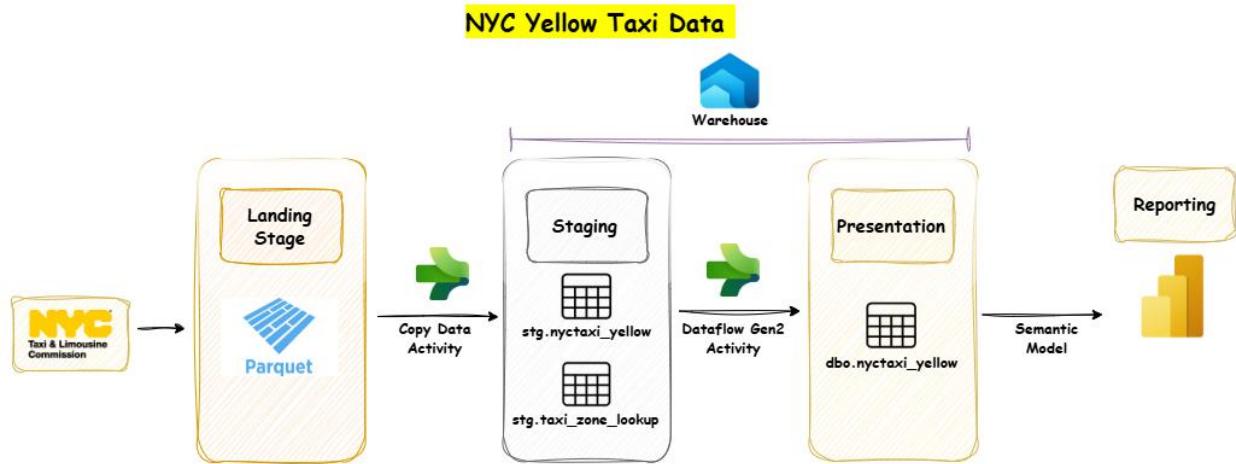


NYC Taxi Data Project Using Microsoft Fabric

Overall Project Architecture,



Create Fabric Workspace by providing relevant name for the workspace.

Create Lakehouse as 'ProjectLakehouse'.

Go to the Lake house and Create sub folders under the Files folder as 'nyctaxi_lookup_zone' and 'nyctaxi_yellow'. Upload Yellow trip data Parquet files to the 'nyctaxi_yellow' folder and Upload taxi_zone_lookup.csv file to the 'nyctaxi_lookup_zone' folder.

The screenshot shows the Microsoft Fabric Workspaces interface. The left sidebar shows the workspace navigation, and the main area displays the 'ProjectLakehouse' workspace. The 'Tables' and 'Files' sections are visible in the Explorer pane. The 'Get data in your' section contains three buttons: 'Upload files', 'Start with sample data', and 'New shortcut'. An 'Upload files' dialog is open, showing the 'nyctaxi_yellow' folder path. The 'Current uploads' table lists five files, all of which are currently waiting for processing:

File Name	Lakehouse Name	Status
yellow_tripdata_2024-05.parquet	ProjectLakehouse	Waiting...
yellow_tripdata_2024-04.parquet	ProjectLakehouse	Waiting...
yellow_tripdata_2024-03.parquet	ProjectLakehouse	Waiting...
yellow_tripdata_2024-02.parquet	ProjectLakehouse	Waiting...
yellow_tripdata_2024-01.parquet	ProjectLakehouse	Waiting...

The screenshot shows the ProjectLakehouse workspace in the Fabric service. On the left, the sidebar includes links for Home, OneLake catalog, Monitor, Real-Time, Workloads, NYC Yellow Taxi Data..., ProjectLakehouse, and Fabric. The main area has a search bar and a message: "A SQL analytics endpoint for SQL querying was created with this item." Below is an Explorer section with a tree view under ProjectLakehouse: Tables and Files. The 'nyctaxi_lookup_zone' folder under Files is highlighted with a red box. To the right is a modal titled "Upload files" with a text input field containing "Files/nyctaxi_lookup_zone/" and an "Upload" button. Below it is a "Current uploads" table:

File Name	Lakehouse Name	Size	Action
taxi_zone_lookup.csv	ProjectLakehouse	12 KB / 12 KB	✓
yellow_tripdata_2024-05.parquet	ProjectLakehouse	59 MB / 59 MB	✓
yellow_tripdata_2024-04.parquet	ProjectLakehouse	56 MB / 56 MB	✓
yellow_tripdata_2024-03.parquet	ProjectLakehouse	57 MB / 57 MB	✓
yellow_tripdata_2024-02.parquet	ProjectLakehouse	48 MB / 48 MB	✓
yellow_tripdata_2024-01.parquet	ProjectLakehouse	47 MB / 47 MB	✓

Create Warehouse as 'ProjectWarehouse' in workspace.

The screenshot shows the NYC Yellow Taxi Data Project workspace in the Fabric service. The sidebar includes links for Home, OneLake catalog, Monitor, Real-Time, NYC Yellow Taxi Data..., ProjectLakehouse, and Fabric. The main area has a search bar and a message: "This project is used to NYC Yellow Taxi Data project. ETL pipeline building, Data transformation, Data Cleansing, Sentiment analysis, and more processes are used." Below is a table of items:

Name	Type	Task	Owner	Refreshed	Next refresh	Endorsement	Sensitivity	Included in app
ProjectLakehouse	Lakehouse	—	User 1	—	—	—	—	—
ProjectLakehouse	Semantic mo...	—	NYC Yellow T...	6/5/2025, 9:25...	N/A	—	—	—
ProjectLakehouse	SQL analytics...	—	User 1	—	—	—	—	—
ProjectWarehouse	Warehouse	—	User 1	—	—	—	—	—
ProjectWarehouse	Semantic mo...	—	NYC Yellow T...	6/5/2025, 9:37...	N/A	—	—	—

A red box highlights the "+ New item" button in the top navigation bar and the "ProjectWarehouse" row in the table.

Create Folder for Pipelines as 'NYC_Taxi_Data_Pipelines'.

