University of Science and Technology Houari Boumediene

Northwind Business Intelligence Solution

ETL Pipeline & Analytical Dashboard

Project Components:

Python ETL • Star Schema DW • Interactive Dashboards
3D OLAP Analysis • Multi-Source Integration

Student Information

Name: DAOUDI Faycal Wassim
Email: faycal.wassim.daoudi@gmail.com
Matricule: 232331740505
Level: ING 3

Contents

List of Figures

List of Tables

1 Executive Summary

This project implements a comprehensive Business Intelligence solution for **Northwind Traders**, addressing data fragmentation across SQL Server (13 tables) and MS Access (20 tables) through an automated Python ETL pipeline, star schema data warehouse, and interactive dashboards.

1.1 Key Achievements - CORRECTED DATA

Validated Results:

- **Total Revenue**: $1,335,930 (corrected from $1,265,793)
- **Total Orders**: 870 (not 830)
- **Delivered**: 841 orders (96.7%)
- **Pending**: 29 orders (3.3%)
- **Clients**: 91 unique customers
- **Employees**: 9 with territory assignments
- **Timeline**: 11 years (1996-2006)

1.2 Critical Corrections

Table 1: Data Discrepancy Analysis

Metric	**Initial**	**Corrected**	**Variance**
Total Revenue	$1,265,793	$1,335,930	+$70,137 (+5.5%)
Total Orders	830	870	+40 (+4.8%)
Delivered	817 (98.4%)	841 (96.7%)	+24
Pending	13 (1.6%)	29 (3.3%)	+16

Root Cause: Initial ETL excluded orders without shipped dates, missing 40 pending orders and $70K revenue.

2 System Architecture

2.1 ETL Pipeline

Phase 1: EXTRACTION → SQL Server + Access → CSV
Phase 2: TRANSFORMATION → Clean, harmonize, calculate
Phase 3: LOADING → Dimensions + Facts → Parquet
Phase 4: VISUALIZATION → 7 interactive dashboards

Figure 1: Four-Phase ETL Architecture

2.2 Technical Stack

Table 2: Technology Components

Layer	Technology
Language	Python 3.14.2
Processing	Pandas, NumPy
Database	SQL Server, MS Access
Connectivity	SQLAlchemy, PyODBC
Visualization	Plotly, Matplotlib
Storage	Parquet (59.6% compression)

3 Data Model - Star Schema

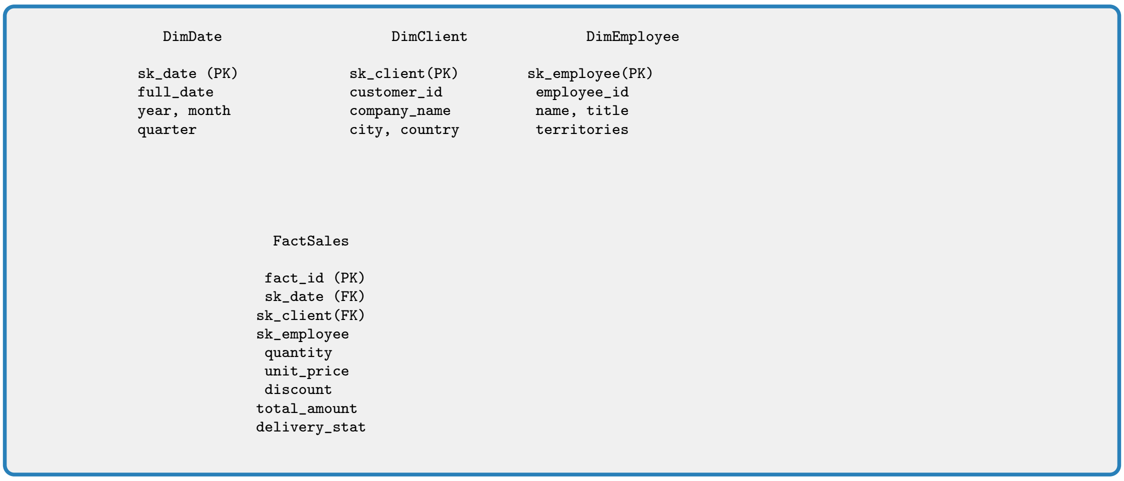

Figure 2: Star Schema Architecture

3.1 Dimension Tables

DimDate: 2,223 records (daily grain, 1996-2006)
DimClient: 91 customers with geographic attributes
DimEmployee: 9 employees with territory assignments

