SQL KONFERENZ 2025

Mastering Spark Notebooks

A Guide for Data Engineers
Version 2025

Tillmann Eitelberg

CEO, oh22information services GmbH
Microsoft Data Platform MVP

Oliver Engels

CEO, oh22data AG
Microsoft Data Platform MVP

Agenda

- Getting Started with Spark Notebooks
- Efficient coding with Python in Spark Notebooks
- Environments – Setting up your Spark workspace
- Mastering data manipulation in Spark
- Delta Tables – Optimizing data storage and performance
- Semantic Link – Connecting your data for a smarter workflow
- Simplifying data preparation with Data Wrangler
- Supercharging your Spark workflow with integrated tools

What we can’t cover today

- Fabric CI/CD
  - Notebook source control and deployment - Microsoft Fabric | Microsoft Learn
  - Overview of Fabric Git integration - Microsoft Fabric | Microsoft Learn
- Microsoft Spark Utilities
  - Introduction to Microsoft Spark utilities - Azure Synapse Analytics | Microsoft Learn
- Python Workshop
  - Documentation (python.org)
  - Python Tutorials – Real Python

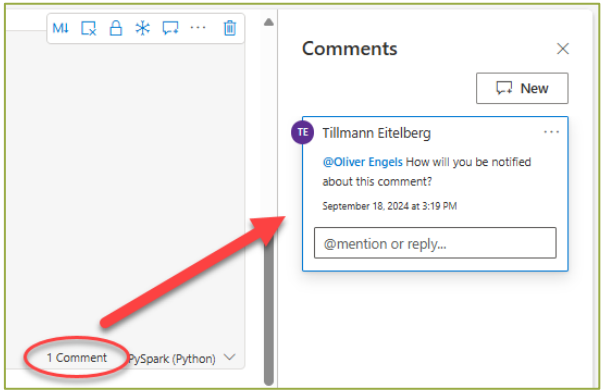
Getting Started with Spark Notebooks

Notebooks – The basics

- Add, delete and move rows
- Lock or freeze cells
- hide input or output
- Merge or split rows
- Shift + Enter / Strg + Enter
- Add Code and Markdown Cells
- Adding ipywidgets (IPython Widgets)
- Toggle parameter cell (used in pipeline activity)
- Fabric notebooks recognize the standard Jupyter .ipynb files
- Fabric Notebook known limitation

Comments

- Creating comments / threads
- Comments are assigned to individual cells and can be accessed from there
- People can be mentioned in comments
- Comments can be closed as a thread



Endorse Fabric Items

- Various objects within Fabric can be endorsed
- By default, users can promote notebooks (and other objects)
- With special permissions, they can also be set as certified
- Individual objects that contain data (currently only lakehouses and semantic models) can also be given a master data badge

Name	Type	Opened	Owner	Endorsement	Sensitivity	Location
#FABCONEUROPE	Workspace	now	—	—	—	Workspaces
03 - Snippets	Notebook	11 minutes ago	Tillmann Eitelberg	Promoted	—	#FABCONEUROPE
02 - Magic Commands	Notebook	34 minutes ago	Tillmann Eitelberg	—	—	#FABCONEUROPE
01 - Basics	Notebook	an hour ago	Tillmann Eitelberg	—	—	#FABCONEUROPE

SQL KONFERENZ 2025

Magic Commands

```
%%pyspark

result = spark.sql("SELECT * FROM random_numbers")

display(result)
```

[11] - Command executed in 1 sec 467 ms by Tillmann Eitelberg on 5:26:57 PM, 9/17/24 PySpark (Python)

Table | Chart | Download | Showing rows 1 - 3 | Inspect | Search

	Spalte1	Spalte2	Spalte3	Spalte4	Summe
1	97	38	96	43	274
2	78	23	34	72	207
3	5	62	46	15	128

```
%%sql
SELECT * FROM random_numbers
```

[12] - Command executed in 1 sec 451 ms by Tillmann Eitelberg on 5:26:59 PM, 9/17/24 Spark SQL

Table | Chart | Download | Showing rows 1 - 3 | Inspect | Search

	Spalte1	Spalte2	Spalte3	Spalte4	Summe
1	97	38	96	43	274
2	78	23	34	72	207
3	5	62	46	15	128

SQL KONFERENZ 2025

Magic Commands

```
1  # %who_ls
2  # %who
3  %whos
```

[13] ✓ - Command executed in 250 ms by Tillmann Eitelberg on 5:26:59 PM, 9/17/24 PySpark (Python)

```
Variable              Type            Data/Info
---------------------------------------------------
HiveContext           type            <class 'pyspark.sql.context.HiveContext'>
StreamingContext      type            <class 'pyspark.streaming<...>ontext.StreamingContext'>
col                   function        <function col at 0x7809f70c0d30>
data                  list            n=3
df                    DataFrame       DataFrame[Spalte1: bigint<...>4: bigint, Summe: bigint]
display               function        <function display at 0x7809eee4d360>
displayHTML           DisplayHTML     <notebookutils.visualizat<...>object at 0x7809eee2b220>
exit                  function        <function trap_exit at 0x7809ed30ae60>
initializeLHContext   function        <function initializeLHContext at 0x7809ed30b910>
mssparkutils          module          <module 'notebookutils.ms<...>ssparkutils/__init__.py'>
notebookutils         module          <module 'notebookutils' f<...>tebookutils/__init__.py'>
prepare               function        <function prepare at 0x7809f6eb6290>
random                module          <module 'random' from '/h<...>ib/python3.10/random.py'>
result                DataFrame       DataFrame[Spalte1: bigint<...>4: bigint, Summe: bigint]
sc                    SparkContext    <SparkContext master=yarn<...>d-46d6-bf9f-e49e9fb8be8c>
spark                 SparkSession    <pyspark.sql.session.Spar<...>object at 0x780a04645180>
spark_sum             function        <function sum at 0x7809f70c1990>
sqlContext            SQLContext      <pyspark.sql.context.SQLC<...>object at 0x780a046452a0>
```

```
1  data
```

[14] ✓ - Command executed in 240 ms by Tillmann Eitelberg on 5:27:00 PM, 9/17/24 PySpark (Python)

```
[(97, 38, 96, 43), (78, 23, 34, 72), (5, 62, 46, 15)]
```

```
1  %xdel data
```

[15] ✓ - Command executed in 283 ms by Tillmann Eitelberg on 5:27:01 PM, 9/17/24 PySpark (Python)

```
1  data
```

[16] ⊗ - Command executed in 276 ms by Tillmann Eitelberg on 5:27:03 PM, 9/17/24 PySpark (Python)

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[47], line 1
----> 1 data

NameError: name 'data' is not defined
```

- List all variables with “%whos”
- Display variable
- Delete variables with “%xdel”
- %%capture - captures the output of the entire cell

SQL KONFERENZ 2025

Magic Commands

```
%%capture install_log
```

- Suppresses or captures the output of a Jupyter cell
- Handles both stdout (regular output) and stderr (error messages)
- Can store the output in a variable for later use
- Useful for hiding long or unnecessary outputs (e.g., pip install)
- Does not affect the code execution

```
# Access the captured output later
print("Captured stdout:", captured_output.stdout)
print("Captured stderr:", captured_output.stderr)
```

Magic Commands

[Format code in Microsoft Fabric notebooks
Microsoft Fabric | Microsoft Learn]

