Beginner's Roadmap: Fleet Management Logistics BI Project

Project Plan

October 29, 2025

Contents

1 Project Overview: The Story

Our goal is to act as a BI Data Analyst for a logistics company. We've been given a large data dump (`Shipments_Report.csv`) and know the company is struggling with late deliveries, rising costs, and potential safety issues.

Our mission: Transform this raw data into an actionable intelligence dashboard. We will tell a story that starts with ”What is happening?” (KPIs), moves to ”Why is it happening?” (Statistical Analysis), and ends with ”What should we do?” (Recommendations).

Core Problems to Solve:

- **On-Time Delivery (OTD):** Why are we missing delivery targets?
- **Safety & Compliance:** Are drivers violating rules (e.g., 'No Go Zones')?
- **Inefficiency:** Are we wasting fuel and time? (Km/h, Fleet Utilization)

2 Phase 1: Data Cleaning & Modeling (Excel & SQL)

This phase is the foundation. We'll create a clean, reliable, and queryable database.

2.1 Step 1: Initial Inspection & Prep (Excel)

1. **Inspect CSV:** Open `Shipments_Report_2025-10-27.xlsx - Data Export.csv`.
2. **Clean Junk:** You noted 3 header rows. Delete rows 1-3. Row 4 is your true header.
3. **Spot Check Columns:** Look at columns like `Otd`, `Otd Status`, `Lead Time`, `Lead Time Sla`, `Total Time Spent In No Go Zones`, and all `...Date...` columns. Check their formats. `Lead Time` (e.g., 34.013) looks like decimal hours, which is excellent.
4. **Create Dimension Files (Critical Thinking):** Your main file is a *fact* table. We need *dimension* tables for context. Create two new, small Excel files:
   - **Dim_Truck.xlsx:**
     - Go to the main file, filter the `Truck Plate` column, and ”Copy Unique Values.”
     - Paste into a new sheet. Add the corresponding `Truck Type`.
     - Your columns should be: `Truck_Plate` (Primary Key), `Truck_Type`.
   - **Dim_Plant.xlsx:**
     - Go to the main file, copy unique values from the `Plant` column.
     - **Feature Engineering Task:** The dataset **lacks plant locations**, which we need for distance calculation (Km/h KPI). You must *create* this data.
     - Add columns: `Plant_Name` (Primary Key), `Plant_Lat`, `Plant_Lon`.
     - *Action:* Go to Google Maps and find realistic Lat/Lon coordinates for the plant names (e.g., if a plant is ”Lagos Depot,” find the lat/lon for Lagos, Nigeria). This demonstrates problem-solving.
5. **Save Clean File:** Save the main data file as `Fact_Shipments_Clean.csv`.

2.2 Step 2: Database Setup (SQL)

Use any SQL database (PostgreSQL, SQL Server Express, or even SQLite).

1. **Create Tables:**
   - `CREATE TABLE Dim_Plant ( Plant_Name VARCHAR(100) PRIMARY KEY, Plant_Lat FLOAT, Plant_Lon FLOAT );`
   - `CREATE TABLE Dim_Truck ( Truck_Plate VARCHAR(50) PRIMARY KEY, Truck_Type VARCHAR(50) );`
   - `CREATE TABLE Fact_Shipments ( ... );` — Define columns to match your CSV file. Use `DATETIME` for dates, `FLOAT` for numbers like `Lead Time`, `VARCHAR` for text.
2. **Import Data:** Use your database's import wizard to load the data from your clean `.csv` and `.xlsx` files into these tables.

2.3 Step 3: Data Transformation (SQL View)

This is where we prepare the data for analysis. We'll create a single, powerful **SQL View**. This view will join our tables and calculate our KPIs.