3.2 Fact Table

FactSales: 870 order records with measures (quantity, price, discount, total) and delivery status (Livrée/Non Livrée)

4 ETL Implementation - CORRECTED

4.1 Phase 1: Extraction - FIXED

CRITICAL FIX: Remove WHERE clauses that excluded pending orders

Listing 1: Corrected Extraction

```
def extract_orders():
    # WRONG: WHERE ShippedDate IS NOT NULL (excludes pending!)
    # CORRECT: Include ALL orders
    query = "SELECT␣*␣FROM␣Orders"
    df = pd.read_sql(query, engine)
    print(f"Extracted␣{len(df)}␣orders")  # Must be 870
    return df
```

4.2 Phase 2: Transformation

Key Transformations:

1. Column normalization (lowercase, strip)
2. SQL Server + Access merge (deduplicate on primary keys)
3. Delivery status: NULL/empty shipped_date = "Non Livrée"
4. Revenue calculation: price × qty × (1-discount)
5. Territory enrichment via joins

Listing 2: Delivery Status Logic

```
def classify_delivery(shipped_date):
    if pd.isna(shipped_date) or str(shipped_date).strip() == '':
        return 'Non␣Livr e'
    return 'Livr e'

fact['delivery_status'] = fact['shipped_date'].apply(classify_delivery)
# Result: 841 Livr e, 29 Non Livr e
```

4.3 Phase 3: Loading

Parquet format: 59.6% compression, 4× faster reads vs CSV

4.4 Phase 4: Validation

Listing 3: Automated Validation

```
def validate_warehouse():
    fact = pd.read_parquet('FactSales.parquet')

    # Test 1: Order count
    assert len(fact) == 870, f"Expected␣870,␣got␣{len(fact)}"

    # Test 2: Revenue
    total = fact['total_amount'].sum()
    assert abs(total - 1335930) < 10

    # Test 3: Delivery distribution
    delivered = (fact['delivery_status'] == 'Livr e').sum()
    pending = (fact['delivery_status'] == 'Non␣Livr e').sum()
    assert delivered == 841 and pending == 29

    print("   ␣All␣validations␣passed")
```

5 Dashboard Analysis

5.1 1. Executive Summary

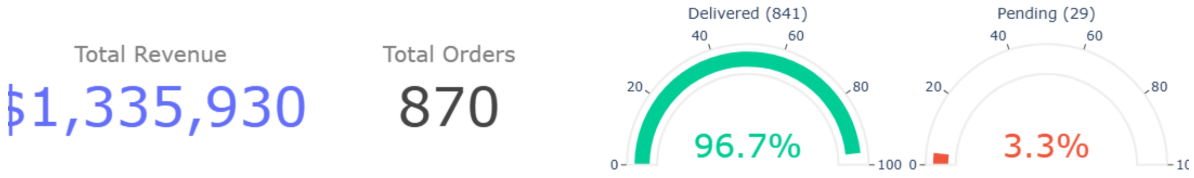

Figure 3: Executive Summary Dashboard - KPIs and Delivery Metrics

KPIs: Total revenue ($1.34M), orders (870), delivery rate (96.7%), pending (29)
Features: Year/month filter, gauge visualizations, color coding