```
%load_ext jupyter_black
```

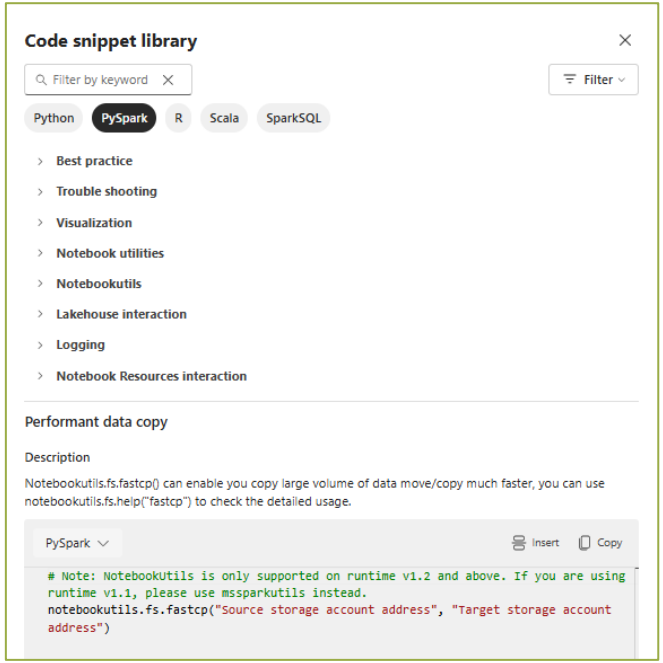
- extension for Jupyter Notebooks that auto-formats Python code using the Black code formatter
- enforces a consistent code style by reformatting code to follow best practices
- removes unnecessary whitespace, corrects indentation, and improves readability

```
data = {"A": np.random.randint(1, 100, 10), "B":
np.random.randint(1, 100, 10), "C": np.random.randint(1, 100,
10)}
df = pd.DataFrame(data)
```

```
data = {"A": np.random.randint(1, 100, 10), "B":
np.random.randint(1, 100, 10),
"C": np.random.randint(1, 100, 10),
}
df = pd.DataFrame(data)
```

Code snippet library

- Library with various example code on individual topics
- Can be filtered by language
- Code can be transferred directly to a notebook as a new line
- Individual variables / placeholders must be filled in the code
- Import of additional libraries is not always clearly defined
- Snippets can also be added via IntelliSense
- The examples provided do not appear to be identical



Copilot

- Load a specific package for copilot
- Install Magic Commands for Copilot

```
#Run this cell to install the required packages for Copilot
%pip install https://aka.ms/chat_magics-0.0.0-py3-none-any.whl
%load_ext chat_magics
```

Mistakes are possible.
Carefully review content created by AI before using it.

Copilot

No pandas imported in the current notebook
File path and name is incorrect
The context menu does a much better job

Home | Edit | Comments | Develop | Share

All sources
Lakehouses
+ Lakehouse
Lakehouse
Tables
dbo
publicholidays
Files
images
sample_datasets

nyc_taxi_green_2020_1.parquet
nyc_taxi_yellow_2020_1.parquet
public_holidays.parquet
us_population_county_area.csv

Context Commands

- `%pin` & `%unpin` - Pin and unpin dataframes to better le
- `%new_task` - Provide overall guidance for generated respons
- `%ignore` - Ignore a cell so that it is not processed by chat-magics

Detailed Usage

For detailed information on each command, use the ? (help operator), e.g. `%%code?` or `%set_sharing_level?`. Also, note that some magic commands may not be available depending on

```
# Welcome to your new notebook
# Type here in the cell editor to add code!
```

[2] 6 sec - Command executed in 272 ms by Tillmann Eitelberg on 2:08:10 PM, 9/17/24 PySpark (Python)

Copilot Preview

Let's work with your data

Choose an option below or type a request to get started.

- Load data from my lakehouse into a DataFrame
- Generate insights from data
- Suggest data visualizations

Load city safety into a DataFrame

You can imp ta from a file to a pandas dataframe using the following code:

Python

```
df = pd.read_csv('/lakehouse/default/Files/city safety.csv')
```

While generating the code, I made the following assumptions:

It is assumed that pandas is imported as pd

What are the top 5 safest cities in the dataset?
Remove any duplicate entries in the DataFrame.

Type a request or click the sparkle icon for ideas.

AI-generated content can have mistakes. Make sure it's accurate and appropriate before using it. Read preview terms

Session ready | AutoSave: On | Selected Cell 3 of 3 cells

SQL KONFERENZ 2025

Copilot

- Prompt “Analyze df and provide insights about the data”
- Output only in the Copilot window
- The summary is “OK”
- No code is copied into the notebook
- Further questions are a nice idea
- Other questions can also create tables, but also only in the Copilot window

Data Insights:

- **Data Types:**
  - The 'dataType' column contains 100 unique values, indicating various types of incidents or responses.
  - The 'category' column has 5189 unique values, representing different locations or addresses.
- **Location Information:**
  - Latitude ranges from 47.4956 to 47.7766, with a mean of approximately 47.6203.
  - Longitude is constant at -122.331.
- **Temporal Information:**
  - The dataset includes entries from various dates and times, but the 'dateTime' column seems to have missing values.
- **Response Details:**
  - Most common response type is 'Aid Response', occurring 3549 times.
- **Missing Values:**
  - Columns like 'dataSubtype', 'dateTime', 'subcategory', and 'status' have missing values.
- **Source Information:**
  - All entries have the same source and extended properties values (2020 and 7 respectively).

These insights provide a basic understanding of the dataset's contents, including the types of incidents recorded, location details, temporal aspects, common response types, and presence of missing values in certain columns. Further analysis could involve exploring relationships between variables or trends over

What are the top 5 categories with the highest number of incidents?

Combine similar subcategories into broader categories for better analysis.

Copilot – using Chat-magics

- Chat Magics bring the magic of Copilot to your notebook
- The `%%chat` command allows you to ask questions about the state of your notebook
- The `%%code` enables code generation for data manipulation or visualization
- `%pin DataFrameName` to help the AI to focus on specific dataframes
- `%new_task` clear the AI to focus on a new task

SQL KONFERENZ 2025

Efficient coding with Python in Spark Notebooks

Modularize Your Code

- Functions and modules allow code reuse, reducing duplication
- Improves readability and maintenance by breaking down complex tasks
- Modular code is easier to scale and modify as projects grow
- Isolate components for easier testing and bug fixing
- Centralize logic in functions/modules to avoid copy-pasting and reduce errors and thereby gain clean structure, better collaboration, version control, and easier maintenance
- Default Parameters: Flexibility in functions with minimal input, reducing repetitive code while allowing customization

- %run <notebook>

  vs.

  mssparkutils.notebook.run("notebook path", <timeoutSeconds>, <parameterMap>)

- The upper limit for notebook activities or concurrent notebooks in notebookutils.notebook.runMultiple() is 50.
- The statement depth for %run is up to 5, with a total of 1000 referenced cells.

Use Decorators for Reusable Logic

- Functions that modify the behavior of other functions or methods, without changing their code
- Allows code reuse for common patterns like logging, timing, authentication, caching, etc.
- Uses the @decorator_name syntax above the function to apply the decorator
- Decorators take a function as an argument and return a new function
- Multiple decorators can be stacked on top of a function

Use Decorators for Reusable Logic

```python
def timeit(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        print(f"Execution time: {time.time() - start_time} seconds")
        return result
    return wrapper
```

[2] ✓ <1 sec - Command executed in 245 ms by Tillmann Eitelberg on 4:58:16 PM, 9/18/24 PySpark (Python)

```python
@timeit
def run_transformation(df):
    return df.filter(df["age"] > 18)
```

[3] ✓ <1 sec - Command executed in 244 ms by Tillmann Eitelberg on 4:58:16 PM, 9/18/24 PySpark (Python)

