Day 1–Day 10: Detailed Step■by■Step Plan

This 10■day plan combines learning and doing in parallel. Each day you will touch Python, SQL, Excel, and a BI tool (Power BI or Tableau) on the SAME core project: a Pharma Sales & Safety Analytics System. Assume 6–8 hours per day. Adjust up/down to your pace.

Day 1 – Get Comfortable With the Data

Goal: Set up the dataset and touch all four tools: Python, SQL, Excel, and BI using the same pharma_sales.csv file.

Step■by■Step Tasks:

- **Setup (30–45 min)**: Create a folder for the project. In Excel, create a file named *pharma_sales.csv* with columns Date, Product, Region, Quantity, Revenue and at least 20–50 rows of example data. Save as CSV.
- **Python – Load & Inspect (1.5–2 hrs)**: Open Jupyter/VS Code, create *day1_python.ipynb*, import pandas, load the CSV with pd.read_csv(), run head(), info(), describe(), then create a new column Price_per_unit = Revenue / Quantity and filter rows with Revenue > 10000.
- **SQL – Basic SELECT (1–1.5 hrs)**: Create/import table pharma_sales from the CSV in SQLite/MySQL. Run SELECT *; SELECT specific columns (Date, Product, Revenue); apply WHERE Region='North' and WHERE Revenue>10000; use LIMIT to view a subset.
- **Excel – Basic Summary (1–1.5 hrs)**: Open the CSV in Excel, convert to a Table, use SUM, AVERAGE, MAX, MIN on Revenue and Quantity; apply Filter and Sort (e.g., highest Revenue first).
- **BI Tool – First Charts (1–1.5 hrs)**: In Power BI/Tableau, import pharma_sales.csv and build two visuals: Revenue by Product and Revenue by Region as bar/column charts.
- **Notes (20–30 min)**: Create a notes file (e.g., day1_notes.txt) summarizing commands, queries, formulas, and screenshots you created.

End■of■Day Deliverables (You should have):

- pharma_sales.csv with at least 20–50 rows of data.
- day1_python.ipynb with CSV load, inspection, a filter, and a new column.
- day1_sql.sql file with SELECT and WHERE examples.
- day1_excel.xlsx with basic summaries and filters applied.
- day1_dashboard.pbix (or Tableau workbook) with at least 2 visuals.
- day1_notes.txt summarizing what you did and learned.

Day 2 – Data Cleaning Across All Tools

Goal: Learn to detect and fix missing, duplicate, and invalid data consistently in Python, SQL, Excel, and BI.

Step■by■Step Tasks:

- **Prepare Dirty Data (20–30 min)**: Intentionally introduce issues in pharma_sales.csv: blank Regions, negative Quantity, zero Revenue, duplicate rows.
- **Python – Data Cleaning (1.5–2 hrs)**: In day2_python.ipynb, load the CSV. Use isna().sum() to check missing values, fill missing Region with 'Unknown', remove rows where Quantity <= 0 or Revenue <= 0, drop duplicates, and save a cleaned file as pharma_sales_clean.csv.
- **SQL – Handling NULLs & Bad Rows (1–1.5 hrs)**: In SQL, inspect rows with NULL Region (WHERE Region IS NULL). Update them to 'Unknown' (if your DB allows UPDATE). Remove bad rows where Quantity <= 0 or Revenue <= 0 using DELETE statements.
- **Excel – Clean & De■duplicate (1–1.5 hrs)**: In Excel, filter blank Region cells and fill them with 'Unknown'. Use an IF formula if needed. Use Data → Remove Duplicates to clean repeated rows. Check that no negative or zero Quantity/Revenue values remain.
- **BI Tool – Clean via Power Query (1–1.5 hrs)**: In Power BI/Tableau, use the data transformation view (Power Query in Power BI) to replace null Regions with 'Unknown' and filter out invalid rows. Apply changes and verify visuals still make sense.
- **Notes (20–30 min)**: Document each type of data issue and how you fixed it in Python, SQL, Excel, and BI.

End■of■Day Deliverables (You should have):

- pharma_sales_clean.csv created from cleaned data.
- day2_python.ipynb with null-handling, row filtering, and saved clean file.
- SQL script with UPDATE and DELETE statements for cleaning.
- Cleaned Excel file with duplicates removed and no blank or invalid values.
- Updated BI model reflecting cleaned data.
- Notes describing your cleaning strategy in all tools.

