instructions for Reproducibility.

- **Data Warehouse:** Star schema dimensional modeling implemented in PostgreSQL
- **ETL Pipeline:** Pentaho Data Integration (PDI)
- **Analytics:** Descriptive, Predictive, and Prescriptive analysis using Python
- **Visualization:** Power BI dashboard

---

2. Prerequisites

Required Software

1. **PostgreSQL**
2. **Pentaho Data Integration (PDI)**
3. **Python**
4. **IDE or Text Editor with Python Support**
5. **Power BI Desktop**
6. **Source Data Files:** CSV files containing Olist e-commerce data

---

3. System Architecture Overview

```
┌─────────────────┐
│   Source Data   │
│   (CSV Files)   │
└────────┬────────┘
         │
         ▼
┌────────────────┐
│  ETL Process   │
│  (Pentaho PDI) │
└────────┬───────┘
         │
         ▼
┌────────────────┐
│  Data Warehouse│
│  (PostgreSQL)  │
└────────┬───────┘
         │
         ├──────────────────┬──────────────────┐
         ▼                  ▼                  ▼
┌────────────────┐   ┌────────────┐   ┌────────────┐
│  Analytical    │   │   Python   │   │   Power BI │
│  SQL Queries   │   │  Analytics │   │  Dashboard │
└────────────────┘   └────────────┘   └────────────┘
```

Data Warehouse Schema

Star Schema with:

Dimension Tables (7):

- `dim_date` - Date dimension with calendar hierarchies
- `dim_geography` - Geographic locations (zip codes, cities, states, regions)
- `dim_customer` - Customer information (SCD Type 2)
- `dim_product` - Product catalog with categories and attributes
- `dim_seller` - Seller information and locations
- `dim_payment_type` - Payment methods
- `dim_order_status` - Order status categories

Fact Tables (3):

- `fact_sales` - Sales transactions (grain: order item level)
- `fact_delivery_performance` - Delivery metrics (grain: order level)
- `fact_customer_reviews` - Customer reviews and ratings (grain: review level)

---

4. Implementation Guide

4.1 Database Setup

For your convience we show the setup in PostgreSQL both with psql and pgAdmin 4: **Option A** using psql command line (if you have it insalled) **Option B** using pgAdmin 4 GUI

Option A: Using psql (Command Line)

Step 1: Open psql Command Line

On Windows:

1. Open Command Prompt or PowerShell
2. Run:

   ```
   psql -U postgres
   ```

3. Enter your PostgreSQL password when prompted

On Linux/Mac:

1. Open Terminal
2. Run:

   ```
   psql -U postgres
   ```

3. Enter your password when prompted

Step 2: Create Database

```
CREATE DATABASE olist_dw;
```

You should see: `CREATE DATABASE`

Step 3: Connect to the Database

```
\c olist_dw
```

You should see: `You are now connected to database "olist_dw" as user "postgres".`

Step 4: Execute Schema Creation Script

Method 1: Using \i command (with your absolute path)

```
\i 'C:/itis/Group1_BID3000_2025/Database/schema_creation.sql'
```

Notes:

- Use **forward slashes (/)**
- Use the **absolute path** to your schema_creation.sql file
- Keep single quotes around the path

Method 2: Copy and paste SQL

1. Open `schema_creation.sql` in a text editor
2. Copy all the SQL content
3. Paste it into the psql prompt
4. Press Enter to execute

Step 5: Verify Table Creation

```
-- to see a table of all the tables
\dt
```

OR

```sql
-- To see a list of tables
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public'
ORDER BY table_name;
```

Expected Output: You should see 10 tables listed:

- dim_customer
- dim_date
- dim_geography
- dim_order_status
- dim_payment_type
- dim_product
- dim_seller
- fact_customer_reviews
- fact_delivery_performance
- fact_sales

Option B: Using pgAdmin 4 (Graphical Interface)

