Delta Lake Tables

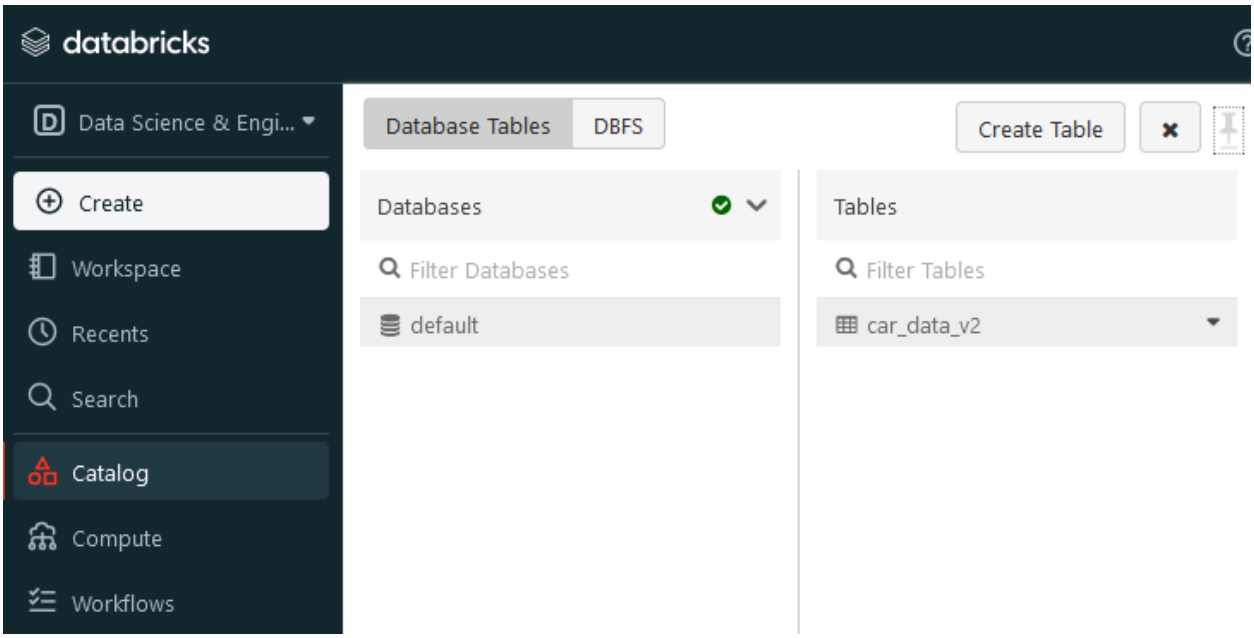
Zoya Shafique, Sarim Asrar
DSE I2450, Spring 2024

Delta Lake tables are essentially Parquet files with additional metadata and transaction logs. They offer features like ACID transactions, scalable metadata handling, data versioning, schema enforcement, and more. These features add structure and organization to Delta Lake, making it the first Lakehouse structure of its kind [1]. Delta Lake tables are particularly useful for data lakes where multiple users and processes need to read and write data simultaneously with reliability and consistency as they help maintain data consistency. The best part is that Delta Lake tables are fully compatible with Spark APIs, meaning that you can use Delta Lake tables as you would use a regular DataFrame.

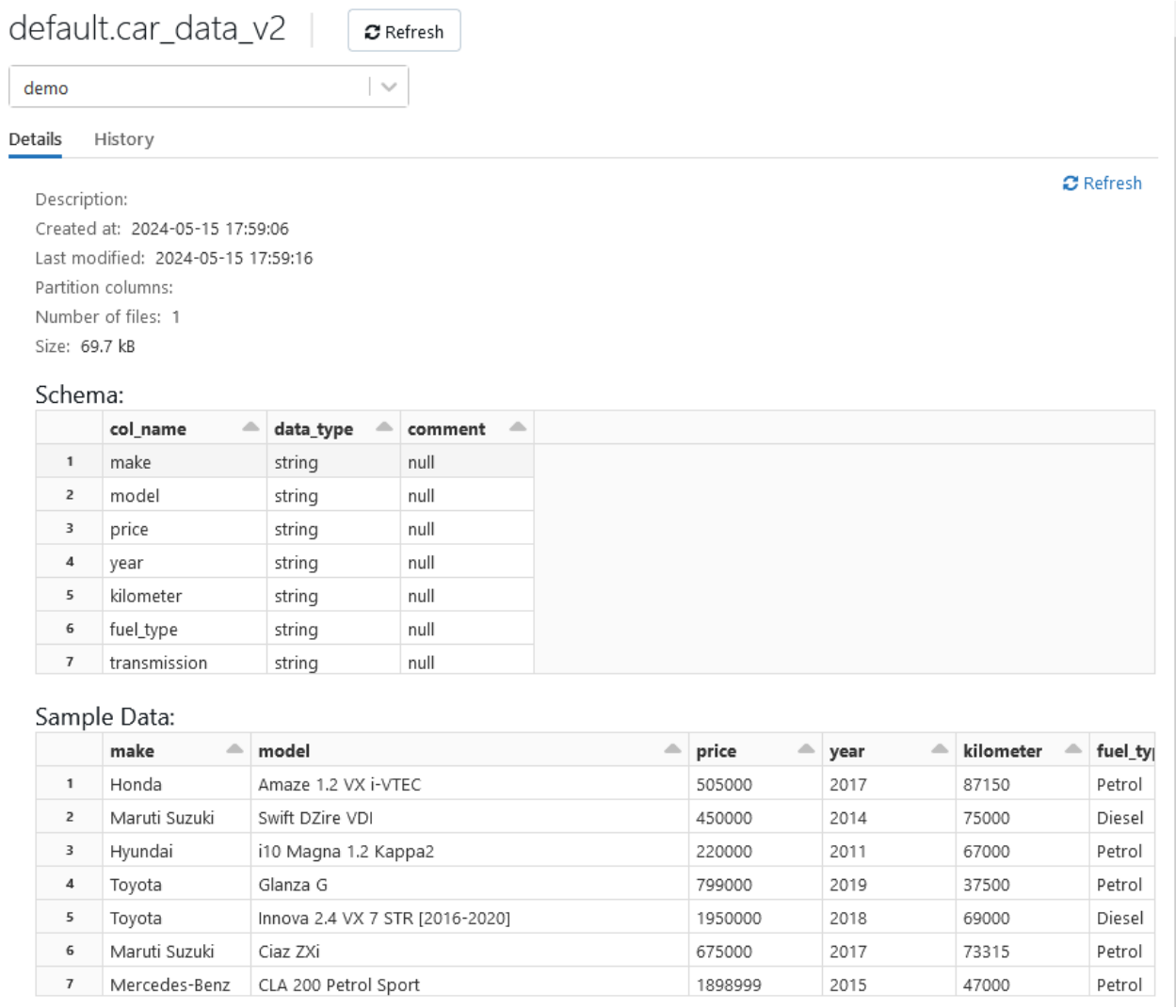
Delta Lake tables can be created in Databricks by calling the 'write.saveAsTable()' function on your DataFrame [as can be seen in this tutorial].

```
# Write the data to a table.
table_name = "car_data_v2"
car_data.write.saveAsTable(table_name)
```