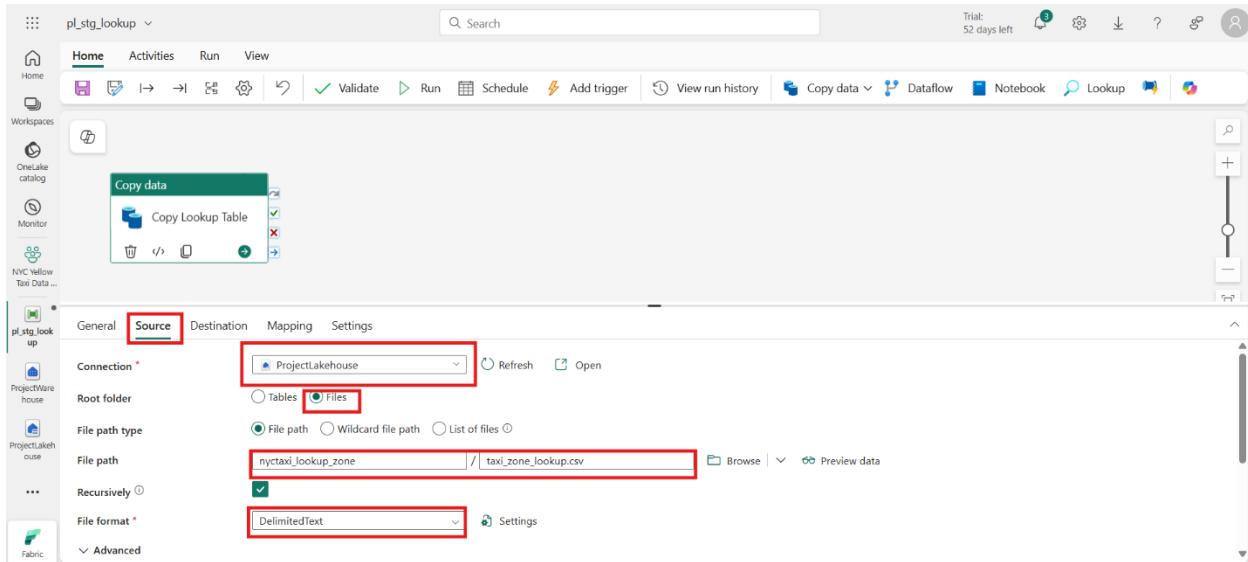
The screenshot shows the Fabric NYC Yellow Taxi Data Project interface. On the left, there's a sidebar with various icons for Home, Workspaces, OneLake catalog, Monitor, Real-Time, and ProjectWare house. The main area displays a list of items under the 'NYC Yellow Taxi Data Project' tab. A new item has been created, and it is highlighted with a red box. The table columns include Name, Type, Task, Owner, Refreshed, Next refresh, Endorsement, Sensitivity, and Included in app. The newly created folder is named 'NYC_Taxi_Data_Pipelines' and is listed under the 'ProjectLakehouse' category.

Name	Type	Task	Owner	Refreshed	Next refresh	Endorsement	Sensitivity	Included in app
NYC_Taxi_Data_Pipelines	Folder	—	—	—	—	—	—	—
ProjectLakehouse	Lakehouse	—	User 1	—	—	—	—	—
ProjectLakehouse	Semantic mo...	—	NYC Yellow T...	6/5/2025, 9:25...	N/A	—	—	—
ProjectWarehouse	SQL analytics...	—	User 1	—	—	—	—	—
ProjectWarehouse	Warehouse	—	User 1	—	—	—	—	—
ProjectWarehouse	Semantic mo...	—	NYC Yellow T...	6/5/2025, 9:37...	N/A	—	—	—

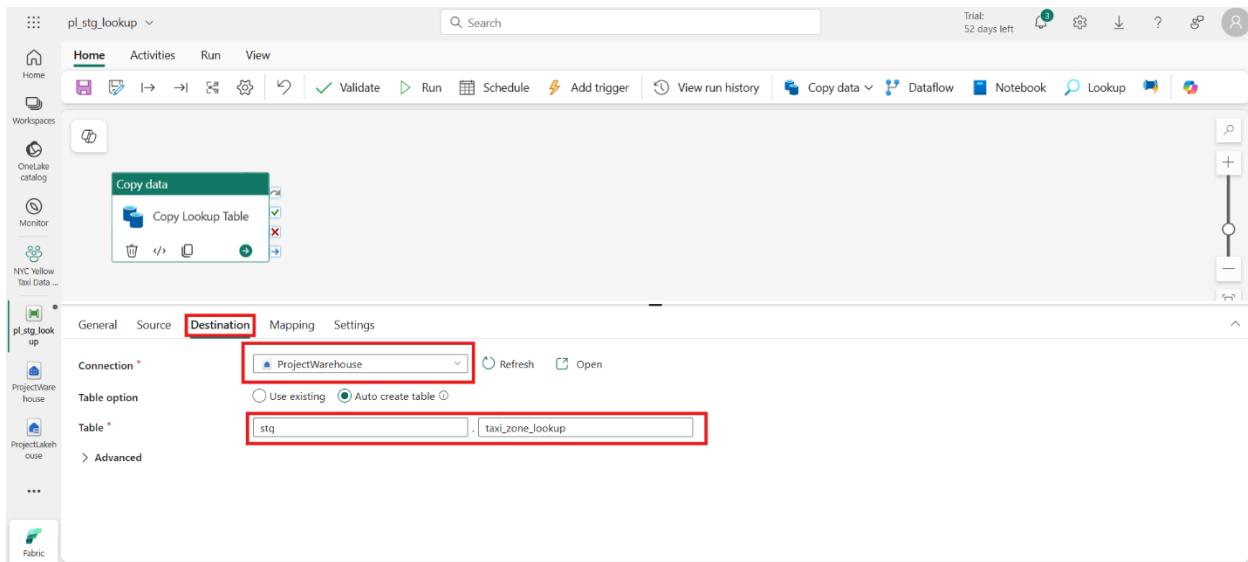
Inside the created folder, Create Pipeline named as 'pl_stg_lookup'. This pipeline's purpose is to Copy all the Data included in the 'nyctaxi_lookup_zone' f

The screenshot shows the configuration page for the 'pl_stg_lookup' pipeline. The top navigation bar includes Home, Activities, Run, View, and several pipeline-specific buttons like Validate, Run, Schedule, Add trigger, View run history, Copy data, Dataflow, Notebook, Lookup, and a plus sign for new activities. The main form is titled 'Copy data' and contains a 'Copy Lookup Table' activity. Below this, the 'General' tab is selected, showing fields for Name (highlighted with a red box), Description, Activity state (Activated), Timeout (0:12:00), and Retry (0). There are also 'Source', 'Destination', 'Mapping', and 'Settings' tabs, and an 'Advanced' section at the bottom.