Step 1: Open pgAdmin 4

1. Launch **pgAdmin 4**
2. Enter password if needed
3. In the left panel, expand **Servers** → **PostgreSQL**

Step 2: Create Database

1. **Right-click on "Databases"** → Select **"Create"** → **"Database..."**
2. In the **"Create - Database"** dialog:

   - **Database:** Enter `olist_dw`
   - **Owner:** Select your PostgreSQL user (usually `postgres`)
3. Click **"Save"**
4. The new database `olist_dw` should appear in the databases list

Step 3: Open Query Tool

1. **Expand "Databases"** in the left panel
2. **Click on "olist_dw"** to select it
3. **Click the "Query Tool" icon** in the toolbar (or right-click → Query Tool)
4. A new Query Tool window opens

Step 4: Load and Execute Schema Creation Script

1. In the Query Tool, click **"Open File"** icon (folder icon) or press **Ctrl+O**
2. **Navigate to the path of your schema_creation.sql file:**

```
C:/Group1_BID3000_2025/Database/schema_creation.sql
```

3. **Select the file** and click **"Open"**
4. The SQL script will load into the Query Tool editor
5. **Click the "Execute" button** (play icon)
6. **Wait for execution** to complete

Step 5: Verify Table Creation

1. In the **Query Tool**, clear the current query
2. **Enter the following query:**

```sql
-- Check that all tables were created
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public'
ORDER BY table_name;
```

3. **Execute the query** or click (F5)
4. **Review the output**

Expected Output: You should see 10 tables listed:

- dim_customer
- dim_date
- dim_geography
- dim_order_status
- dim_payment_type
- dim_product

- dim_seller
- fact_customer_reviews
- fact_delivery_performance
- fact_sales

Alternative Verification (Visual):

1. In the left Browser panel, expand: **Databases** → **olist_dw** → **Schemas** → **public** → **Tables**
2. You should see all 10 tables listed
3. Right-click any table → **"Properties"** to view details

Expected Result: All dimension and fact tables are created with their constraints and indexes.

4.2 ETL Process Configuration

Our ETL process uses Pentaho Data Integration (PDI) to load data from CSV files into the data warehouse.

Step 1: Open Pentaho Data Integration

1. **Launch PDI**
    - Windows: Navigate to PDI installation folder → Run `Spoon.bat` or search for Pentaho Data Integration in the search bar
    - Linux/Mac: Run `spoon.sh`
2. **Wait for the application to start** (can take a few minutes)

Step 2: Configure Database Connection

1. **Open the .ktr files one at a time** from the ETL folder
2. **In the left panel, right-click on database connections** → Select your PostgreSQL connection → Edit
3. **Update connection parameters:**
    - **Connection Name:** Keep as configured
    - **Connection Type:** PostgreSQL
    - **Access:** Native (JDBC)
    - **Host Name:** `localhost` (or your PostgreSQL server address)
    - **Database Name:** `olist_dw`
    - **Port Number:** `5432` (default PostgreSQL port)
    - **Username:** Your PostgreSQL username (e.g., `postgres`)
    - **Password:** Your PostgreSQL password
4. **Test Connection** → Click "Test" button to verify connectivity
5. **Save** the connection settings

Step 3: Configure CSV File Paths

For each .ktr file, you need to update the CSV file input paths:

Transformation Files to Configure:

1. `tr_load_dim_date.ktr`
2. `tr_load_dim_geography.ktr`
3. `tr_load_dim_customer.ktr`
4. `tr_load_dim_product.ktr`
5. `tr_load_dim_seller.ktr`
6. `tr_load_dim_payment_type.ktr`
7. `tr_load_dim_order_status.ktr`
8. `tr_load_fact_sales.ktr`
9. `tr_load_fact_delivery_performance.ktr`
10. `tr_load_fact_customer_reviews.ktr`

For each transformation:

1. **Open the .ktr file** in PDI

2. **Find the CSV Input step** we placed them mostly on the top left area

3. **Double-click the step** to open properties and see which file is used prior to updating the file names are kept in their original form as downloaded from kaggle for easy access locally.