```python
df = spark.sql("SELECT * FROM lakehouse.dbo.fabcon01 LIMIT 1000")
df_trans = run_transformation(df)
```

[8] ✓ 1 min 3 sec - Command executed in 1 min 2 sec 912 ms by Tillmann Eitelberg on 5:06:15 PM, 9/18/24 PySpark (Python)

Log

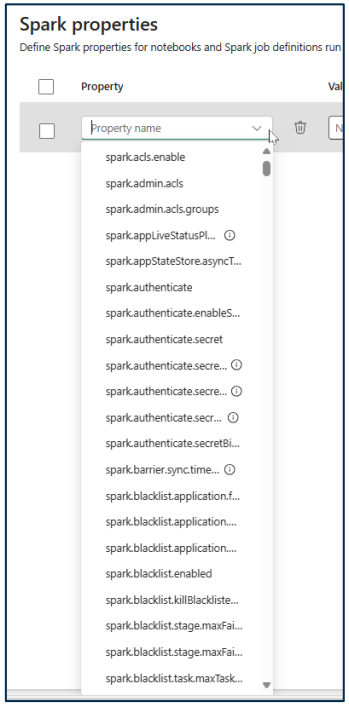
```
Execution time: 0.0027866363525390625 seconds
```

SQL KONFERENZ 2025

Environments – Setting up your Spark workspace

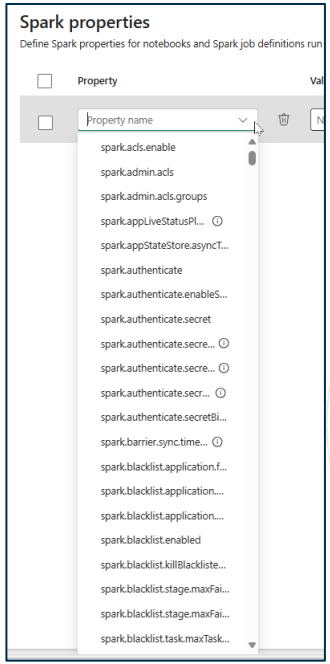
Environments – Setting up your Spark workspace

- Enables the summarization of environmental parameters
- Add public libraries via PyPi in fixed versions (including older ones)
- Add your own non-publicly hosted packages
- Configuration of the Spark cluster from the number of driver cores to memory and executor instances
- Definition of various Spark properties
- Additional files can be added and accessed from the notebook.
- Environments must be saved and published
- Publishing takes several minutes
- Caution: a Spark Session with an environment that contains additional libraries takes longer to start
  (But the installation in the notebook otherwise also)

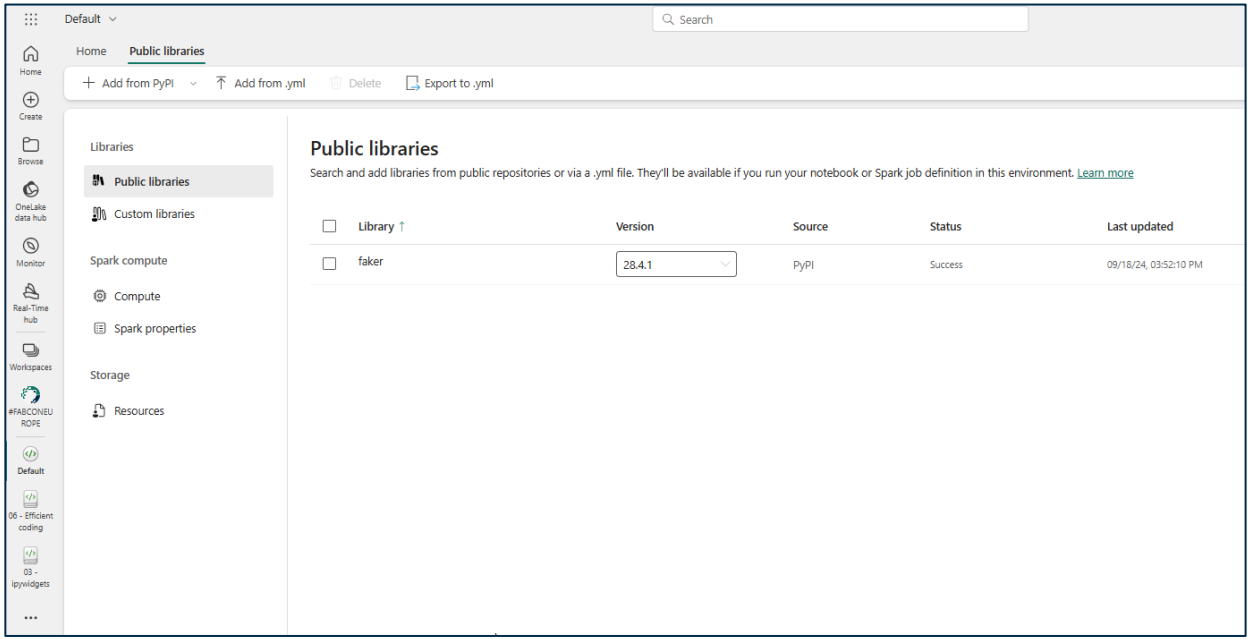


Environments – Setting up your Spark workspace

- Enables the summarization of environmental parameters
- Add public libraries via PyPi in fixed versions (including older ones)
- Add your own non-publicly hosted packages
- Configuration of the Spark cluster from the number of driver cores to memory and executor instances
- Definition of various Spark properties
- Additional files can be added and accessed from the notebook.
- Environments must be saved and published
- Publishing takes several minutes
- Caution: a Spark Session with an environment that contains additional libraries takes longer to start
  (But the installation in the notebook otherwise also)



Environments – Setting up your Spark workspace



SQL KONFERENZ 2025

Mastering data manipulation in Spark

DataFrames

DataFrames

- Two-dimensional, tabular data structures with labeled axes (rows and columns), similar to a spreadsheet or SQL table
- Widely used for data manipulation, analysis, and processing tasks
- The primary framework in Spark is, unsurprisingly, Spark DataFrames
- Nevertheless, Pandas is the most widely used DataFrame framework
  - Pandas can be used in both Python and pySpark
  - Widest support through various libraries
- Developers have to be careful which DataFrame they use
  - Reconversion costs a lot of time and resources
  - Not all functions are directly supported by all data frames.

Mastering data manipulation in Spark

- Each DataFrame has a schema that defines the structure, meaning the column names and data types are known
- Load data from various sources (CSV, JSON, Parquet, Hive, etc.)
- Perform operations like filtering, selecting, joining, and aggregation to transform raw data into useful insights
- Save transformed data in formats like Parquet, ORC, CSV, or to databases like Hive or relational databases via JDBC

Mastering data manipulation in Spark

- **Spark DataFrames**: DataFrames are processed in parallel across a cluster, making them highly scalable for big data
- **Pandas**: DataFrames are processed locally on a single machine, suitable for small to medium data workloads
- Both have more or less the same functions, but different syntax
- Due to different functions and different internal structures, not all pySpark / Python functions and modules can always handle both DataFrames identically

SQL KONFERENZ 2025

Mastering data manipulation in Spark

- When converting data between Spark DataFrames and Pandas, several issues can arise:
  - Memory limits: Pandas operates in-memory, so converting a large Spark DataFrame into Pandas may exceed memory limits
  - Type mismatches: Data types supported by Spark may not map directly to Pandas (e.g., null handling, categorical data)
  - Performance: Spark is optimized for distributed systems, while Pandas is not.
    Converting large datasets to Pandas can slow down performance significantly

Mastering data manipulation in Spark

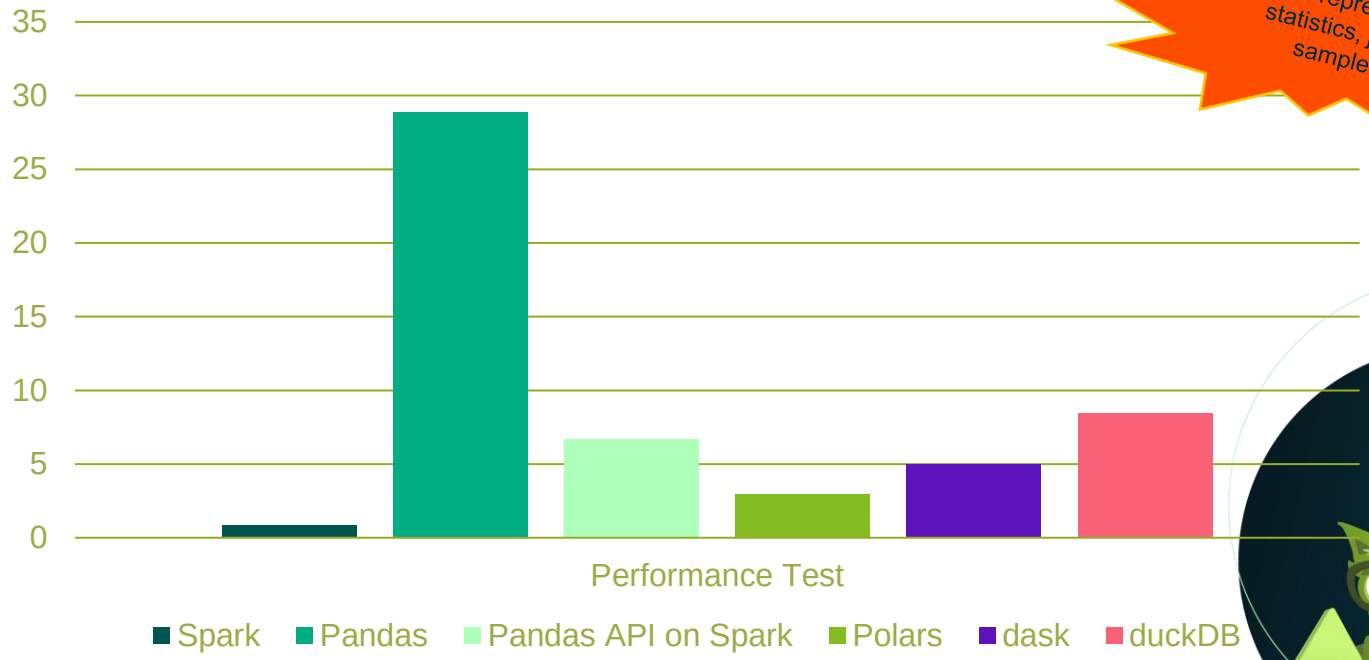
Description	Pandas Commands	Spark Commands
Select specific columns	df[['column']]	df.select('column')
Filter rows based on conditions	df[df['column'] > value]	df.filter(df['column'] > value) df.where(df['column'] > value)
Group data and perform aggregations	df.groupby('column').agg()	df.groupBy('column').agg()
Add or modify a column	df['new_column'] = df['existing_column'] * 2	df.withColumn('new_column', df['existing_column'] * 2)
Remove columns	df.drop(columns=['column'])	df.drop('column')
Join two DataFrames	df.merge(other_df, on='key')	df.join(other_df, on='key')
Sort data	df.sort_values(by='column')	df.orderBy('column') or df.sort('column')
Remove duplicate rows	df.drop_duplicates()	df.dropDuplicates()
Select unique rows	df['column'].unique()	df.distinct()
Limit the number of rows	df.head(n)	df.limit(n)
Replace NULL/NaN values	df.fillna(value)	df.fillna(value)
Remove rows with NULL/NaN values	df.dropna()	df.na.drop()
