ChatGPT

End-to-End Data Analytics Project with Microsoft Fabric

Project Overview

This project is a comprehensive **retail analytics scenario** built on Microsoft Fabric. It will guide you through the entire data journey – from **data ingestion** to **data visualization** – leveraging key Fabric services in one hands-on project. Using the fictional **Wide World Importers (WWI)** dataset (a sample retail database), you will ingest raw sales and dimension data, transform it through a *medallion architecture* (bronze/silver/gold layers), and build interactive reports. This end-to-end project touches on all major Fabric components: Data Factory pipelines for ingestion, a Lakehouse (or Data Warehouse) for storage, Spark notebooks for data preparation, Power BI for modeling and visualization, and Pipelines for orchestration. In essence, it demonstrates the process **"from data acquisition to data consumption,"** showcasing how different Fabric experiences integrate in a single workflow [1]. By completing this project, you'll gain a practical understanding of Fabric's unified analytics capabilities in a real-world context.

Required Services and Data

- **Microsoft Fabric Tenant**: You'll need access to Microsoft Fabric (e.g. a Power BI Premium or Fabric trial account). Ensure you have a Fabric-enabled workspace (the project assumes you can create Fabric items like Lakehouses, pipelines, notebooks, etc.) [2]. If you don't have Fabric access, sign up for a free trial [3].
- **Data Factory (Pipelines)**: The Data Factory experience in Fabric will be used to create pipelines for data ingestion and workflow orchestration. No separate Azure Data Factory service is needed – Fabric's integrated Data Factory provides 200+ connectors for various sources [4].
- **Lakehouse / Storage**: We will use a **Fabric Lakehouse** to store data. Fabric Lakehouse is a OneLake-based data lake with Delta tables. (Alternately, you could use a Fabric Data Warehouse for a purely SQL-based approach [5], but this guide uses a Lakehouse for a lakehouse-style architecture).
- **Notebook (Spark runtime)**: Fabric's notebook experience (powered by Spark) will be used for data exploration and transformation. PySpark will help convert raw files into cleaned Delta tables and create aggregated data. Fabric provides a built-in Spark runtime ("Live Pool") available in each workspace [6] [7].
- **Power BI (Analytics)**: Power BI is deeply integrated into Fabric for modeling and visualization. We will create a Power BI **semantic model** directly connected to the Lakehouse (using the new **DirectLake** mode) and build a report on top of it [8]. This will enable interactive analysis of the transformed data without needing to export or duplicate data.
- **Sample Dataset**: The project uses the **Wide World Importers (WWI)** sample data – a realistic retail dataset provided by Microsoft. The data (sales facts and related dimensions) is available in a public Azure Blob Storage container as Parquet files [9]. We will connect to this external source to pull data into Fabric. (You can substitute any similar open dataset if you prefer, but WWI is conveniently provided for Fabric tutorials [10] [11].) No local data download is required – the pipeline will read from the blob URL directly.

Implementation Steps

Below is a step-by-step plan to implement the project. Each step corresponds to a key stage in the data pipeline, leveraging a different Fabric component:

1. **Create a Fabric Workspace and Lakehouse** – Begin by creating a new **Fabric workspace** (using the Power BI portal). In the workspace, create a **Lakehouse** to serve as your data lake store. This Lakehouse will automatically provision storage in OneLake and a SQL endpoint for connectivity. *(In Fabric, the Lakehouse will appear as an item with a "Tables" section for Delta tables and a "Files" section for raw files.)* [5] [12]. Give the Lakehouse a name (e.g., "WWI Retail Lakehouse"). This sets up the storage foundation for ingesting data.

