

# Big Data Processing with Spark or Hive

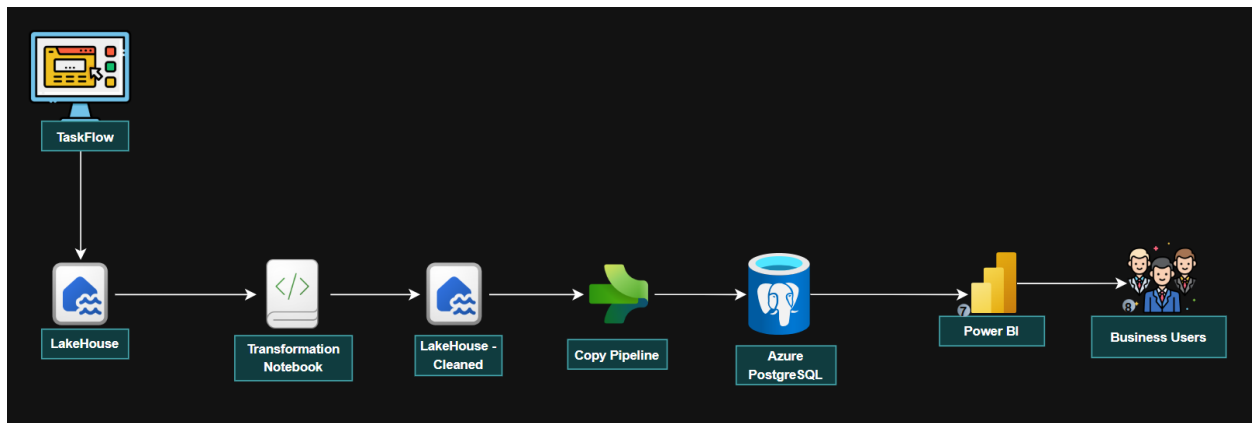
## Introduction

This section is broken down into 3 parts, with the first part having to do with the data generation of over 20 million records, then transforming and loading of the data into PostgreSQL Database

## Project Architecture: Enterprise Task Management System

Background story - Assuming we have an application called **TaskFlow** which is a mobile enterprise application used by a multinational technology consulting firm with offices across multiple time zones. The application allows employees to track, manage, and log their work on various client projects.

Let's assume we have an Enterprise solution called TaskFlow that generates over 20 million records of data and is stored in the Lakehouse File Folder in Microsoft Fabric. We will be using Fabric due to its cost and cluster without needing to spend extra on another **Platform as a Service (PaaS)**.