Rename a column	df.rename(columns={'old_name': 'new_name'})	df.withColumnRenamed('old_name', 'new_name')
Convert DataFrame to Pandas (Spark only)	N/A	df.toPandas()

DataFrames

DataFrames - Pandas-on-Spark, Dask, Polars

- **Pandas-on-Spark**: Combines the simplicity of Pandas with Spark's scalability, enabling Pandas-like operations on large, distributed data
- **Dask**: Scales Pandas-like operations across multiple cores or machines, handling datasets larger than memory
- **Polars**: A high-performance DataFrame library optimized for speed and multi-threading, handling both small and large datasets efficiently
- **DuckDB**: A lightweight, in-process SQL database optimized for fast analytical queries on large datasets, especially columnar formats like Parquet

Mastering data manipulation in Spark



Not representative statistics, just a few sample tests

SQL KONFERENZ 2025

Mastering data manipulation in Spark

- When using multiple DataFrame frameworks in the same notebook (e.g., Pandas, Spark, Dask, Polars), issues include:
  - Inconsistent APIs: Each framework has its own methods and syntax, leading to confusion or errors when switching
  - Data transfer overhead: Converting data between frameworks (e.g., Spark to Polars) can be time-consuming and resource-intensive
  - Memory management: Different frameworks handle memory differently, potentially leading to inefficiencies or crashes if not managed carefully
  - Library version conflicts: Different versions of libraries may be incompatible, leading to errors or issues that are sometimes difficult or impossible to resolve

SQL KONFERENZ 2025

Delta Tables – Optimizing data storage and performance

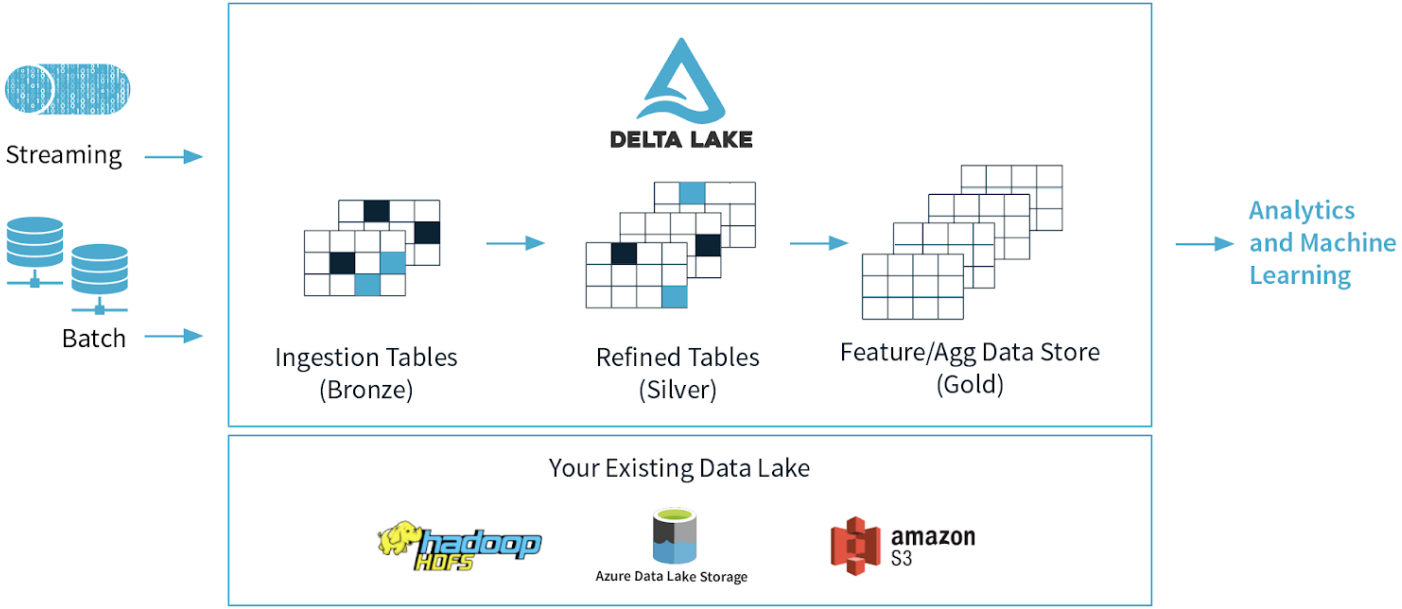
Delta Tables – Optimizing data storage and performance

DELTA LAKE

- Delta Tables combine the scalability of data lakes with ACID transactions for reliable data management
- Maintain historical versions of data, enabling time travel and auditability
- Support for upserts (MERGE), deletes, and updates directly in the table
- Can automatically evolve to accommodate new data without manual schema adjustments
- Use data indexing, caching, and compaction techniques to improve query speed
- Integrates seamlessly with Spark, Microsoft Fabric, Databricks and various cloud platforms for easy adoption

SQL KONFERENZ 2025

Delta Tables – Optimizing data storage and performance



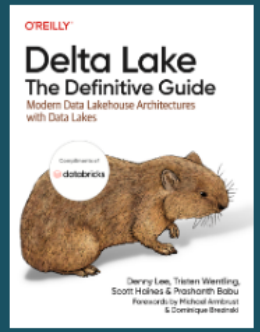
Delta Tables – Example commands

SQL Command	Description
CREATE TABLE AS (Delta)	Creates a new Delta table from the results of a query.
MERGE INTO	Performs upserts by merging new data into an existing Delta table based on a condition.
DESCRIBE DETAIL	Provides detailed information about the Delta table, such as version history and storage details.
DESCRIBE HISTORY	Shows the version history and metadata of a Delta table, useful for time travel and audit trails.
OPTIMIZE	Optimizes the performance by combining smaller files into larger ones in a Delta table.
VACUUM	Removes old versions of data files that are no longer needed to reclaim storage space.
RESTORE	Restores a Delta table to an earlier version.
TIMESTAMP AS OF	Queries a Delta table as of a specific timestamp.
VERSION AS OF	Queries a Delta table as of a specific version number.

Delta Tables – V-Order

- The Lakehouse and the Delta Lake table format are central to Microsoft Fabric
- V-Order is a write time optimization to the parquet file format that enables lightning-fast reads under the Microsoft Fabric compute engines, such as Power BI, SQL, Spark, and others
- V-Order sorting has a 15% impact on average write times but provides up to 50% more compression
- 100% open-source parquet format compliant
- Enabled by default in Microsoft Fabric
  Delta Lake table optimization and V-Order - Microsoft Fabric | Microsoft Learn

Delta Tables – Optimizing data storage and performance



Delta Lake: The Definitive Guide

Building modern data lakehouse architectures with Delta Lake with forewords by Michael Armbrust and Dominique Brezinski

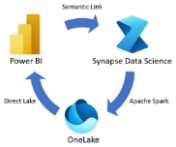
https://delta.io/pdfs/dldg_databricks.pdf

Semantic Link – Connecting your data for a smarter workflow

Semantic Link

- The semantic link feature connects semantic models with Synapse Data Science in Microsoft Fabric
- Enhance data connectivity
- Propagate semantic information
- Bridges the gap between Power BI and the Data Science experience
- **SemPy** is the corresponding Python package that provides all the functions within Fabric and is Lib is preinstalled!!
- Semantic Link Labs – Additional library with more functions to simplify working with Microsoft Fabric
  microsoft/semantic-link-labs: Early access to new features for Microsoft Fabric's Semantic Link. (github.com)

Semantic Link [-Labs]



- From your Fabric Notebook you can:

Home Edit Run View

Run all | Connect | PySpark (Python) | env_sempy | Data Wrangler | Copilot

Explorer

Get Workspaces combined with Semantic Models

This sempy code produces an overview on all your workspaces including the semantic models