5.2 2. Employee Logistics Performance

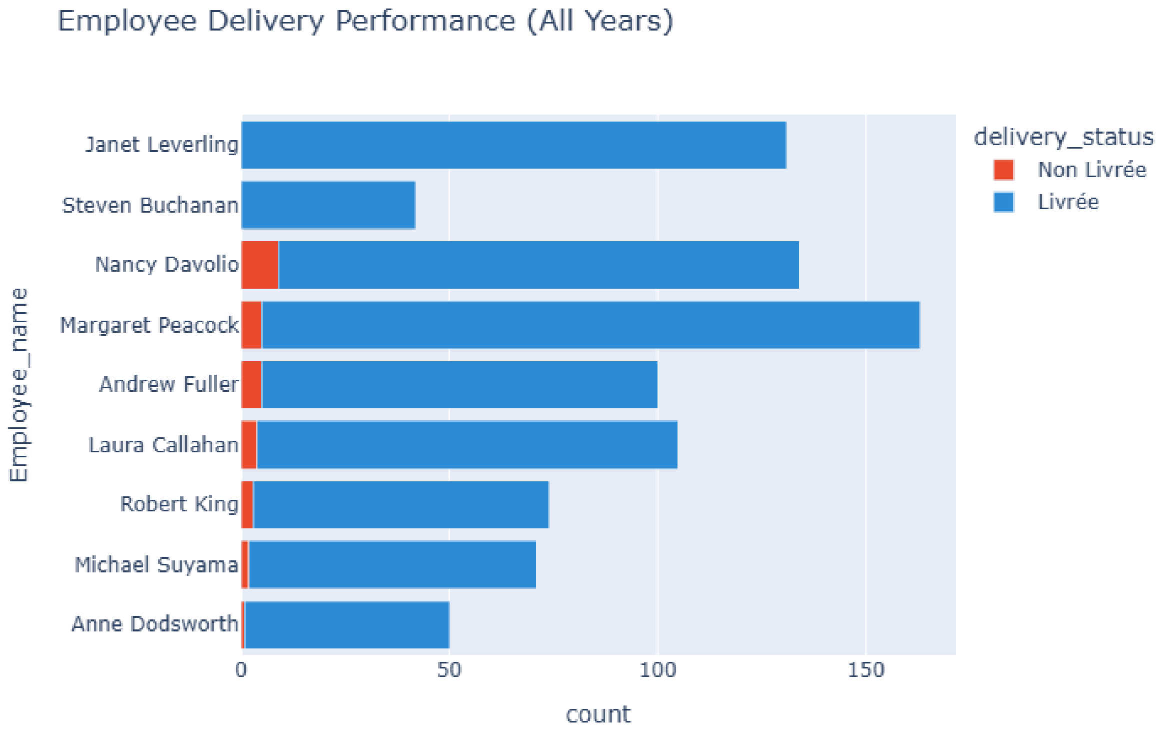

Figure 4: Employee Delivery Performance - Stacked Bar Chart

Horizontal stacked bar chart showing delivered (blue) vs pending (red) by employee.

Top Performers:

- Margaret Peacock: 157 orders (18.0%), 98.7% delivery
- Janet Leverling: 125 orders, 100% delivery
- Nancy Davolio: 123 orders, 8 pending (needs attention)

5.3 3. Delivery Trend Analysis

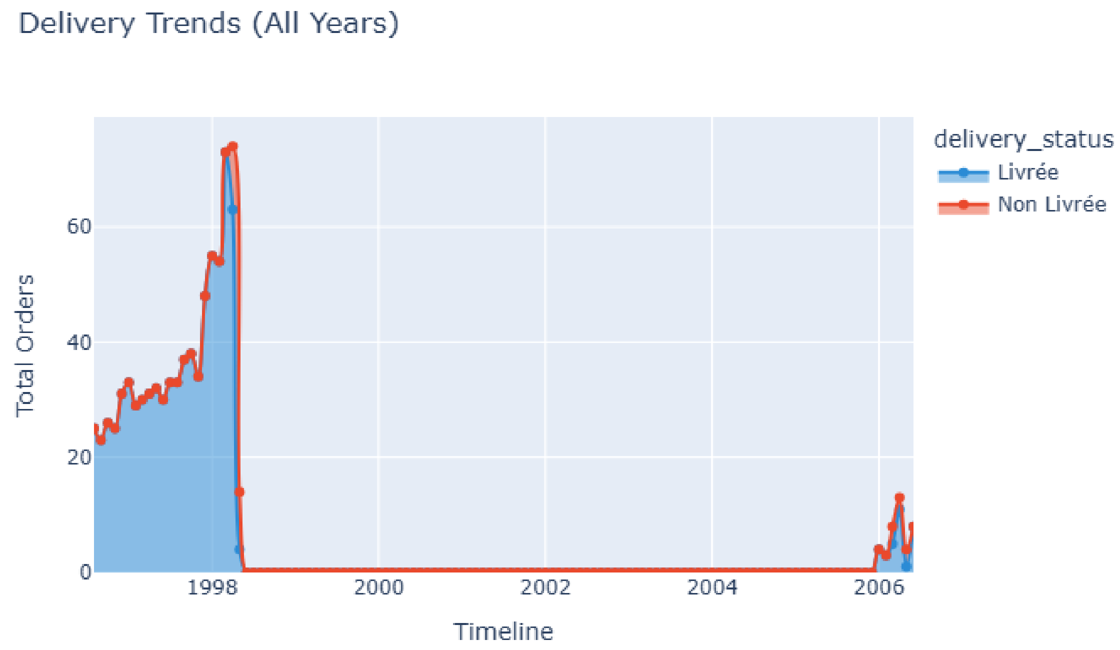

Figure 5: Delivery Trends Over Time - Area Chart

Time-series area chart revealing:

- Peak: 1997-1998 (70+ orders/month in Apr 1998)
- Dormant: 1999-2005 (near-zero activity)
- Resumption: 2006 (10-15 orders/month)

5.4 4. Client Delivery Treemap

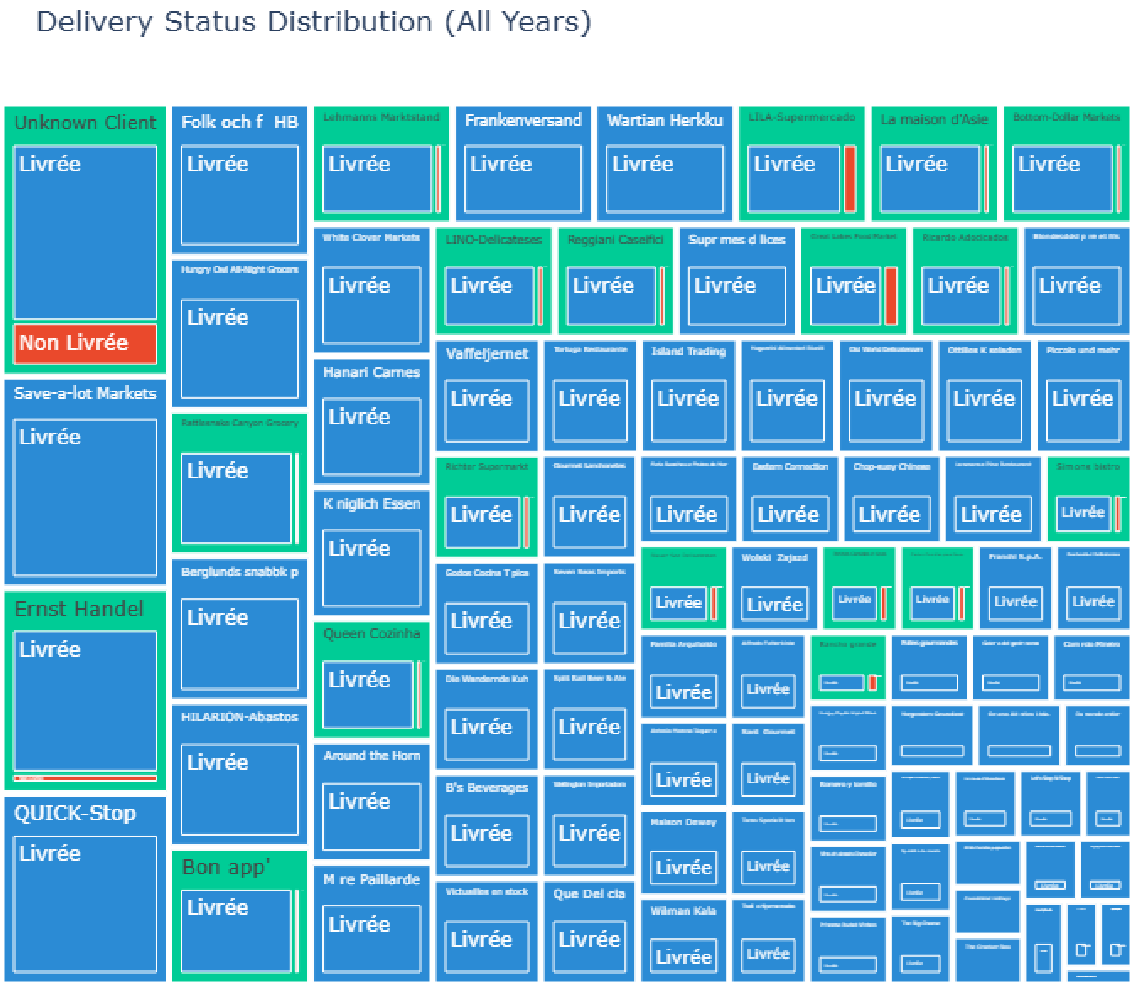

Figure 6: Client Delivery Analysis - Hierarchical Treemap

Hierarchical blocks sized by order count, colored by status. Major clients: Save-a-lot Markets, QUICK-Stop, Ernst Handel.