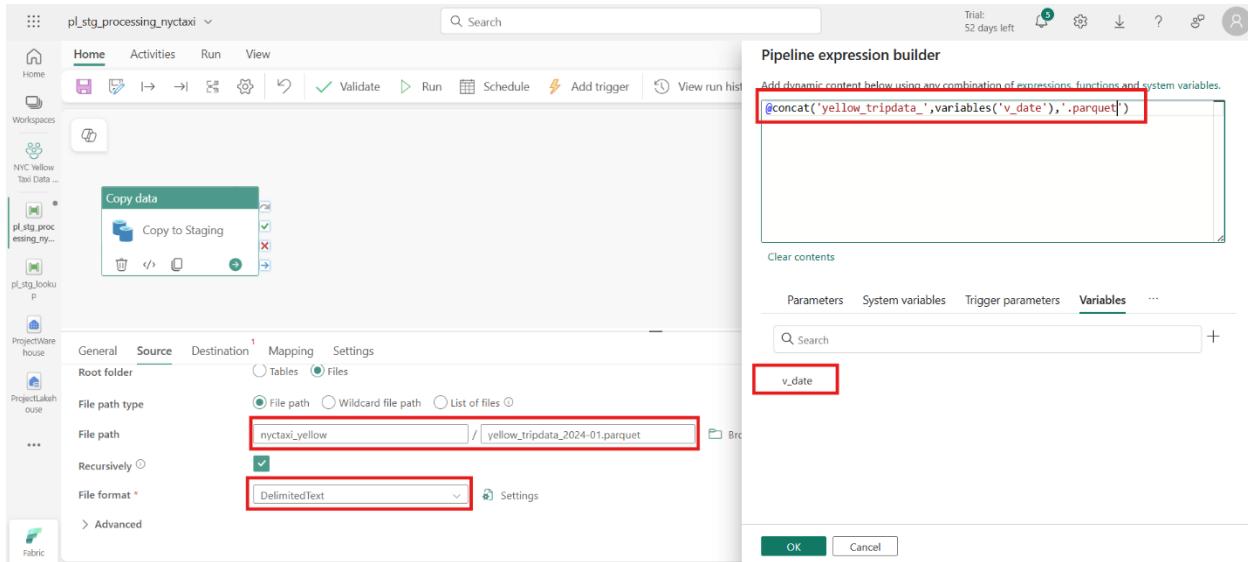
Enter relevant details to the copy data activity source tab.



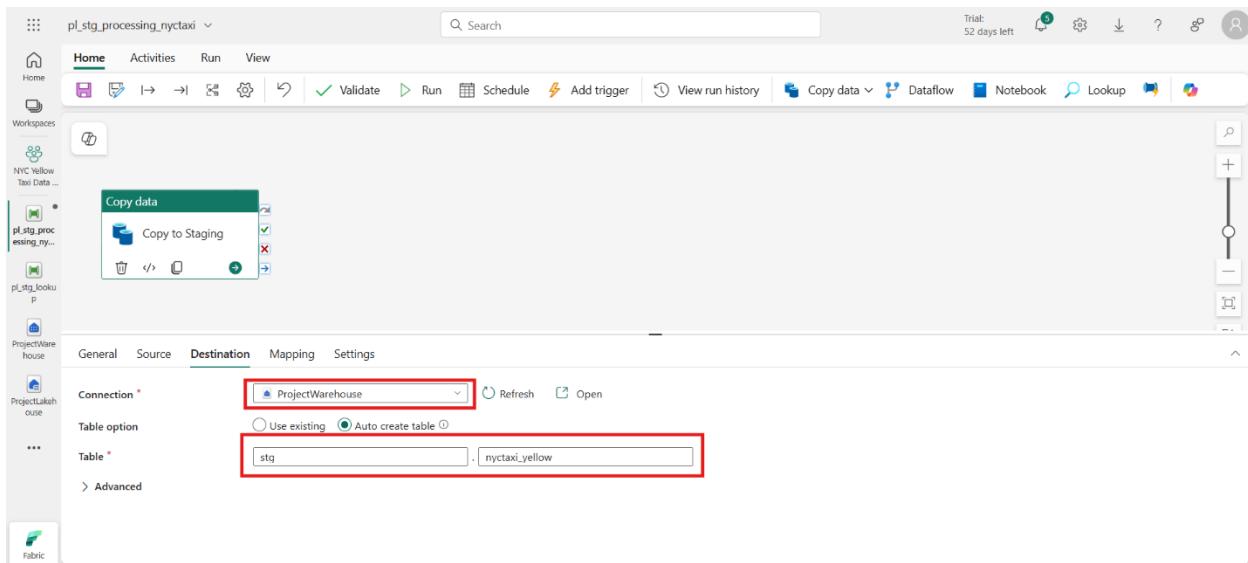
Enter relevant destination details to the destination tab as shown in the below figure.



Create another pipeline Inside the ‘NYC_Taxi_Data_Pipelines’ folder. Give pipeline name as ‘pl_stg_processing_nyctaxi’. This pipeline is used to copy ‘Yellow_Tripdata_2024-01.parquet’ file to data warehouse.



Give relevant details to the destination tab.



Set value for the v_date variable manually.

The screenshot shows the Azure Data Factory pipeline editor interface. The pipeline is named 'pl_stg_processing_nyctaxi'. In the center, there's a 'Copy data' activity labeled 'Copy to Staging'. Below it, the 'Variables' tab is selected in the navigation bar. A variable named 'v_date' is listed with a type of 'String' and a default value of '2024-01'. This row is highlighted with a red box. On the left sidebar, there are workspace items like 'NYC Yellow Tax Data ...', 'pl_stg_processing_ny...', 'pl_stg_looku...', and 'ProjectW...', along with a 'Fabric' icon.

When run the pipeline, if it run successfully, its look like this.

The screenshot shows the Azure Data Factory pipeline editor interface after a successful run. The pipeline status is now 'Succeeded', indicated by a green checkmark. A toast notification on the right says 'Run Succeeded Successfully ran pl_stg_processing_nyctaxi...'. The 'Output' tab is selected, showing a single run entry. The run details table includes columns for Activity name, Activity status, Run start, Duration, Input, and Output. The 'Copy to Staging' activity is listed with a status of 'Succeeded' and a run start time of 6/5/2025, 10:00:28 AM. The duration was 47s. The input and output sections are empty.

Run this SQL query to identify if there exist any irrelevant data file. As shown in the figure, there exist data relevant to the 2002.