```
# List your accesible workspaces
wspaces = fabric.list_workspaces().sort_values(by='Name', ascending=True)

# Filter the list only for workspaces on a dedicated capacity
filtered_ws = pd.DataFrame(wspaces).query('`Is On Dedicated Capacity` == True')[["Name", "Id"]]

#Loop the workspace list and build extract the underlying semantic models (incl. additional XMLA Props)
datasets = [
            fabric.list_datasets(ws,additional_xmla_properties=['Model.DefaultMode', 'Model.DirectLakeBehavior', 'CompatibilityLevel']).assign(workspace=ws)
            for ws in filtered_ws['Name']
        ]

#Build a simple data catalog of all the datasets in your workspaces
ds_catalog = pd.concat(datasets, ignore_index=True)
display(ds_catalog)
```

[9] - Command executed in 51 sec 792 ms by Oliver Engels on 8:27:22 PM, 8/13/24

Table | Chart | Download | Showing rows 1 - 28

	Dataset Name	Dataset ID	Created Timestamp	Last Update	Model Default Mode	Model Direct Lake Behavior	Compatibility Level	workspace
12	Adventure Works ...	7410f99-0f6d-...	2019-09-17T05:50:29.000	NULL	Import	Automatic	1567.0	Market_DP_DEV
13	Adventure Works ...	8294215f-a511...	2019-09-17T05:50:29.000	NULL	Import	Automatic	1567.0	Market_DP_PRD
14	Adventure Works ...	5e8413d5-4c35...	2019-09-17T05:50:29.000	NULL	Import	Automatic	1567.0	Market_DP_QAS

- https://github.com/microsoft/semantic-link-labs/tree/main/notebooks

Functional Dependencies in your data

Insides in your Fabric landscape

Gover/Administer Fabric landscape

Manage Semantic Models at Scale

Monitoring your Fabric landscape

Quality Gates at scale

Semantic Link Labs

- Innovator: Michael Kovalsky, Microsoft CAT team
- Library has > 300 and more functions
- You don’t need to be a python expert