Once the table has been created, it can be accessed by going to the 'Catalog' tab in the sidebar.



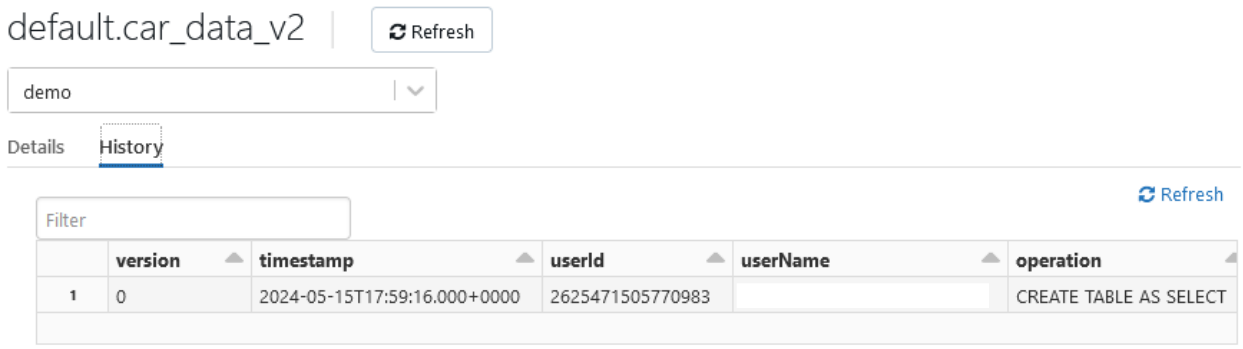
Opening the table brings us to the following interface:



Here we see

- The path at which the table is saved: default.car_data_v2
- Details such as the size, when the table was last edited, the number of files within the data. We can add in a description as well.
- The table's schema and information about data types.
- There is a tab where we can view the history of edits to the table.

As we have just created the table, the History table contains only one row.



	version	timestamp	userId	userName	operation
1	0	2024-05-15T17:59:16.000+0000	2625471505770983		CREATE TABLE AS SELECT

After we edit our table and clean up our datatypes, we can see that the schema updates:

Schema:

	col_name	data_type	comment
1	make	string	null
2	model	string	null
3	price	double	null
4	year	int	null
5	kilometer	double	null
6	fuel_type	string	null
7	transmission	string	null

We can also check the history and see our comments:

		userMetadata	engineInfo
1	umOutputRows": "2059", "numOutputBytes":	overwritten-for-fixing-incorrect-dtypes-in-columns	Databricks-Runtime/12.2.x-scal
2			

The ‘userMetadata’ can be used to commit comments with our changes to the data. This helps keep a record of changes made. In addition to ensuring data quality and maintenance, data versioning also aids reproducibility.

The main benefit of using Delta Lake tables is their metadata handling and data versioning capabilities, as it helps us maintain a record of how the data has changed and ensure data quality and consistency – which is crucial for machine learning applications. Delta Lake tables also contain schema enforcement, which guards against data corruption. This added layer of governance and organization on top of the data lake storage is what makes Databricks a Lakehouse platform as opposed to a data warehouse or a data lake.

References

[1] Mazumdar, Dipankar, Jason Hughes, and J. B. Onofre. "The Data Lakehouse: Data Warehousing and More." *arXiv preprint arXiv:2310.08697* (2023). [link]