peopleTECH
A Quest Global Company

Modernizing Analytics with Microsoft Fabric:

Unlocking Power BI’s Full Potential

Table of Contents

Executive Summary

As organizations look to modernize their analytics platforms, the combination of Microsoft Power BI and Microsoft Fabric offers a powerful opportunity to simplify architectures, reduce cost, and enable real-time, scalable insights. For customers who have invested in star-schema data warehouses—especially those built on Snowflake or SQL Server—Fabric provides a unified platform that tightly integrates Power BI with storage, transformation, and governance layers.

People Tech Group has developed advanced migration solutions, including a proprietary T-SQL to PySpark migration framework, to support customers transitioning legacy data systems to Microsoft Fabric's lake-first architecture.

1. Why Microsoft Fabric?

The Challenge

Many organizations have built data platforms using:

- ➢ Snowflake, Synapse, or SQL Server for storage
- ➢ SSIS and T-SQL for data transformation
- ➢ Power BI for reporting

snowflake | Azure Synapse Analytics | Microsoft SQL Server | Power BI

While these architectures work, they involve:

Complex ETL pipelines and data movement
with scattered scheduling, limited observability, and high maintenance overhead.

Redundant storage for Power BI imports
along with high latency due to repeated data movement and lack of Direct Lake access.

Siloed governance, fragmented security and scattered metadata
across tools, making lineage and compliance difficult.

High operational and licensing costs
from tightly coupled compute/storage, inefficient scaling, and overlapping tools.

The Solution

01 **OneLake** as the universal storage layer

02 **Lakehouse** and Warehouse for compute

03 **Data Factory** for orchestration

04 **Power BI** deeply integrated for analytics

05 **Purview** for security and governance

2. Power BI and Fabric: A Tightly Integrated Future

2.1 DirectLake Mode: Performance Without Duplication

Power BI’s new DirectLake mode allows reports to read delta-parquet files directly from OneLake—eliminating the need for import mode or DirectQuery workarounds.

- ➢ Near real-time performance
- ➢ No refresh scheduling
- ➢ Zeros data duplication

2.2 Semantic Models in Fabric

Semantic models (datasets) are now native to Fabric:

- ➢ Stored, versioned, and deployed in the same workspace
- ➢ Managed alongside Lakehouses and Warehouses
- ➢ Integrated with Git for CI/CD workflows

2.3 Governance, Lineage, and Security

Fabric natively integrates with Microsoft Purview:

- Full data lineage from source to report
- Centralized catalog and metadata management
- Enterprise-grade access controls via Azure AD

2.4 AI & Copilot Enhancements

Fabric enables Power BI Copilot to:

- Generate semantic models from raw tables
- Suggest DAX measures
- Build reports from natural language prompts

3. The Need for a Structured Migration Strategy

Assessment First

Migrating to Fabric isn't a lift-and-shift—it requires a strategic roadmap:

- Inventory of current T-SQL logic, SSIS packages, and Power BI datasets
- Understanding performance and refresh pain points
- Identifying critical data domains to pilot

Component Mapping

Existing Stack -> Microsoft Fabric Equivalent

- SQL Server / Snowflake -> Fabric Warehouse / Lakehouse
- SSIS Pipelines -> Fabric Data Factory Pipelines
- Stored Procedures / T-SQL -> PySpark in Notebooks / T-SQL in Warehouse
- Power BI Premium -> Power BI in Fabric with DirectLake