The screenshot shows the ProjectWarehouse interface. In the Explorer pane, under the 'stg' warehouse, the 'Tables' section contains 'nyctaxi_yellow' and 'taxi_zone_lookup'. In the 'Queries' section, 'SQL query 1' is selected. The query is:

```
SELECT max(tpep_pickup_datetime), min(tpep_dropoff_datetime) from stg.nyctaxi_yellow
```

The results pane shows a single row of data:

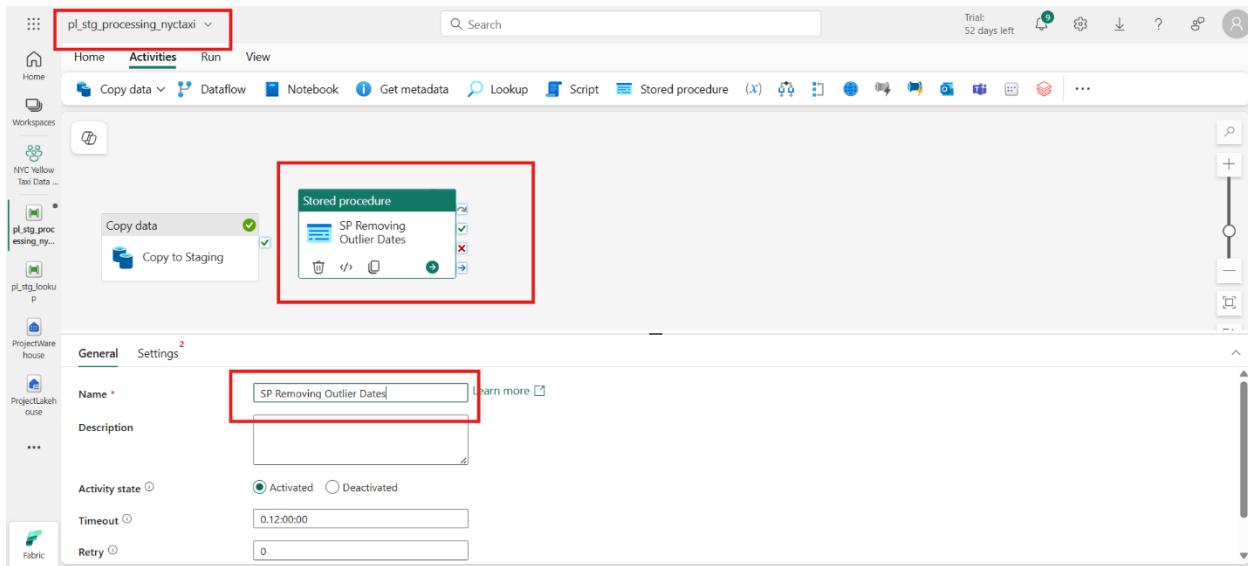
	max(tpep_pickup_datetime)	min(tpep_dropoff_datetime)
1	2024-02-01 00:01:15.000000	2002-12-31 23:05:41.000000

To remove irrelevant data run this query in your warehouse query space. Here create stored procedure as 'data_cleansing_stg'.

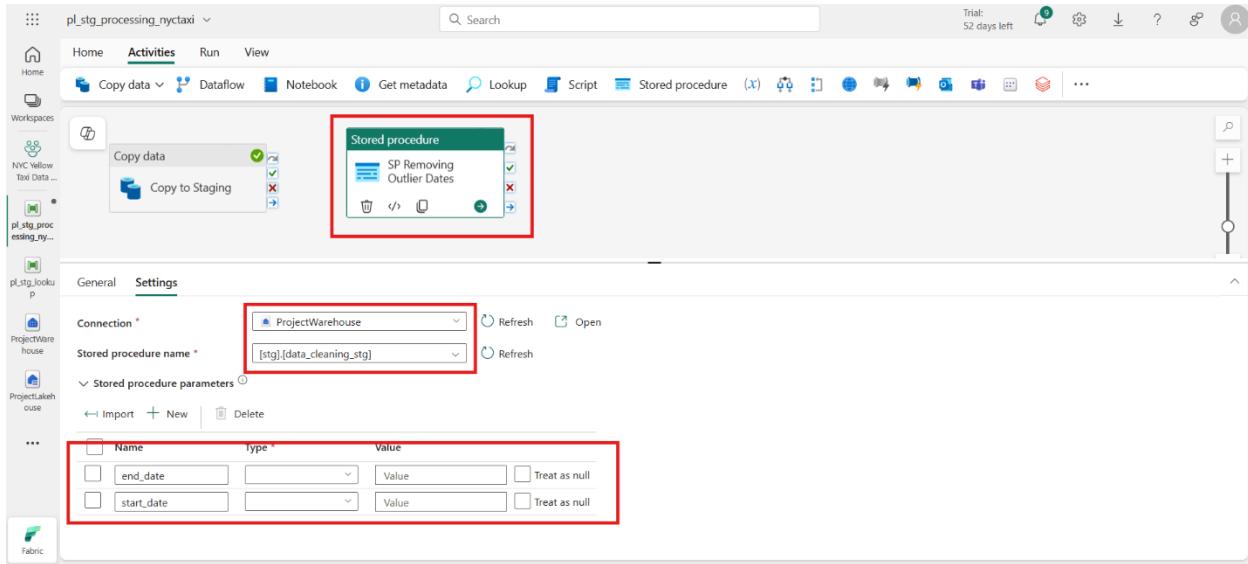
The screenshot shows the ProjectWarehouse interface. In the Explorer pane, under the 'stg' warehouse, the 'Stored Procedures' section contains 'data_cleaning_stg'. In the 'Queries' section, 'SQL query 2' is selected. The query is:

```
CREATE PROCEDURE stg.data_cleaning_stg
@start_date DATETIME2,
@end_date DATETIME2
AS
DELETE FROM stg.nyctaxi_yellow WHERE tpep_pickup_datetime > @end_date OR tpep_pickup_datetime < @start_date
```

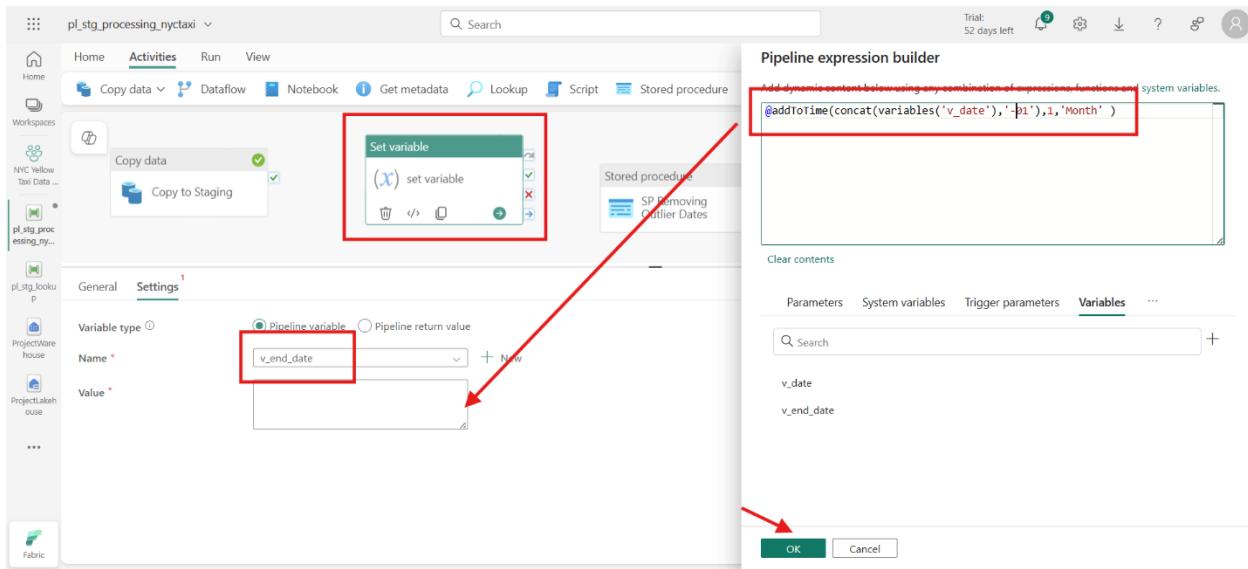
Back to the pl_stg_processing_nyctaxi pipeline and create stored procedure activity named as ‘SP_Removing_Outlier_Dates’.