Day 3 – Grouping and Summaries

Goal: Summarize data by Product and Region and understand how aggregation works in each tool.

Step■by■Step Tasks:

- **Python – Grouping & Aggregation (1.5–2 hrs)**: In a new notebook (day3_python.ipynb), load pharma_sales_clean.csv and use groupby() to compute total Revenue by Product and by Region, and average Quantity by Product. Store results in separate DataFrames and print them.
- **SQL – GROUP BY (1–1.5 hrs)**: Write queries to compute SUM(Revenue) and AVG(Quantity) grouped by Product and by Region. Experiment with ORDER BY to sort the results by total Revenue descending.
- **Excel – Pivot Tables (1–1.5 hrs)**: From the cleaned Excel data, insert a Pivot Table. Create one pivot showing sum of Revenue by Product and another showing sum of Revenue by Region. Optionally add a chart from each pivot.
- **BI Tool – Aggregated Visuals (1–1.5 hrs)**: Build visuals aligned with your grouped results: bar charts for total Revenue by Product and total Revenue by Region. Add a card showing Grand Total Revenue.
- **Notes (20–30 min)**: Write down how Python groupby(), SQL GROUP BY, Excel pivot tables, and BI visuals all express the same business question in different tools.

End■of■Day Deliverables (You should have):

- day3_python.ipynb with groupby aggregations.
- SQL file with GROUP BY queries and sorted results.
- Excel workbook with at least two pivot tables and optional pivot charts.
- BI report updated to show aggregated revenue charts.
- Notes mapping aggregation concepts across tools.

Day 4 – Joining Sales with Product Master

Goal: Combine multiple tables to enrich your sales data with product categories and prices.

Step■by■Step Tasks:

- **Create product_master.csv (20–30 min)**: Create a small reference file product_master.csv with columns Product, Category, Price_per_unit (e.g., DrugA=Antibiotic, DrugB=Analgesic, etc.).
- **Python – Merge Tables (1.5–2 hrs)**: In day4_python.ipynb, load pharma_sales_clean.csv and product_master.csv. Use merge() on Product with how='left' to add Category and Price_per_unit to each sales row. Then aggregate Revenue by Category.
- **SQL – JOIN (1.5–2 hrs)**: Create a product_master table in SQL and insert the same data. Write a LEFT JOIN query to combine pharma_sales and product_master on Product. Then compute total Revenue by Category using GROUP BY.
- **Excel – Lookup (1–1.5 hrs)**: In Excel, use VLOOKUP or XLOOKUP to bring Category and Price_per_unit from product_master into the sales sheet. Verify results, then create a pivot of Revenue by Category.
- **BI Tool – Relationships (1–1.5 hrs)**: Load both pharma_sales_clean and product_master into your BI tool. Create a relationship on Product, and build a chart of Revenue by Category. Confirm that Category comes from the master table while Revenue comes from the sales table.
- **Notes (20–30 min)**: Summarize how joins/relationships work in SQL, Python, Excel, and BI and how they let you enrich raw sales data with business context.

End■of■Day Deliverables (You should have):

- product_master.csv with product categories and prices.
- day4_python.ipynb with merge and category-level revenue.
- SQL join queries and category revenue aggregation.
- Excel workbook with successful VLOOKUP/XLOOKUP and category pivot.
- BI report visualizing revenue by Category.
- Notes explaining join concepts across tools.

Day 5 – Time■Series and Trend Analysis

Goal: Add time awareness to your analysis and visualize monthly revenue trends.

Step■by■Step Tasks:

- **Ensure Dates Are Well■Formatted (20–30 min)**: Confirm the Date column is in a consistent format (e.g., YYYY-MM-DD) in your CSV and Excel files.
- **Python – Time Features (1.5–2 hrs)**: In day5_python.ipynb, parse Date as datetime, create Month and possibly Week or Quarter columns using dt accessors, then aggregate Revenue by Month to produce a time series of monthly revenue.
- **SQL – Time■Based Grouping (1–1.5 hrs)**: In SQL, extract the year■month portion of Date (e.g., using substr() or DATE_FORMAT depending on DB) and GROUP BY that to compute monthly total Revenue. ORDER BY the month column.
- **Excel – Month Column & Charts (1–1.5 hrs)**: Add a Month column using a TEXT or DATE function, then build a line chart showing Revenue by Month. Check for upward or downward trends.
- **BI Tool – Trend Visual (1–1.5 hrs)**: Mark Date as a date type, place it on the axis of a line chart with Revenue as the value. Optionally add slicers for Product or Region to examine trends by segment.
- **Notes (20–30 min)**: Write down how to create and use date/time fields in each tool and what trends you observe in the data.

