AkasaAir Data Engineer Task 1

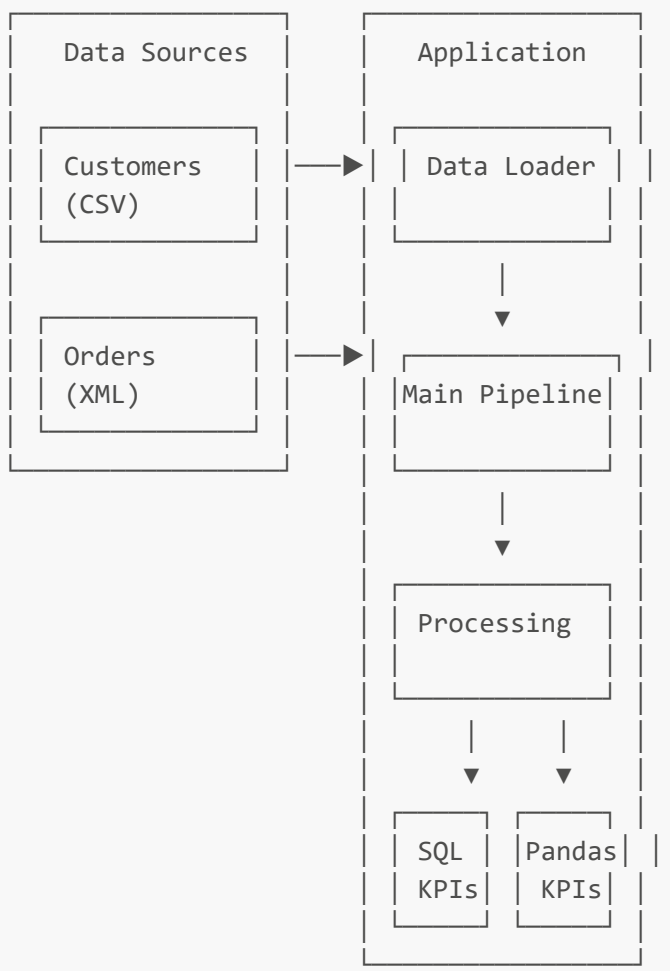
Project Overview

This project implements a data processing pipeline for analyzing customer and order data. The application demonstrates two different methods for data processing and KPI calculation:

1. **Database (SQL) Approach** - Using MySQL for data storage and SQL queries for analytics
2. **In-Memory (Pandas) Approach** - Using pandas DataFrames for data processing

System Architecture

High-Level Design



Component Architecture

```
src/
├── main.py                 # Main
├── load_data.py           # Data ingestion module
├── db_approach.py         # SQL-based processing
```

```
├── in_memory_approach.py  # Pandas-based processing
└── __init__.py           # Package initialization
```

Implementation Details

1. Data Loading (`load_data.py`)

Purpose: Handles data ingestion from multiple file formats

Key Functions:

- `load_customers(file_path)`: Loads customer data from CSV
- `load_orders(file_path)`: Loads and deduplicates order data from XML

```
# Example data cleaning process
customers['mobile_number'] = customers['mobile_number'].astype(str)
customers['customer_name'] = customers['customer_name'].str.strip()
```

2. Database Approach (`db_approach.py`)

Purpose: Implements SQL-based data processing using MySQL

Architecture:

- **Connection Management**: Secure database connections using environment variables
- **Database Creation**: Automatic database and table creation
- **SQL Analytics**: Aggregation queries for KPI calculation

Database Schema:

```
customers (
    customer_id VARCHAR(50) PRIMARY KEY,
    customer_name VARCHAR(100),
    mobile_number VARCHAR(20) INDEX,
    region VARCHAR(50)
)

orders (
    order_id VARCHAR(50) PRIMARY KEY,
    mobile_number VARCHAR(20) FOREIGN KEY,
    order_date_time DATETIME,
    total_amount FLOAT,
    INDEX idx_order_date (order_date_time)
)
```

3. In-Memory Approach (`in_memory_approach.py`)

Purpose: Implements pandas-based data processing

Key Functions:

- `merge_data()`: Joins customer and order data
- `calculate_kpis()`: Computes business metrics using pandas operations

4. Main Pipeline (`main.py`)

Purpose: Orchestrates the entire data processing workflow

Execution Flow:

1. **Data Ingestion**: Load customer and order data
2. **Validation**: Check data integrity
3. **Database Processing**: Execute SQL-based analytics
4. **In-Memory Processing**: Execute pandas-based analytics
5. **Results Comparison**: Display results from both approaches

Business Metrics (KPIs)

The application calculates four key business metrics:

1. Repeat Customers

- **Definition**: Customers who have placed more than one order
- **Business Value**: Identifies loyal customer base
- **Output**: Customer names with order counts

2. Monthly Sales Trends

- **Definition**: Order volume and revenue aggregated by month
- **Business Value**: Tracks business growth patterns
- **Output**: Monthly order counts and total revenue

3. Regional Revenue Analysis

- **Definition**: Total revenue breakdown by geographic region
- **Business Value**: Identifies high-performing markets
- **Output**: Revenue totals sorted by region

4. Top Spenders (Last 30 Days)

- **Definition**: Top 10 customers by spending in recent period
- **Business Value**: Identifies high-value customers for retention
- **Output**: Customer names with recent spending amounts

How to Run

Prerequisites

1. **Python 3.7+** installed
2. **MySQL Server** running (for database approach)

3. **Git** (for cloning repository)

Setup Instructions

1. Clone the Repository

```
git clone https://github.com/jishnu3000/AkasaAir-DataEngineer-Task1.git
cd AkasaAir-DataEngineer-Task1
```