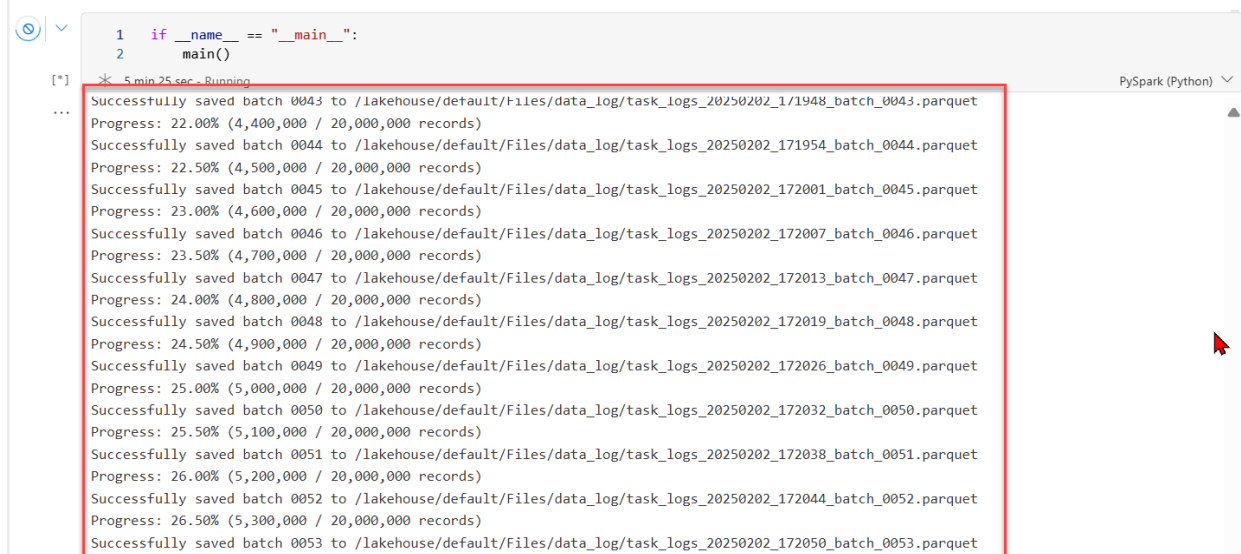
*TaskFlow Architecture*

# Section 1: Generate 20 Million Record from TaskFlow Application

From the architecture above you will notice the amount of data being generated using the ***Faker library in Python*** and running the code using the Fabric Spark Cluster which allows us to process as much data as possible

## Data Generated to Lakehouse












You will notice the image below shows how the data are being generated and stored to the Microsoft Fabric Lakehouse File Folder Directory.



```
1 if __name__ == "__main__":
2     main()

[*] * 5 min 25 sec - Running
...
Successfully saved batch 0043 to /lakehouse/default/Files/data_log/task_logs_20250202_171948_batch_0043.parquet
Progress: 22.00% (4,400,000 / 20,000,000 records)
Successfully saved batch 0044 to /lakehouse/default/Files/data_log/task_logs_20250202_171954_batch_0044.parquet
Progress: 22.50% (4,500,000 / 20,000,000 records)
Successfully saved batch 0045 to /lakehouse/default/Files/data_log/task_logs_20250202_172001_batch_0045.parquet
Progress: 23.00% (4,600,000 / 20,000,000 records)
Successfully saved batch 0046 to /lakehouse/default/Files/data_log/task_logs_20250202_172007_batch_0046.parquet
Progress: 23.50% (4,700,000 / 20,000,000 records)
Successfully saved batch 0047 to /lakehouse/default/Files/data_log/task_logs_20250202_172013_batch_0047.parquet
Progress: 24.00% (4,800,000 / 20,000,000 records)
Successfully saved batch 0048 to /lakehouse/default/Files/data_log/task_logs_20250202_172019_batch_0048.parquet
Progress: 24.50% (4,900,000 / 20,000,000 records)
Successfully saved batch 0049 to /lakehouse/default/Files/data_log/task_logs_20250202_172026_batch_0049.parquet
Progress: 25.00% (5,000,000 / 20,000,000 records)
Successfully saved batch 0050 to /lakehouse/default/Files/data_log/task_logs_20250202_172032_batch_0050.parquet
Progress: 25.50% (5,100,000 / 20,000,000 records)
Successfully saved batch 0051 to /lakehouse/default/Files/data_log/task_logs_20250202_172038_batch_0051.parquet
Progress: 26.00% (5,200,000 / 20,000,000 records)
Successfully saved batch 0052 to /lakehouse/default/Files/data_log/task_logs_20250202_172044_batch_0052.parquet
Progress: 26.50% (5,300,000 / 20,000,000 records)
Successfully saved batch 0053 to /lakehouse/default/Files/data_log/task_logs_20250202_172050_batch_0053.parquet
```

You will notice the data are stored in parquet (columnar) file format; this helps compress the data to a smaller form.