End■of■Day Deliverables (You should have):

- day5_python.ipynb with Month/Date-based aggregations.
- SQL script for monthly revenue analysis.
- Excel file with Month column and line chart.
- BI line chart showing revenue trend over time.
- Notes on trends and time handling.

Day 6 – KPIs and Basic Statistics

Goal: Turn raw numbers into meaningful business KPIs and basic statistical insights.

Step■by■Step Tasks:

- **Define KPIs (15–20 min)**: Decide on a few KPIs, e.g., Total Revenue, Average Revenue per Order, Total Quantity Sold, Revenue by Top Product/Region.
- **Python – Basic Statistics (1.5–2 hrs)**: In day6_python.ipynb, compute mean, median, min, max, and standard deviation for Revenue and Quantity. Compute correlation between Quantity and Revenue and interpret it briefly in comments.
- **SQL – KPI Aggregations (1–1.5 hrs)**: Use AVG(), SUM(), MIN(), MAX() to compute core KPIs in SQL. Optionally create a view that stores these KPIs for easy reuse.
- **Excel – KPI Summary (1–1.5 hrs)**: Build a small KPI block (cells) showing Total Revenue, Average Revenue per Order, Total Quantity, and perhaps Revenue per Region. Use cell formatting to make them visually stand out.
- **BI Tool – KPI Cards (1–1.5 hrs)**: In BI, add KPI card visuals for your main KPIs and place them at the top of the dashboard (e.g., Total Revenue, Average Revenue per Order, Total Orders). Ensure they filter correctly when using slicers.
- **Notes (20–30 min)**: Describe each KPI in plain language and note how each tool helps you compute and display it.

End■of■Day Deliverables (You should have):

- day6_python.ipynb with descriptive statistics and correlation.
- SQL script computing KPI metrics.
- Excel sheet with a formatted KPI summary block.
- BI dashboard updated with KPI cards at the top.
- Notes listing KPIs and their business meaning.

Day 7 – Introduce Safety / Adverse Events Data

Goal: Enrich your project with basic pharmacovigilance style analysis by combining sales and AE data.

Step■by■Step Tasks:

- **Create adverse_events.csv (20–30 min)**: Build a new file with columns AE_ID, Date, Product, Region, Severity, Cases, with at least 20–30 rows representing adverse event counts.
- **Python – Safety Aggregation (1.5–2 hrs)**: In day7_python.ipynb, load adverse_events.csv, aggregate total Cases by Product and by Severity, then merge the product■level Cases with the sales data to compare Revenue vs Cases per Product.
- **SQL – AE Analysis (1.5–2 hrs)**: Create an adverse_events table in SQL. GROUP BY Product and Severity to get total Cases. Join with pharma_sales by Product to see Revenue and Cases together. Identify products with high revenue and high AE counts.
- **Excel – Comparison Charts (1–1.5 hrs)**: In Excel, create a small combined summary table for each Product: Total Revenue and Total Cases. Build a bar or combo chart comparing Revenue vs Cases by Product.
- **BI Tool – Safety Dashboard Section (1–1.5 hrs)**: Load adverse_events.csv into BI, relate it to sales on Product, and create visuals such as Cases by Product and Cases by Severity. Add these to a new 'Safety' page or section of your dashboard.
- **Notes (20–30 min)**: Reflect on how safety data changes the interpretation of sales data and why this matters in pharma.

End■of■Day Deliverables (You should have):

- adverse_events.csv representing basic safety data.
- day7_python.ipynb combining sales and AE data.
- SQL scripts for AE aggregation and sales–AE joins.
- Excel charts comparing revenue and cases.
- BI dashboard with a dedicated safety/AE section.
- Notes discussing business insights from combined sales and safety.

Day 8 – Dashboard Design and Storytelling

Goal: Turn your BI views into a clear, manager■friendly dashboard with better layout and interactivity.