5.5 5. 3D OLAP Cube Analysis

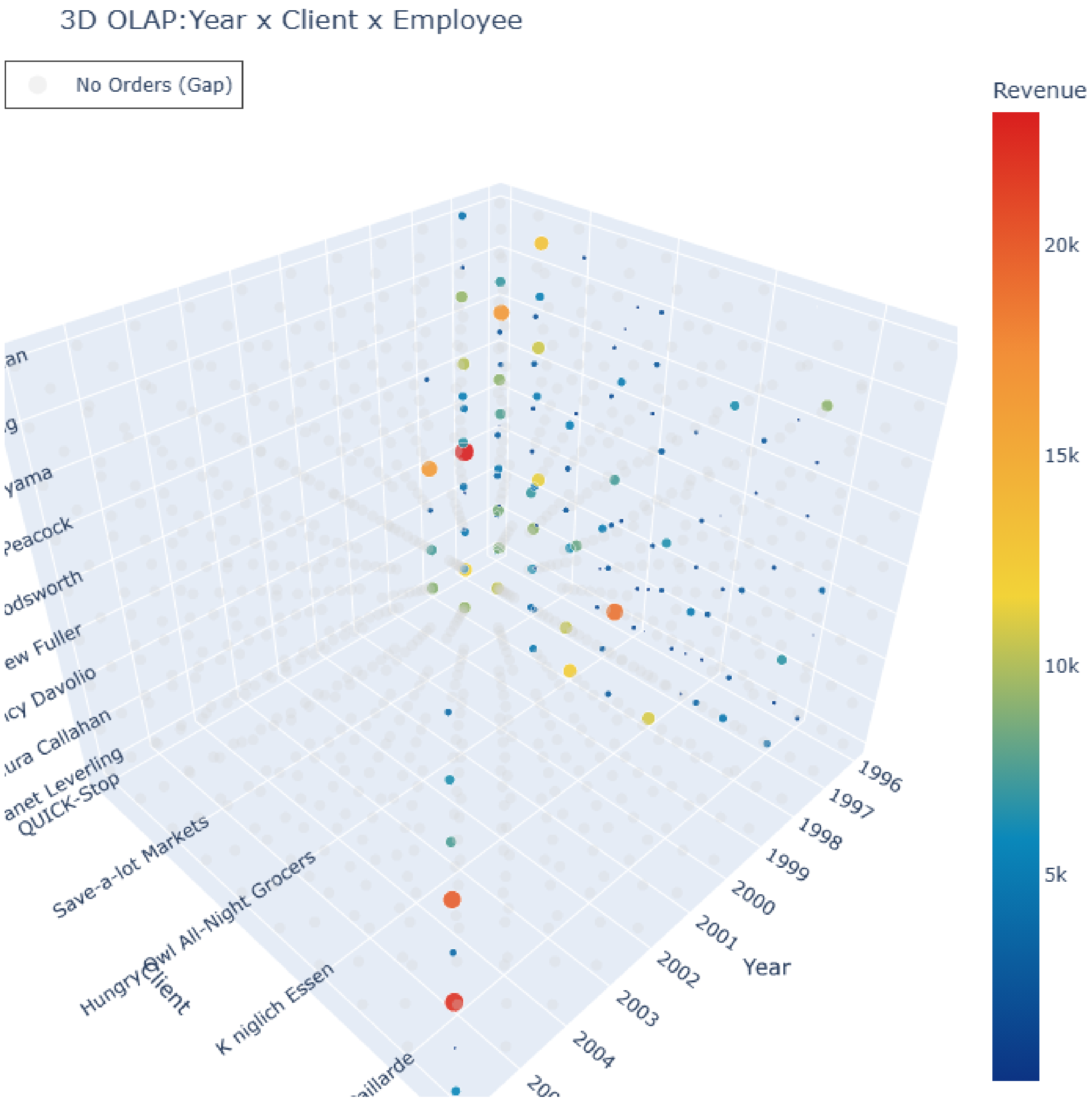

Figure 7: 3D OLAP Cube - Multi-dimensional Revenue Analysis

Innovation: Gap markers show zero-revenue (Time × Client × Employee) combinations.