- You don’t need to deal with complex API
- The swiss knife for all Fabricators

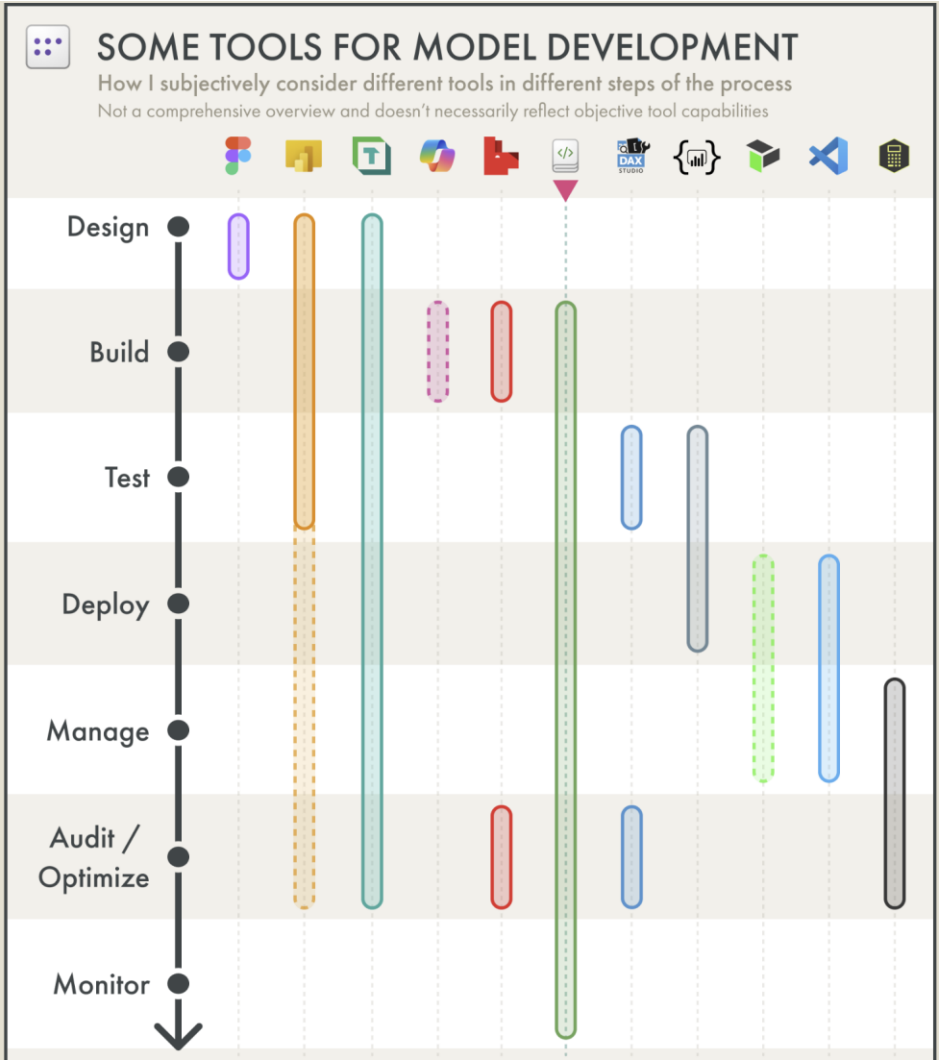
/ sempy_labs

sempy_labs

- sempy_labs package
  - Subpackages
    - sempy_labs.admin package
      - Module contents
    - sempy_labs.directlake package
      - Module contents
    - sempy_labs.lakehouse package
      - Module contents
    - sempy_labs.migration package
      - Module contents
    - sempy_labs.report package
      - Module contents
    - sempy_labs.tom package
      - Module contents
  - Module contents
    - `ConnectLakehouse`
    - `ConnectWarehouse`
    - `add_user_to_workspace()`
    - `assign_workspace_to_capacity()`
    - `assign_workspace_to_dataflow_storage()`
    - `backup_semantic_model()`
    - `cancel_dataset_refresh()`
    - `check_fabric_capacity_name_availablility()`
    - `clear_cache()`
    - `commit_to_git()`
    - `connect_workspace_to_git()`
    - `copy_semantic_model_backup_file()`
    - `create_abfss_path()`

Semantic Link [-Labs]

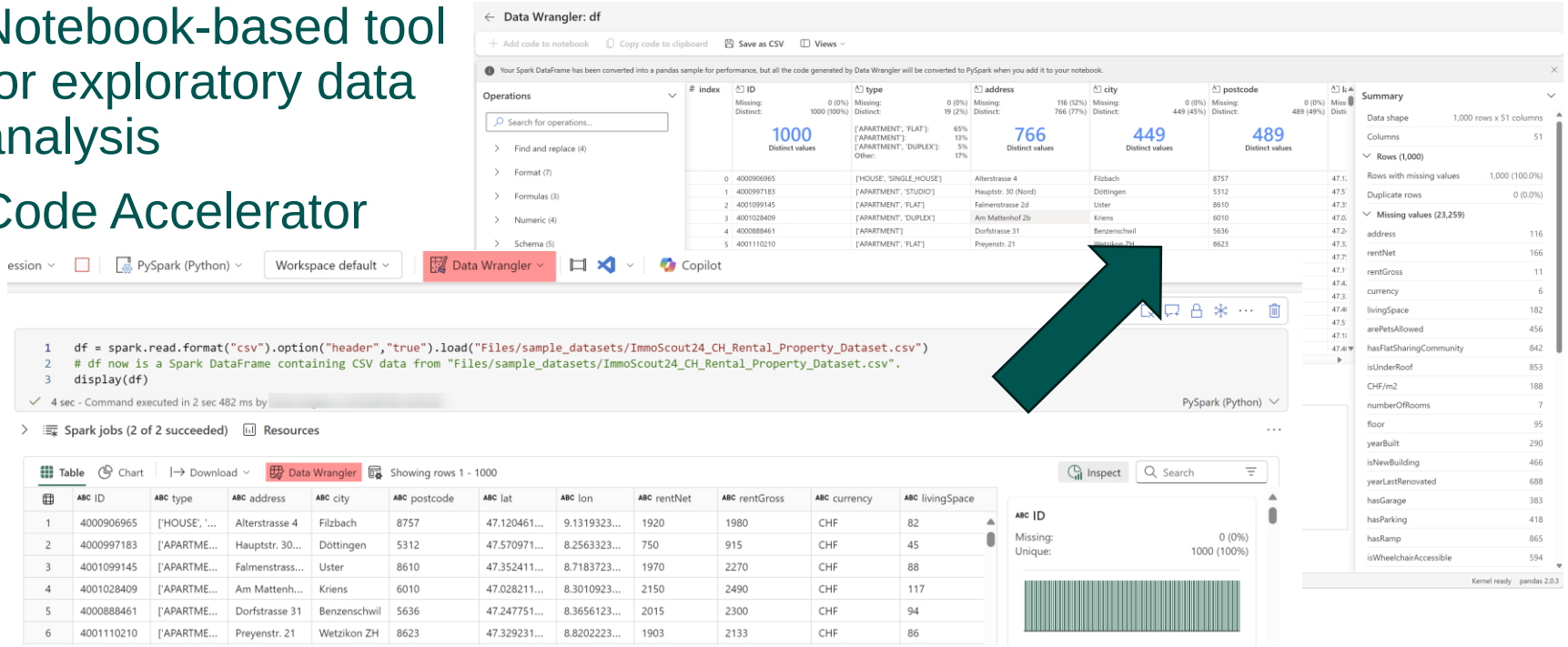
- If you are a “Semantic” – Data Engineer or modeler
- Kurt Buhler has an excellent blog on tools for model development
  https://data-goblins.com/power-bi/semantic-link-labs
- The only place were notebooks can’t support you is design



Simplifying data preparation with Data Wrangler

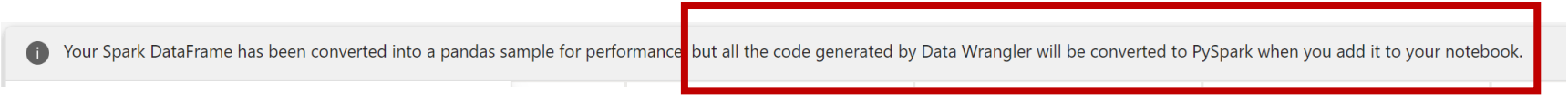
Simplifying data preparation with Data Wrangler

- Notebook-based tool for exploratory data analysis
- Code Accelerator

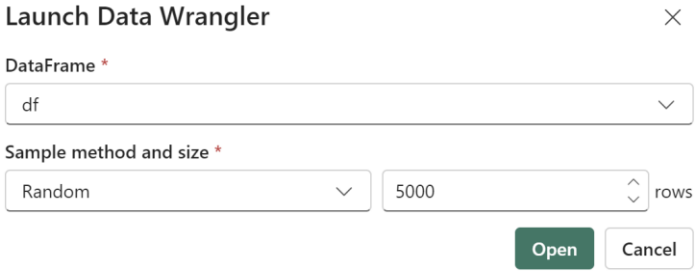


Simplifying data preparation with Data Wrangler

- Works with Pandas **and** Spark DataFrames
- Automatically converts Spark DataFrames to pandas samples for performance reason