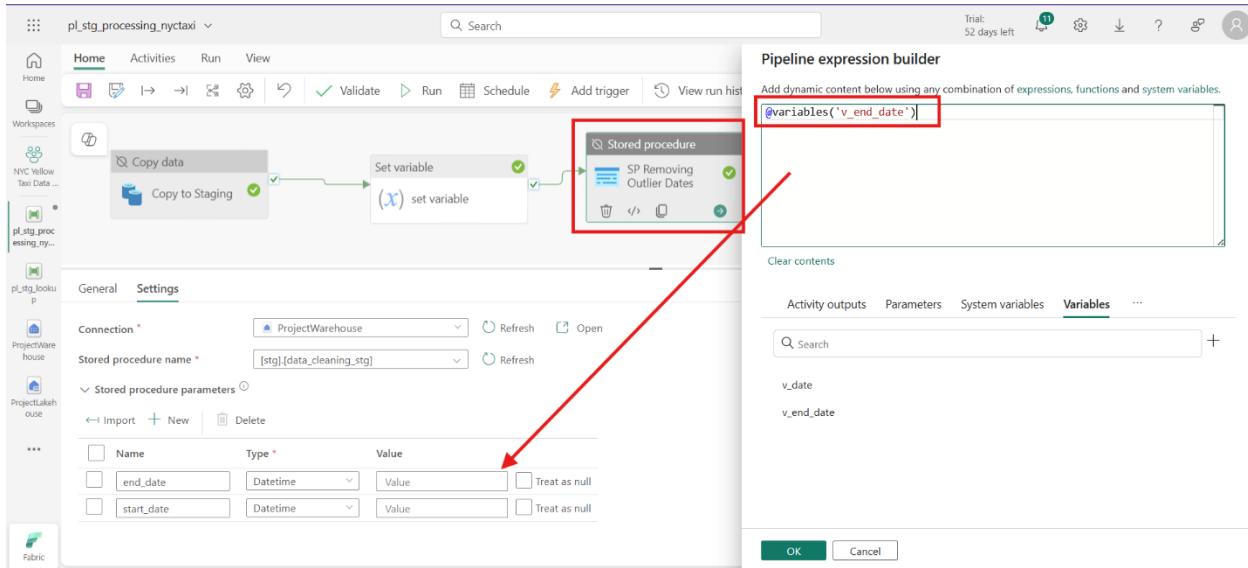
Provide details to the Settings tab as below figure.

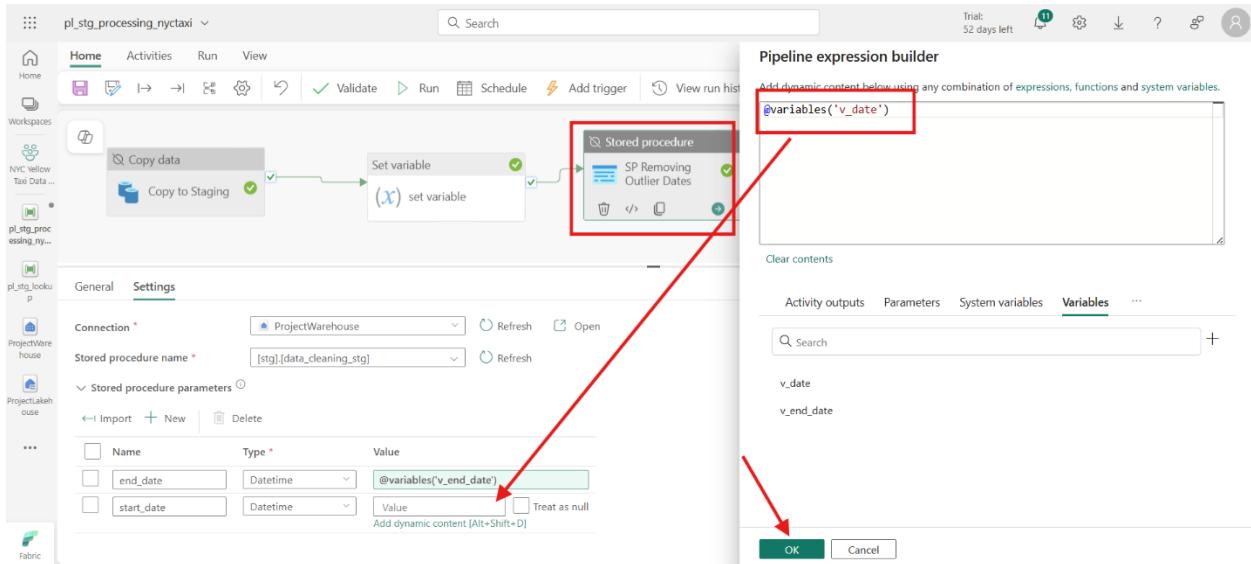


Create set variable activity to store v_end_date value. This variable used to increment month one by one.