2. **Ingest Data using Data Factory Pipeline** – Next, use Fabric's Data Factory experience to ingest the sample data into the Lakehouse. In your workspace, create a new **Pipeline**. Inside the pipeline, add a **Copy Data** activity (under Data Factory's drag-and-drop activities) to transfer data from the source into the Lakehouse. For the source, choose *Azure Blob Storage* and use the provided public URL for the WWI sample data. (The WWI Parquet files are in a public blob container, so you can connect with anonymous access [9]. For example, the container URL is `https://fabrictutorialdata.blob.core.windows.net/sampledata/` as given in Microsoft's tutorial.) Configure the Copy activity to read from the `WideWorldImportersDW/parquet` directory in that container (which contains multiple tables) [13]. For the destination, select your Lakehouse and specify the **Files** section (e.g., copy into a folder like `Files/wwi-raw-data`). This pipeline will effectively land all source Parquet files into your Lakehouse's file system. Save and **Run** the pipeline. After execution (it may take a minute for all files to copy), verify in the Lakehouse > Files section that a new folder (e.g. *wwi-raw-data*) with the data has appeared [14]. *Result:* you have ingested the raw data (bronze layer) into Fabric's one lake.

3. **Prepare and Transform Data with Notebooks (PySpark)** – With raw data in place, the next step is to create clean Delta tables (silver/gold layers). Fabric allows two approaches here – a code-first approach using notebooks/Spark, or a low-code approach using Dataflows or Pipeline data transformations [5] [15]. In this project, we'll use a **Notebook with PySpark** for hands-on experience. Create a new Notebook in your workspace and attach it to the Lakehouse (you can create the notebook from the Lakehouse UI, ensuring the notebook's *default Spark pool* is used). In the notebook, use PySpark to read the Parquet files from the Lakehouse *Files* and write them out as **Delta tables** in the Lakehouse *Tables* section. For example, you might load the fact sales data and partition it by date before saving as a Delta table [16] [17]. Do the same for dimension tables: read each Parquet file (e.g., `dimension_customer`, `dimension_date`, etc.), drop any unused columns, and write to Tables with the same name [18] [19]. Fabric notebooks support standard Spark APIs – when you write DataFrames to a `Tables/` path in the Lakehouse, Fabric's **automatic table registration** will make them appear as tables in the Lakehouse item (no manual `CREATE TABLE` needed) [20]. After running the notebook, refresh the Lakehouse to confirm that your Delta tables (fact and dimension tables) are now listed under the Tables section [21].

Data Transformation (Business Logic): Once the base tables are created, you can perform further transformations or create aggregate tables for reporting. For example, join the fact and dimension tables to create a **daily sales aggregate**. You can either use PySpark code or Spark SQL for this step – Fabric supports both, so choose based on your familiarity [22]. In PySpark, you might create DataFrames from each table and then perform joins and groupBy aggregations to compute totals and profits by date and city [23] [24], writing the results as a new Delta table (e.g., `aggregate_sale_by_date_city`) [25]. Alternatively, using Spark SQL, you can create a temporary view joining the tables and use a SQL query to aggregate, then save that as a Delta table [26] [27]. The official tutorial demonstrates both a PySpark approach and a Spark SQL approach that yield similar results [28] [29]. After this step, you should have one or more curated **Gold tables** ready for analysis (for instance, an aggregated sales table, along with the cleaned dimension tables).

1. **Create a Power BI Data Model (DirectLake) and Visualize** – With the transformed data in the Lakehouse, you can now leverage Power BI for modeling and reporting – all within the Fabric environment. Open the Lakehouse and switch to the **SQL Analytics Endpoint** (this is the built-in SQL interface for the Lakehouse) [30]. You will see all your Lakehouse tables (fact, dimensions, aggregates) available via this endpoint. Click **New Semantic Model** to create a Power BI dataset directly on these tables [31]. In the dialog, give the model a name and select the tables to include (e.g., the fact table and related dimensions). Choose *DirectLake* mode if prompted – this enables Power BI to load data from the Lakehouse files directly into memory for extremely fast querying [32]. After the dataset is created, open it in the Power BI web modeling view. Here, define relationships between your fact and dimension tables (just like designing a star schema data model). For example, join `fact_sale`'s CityKey to `dimension_city`'s CityKey, SalespersonKey to the Employee dimension, CustomerKey to Customer dimension, etc. (all as one-to-many relationships) [33] [34]. Once relationships are set, the data model is essentially complete – the DirectLake dataset will fetch the data from the Lakehouse parquet files on-the-fly and keep them updated. Now click **New Report** to create a Power BI report on this dataset. In the report canvas, you can drag fields from your tables to build visuals. For instance, you might create a card showing total profit, a bar chart of profit by product category or by city, and a line chart of sales over time. Using the WWI sample data, you could visualize metrics like total sales by month, profit by territory, etc. Design the report with a title and a few visuals as you see fit [35]. **Result:** You have built an interactive Power BI report that directly reads the Fabric Lakehouse data (no manual exports needed). Any changes to the Lakehouse data (e.g., if new data is ingested later) can be reflected in near real-time via DirectLake. This demonstrates Fabric's tight integration of lakehouse and BI components.