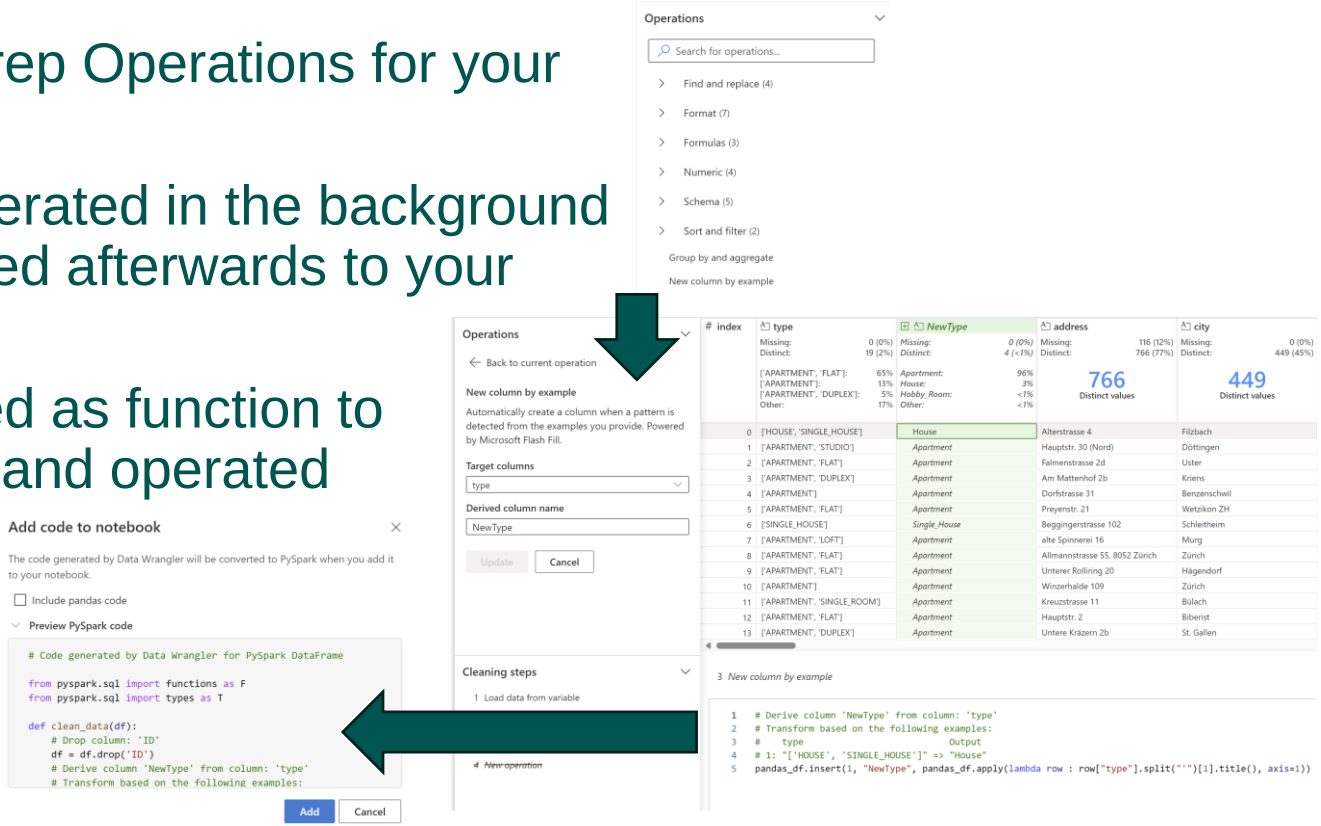


- Custom Sample possibility



Simplifying data preparation with Data Wrangler

- No Code Data Prep Operations for your DataFrame
- Code will be generated in the background and can be applied afterwards to your notebook
- Code is generated as function to be easily reused and operated



Supercharging your Spark workflow with integrated tools

Working with VSCode

- 2 different MS Fabric extensions are available
- Data Wrangler also exists as a standalone VSCode extension
- In addition to Microsoft extensions, extensions from the community are also available
- To best use the add-in locally, devcontainers are used
  devcontainers (github.com)
- Container image must be built locally
  - VS Code extension with Docker support - Microsoft Fabric | Microsoft Learn
  - SynapseVSCode/samples/.devcontainer at main · microsoft/SynapseVSCode (github.com)
- VSCode on the web from Fabric currently with version bugs

SQL KONFERENZ 2025

Working with VSCode

Synapse VSCode

Synapse VS Code - Visual Studio Marketplace

Synapse VS Code - Remote

Synapse VS Code - Remote - Visual Studio Marketplace

Fabric Studio

Fabric Studio - Visual Studio Marketplace

OneLake VSCode

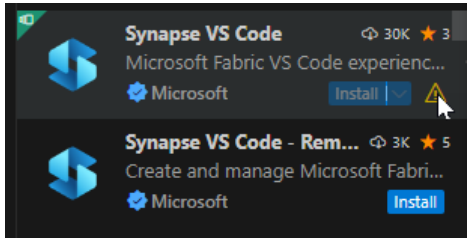
OneLake VSCode - Visual Studio Marketplace

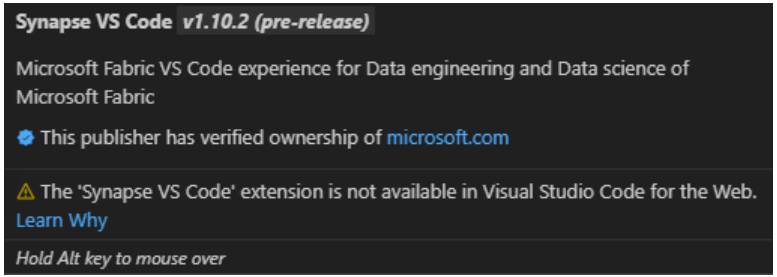
Data Wrangler

Data Wrangler - Visual Studio Marketplace

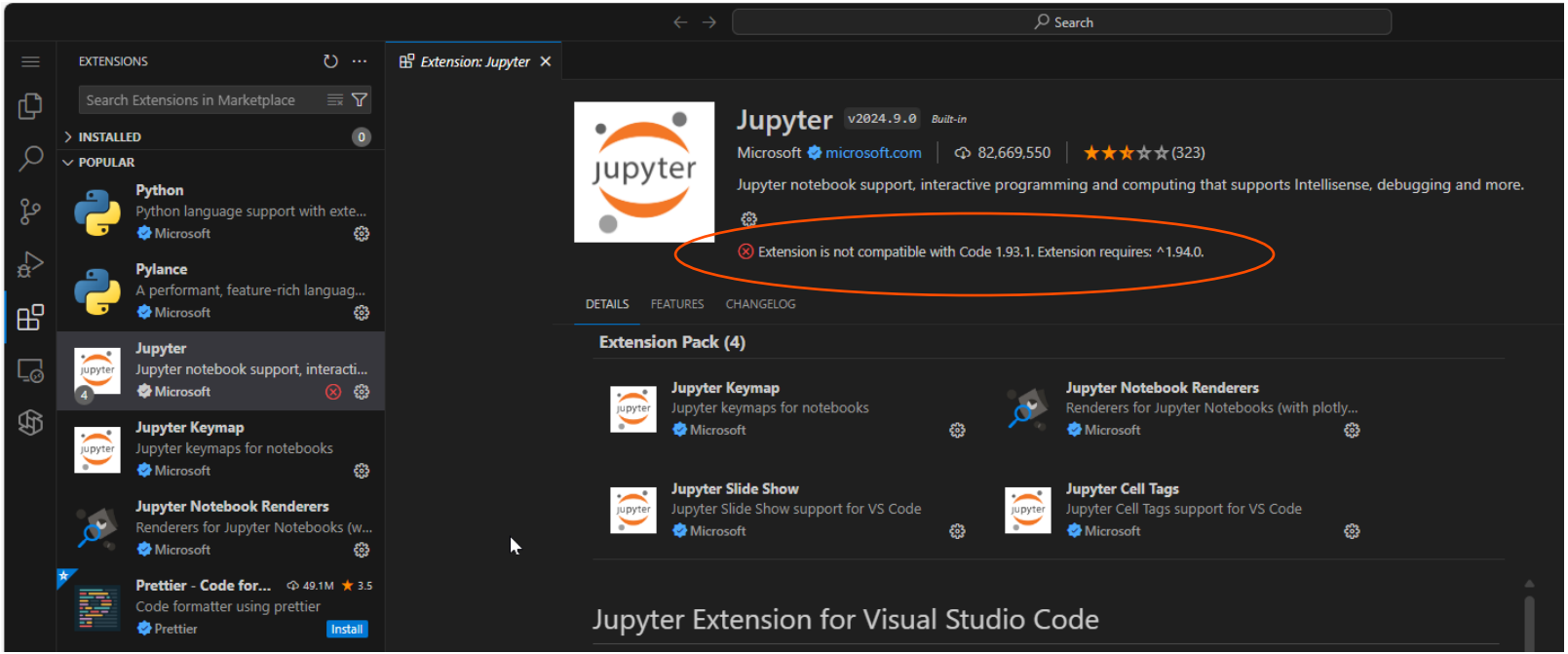
Working with VSCode

- Different versions for the desktop and web versions
- Synapse VS Code – Remote is intended as the extension for the web

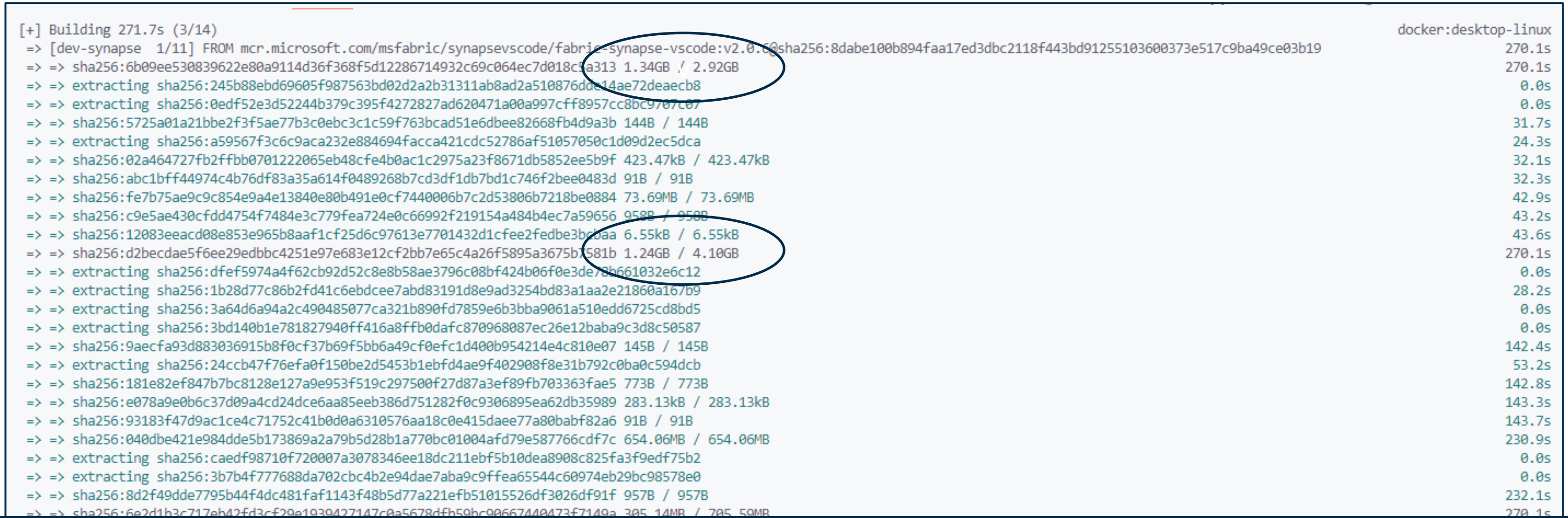




Working with VSCode



Working with VSCode

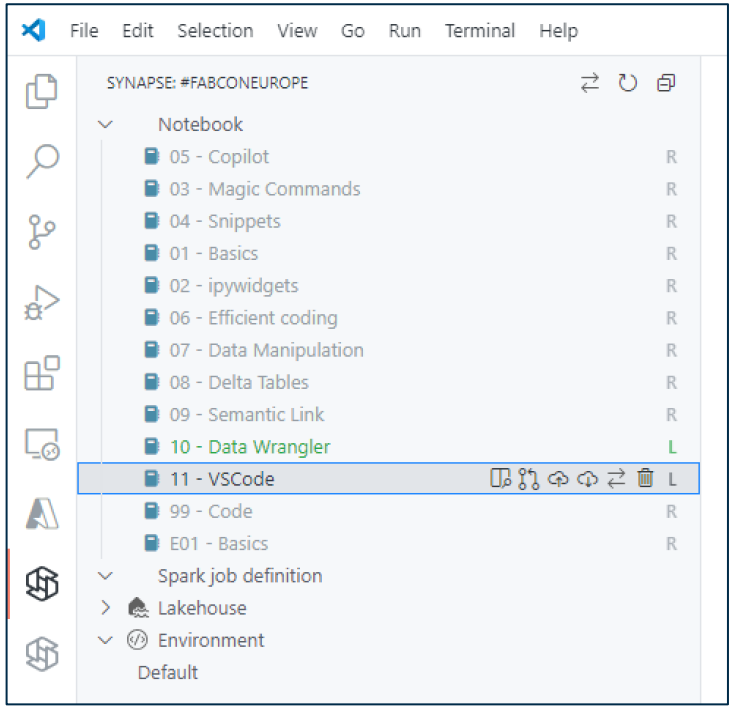