4. People Tech Group: Accelerating Your Fabric Journey

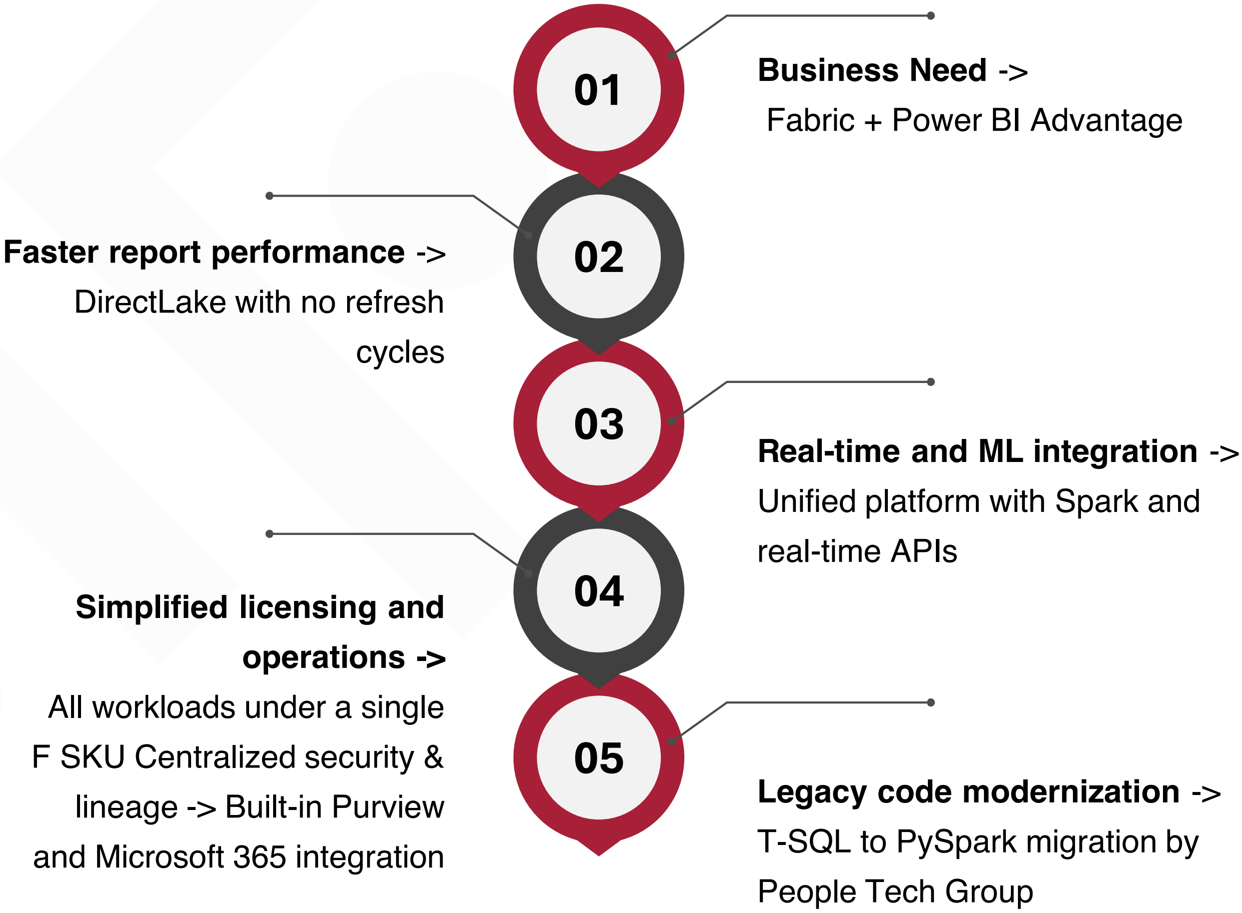
People Tech Group has developed a T-SQL to PySpark Migration Framework that automates and optimizes the conversion of legacy SQL code into scalable PySpark logic for Fabric Lakehouses. Key features include:

- Syntax and pattern detection
- Automated translation of stored procedures and transformations
- Integration with Fabric Notebooks and Pipelines
- Support for incremental loads, joins, and complex transformations

Benefits:

- Accelerates cloud migration timelines
- Reduces reliance on legacy SSIS and on-prem SQL logic
- Optimized for Microsoft Fabric’s Spark-native architecture

5. Value Drivers for Migration



01 **Business Need ->** Fabric + Power BI Advantage

02 **Faster report performance ->** DirectLake with no refresh cycles

03 **Real-time and ML integration ->** Unified platform with Spark and real-time APIs