2. **Orchestrate and Automate with Pipelines** – Finally, to make this a truly production-style solution, you can automate the data refresh process using Fabric pipelines. In a real scenario, you might schedule the pipeline to run daily or when new files arrive. Using the Data Factory pipeline created earlier, you can add more steps to orchestrate the entire flow. For example, after the **Copy Data** activity that ingests raw files, add a **Notebook** activity to run your transformation notebook within the pipeline (ensuring that after data lands, it gets processed into tables). You could also add a **Data Flow** activity if you prefer a no-code transformation for some steps, or include **Notification** steps (such as sending an email on success/failure). Fabric pipelines support scheduling triggers and alerting. In the official Data Factory end-to-end tutorial, Microsoft shows how to "use automation and notification to create a complete data integration scenario" after ingesting and transforming data [36]. In our project, an automated pipeline could look like: *Copy data from Blob -> Run Notebook to process data -> (Optionally) Refresh the Power BI dataset*. Since our Power BI dataset uses DirectLake, it doesn't require an explicit refresh for new data; however, if using a warehouse or if you wanted to trigger a dataset refresh, you could add an API call or use Fabric's integration to refresh the report. The key is that **Fabric's pipeline can orchestrate and schedule the end-to-end flow** – you can run it on a schedule or on-demand, and monitor the runs centrally. This way, the entire solution – from data ingestion, through transformation, to report update – is automated. (As noted in the Fabric tutorial, you can *"orchestrate and schedule data ingestion and transformation flow with a pipeline"* [37].)

Learning Outcomes

By completing this project, you will gain experience with most core features of Microsoft Fabric in an integrated scenario:

- **Data Ingestion with Data Factory**: Learn how to connect to external data sources (using Fabric's connectors) and ingest data into OneLake. You saw how to use a Copy Data activity to bring in data from a public blob storage into the Fabric Lakehouse [38]. This demonstrates Fabric's ability to use over 200 connectors for seamless data movement [4].
- **Lakehouse Storage & Delta Lake**: Understand how Fabric's Lakehouse stores data in Delta format, enabling a *medallion architecture* (raw bronze data in Files, refined silver and gold data in Delta tables). All Fabric engines (Spark, SQL, Power BI) can access the same data without duplication [5]. You practiced creating Delta tables and observed the automatic catalog registration in action.
- **Data Transformation with Notebooks (Spark)**: Get hands-on with Fabric notebooks and PySpark to clean and transform data. You used Python/PySpark to read Parquet files, generate new columns, join data, and write results as Delta tables [16] [23]. This illustrated Fabric's support for code-first data engineering and how Spark integrates with OneLake. You also learned that Fabric offers low-code dataflows as an alternative for transformations, and even Spark SQL for those more comfortable with SQL syntax [26] [29].
- **Power BI Integration (DirectLake Mode)**: See the power of native integration with Power BI. Using DirectLake, you built a semantic model directly on the lakehouse data, eliminating the need to import or refresh data in the traditional sense [32]. This taught you how to define relationships and metrics in Power BI on top of data lake files and create interactive visuals. You experienced how changes in the underlying data could be reflected quickly in reports, combining the freshness of DirectQuery with the speed of import mode [39] [40].
- **Orchestration and Automation**: Learn how to tie everything together with Fabric pipelines. By automating the copy and notebook processes in a pipeline, you saw how a single pipeline can run the entire workflow. You also learned about scheduling pipelines and adding notifications or triggers to build a production-ready data pipeline [41] [36]. Monitoring pipeline runs in Fabric's interface is another valuable skill from this project.
- **End-to-End Solution Architecture**: Overall, this project gave you a blueprint of a real-world analytics solution in Fabric – from raw data ingestion to BI reporting in one platform. It highlighted Fabric's **unified environment**, where data engineers and BI developers can collaborate on one copy of data. You touched on multiple roles: data engineer (ingesting and transforming data), data analyst (modeling and reporting in Power BI), and data ops (scheduling and managing the pipeline). This holistic understanding will be directly applicable to designing your own Fabric solutions that span the full analytics lifecycle [1] [42].

