<https://learn.microsoft.com/en-us/training/modules/use-apache-spark-work-files-lakehouse/>

https://learn.microsoft.com/en-us/training/modules/ingest-data-with-spark-fabric-notebooks/

* Distributes data across multiple nodes for faster processing
* Cluster management object is called SparkContext
* Each workspace is assigned a spark cluster

In workspace, you can find Spark Compute (spark settings)  
- Node family  
- Runtime version  
- Spark properties

**Libraries** -  
- Built-in – can be found in Apache Spark runtime in Fabric  
- Public  
- Custom

<https://learn.microsoft.com/en-us/fabric/data-engineering/library-management>  
<https://learn.microsoft.com/en-us/fabric/data-engineering/runtime>

**Spark Job definition** -  **-** can be used to ingest and transform data as part of automated process  
- spark run script on-demand or based on schedule

Loading data into a dataframe -  
  
%%pyspark  
df = spark.read.load(‘Files/data/products.csv’,  
 format=’csv,  
 header=True  
)  
display(df.limit(10))

**Specifying an explicit schema -**  
from pyspark.sql.types import \*  
from pyspark.sql.functions import \*  
  
productSchema = StructType([  
 StructField(“ProductId”, IntegerType()),  
 StructField(“ProductName”, StringType()),  
 StructFiled(“Category”, StringType(),  
 StructFiled(“ListPrice”, FloatType())  
 ])  
df = spark.read.load(‘Files/data/product-data.cv’,  
 format=’csv’,  
 schema=productSchema,  
 header=False)  
display(df.limit(10))  
  
  
**Select “ProductID” and “ListPrice” columns in a dataframe –**

Pricelist\_df = df.select(“ProductID”, “ListPrice”) or  
Pricelist\_df = df[“ProductID”, “ListPrice”]  
  
  
**Filter *Category* to “Mountain Bikes” and “Road Bikes” -**  
  
df\_bikes = df.select(“Product Name”, “Category”, “List Price”).where((df[“Category”]==“Mountain Bikes”) | df[“Category”]==“Road Bikes”))  
display(df\_bikes)  
  
  
**Group -**

counts\_df = df.select(“ProductID”, “Category”).groupBy(“Category”).count()  
display(counts\_df)  
  
  
**Saving a dataframe -**  
  
df\_bikes.write.mode(“overwrite”).parquet(“Files/products-data/bikes.parquet”)  
  
  
**Partitioning the output file -**  
bikes\_df.write.partitionBy(“Category”).mode(“overwrite”).parquet(“Files/bike\_data”)  
  
  
**Load partitioned data -**  
  
road\_bikes\_df = spark.read.parquet(‘Files/bike\_data/Category=Road Bikes’)  
display(road\_bikes\_df.limit(5))  
  
\*\* the partitioning column included in the file path won’t include in the dataframe

**Read data from source available in parquet format-**  
df = spark.read.parquet(“location to read from”)  
  
“location to read from” 🡪 Copy ABFS path – absolute path if source is in default lakehouse  
“location to read from” 🡪 Copy relative path – if source is in other lakehouse  
  
  
**Load/save dataframe as csv to Files section of the default lakehouse -**  
  
df.write.mode(“overwrite”).format(“csv”).save(“Files/table\_name”)  
  
  
**Load/save dataframe as parquet files to Files sections of the default lakehouse -**  
df.write.mode(“overwrite”).format(“parquet”).save(“Files/table\_name”)  
  
  
**Save dataframe as delta lake parquet tables (delta tables) to Tables section of default Lakehouse -**  
  
df.write.mode(“overwrite”).format(“delta”).saveAsTable(delta\_table\_name)  
df.write.format(“delta”).saveAsTable(delta\_table\_name)  
  
  
**Save dataframe as delta table appending data to existing delta table -**  
  
df.write.mode(“append”).format(“delta”).saveAsTable(delta\_table\_name)  
  
  
  
  
  
  
  
  
  
**Delta tables**   
  
  
- Delta lake supports the creation of both **managed** and **external** tables  
- Data for the **managed** tables are stored in the **Tables** storage location associated with the catalog.  
- we can create an empty table in managed delta table using **spark.catalog.createTable** method  
- we can save a dataframe as delta table using **saveAsTable** method  
- deleting a managed tables deletes the underlying data  
  
- we can also create external tables using **spark.catalog.createExternalTable** method. External tables define metadata in catalog. Stored in **Files** in storage location. Deleting external tables   
do not delete underlying data.

**Save dataframe as external delta table in Files sections of lakehouse -**  
df.write.format(“delta”).saveAsTable(“external\_product”, path=”abfs\_path/external\_products”)  
  
abfs\_path: Copy ABFS path

The high concurrency mode for Fabric notebooks is set at the workspace level. It is on by default; however, it can be turned off in scenarios where notebooks require dedicated compute resources.