04 **Simplified licensing and operations ->** All workloads under a single F SKU Centralized security & lineage -> Built-in Purview and Microsoft 365 integration

05 **Legacy code modernization ->** T-SQL to PySpark migration by People Tech Group

Sample Code Conversion from T-SQL to PySpark

Scenario: Monthly Sales Report with Joins, Window Functions, and CTEs

T-SQL Code

```
WITH MonthlySales AS (
    SELECT
        s.SalespersonID,
        s.Region,
        MONTH(t.TransactionDate) AS SalesMonth,
        YEAR(t.TransactionDate) AS SalesYear,
        SUM(t.SalesAmount) AS MonthlyTotal
    FROM
        Sales.SalesTransactions t
    INNER JOIN
        Sales.Salespeople s ON t.SalespersonID = s.SalespersonID
    WHERE
        t.TransactionDate BETWEEN '2023-01-01' AND '2023-12-31'
    GROUP BY
        s.SalespersonID, s.Region, MONTH(t.TransactionDate), YEAR(t.TransactionDate)
),
RankedSales AS (
    SELECT
        *,
        RANK() OVER (PARTITION BY Region, SalesYear, SalesMonth ORDER BY
MonthlyTotal DESC) AS RankInRegion
    FROM
        MonthlySales
)
SELECT *
FROM RankedSales
WHERE RankInRegion <= 3
ORDER BY SalesYear, SalesMonth, Region, RankInRegion;
```

Sample Code Conversion from T-SQL to PySpark

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import month, year, sum, rank
from pyspark.sql.window import Window

# Initialize Spark session
spark = SparkSession.builder.getOrCreate()

# Load tables (assume delta format in Fabric Lakehouse)
transactions = spark.read.format("delta").table("Sales.SalesTransactions")
salespeople = spark.read.format("delta").table("Sales.Salespeople")

# Filter transaction date range
filtered = transactions.filter("TransactionDate BETWEEN '2023-01-01' AND '2023-12-31'")

# Join salespeople
joined = filtered.join(salespeople, on="SalespersonID", how="inner")

# Create derived columns and aggregate
monthly_sales = (
    joined
    .withColumn("SalesMonth", month("TransactionDate"))
    .withColumn("SalesYear", year("TransactionDate"))
    .groupBy("SalespersonID", "Region", "SalesMonth", "SalesYear")
    .agg(sum("SalesAmount").alias("MonthlyTotal"))
)

# Define window for ranking
window_spec = Window.partitionBy("Region", "SalesYear",
"SalesMonth").orderBy(monthly_sales["MonthlyTotal"].desc())

# Apply rank function
ranked_sales = monthly_sales.withColumn("RankInRegion", rank().over(window_spec))

# Filter top 3 per region/month
top_sales = ranked_sales.filter("RankInRegion <= 3")

# Final sort
result = top_sales.orderBy("SalesYear", "SalesMonth", "Region", "RankInRegion")

# Optionally display or write
result.show()
```

6. Migration Notes

T-SQL Feature	PySpark Equivalent
CTEs	Layered DataFrame transformations
JOIN	.join()
MONTH(), YEAR()	month(), year() from pyspark.sql.functions
RANK() OVER (...)	rank().over(Window(...))
PARTITION BY, ORDER BY	Window.partitionBy().orderBy()
WHERE, GROUP BY	.filter(), .groupBy().agg()
Final SELECT/ORDER	.select(...), .orderBy(...)

7. Conclusion: Power BI’s Next Evolution

For customers already using Power BI and a star-schema data model, Microsoft Fabric is more than a new platform—it’s the next stage of analytics evolution. By combining performance, governance, and simplified operations, it unlocks a future-ready data environment.

People Tech Group is ready to partner with you through this transformation—modernizing your architecture and helping you fully realize the benefits of Fabric’s deep Power BI integration.

Thank You !

USA : +1 206-858-9902
INDIA : +91 40 41239999

18300 NE Union Hill Road
Suite 210 Redmond, WA 98052

Info@peopletech.com
www.peopletech.com