References and Further Reading:

- Microsoft Fabric End-to-End Tutorial: *Lakehouse Scenario (Wide World Importers)* – Official step-by-step guide covering a similar project [43] [42]. Includes instructions for setting up the workspace, ingesting WWI data, transforming with notebooks, and building a Power BI report.
- Microsoft Fabric Data Warehouse Tutorial – An alternative end-to-end project focusing on a SQL Warehouse (if you prefer T-SQL for transformation) [44] [45]. It covers pipeline ingestion, T-SQL transformations, time-travel, and creating a Power BI report via a warehouse.
- Fabric Sample GitHub Repository – Contains **sample notebooks and pipeline JSON** for the WWI tutorial (Lakehouse scenario) [46]. You can download ready-made notebooks ("01 - Create Delta Tables", etc.) from the repository to accelerate the development [47].

- Community Example – *“My First End-to-End Project in Microsoft Fabric”* (Reddit/YouTube) – A user’s walkthrough of a Fabric project similar to this guide, using pipelines for ingestion, dataflows and notebooks for transformation, a Lakehouse + Warehouse combination, and Power BI reporting [48] [49]. This can provide additional context and tips for building Fabric projects in practice.

---

[1] [2] [3] [4] [5] [8] [10] [11] [12] [15] [37] [41] [42] [43] Lakehouse end-to-end scenario: overview and architecture - Microsoft Fabric | Microsoft Learn
https://learn.microsoft.com/en-us/fabric/data-engineering/tutorial-lakehouse-introduction

[6] [7] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [46] [47] Lakehouse tutorial - Prepare and transform lakehouse data - Microsoft Fabric | Microsoft Learn
https://learn.microsoft.com/en-us/fabric/data-engineering/tutorial-lakehouse-data-preparation

[9] [13] [14] [38] Lakehouse tutorial - Ingest data into the lakehouse - Microsoft Fabric | Microsoft Learn
https://learn.microsoft.com/en-us/fabric/data-engineering/tutorial-lakehouse-data-ingestion

[30] [31] [32] [33] [34] [35] [39] [40] Lakehouse tutorial - Build a report - Microsoft Fabric | Microsoft Learn
https://learn.microsoft.com/en-us/fabric/data-engineering/tutorial-lakehouse-build-report

[36] End-to-end tutorials in Microsoft Fabric - Microsoft Fabric | Microsoft Learn
https://learn.microsoft.com/en-us/fabric/fundamentals/end-to-end-tutorials

[44] [45] Data Warehouse Tutorial: Introduction - Microsoft Fabric | Microsoft Learn
https://learn.microsoft.com/en-us/fabric/data-warehouse/tutorial-introduction

[48] [49] My First End-to-End Project in Microsoft Fabric – Full Walkthrough with Lakehouse + DataWarehouse + Power BI : r/MicrosoftFabric
https://www.reddit.com/r/MicrosoftFabric/comments/1kpbiuj/my_first_endtoend_project_in_microsoft_fabric/