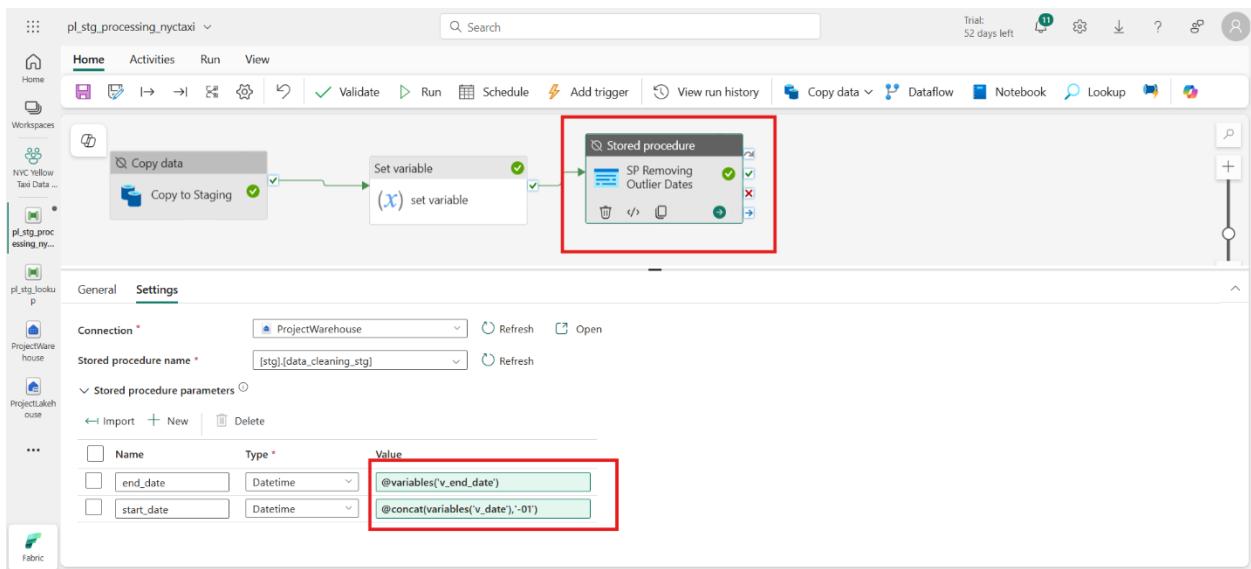


Go to the stored procedure activity and import variables and set values as shown in the figure.

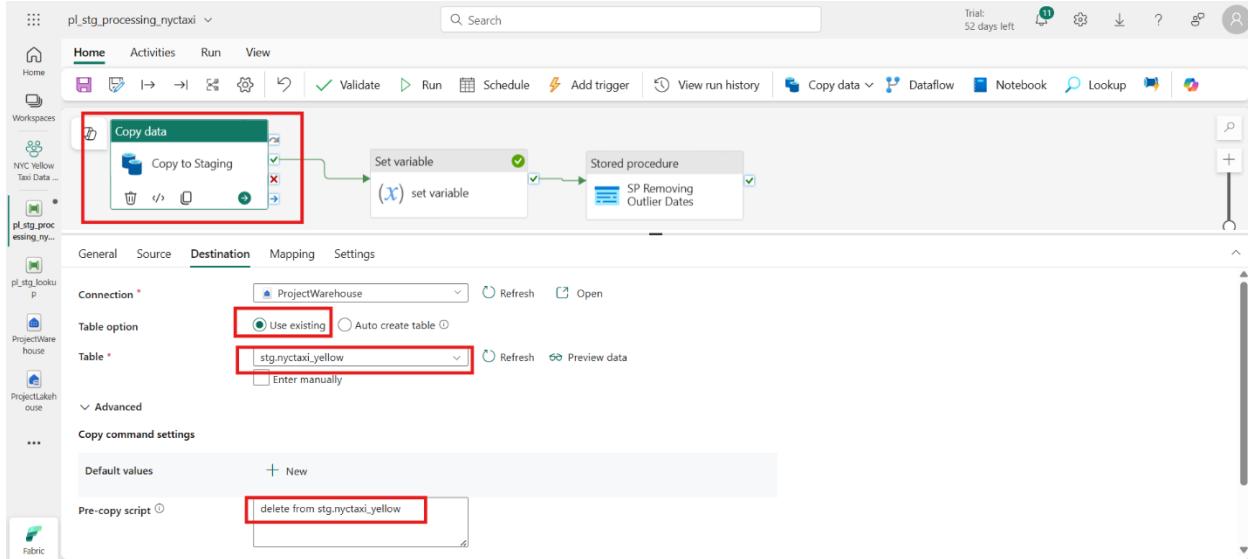




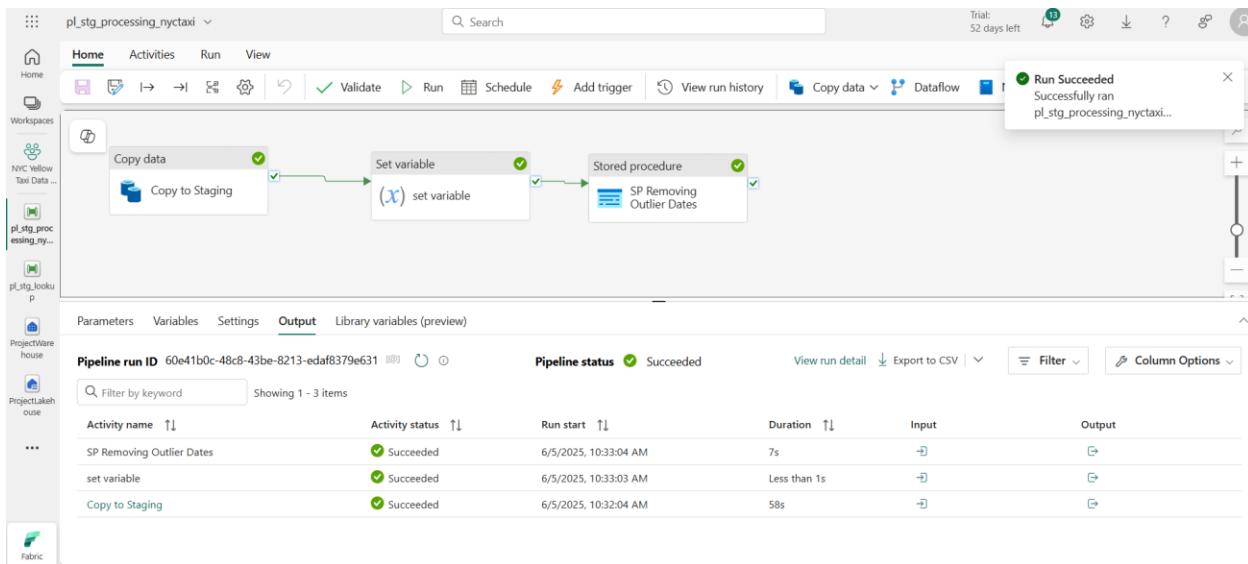
Finally stored procedure activity looks like this.



Back to the Copy Data activity change data as shown in the figure. Pre -copy script used to ensure delete previous data from `stg.nyctaxi_yellow` table. Still fabric doesn't support truncate data function that's why use this script here.



When run the created pipeline it's run successfully as shown inn the figure.



Create sql query to create metadata schema and under the metadata schema create table as metadata.processing_log.