4. **Update the file path:**

   - Click **Browse** button
   - Navigate to your CSV source data location
   - Select the correct CSV file for this transformation
   - Click **OK**

5. **Verify field mappings: This step is NOT recommended as we filter fields in most ETL's**

   - Click **Get Fields** button (if needed)
   - Ensure field names match the CSV file

6. **Save the transformation** (Ctrl+S or File → Save)

Step 4: Configure Table Lookups

For fact table transformations, you need to configure database lookups:

Fact tables with lookups:

- `tr_load_fact_sales.ktr`
- `tr_load_fact_delivery_performance.ktr`
- `tr_load_fact_customer_reviews.ktr`

For each fact table transformation:

1. **Open the transformation** in PDI

2. **Find lookup steps** (e.g., "Database lookup", "Dimension lookup/update")

3. **For each lookup step:**
   - Double-click to open properties
   - **Connection:** Verify it points to your PostgreSQL connection
   - **Lookup schema:** `public`
   - **Lookup table:** Verify the correct dimension table name
   - **Keys:** Verify lookup key fields match
   - **Return values:** Verify return fields are correct
4. **Save the transformation**

Step 5: Execute ETL Transformations

Important: Load dimensions first,and then the fact tables

Recommended Execution Order:

1. **Load Dimension Tables:**

   ```
   tr_load_dim_date.ktr
   tr_load_dim_geography.ktr
   tr_load_dim_payment_type.ktr
   tr_load_dim_order_status.ktr
   tr_load_dim_product.ktr
   tr_load_dim_seller.ktr
   tr_load_dim_customer.ktr
   ```

2. **Load Fact Tables:**

   ```
   tr_load_fact_sales.ktr
   tr_load_fact_delivery_performance.ktr
   tr_load_fact_customer_reviews.ktr
   ```

To execute each transformation:

1. **Open the .ktr file** in PDI
2. **Click the Run button** green play icon
3. **Configure run settings:**
   - Logging level: Basic or Detailed
4. **Click Launch**
5. **Monitor execution:**
   - Watch the Execution Results tab at bottom
   - Green check marks indicate successful steps
   - Red X marks indicate errors
6. **Verify row counts** in the execution log

Alternative: Use the Job File

You can execute all transformations in sequence using the job file:

1. **Open:** `extract_transform_load.kjb`
2. **Verify:** All transformation paths are correct
3. **Run the job**
4. **Monitor:** Job will execute all transformations in proper order

Step 6: Verify Data Load

Verify data in PostgreSQL:

```sql
-- Check record counts
SELECT 'dim_date' as table_name, COUNT(*) as record_count FROM dim_date
UNION ALL
SELECT 'dim_customer', COUNT(*) FROM dim_customer
UNION ALL
SELECT 'dim_product', COUNT(*) FROM dim_product
UNION ALL
SELECT 'dim_seller', COUNT(*) FROM dim_seller
UNION ALL
SELECT 'fact_sales', COUNT(*) FROM fact_sales
UNION ALL
SELECT 'fact_delivery_performance', COUNT(*) FROM fact_delivery_performance
UNION ALL
SELECT 'fact_customer_reviews', COUNT(*) FROM fact_customer_reviews
ORDER BY table_name;
```

Expected Results:

- All tables should have records and without any errors in the output

---

4.3 Analytical Queries Execution

Step 1: Open SQL Query Tool

- **In pgAdmin:** Right-click database → Query Tool
- **In psql:** Connect to `olist_dw` database

Step 2: Execute Analytical Queries

1. **Navigate to:** analytical_queries.sql file

2. **Open the file** in your SQL editor

3. **Execute queries individually:**

   - The file contains 10 analytical queries
   - Each query includes a header comment explaining its purpose

Key Analytical Queries:

Query	Purpose
Query 1	Year-over-Year Revenue Growth
Query 2	Seasonal Pattern Analysis
Query 3	Time Hierarchy Drill-down
Query 4	Geographic Hierarchy Drill-down
Query 5	Customer Revenue Ranking
Query 6	Moving Average Analysis
Query 7	Multi-dimensional Filtering
Query 8	Above Average Customers
Query 9	Customer Profitability Analysis
Query 10	Seller Performance KPIs

4.4 Python Analytics Setup

The Python analytics notebooks perform three types of analysis:

1. **Descriptive Analytics** - Statistical summaries and correlations
2. **Predictive Analytics** - Customer satisfaction prediction using Decision Trees
3. **Prescriptive Analytics** - Delivery route optimization using Linear Programming

Step 1: Create Environment Variable File

1. **Navigate to the Analytics folder:**

   ```
   C:/Group1_BID3000_2025/Analytics/
   ```

2. **edit the `.env`** in this folder

3. **Add your database connection string:**

   ```
   DB_URL=postgresql+psycopg2://username:password@localhost:5432/olist_dw
   ```

   Replace:

   - `username` - Your PostgreSQL username
   - `password` - Your PostgreSQL password
   - `localhost` - Your database host
   - `5432` - PostgreSQL port (default is 5432)

  - `olist_dw` - Your database name

Example:

```
DB_URL=postgresql+psycopg2://postgres:mypassword@localhost:5432/olist_dw
```

Step 2: Install Python Dependencies

Open a terminal in the Analytics folder and run:

```
pip install pandas numpy matplotlib seaborn sqlalchemy psycopg2-binary python-dotenv scipy scikit-learn pulp
```

Package Breakdown by Notebook:

For all notebooks:

```
pip install pandas numpy matplotlib seaborn sqlalchemy psycopg2-binary python-dotenv
```

Additional for descriptive_analysis.ipynb:

```
pip install scipy
```

Additional for predictive_analysis.ipynb:

```
pip install scikit-learn
```

Additional for prescriptive_analysis.ipynb:

```
pip install pulp
```

Complete installation command (all packages):

```
pip install pandas numpy matplotlib seaborn sqlalchemy psycopg2-binary python-dotenv scipy scikit-learn pulp
```

Step 3: Open Notebooks

1. **Launch your IDE or Jupyter:**
   - **Visual Studio Code:** Open folder → Open .ipynb file
   - **Jupyter Notebook:** Run `jupyter notebook` in terminal
   - **JupyterLab:** Run `jupyter lab` in terminal
2. **Select Python kernel** (if prompted)

Step 4: Execute Analytics Notebooks

A. Descriptive Analysis

File: `descriptive_analysis.ipynb` Statistical summaries, correlation analysis, and data exploration

- Open the notebook and run all cells sequentially

Outputs Generated:

- `descriptive_exports/statistical_summary.csv`
- `descriptive_exports/correlation_analysis.csv`
- `descriptive_exports/correlation_significance.csv`
- `descriptive_exports/monthly_analytics.csv`
- `descriptive_exports/monthly_geography_analytics.csv`
- `descriptive_exports/dim_geography.csv`
- `descriptive_exports/correlation_matrix_latest.png`

Key Insights:

- Statistical distributions of business metrics
- Correlation between revenue, orders, and customer behavior
- Geographic performance patterns
- Seasonal trends

B. Predictive Analysis

File: `predictive_analysis.ipynb` Predict customer satisfaction using Decision Tree Classifier

- Open the notebook and run all cells sequentially

Outputs Generated:

- `predictive_exports/satisfaction_model_performance.csv`
- `predictive_exports/satisfaction_feature_importance.csv`
- `predictive_exports/satisfaction_confusion_matrix.csv`
- `predictive_exports/satisfaction_predictions_test_set.csv`
- `predictive_exports/satisfaction_business_metrics.csv`
- `predictive_exports/satisfaction_regional_analysis.csv`
- `predictive_exports/customer_satisfaction_analysis_*.png`
- `predictive_exports/confusion_matrix_satisfaction_*.png`
- `predictive_exports/cross_validation_results_*.png`
- `predictive_exports/feature_importance_satisfaction_*.png`