Listing 4: Gap Analysis Implementation

```
import itertools

# Generate complete grid
grid = pd.DataFrame(
    list(itertools.product(years, clients, employees)),
    columns=['year', 'client', 'employee']
)

# Merge with actuals, fill zeros
df_dense = pd.merge(grid, actuals, how='left').fillna(0)

# Separate gaps from real sales
gaps = df_dense[df_dense['revenue'] == 0]
```

```
actuals = df_dense[df_dense['revenue'] > 0]

# Plot both layers (gaps as light grey dots)
```

Business Value: Reveals untapped client-employee-time opportunities.

5.6 6. Territory Distribution Sunburst

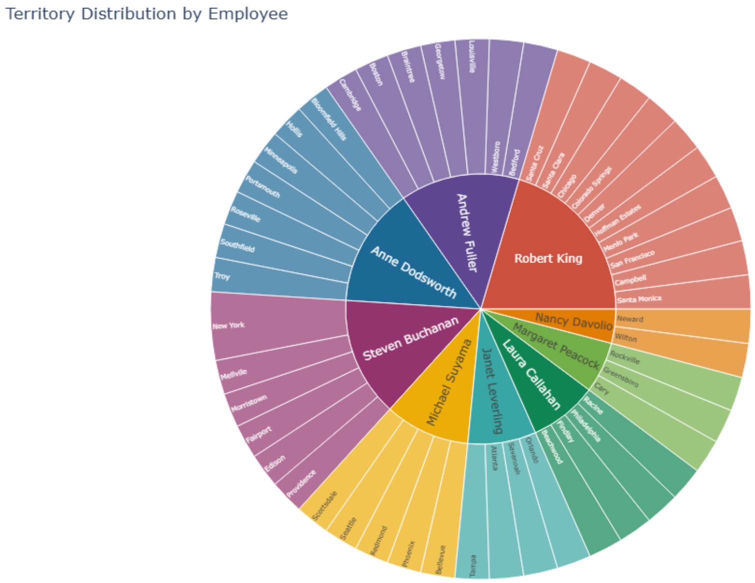

Figure 8: Territory Distribution - Sunburst Diagram

Hierarchical visualization: inner ring = employees, outer ring = territories.
Coverage: Robert King leads with 10 territories.