The screenshot shows the Azure Data Studio interface. In the Explorer pane, under the 'ProjectWarehouse' warehouse, the 'Schemas' node is expanded, and the 'metadata' schema is selected. In the main SQL editor pane, the following SQL code is run:

```
create schema metadata;
create table metadata.processing_log
(
    pipeline_run_id varchar(255),
    table_processed varchar(255),
    rows_processed INT,
    latest_processed_pickup datetime2(6),
    processed_datetime datetime2(6)
);
```

The code is highlighted with a red box. The execution messages show the command was successful.

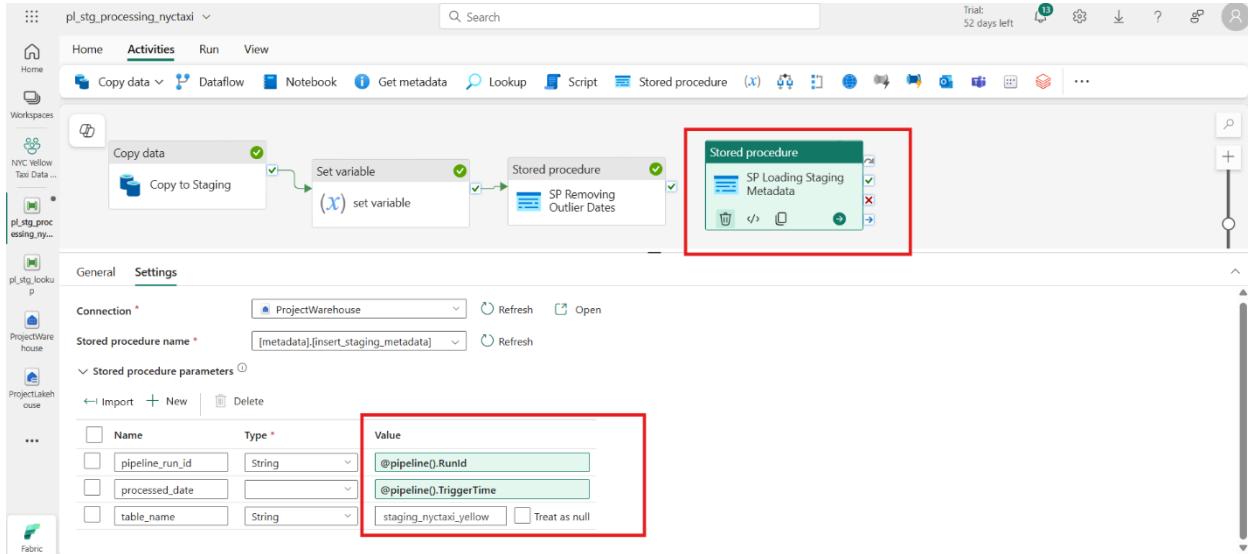
Create stored procedure to store data to metadata.processing_log table.

The screenshot shows the Azure Data Studio interface. In the Explorer pane, under the 'ProjectWarehouse' warehouse, the 'Stored Procedures' node is selected. In the main SQL editor pane, the following SQL code is run:

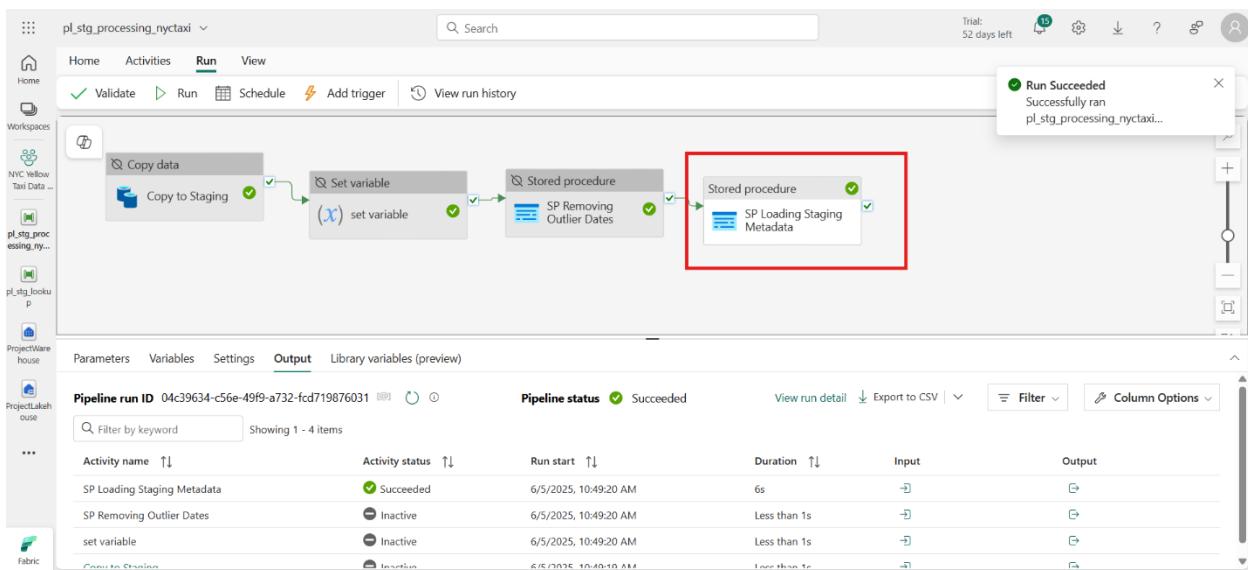
```
CREATE PROCEDURE metadata.insert_staging_metadata
    @pipeline_run_id VARCHAR(255),
    @table_name VARCHAR(255),
    @processed_date DATETIME2
AS
    INSERT INTO metadata.processing_log (pipeline_run_id, table_processed, rows_processed, latest_processed_pickup, processed_datetime)
    SELECT
        @pipeline_run_id AS pipeline_id,
        @table_name AS table_processed,
        COUNT(*) AS rows_processed,
```

The code is highlighted with a red box. The execution messages show the command was successful.