Step■by■Step Tasks:

- **Review Existing Dashboard (20–30 min)**: Open your current BI report and note what looks cluttered, what's missing, and what a manager would care about most.
- **Design Layout on Paper (20–30 min)**: Sketch a simple layout: top row KPIs, middle row sales charts, bottom row safety charts and filters/slicers.
- **BI Tool – Dashboard Refinement (2–3 hrs)**: Rearrange visuals according to your sketch. Ensure KPIs are at the top, core charts are central, and filters/slicers for Date, Product, Category, and Region are placed logically. Adjust titles, labels, and axis names for clarity.
- **Add Interactivity (1–1.5 hrs)**: Configure slicers/filters to control multiple visuals. Test scenarios such as filtering on a single Product or Region and checking that KPIs and charts update as expected.
- **Excel – Simple Management Summary (1–1.5 hrs)**: On a new sheet, create a compact summary using key charts (copy/paste) and KPI values, as if you were sending a one■page report to a manager.
- **Notes (20–30 min)**: Write what story your dashboard tells and how a pharma manager might use it to make decisions.

End■of■Day Deliverables (You should have):

- A cleaner, more structured BI dashboard layout.
- Functional slicers/filters for key dimensions.
- An Excel one■page summary sheet with key charts and KPIs.
- Notes describing the narrative of your dashboard.

Day 9 – Automation with Python

Goal: Automate the most repetitive parts of your analysis so new reports can be produced quickly.

Step■by■Step Tasks:

- **Define a Reporting Workflow (20–30 min)**: Decide what you want your Python script to do: load latest CSVs, clean them, compute summaries, and export a daily/weekly report.
- **Python – Automation Script (2–3 hrs)**: In day9_automation.py or a notebook, write code that loads pharma_sales.csv and adverse_events.csv, applies the cleaning steps from earlier days, computes key aggregations (e.g., revenue by month, revenue and cases by product), and saves the results to new CSV or Excel files in an 'outputs' folder.
- **Optional Python – Basic Plots (1–1.5 hrs)**: Use matplotlib (or similar) to generate and save a couple of static PNG charts (e.g., Revenue by Product, Cases by Product) automatically as part of the script.
- **BI / Excel – Connect to Outputs (1–1.5 hrs)**: Optionally point Excel or BI to the cleaned/aggregated output files so refreshing data in the future will be fast and consistent.
- **Notes (20–30 min)**: Document how to run your script (inputs, outputs, and steps) as if handing it to another analyst.

End■of■Day Deliverables (You should have):

- day9_automation.py (or notebook) implementing your end■to■end data preparation and summary pipeline.
- Generated output files (cleaned and aggregated data).
- Optionally, automatically generated charts saved as PNG images.
- Notes describing how your automation works and how to reuse it.

Day 10 – Finalize and Package the Project

Goal: Turn all your work into a polished, portfolio■ready project that you can show to recruiters and hiring managers.

Step■by■Step Tasks:

- **Review All Artifacts (30–45 min)**: Collect your main files: notebooks, SQL scripts, Excel workbooks, BI report, CSVs, and notes. Organize them into a logical folder structure.
- **Polish Code and Queries (1.5–2 hrs)**: Clean up Python notebooks/scripts by adding comments and removing unused code. Tidy SQL scripts and label sections for different analyses.
- **Finalize Dashboard (1–1.5 hrs)**: Make sure all dashboard visuals are named, labeled, and arranged clearly. Add a title, a subtitle describing the scope (e.g., 'Pharma Sales & Safety Analytics'), and ensure slicers work correctly.
- **Write a Short Project Report (1–1.5 hrs)**: In a document (e.g., PDF or Word), write 1–2 pages covering: Problem Statement, Data Sources, Tools Used, Key Steps (Cleaning, Aggregation, AE analysis, Automation), and Key Insights. Include 2–4 screenshots of important charts.
- **Portfolio Packaging (1–1.5 hrs)**: Decide how you will share this project (GitHub, PDF, BI file). Rename files clearly (e.g., Pharma_Analytics_Project_1) and prepare a short description you can paste into your resume or LinkedIn.
- **Notes & Reflection (20–30 min)**: Reflect on what you found hardest, what came naturally, and which areas you want to deepen in the next 10■day cycle.

End■of■Day Deliverables (You should have):

- A well■organized project folder containing all relevant code, data, and reports.
- Cleaned and commented Python notebooks/scripts and SQL files.
- Final BI dashboard file ready to demo.
- A 1–2 page project report document with screenshots.
- A short portfolio description you can reuse on your resume/LinkedIn.
- Reflection notes guiding what to focus on in the next learning cycle.