2. Create Virtual Environment

```
# Windows
python -m venv venv
venv\Scripts\activate
```

3. Install Dependencies

```
pip install -r requirements.txt
```

4. Database Configuration

Create a `.env` file in the project root:

```
DB_HOST=localhost
DB_PORT=3306
DB_USER=your_mysql_username
DB_PASSWORD=your_mysql_password
DB_NAME=akasa_db
```

MySQL Setup Options:

Option A: Use existing MySQL user

```
DB_USER=root
DB_PASSWORD=your_root_password
```

Option B: Create new MySQL user

```
-- Connect to MySQL as root
CREATE USER 'akasa_user'@'localhost' IDENTIFIED BY 'secure_password';
```

```
GRANT ALL PRIVILEGES ON *.* TO 'akasa_user'@'localhost';
FLUSH PRIVILEGES;
```

5. Run the Application

```
# From project root directory
python -m src.main
```

Expected Output

```
Starting data processing...
Loading customer data...
Loaded 5 customers
Loading order data...
Loaded 3 orders
Loaded 5 customers and 3 orders

==================================================
RUNNING DATABASE (SQL) APPROACH
==================================================
Database connection successful!
Database setup complete!

Database KPI Results:

REPEAT CUSTOMERS:
------------------------------
customer_name  order_count
  Aarav Mehta            2

MONTHLY TRENDS:
------------------------------
 total_orders  total_revenue
            1         8930.0
            1         7450.0
            1         5299.0

REGIONAL REVENUE:
------------------------------
region  regional_revenue
  West           12749.0
 North            8930.0

TOP SPENDERS:
------------------------------
customer_name  recent_spend
  Aarav Mehta       12749.0

==================================================
```

```
RUNNING IN-MEMORY (PANDAS) APPROACH
================================================
Merging customer and order data...
Merged data: 3 records
Successfully merged data: 3 records
Calculating business metrics...
Successfully calculated all metrics

In-Memory KPI Results:

REPEAT CUSTOMERS:
------------------------------
customer_name  number_of_orders
  Aarav Mehta                 2

MONTHLY TRENDS:
------------------------------
  month  total_orders  total_revenue
2025-09             1         8930.0
2025-10             1         7450.0
2025-11             1         5299.0

REGIONAL REVENUE:
------------------------------
region  total_revenue
  West        12749.0
 North         8930.0

TOP SPENDERS:
------------------------------
customer_name  total_spent
  Aarav Mehta      12749.0
...
```

Configuration Options

Environment Variables

- `DB_HOST`: MySQL server hostname (default: localhost)
- `DB_PORT`: MySQL server port (default: 3306)
- `DB_USER`: MySQL username
- `DB_PASSWORD`: MySQL password
- `DB_NAME`: Database name (default: akasa_db)

File Paths

- Customer data: `data/task_DE_new_customers.csv`
- Order data: `data/task_DE_new_orders.xml`

Troubleshooting

Common Issues

1. MySQL Connection Error

```
Error: (mysql.connector.errors.ProgrammingError) 1045 (28000): Access denied
```

Solution: Verify MySQL credentials in `.env` file

2. Module Import Error

```
ModuleNotFoundError: No module named 'src'
```

Solution: Run from project root using `python -m src.main`

3. Missing Dependencies

```
ModuleNotFoundError: No module named 'pandas'
```

Solution: Install requirements: `pip install -r requirements.txt`

4. Database Creation Error

```
Error: Database 'akasa_db' doesn't exist
```

Solution: The application automatically creates the database. Ensure MySQL user has CREATE privileges.

Fallback Mode

If database connection fails, the application will:

1. Log the database error
2. Skip SQL approach
3. Continue with pandas approach
4. Still provide analytics results

Project Structure

```
AkasaAir-DataEngineer-Task1/
├── .env                          # Environment configuration
├── .gitignore                    # Git ignore rules
├── README.md                     # Project documentation
├── requirements.txt              # Python dependencies
```

```
├── data/                          # Data files
│   ├── task_DE_new_customers.csv # Customer data
│   └── task_DE_new_orders.xml    # Order data
├── src/                           # Source code
│   ├── __init__.py               # Package init
│   ├── main.py                   # Main application
│   ├── load_data.py              # Data loading
│   ├── db_approach.py            # SQL processing
│   └── in_memory_approach.py     # Pandas processing
└── venv/                          # Virtual environment
```

Testing

Data Validation Tests

```
# Test data loading
python -c "from src.load_data import load_customers;
print(load_customers('data/task_DE_new_customers.csv').shape)"

# Test database connection
python -c "from src.db_approach import get_db_engine; print('Connected!' if
get_db_engine() else 'Failed')"
```

Future Improvements

Scalability

- **Distributed Processing**: Apache Spark integration
- **Cloud Database**: AWS RDS or Azure SQL support

Analytics

- **Visualization**: Interactive dashboards
- **Machine Learning**: Predictive analytics integration

Infrastructure

- **Containerization**: Docker deployment
- **CI/CD Pipeline**: Automated testing and deployment

Dependencies

Package	Version	Purpose
pandas	Latest	Data manipulation and analysis
sqlalchemy	Latest	SQL toolkit and ORM
pymysql	Latest	MySQL database connector

Package	Version	Purpose
python-dotenv	Latest	Environment variable management
lxml	Latest	XML parsing
tabulate	Latest	Table formatting

Author: Jishnu Plavinchottil Jayaraj
Repository: AkasaAir-DataEngineer-Task1
Date: November 6, 2025