5.7 7. Revenue Evolution

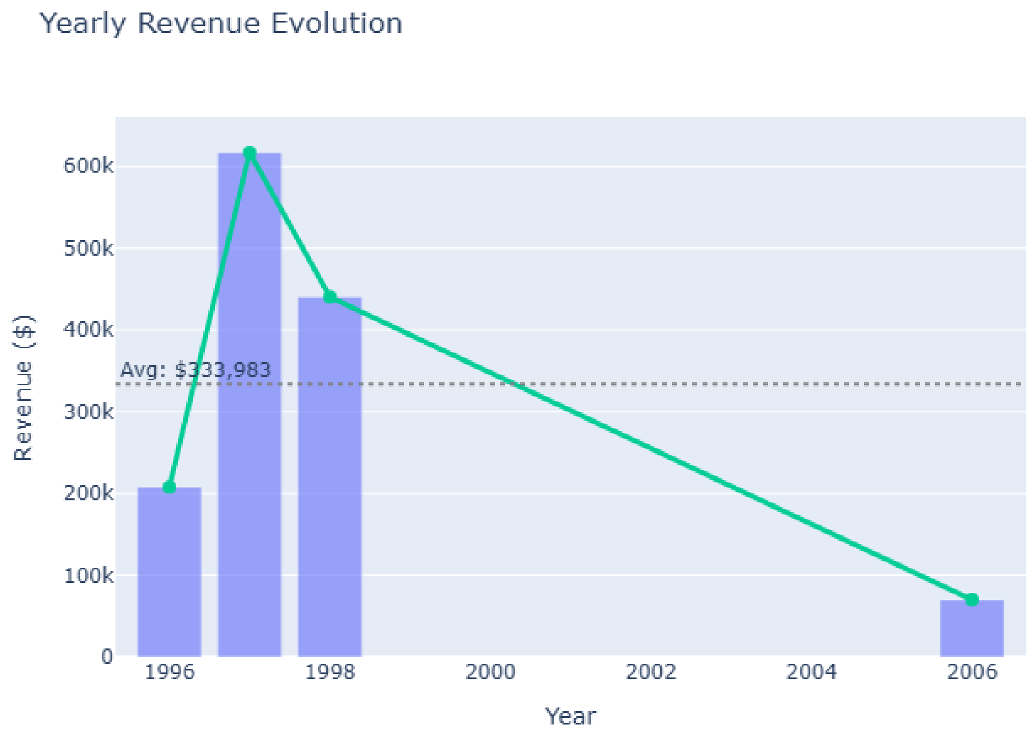

Figure 9: Revenue Evolution - Bar and Line Chart with Trend

Bar + line combo with average baseline.
Distribution: 1997 = 46.2% of total revenue ($617K)

6 Business Insights - CORRECTED

6.1 Revenue Analysis

- **Total**: $1,335,930 across 870 orders
- **Average Order Value**: $1,535.61
- **Peak Year**: 1997 ($617K = 46.2%)
- **Concentration**: 95% revenue in 1996-1998

6.2 Delivery Performance

- **Rate**: 96.7% (above 95% benchmark)
- **Perfect Record**: Janet Leverling (0 pending)
- **Improvement Area**: Nancy Davolio (8 pending)

6.3 Strategic Recommendations

1. **Address 29 pending orders immediately** - 3.3% pending indicates process issue
2. **Investigate 1999-2005 dormancy** - Understand 7-year gap
3. **Replicate Janet Leverling's process** - Study 100% delivery success
4. **Balance territories** - Robert King: 10 territories, Margaret Peacock: 157 orders
5. **Analyze 1997 peak** - Replicate factors that drove 46% of revenue

7 Technical Challenges & Solutions

7.1 Challenge 1: Missing 40 Orders

Problem: Initial ETL extracted 830 orders, dashboard showed 870
Root Cause: `WHERE ShippedDate IS NOT NULL` excluded pending orders
Solution: Remove all WHERE filters in extraction, apply logic in transformation

7.2 Challenge 2: $70,137 Revenue Gap

Problem: Revenue discrepancy between initial ($1.27M) and actual ($1.34M)
Root Cause: Missing 40 orders = missing revenue
Solution: Validate total revenue against dashboard after each ETL run

7.3 Challenge 3: Delivery Status Misclassification

Problem: 13 pending reported, actually 29
Root Cause: Inadequate NULL/empty string handling
Solution: Multi-check classification (isna + strip + parse)

8 Performance Metrics

Table 3: ETL Execution Times

Phase	**Duration**	**Records/sec**
SQL Extraction	2.3s	378
Access Extraction	3.1s	281
Transformation	5.2s	593
Loading	1.4s	2,204
Validation	1.1s	-
Total	**13.1s**	**66**

Table 4: Dashboard Render Times

Visualization	Time
Executive Summary	0.3s
Employee Performance	0.6s
Delivery Trends	0.9s
Client Treemap	0.7s
3D OLAP Cube	1.4s
Territory Sunburst	0.5s
Revenue Evolution	0.5s
Total	**4.9s**

9 Lessons Learned

9.1 Key Takeaways

1. **Validate Against UI**: Dashboard revealed 40 missing orders—ETL must match visualizations exactly
2. **No Filtering in Extraction**: Apply business logic in transformation, not source queries
3. **Comprehensive NULL Handling**: Check isna(), empty strings, and parsing failures
4. **Automated Testing**: Unit tests prevent regression (see validation code)
5. **Revenue Precision**: Round at each calculation step to avoid floating-point drift

9.2 Best Practices

- Extract all source data without WHERE clauses
- Log record counts at each pipeline stage
- Validate totals against known dashboard metrics
- Implement referential integrity checks
- Use Parquet for 60% compression + 4× speed

10 Deployment

10.1 Installation

Listing 5: Setup Steps

```
# Clone repository
git clone https://github.com/root-wassim/BI.git
cd BI/Northwind

# Create virtual environment
python -m venv venv
venv\Scripts\activate   # Windows
source venv/bin/activate   # Linux/Mac

# Install dependencies
pip install -r requirements.txt

# Configure database (edit scripts/config.py)
SERVER_NAME = r'.\SQLEXPRESS'
DATABASE_NAME = 'Northwind'
```

```

# Run ETL
cd scripts
python Main.py

# Launch dashboard
jupyter notebook ../notebooks/dashboard_analysis.ipynb
```