1. **Haversine Function (for Distance):** We must calculate distance from our Lat/Lon data. Create this function in your SQL database. (Google "Haversine function SQL" for your specific database flavor).
   - *Example (T-SQL):* `CREATE FUNCTION dbo.GetDistanceKM (@lat1 FLOAT, @lon1 FLOAT, @lat2 FLOAT, @lon2 FLOAT) RETURNS FLOAT AS ... [body of function] ...`
2. **Create the Analysis View:**

```
CREATE VIEW v_Shipment_Analysis AS
SELECT
    fs.*, -- Select all from fact table
    dt.Truck_Type,
    dp.Plant_Lat,
    dp.Plant_Lon,

    -- KPI 1: On-Time Delivery & SLA
    CASE WHEN fs.Otd = 'Yes' THEN 1 ELSE 0 END AS is_OTD,
    CASE WHEN fs.[Lead Time] <= fs.[Lead Time Sla] THEN 1 ELSE 0 END AS is_SLA_Compliant,
    (fs.[Lead Time Sla] - fs.[Lead Time]) AS Sla_Time_Delta_Hours,

    -- KPI 2: Violation Rate
    CASE WHEN fs.[Total Time Spent In No Go Zones] > 0 THEN 1 ELSE 0 END AS is_No_Go_Viola

    -- KPI 3: Kilometers per Hour
    dbo.GetDistanceKM(dp.Plant_Lat, dp.Plant_Lon,
                      fs.Offloaded_Latitude, fs.Offloaded_Longitude) AS Distance_KM,
    NULLIF(fs.[Time Spent In Transit], 0) AS Transit_Hours,
    (dbo.GetDistanceKM(dp.Plant_Lat, dp.Plant_Lon,
                       fs.Offloaded_Latitude, fs.Offloaded_Longitude)
     / NULLIF(fs.[Time Spent In Transit], 0)) AS KPH,

    -- KPI 4 & 5: Fleet Utilization & Other Metrics
```

```
        fs.[Total Journey Duration] AS Total_Journey_Hours,
        fs.[Time Spent At Customer Site] AS Customer_Time_Hours

    FROM
        Fact_Shipments fs
    LEFT JOIN
        Dim_Truck dt ON fs.[Truck Plate] = dt.Truck_Plate
    LEFT JOIN
        Dim_Plant dp ON fs.Plant = dp.Plant_Name;
```

Result: We now have a single, clean source (`v_Shipment_Analysis`) that Power BI and KNIME can connect to.

3 Phase 2: Analysis Workflows (KNIME)

Now we move from "what" to "why." KNIME's visual workflow is perfect for this.

3.1 Step 4: Connect & Prep

1. **Nodes:** `Database Connector` (point to your SQL DB) -> `Database Query` (write `SELECT * FROM v_Shipment_Analysis`).
2. **Clean:** Use the `Missing Value` node to handle any nulls (e.g., for `KPH` where `Transit_Hours` was 0). You can impute the mean or remove the row.
3. **Format:** Use `String to Number` or `Category to Number` for any columns that are read incorrectly.

3.2 Step 5: Statistical Analysis

1. **Logistic Regression (Predicting OTD):**
   - **Goal:** Predict `is_OTD` (our 0/1 variable).
   - **Story:** "What factors most increase the *risk* of a late delivery?"
   - **Nodes:**
     - `Partitioning`: Split data 80% train, 20% test.
     - `Logistic Regression Learner`: Connect to 80% train. Select `is_OTD` as the target. Use features like `Distance_KM`, `Truck_Type`, `Region`, `Customer_Time_Hours`, `Quantity`.
     - `Logistic Regression Predictor`: Connect model and 20% test set.
     - `Scorer`: See how accurate your model is.
   - **Output:** The "Coefficients" from the Learner node are your story. A large positive coefficient for `Customer_Time_Hours` means "For every hour spent at the customer site, the odds of being late increase by X%."
2. **ANOVA (Comparing Groups):**
   - **Goal:** Compare the mean of a number across 3+ groups.

- **Story:** ”Is there a *statistically significant* difference in `Lead Time` between our different `Region`s?”
- **Nodes:** `ANOVA` node. Input `Lead Time` as the test column and `Region` as the category column.
- **Output:** The ’p-value’. If $p < 0.05$, the answer is ”Yes, the difference is real and not due to random chance.”

3. **PCA (Principal Component Analysis):**
   - **Goal:** Simplify complex data.
   - **Story:** ”Our ’time’ variables (`Time Spent In Transit`, `Offloading Duration`, `Return Journey Duration`) are all related. Can we combine them into a single ’Trip Complexity’ score?”
   - **Nodes:** `PCA` node on just those numeric time columns. `Output:` You’ll see ”Principal Component 1” explains (e.g.,) 70% of the variance. You can use this new ”PC1” score in other models.

4 Phase 3: Dashboard Design (Power BI)

This is our ”storytelling” phase. We present our findings to leadership.