Files > data_log				
Name	Date modified	Type	Size	
 task_logs_20250202_171518_batch_0000.parquet	2/2/2025 6:15:20 PM	parquet	5 MB	
 task_logs_20250202_171526_batch_0001.parquet	2/2/2025 6:15:26 PM	parquet	5 MB	
 task_logs_20250202_171532_batch_0002.parquet	2/2/2025 6:15:32 PM	parquet	5 MB	
 task_logs_20250202_171538_batch_0003.parquet	2/2/2025 6:15:39 PM	parquet	5 MB	
 task_logs_20250202_171545_batch_0004.parquet	2/2/2025 6:15:45 PM	parquet	5 MB	
 task_logs_20250202_171551_batch_0005.parquet	2/2/2025 6:15:51 PM	parquet	5 MB	
 task_logs_20250202_171557_batch_0006.parquet	2/2/2025 6:15:57 PM	parquet	5 MB	
 task_logs_20250202_171603_batch_0007.parquet	2/2/2025 6:16:04 PM	parquet	5 MB	
 task_logs_20250202_171610_batch_0008.parquet	2/2/2025 6:16:10 PM	parquet	5 MB	
 task_logs_20250202_171616_batch_0009.parquet	2/2/2025 6:16:16 PM	parquet	5 MB	
 task_logs_20250202_171622_batch_0010.parquet	2/2/2025 6:16:22 PM	parquet	5 MB	

## Section 2: Data Transformation Using PySpark

After successfully generating and loading the data of batch 100,000 records each into the **lakehouse/Files/data\_log** directory in Fabric Notebook.

***The following transformations were performed on the Parquet files:***

All transformations done are made available on the Python code.

### Time Conversions

- Converts Unix timestamps to DateTime
- Adds columns for day of week, hour, and duration

### Data Quality Checks:

- Counts null values
- Shows status distribution

### Project Analytics:

- Tasks per project
- Completion rates
- Average durations

- Total hours logged

### Employee Analytics:

- Individual performance metrics
- Productivity scores
- Completion rates

### Time-based Analysis:

- Task distribution by day and hour
- Average durations
- Total hours by time

### Priority Analysis:

- Task counts by priority
- Average durations
- Completion rates

## Load Transformed File New Directory

After successfully transforming and splitting the files into their respective sub-data, we then loaded the file into a new directory called ***/Files/clean\_data\_log***

Root folder > clean\_data\_log

📁 employee\_metrics

📁 priority\_metrics

📁 project\_metrics

📁 time\_analysis

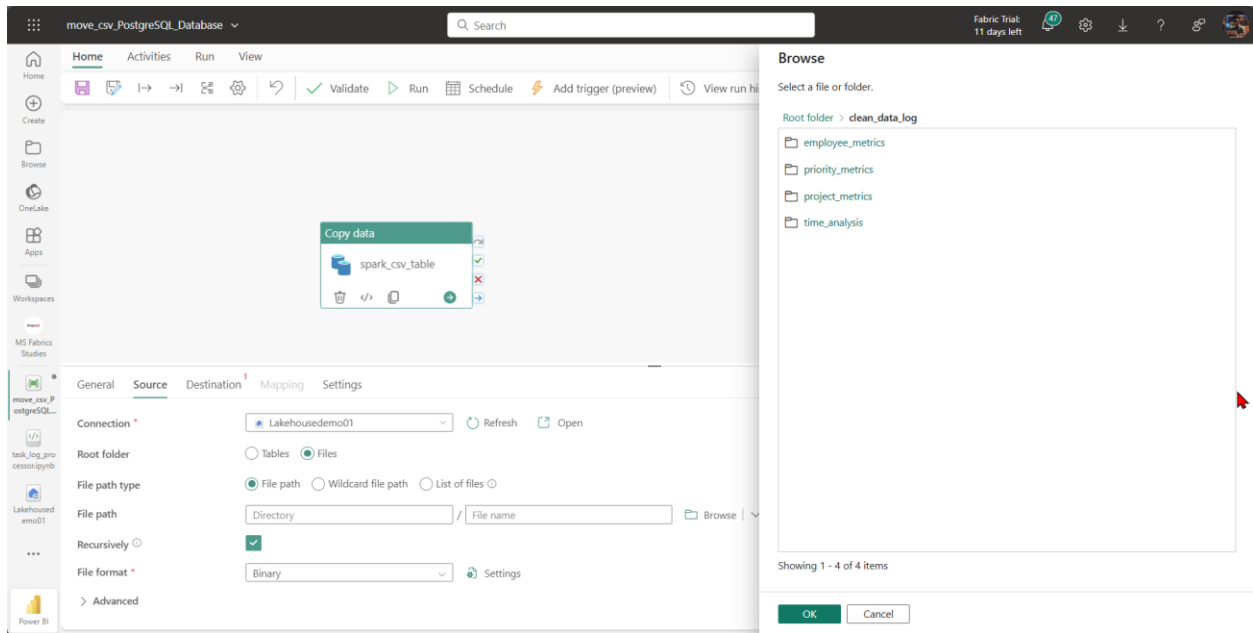
## Section 3: Load Transformed Data to Azure PostgreSQL Database