```
[+] Building 271.7s (3/14)                                                                                                                                                  docker:desktop-linux
 => [dev-synapse  1/11] FROM mcr.microsoft.com/msfabric/synapsevscode/fabric-synapse-vscode:v2.0.6@sha256:8dabe100b894faa17ed3dbc2118f443bd91255103600373e517c9ba49ce03b19          270.1s
 => => sha256:6b09ee530839622e80a9114d36f368f5d12286714932c69c064ec7d018c5a313 1.34GB / 2.92GB                                                                                    270.1s
 => => extracting sha256:245b88ebd69605f987563bd02d2a2b31311ab8ad2a510876dde14ae72deaecb8                                                                                          0.0s
 => => extracting sha256:0edf52e3d52244b379c395f4272827ad620471a00a997cff8957cc8bc9707c07                                                                                          0.0s
 => => sha256:5725a01a21bbe2f3f5ae77b3c0ebc3c1c59f763bcad51e6dbee82668fb4d9a3b 144B / 144B                                                                                         31.7s
 => => extracting sha256:a59567f3c6c9aca232e884694facca421cdc52786af51057050c1d09d2ec5dca                                                                                         24.3s
 => => sha256:02a464727fb2ffbb0701222065eb48cfe4b0ac1c2975a23f8671db5852ee5b9f 423.47kB / 423.47kB                                                                                 32.1s
 => => sha256:abc1bff44974c4b76df83a35a614f0489268b7cd3df1db7bd1c746f2bee0483d 91B / 91B                                                                                           32.3s
 => => sha256:fe7b75ae9c9c854e9a4e13840e80b491e0cf7440006b7c2d53806b7218be0884 73.69MB / 73.69MB                                                                                   42.9s
 => => sha256:c9e5ae430cfdd4754f7484e3c779fea724e0c66992f219154a484b4ec7a59656 958B / 958B                                                                                         43.2s
 => => sha256:12083eeacd08e853e965b8aaf1cf25d6c97613e7701432d1cfee2fedbe3bcbaa 6.55kB / 6.55kB                                                                                     43.6s
 => => sha256:d2becdae5f6ee29edbbc4251e97e683e12cf2bb7e65c4a26f5895a3675b7581b 1.24GB / 4.10GB                                                                                    270.1s
 => => extracting sha256:dfef5974a4f62cb92d52c8e8b58ae3796c08bf424b06f0e3de78b661032e6c12                                                                                          0.0s
 => => extracting sha256:1b28d77c86b2fd41c6ebdcee7abd83191d8e9ad3254bd83a1aa2e21860a167b9                                                                                         28.2s
 => => extracting sha256:3a64d6a94a2c490485077ca321b890fd7859e6b3bba9061a510edd6725cd8bd5                                                                                          0.0s
 => => extracting sha256:3bd140b1e781827940ff416a8ffb0dafc870968087ec26e12baba9c3d8c50587                                                                                          0.0s
 => => sha256:9aecfa93d883036915b8f0cf37b69f5bb6a49cf0efc1d400b954214e4c810e07 145B / 145B                                                                                        142.4s
 => => extracting sha256:24ccb47f76efa0f150be2d5453b1ebfd4ae9f402908f8e31b792c0ba0c594dcb                                                                                         53.2s
 => => sha256:181e82ef847b7bc8128e127a9e953f519c297500f27d87a3ef89fb703363fae5 773B / 773B                                                                                        142.8s
 => => sha256:e078a9e0b6c37d09a4cd24dce6aa85eeb386d751282f0c9306895ea62db35989 283.13kB / 283.13kB                                                                                143.3s
 => => sha256:93183f47d9ac1ce4c71752c41b0d0a6310576aa18c0e415daee77a80babf82a6 91B / 91B                                                                                          143.7s
 => => sha256:040dbe421e984dde5b173869a2a79b5d28b1a770bc01004afd79e587766cdf7c 654.06MB / 654.06MB                                                                                230.9s
 => => extracting sha256:caedf98710f720007a3078346ee18dc211ebf5b10dea8908c825fa3f9edf75b2                                                                                          0.0s
 => => extracting sha256:3b7b4f777688da702cbc4b2e94dae7aba9c9ffea65544c60974eb29bc98578e0                                                                                          0.0s
 => => sha256:8d2f49dde7795b44f4dc481faf1143f48b5d77a221efb51015526df3026df91f 957B / 957B                                                                                        232.1s
 => => sha256:6e2d1b3c717eb42fd3cf29e1939427147c0a5678dfb59bc90667440473f7149a 305.14MB / 705.59MB                                                                                270.1s
```

☐ **devcontainer-dev-synapse** e2c2dcf58b18 | latest | In use | 12 minutes ag | 25.26 GB

Working with VSCode

- Open Notebook Folder
- Update Notebook
- Publish Resource Folder
- Update Resource Folder
- Switch Notebook Environment
- Delete Notebook



Links

- GitHub Repository
  TEitelberg/MasteringSparkNotebooks
- Semantic Link Labs
  microsoft/semantic-link-labs

Session Feedback

SQL
KONFERENZ
2025

SQL
KONFERENZ
2025

SQL
KONFERENZ
2025


SQL
KONFERENZ
2025


SQL
KONFERENZ
2025

SQL
KONFERENZ
2025

SQL
KONFERENZ
2025


SQL
KONFERENZ
2025


SQL
KONFERENZ
2025


SQL
KONFERENZ
2025


SQL
KONFERENZ
2025


SQL
KONFERENZ
2025


SQL
KONFERENZ
2025


SQL
KONFERENZ
2025

SQL
KONFERENZ
2025