4.1 Step 6: Connect & Create Measures (DAX)

1. **Connect:** Open Power BI Desktop. Get Data -> SQL Server Database. Connect to your DB and select `v_Shipment_Analysis`, `Dim_Truck`, and `Dim_Plant`.
2. **Model:** Go to the ”Model” view. Power BI should auto-detect the relationships. If not, drag `Truck_Plate` to `Truck Plate` and `Plant_Name` to `Plant`.
3. **Create DAX Measures:** This is vital. Don’t just drag columns. Create measures.

```
-- Create a new 'Measures' table to hold these
Total Shipments = COUNTROWS(v_Shipment_Analysis)

OTD % = AVERAGE(v_Shipment_Analysis[is_OTD]) -- Format as %

Violation Rate % = AVERAGE(v_Shipment_Analysis[is_No_Go_Violation]) -- Format as %

Avg KPH = AVERAGE(v_Shipment_Analysis[KPH]) -- Format as number

Avg Fleet Utilization (Trips/Truck) =
    DIVIDE(
        [Total Shipments],
        DISTINCTCOUNT(v_Shipment_Analysis[Truck Plate])
    )
```

4.2 Step 7: Build Your Dashboard (3 Pages)

Page 1: Executive KPI Overview

- **Title:** ”Fleet Operations at a Glance”

- **Visuals:**
  - **KPI Cards:** 5 cards at the top for each measure you just created.
  - **Map (Map Visual):** Use `Offloaded_Latitude` and `Offloaded_Longitude`. Use `Ship To City` as location. Use `Total Shipments` for bubble size. Use `OTD %` for color saturation (red-to-green).
  - **Bar Chart:** `OTD %` (Value) by `Region` (Axis).
  - **Line Chart:** `Total Shipments` and `Violation Rate %` by `Dispatch Date` (Axis, set to Month).

Page 2: Operations Deep Dive • **Title:** ”Performance & Inefficiency Analysis”

- **Visuals:**
  - **Slicers (Filters):** `Region`, `Truck_Type`, `Customer Name`.
  - **Scatter Plot:** `Distance_KM` (X-Axis) vs. `KPH` (Y-Axis). `Truck Plate` (Details). This helps you spot slow trucks/trips.
  - **Bar Chart (Clustered):** `Avg(Lead Time)` and `Avg(Lead Time Sla)` by `Customer Name`. (Sort by worst offenders).
  - **Table:** ”Violation Report” - Show `Shipment Number`, `Truck Plate`, `Total Time Spent In No Go Zones` (Filter for `is_No_Go_Violation = 1`).

Page 3: Fleet & Safety Focus • **Title:** ”Truck & Driver Performance”

- **Visuals:**
  - **Bar Chart:** ”Avg KPH by Truck Type”.
  - **Bar Chart:** ”Violation Rate % by Truck Plate” (Sort to find worst offenders).
  - **Table:** ”Fleet Scorecard” - Rows: `Truck Plate`, `Truck_Type`. Values: `Total Shipments`, `OTD %`, `Avg KPH`, `Violation Rate %`.

5 Phase 4: (Optional) Front-End Visualization

This is your ”wow” factor. We’ll simulate a live-tracking map.

1. **Data Simulation:** Since your data is static, we’ll ”replay” it. Use a simple Python script with `pandas` and `flask` (or even just KNIME’s ”Wait” node in a loop) to write a `locations.json` file every 5 seconds. This file would contain a list of trucks and their ”current” simulated Lat/Lon based on their trip data.
2. **HTML/CSS/JS (Single File):**
   - **HTML:** Create a `<div>` with `id="map"`.
   - **Library:** Use **Leaflet.js** (a free, simple map library).
   - **JavaScript:**
     (a) Initialize the Leaflet map.
     (b) Create a `setInterval()` function that runs every 5 seconds.
     (c) Inside the interval, use `fetch('locations.json')` to get the new data.

(d) Loop through the JSON data. For each truck:

(e) Check if a map marker for that `Truck Plate` already exists.

(f) If yes: use `marker.setLatLng([newLat, newLon])` to move it.

(g) If no: create a new `L.marker(...)` and add it to the map.

This four-phase plan takes you from a messy CSV to a predictive, insightful, and visually compelling BI project that directly addresses real-world business problems. Good luck!