For this final step we will be loading the cleaned/transformed data to Azure PostgreSQL Database table, we created a schema called Spark which we will use in the loading.

The Fabric Pipeline is used in moving data from the source which is the Lakehouse folder to the destination which is the Azure PostgreSQL Database.

### Create Copy Pipeline

Using the Pipeline in Microsoft Fabric, we are going to create a copy activity(Action) and move the data from the source to the destination. We are going to repeat the process multiple times for the different folders we have.



Click on the run button after completing all necessary connections.

The screenshot shows the Azure Databricks interface. At the top, the 'Run' button is highlighted with a red arrow. Below the main workspace, the 'Output' tab is active, showing a table with one row of data for the 'spark\_csv\_table' activity.

Activity name	Activity status	Run start	Duration	Input	Output
spark_csv_table	Succeeded	2/2/2025, 7:31:44 PM	26s	-	-

## Confirm Data Load

Confirm data load by heading to the Azure PostgreSQL database table and running the select command.

The screenshot shows a SQL query editor with a select statement and a table of results. The table has columns: project\_name, total\_tasks, completed\_tasks, avg\_duration, total\_hours\_logged, avg\_hours\_per\_task, and completion\_rate.

```
select * from spark.time_analysis
select * from spark.project_metrics
select * from spark.employee_metrics
select * from spark.time_analysis
```

project_name	total_tasks	completed_tasks	avg_duration	total_hours_logged	avg_hours_per_task	completion_rate
still	20,619	6,920	4.24	92,628	4.49	33.56
travel	20,587	6,898	4.26	92,138	4.48	33.51
art	20,559	6,820	4.25	92,150	4.48	33.17
hope	20,441	6,768	4.25	92,026	4.5	33.11
those	20,718	6,928	4.24	93,098	4.49	33.44
few	20,449	6,741	4.26	91,658	4.48	32.96
some	20,606	6,800	4.26	93,042	4.52	33
recognize	20,524	6,909	4.25	92,487	4.51	33.66
include	20,677	6,887	4.26	93,190	4.51	33.31
operation	20,815	6,956	4.26	93,637	4.5	33.42
television	20,616	6,825	4.25	92,220	4.47	33.11
staff	20,615	6,972	4.25	92,779	4.5	33.82
positive	20,267	6,687	4.23	91,149	4.5	32.99
often	20,233	6,687	4.26	90,987	4.5	33.05
explain	20,606	6,878	4.25	92,960	4.51	33.38
watch	20,420	6,847	4.25	92,003	4.51	33.53
growth	20,568	6,928	4.24	92,678	4.51	33.68
film	20,725	6,951	4.26	93,250	4.5	33.54

# Conclusion

From this, we have been able to generate millions of records of data using the Faker library, load the information into Microsoft Fabric Lakehouse, transform the data using PySpark and finally load the data into Azure PostgreSQL using the Fabric Pipeline.