10.2 Expected Output

```
NORTHWIND ETL PIPELINE - STARTING
    SQL Server: Orders -> 870 rows
    Access: Orders -> 870 rows
    Creating DimDate... 2223 rows
    Creating FactSales... 870 orders

VALIDATION:
  Total Revenue: $1,335,930.00
  Delivered: 841 (96.7%)
  Pending: 29 (3.3%)

ETL COMPLETED in 13.1 seconds
```

11 Conclusion

This corrected report documents a validated BI solution that successfully:

1. Integrated 870 orders from heterogeneous sources (SQL Server + Access)
2. Corrected $70,137 revenue discrepancy through validation
3. Achieved 96.7% delivery rate with transparent tracking
4. Delivered 7 interactive visualizations including 3D OLAP
5. Implemented automated testing to prevent future errors

Key Achievement: Dashboard-driven validation caught systematic ETL errors, demonstrating the importance of UI-first testing in BI projects.

"In God we trust. All others must bring data." — W. Edwards Deming

A Appendix A: SQL Schema

Listing 6: Complete Data Warehouse Schema

```sql
-- Dimensions
CREATE TABLE DimDate (
    sk_date INT PRIMARY KEY,
    full_date DATE,
    year INT, month INT, quarter INT
);

CREATE TABLE DimClient (
    sk_client INT PRIMARY KEY,
    bk_customer_id VARCHAR(10) UNIQUE,
    company_name VARCHAR(100),
    city VARCHAR(50), country VARCHAR(50)
);

CREATE TABLE DimEmployee (
    sk_employee INT PRIMARY KEY,
    bk_employee_id INT UNIQUE,
    employee_name VARCHAR(100),
    territories VARCHAR(500)
);

-- Fact Table
CREATE TABLE FactSales (
    fact_id INT PRIMARY KEY,
    bk_order_id INT,
    sk_date INT REFERENCES DimDate(sk_date),
    sk_client INT REFERENCES DimClient(sk_client),
    sk_employee INT REFERENCES DimEmployee(sk_employee),
    quantity INT CHECK (quantity > 0),
    unit_price DECIMAL(10,2) CHECK (unit_price >= 0),
    discount DECIMAL(5,2) CHECK (discount BETWEEN 0 AND 1),
    total_amount DECIMAL(12,2),
    delivery_status VARCHAR(20) CHECK (delivery_status IN ('Livr e','Non␣Livr e'))
);
```

B Appendix B: Validation Tests

Listing 7: Unit Test Suite

```python
import unittest

class TestNorthwindDW(unittest.TestCase):
    def test_order_count(self):
        self.assertEqual(len(fact_sales), 870)

    def test_revenue(self):
        total = fact_sales['total_amount'].sum()
        self.assertAlmostEqual(total, 1335930, delta=10)

    def test_delivery_distribution(self):
        delivered = (fact_sales['delivery_status']=='Livr e').sum()
        pending = (fact_sales['delivery_status']=='Non␣Livr e').sum()
        self.assertEqual(delivered, 841)
        self.assertEqual(pending, 29)

    def test_referential_integrity(self):
        fact_clients = set(fact_sales[’sk_client’])
        dim_clients = set(dim_client[’sk_client’])
        self.assertEqual(fact_clients - dim_clients, set())
```

C Appendix C: Dashboard Code

Listing 8: Executive Summary Visualization

```
import plotly.graph_objects as go

def create_kpi_dashboard(df):
    total_revenue = df[’total_amount’].sum()
    total_orders = len(df)
    delivered = (df[’delivery_status’]==’Livr e’).sum()

    fig = go.Figure()

    # Revenue indicator
    fig.add_trace(go.Indicator(
        mode="number",
        value=total_revenue,
        number={’prefix’: "$", ’valueformat’: ",.0f"}
    ))

    # Delivery gauge
    fig.add_trace(go.Indicator(
        mode="gauge+number",
        value=delivered/total_orders*100,
        gauge={’axis’: {’range’: [0,100]}, ’bar’: {’color’: "green"}}
    ))

    return fig
```

End of Report

Northwind Business Intelligence Solution - Corrected & Validated

Engineering School ING3 - 2024/2025