- `predictive_exports/decision_tree_satisfaction_*.png`

Key Results:

- Model accuracy: ~75-80%
- Main satisfaction factors are: delivery performance, freight costs
- Customer segmentation by satisfaction risk

C. Prescriptive Analysis

File: `prescriptive_analysis.ipynb` Optimize delivery routes using Linear Programming

- Open the notebook and run all cells sequentially

Outputs Generated:

- `prescriptive_exports/delivery_optimization_summary.csv`
- `prescriptive_exports/delivery_optimal_routes.csv`
- `prescriptive_exports/delivery_warehouse_performance.csv`
- `prescriptive_exports/delivery_cost_matrix.csv`
- `prescriptive_exports/delivery_implementation_roadmap.csv`
- `prescriptive_exports/delivery_impact_projections.csv`
- `prescriptive_exports/delivery_optimization_dashboard.png`

Key Recommendations:

- Optimal warehouse-to-customer assignments
- Expected cost savings: 7-8%
- Delivery time reduction projections
- Potential roadmap

Step 5: Review Export Files

All outputs are saved at the export folders:

- `Analytics/descriptive_exports/`
- `Analytics/predictive_exports/`
- `Analytics/prescriptive_exports/`

4.5 Power BI Dashboard

Step 1: Open: `BI_Dashboard.pbix`

Step 2: Explore Dashboard

Dashboard Features:

- **Executive Summary:** KPIs, Monthly revenue trend, Top performing states by revenue, current vs target seasonal index

- **Operational Dashboard:** Orders vs. Revenue Correlation, Customer Acquisition Trend, Region Product Performance, Revenue Per Customer
- **Analytical Deep-Dive:** Key measures, Revenue trend (3 month rolling), Volume vs. Value by Region, Freight vs. Revenue & Rate, Route cost breakdown, Seller & Customer Region, Drivers of positive reviews
- **What-If Analysis:** Price Elasticity Analysis

Interactive Elements:

- **Slicers:** Filter by date and region
- **Cross-filtering:** Click on charts to filter other visuals
- **Drill-down:** Use hierarchy levels on operational dashboard (Year → Quarter → Month)/(Region → State → City) by clicking on data points inside visualizations or using the arrows on the top right of the visual.
- **Tooltips:** Hover over data points for details

Step 3: Export and Share

To export reports:

1. **File** → **Export to PDF** - For static reports
2. **File** → **Publish to Power BI Service** - For online sharing
3. **Screenshot individual visuals** - For presentations

---

6. Project Structure Reference

```
Group1_BID3000_2025/
│
├── Analytics/                          # Python analytics notebooks
│   ├── descriptive_analysis.ipynb     # Statistical analysis
│   ├── predictive_analysis.ipynb      # ML predictions
│   ├── prescriptive_analysis.ipynb    # Optimization
│   ├── .env                            # Database credentials
│   ├── descriptive_exports/           # Descriptive outputs
│   ├── predictive_exports/            # Prediction outputs
│   └── prescriptive_exports/          # Optimization outputs
│
├── Dashboard/                          # Power BI dashboard
│   ├── BI_Dashboard.pbix              # Main dashboard file
│   └── dashboard_screenshots/         # Dashboard images
│
├── Database/                           # SQL scripts
│   ├── schema_creation.sql            # DW schema DDL
│   └── analytical_queries.sql         # Business queries
│
├── Documentation/                      # Project documentation
│   ├── data_dictionary.pdf            # Field definitions
│   └── ERD.pdf                        # Entity-Relationship Diagram
│
├── ETL/                                # Pentaho transformations
│   ├── extract_transform_load.kjb     # Main job file
│   ├── load_dim_*.ktr              # Dimension loads (7 files)
│   └── load_fact_*.ktr             # Fact loads (3 files)
│
├── Report/                             # Final report
│
└── USER_GUIDE.md                       # This file
```