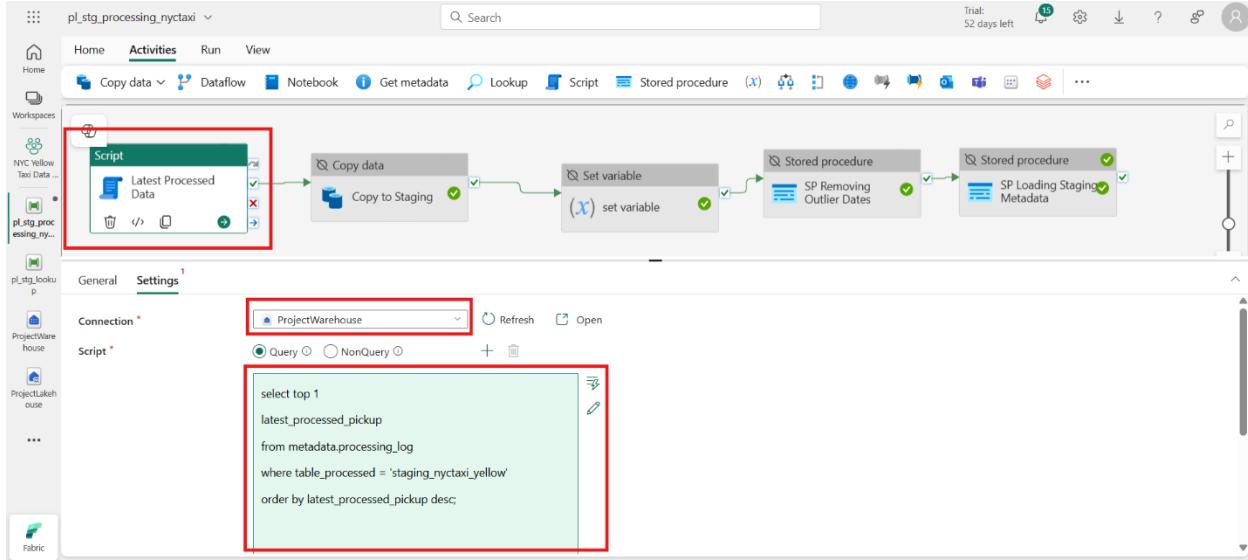
Create another stored procedure activity in pl_stg_processing_nyctaxi pipeline. Provide relevant details as shown in the below figure.



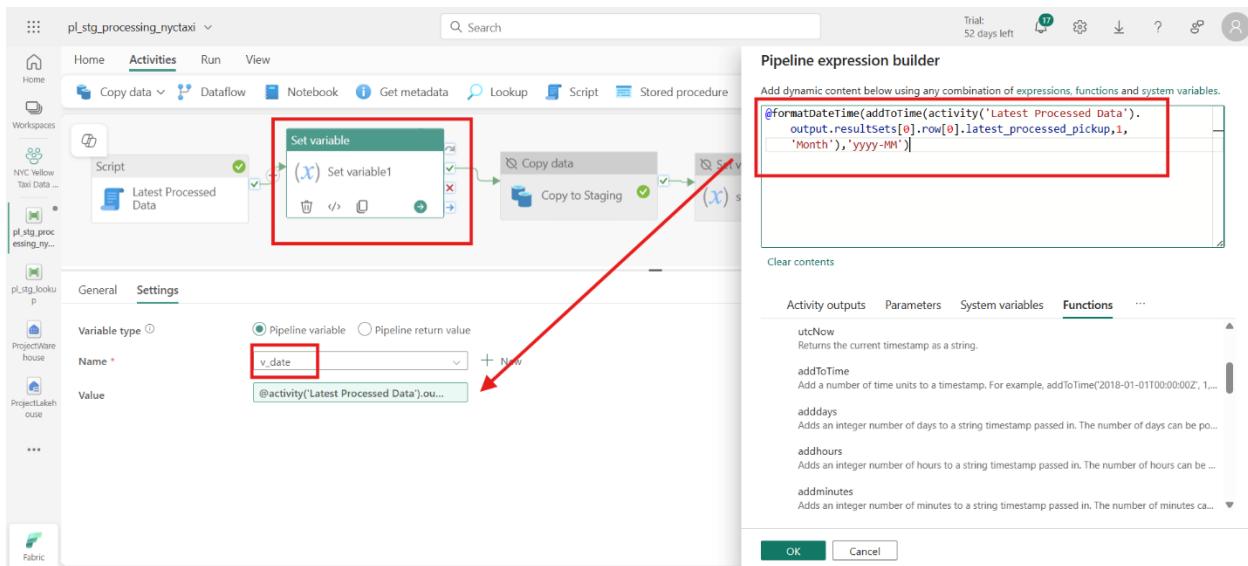
Deactivate remaining activities in the pipeline and run stored procedure activity only. It's run successfully as shown in the figure.



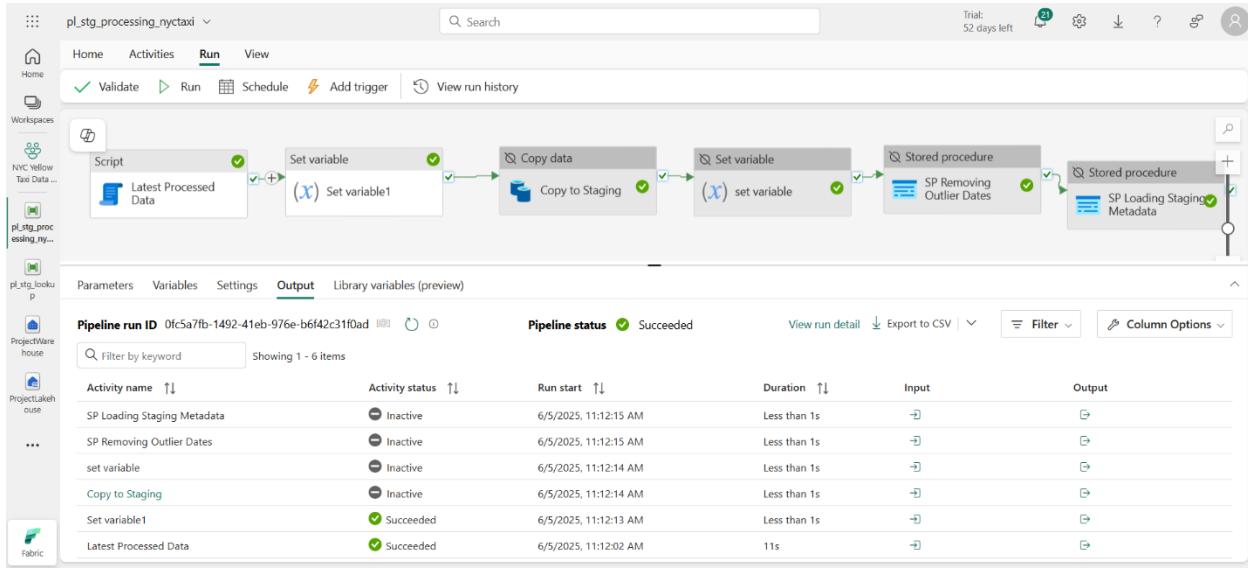
Create Script activity to get latest_processed_pickup top value. We only need the top value to further process.



Create another set variable activity after the script activity store that value. Provide details according to the shown figure.



Deactivate all the remaining activities in the pipeline and run only script and set variable activity.



Create Dataflow Gen2 activity to clean, transform and aggregated data.

After clean stg.nyctaxi_yellow table and stg.taxi_zone_lookup table merge both tables using left outer join. pu_location_id(stg.nyctaxi_yellow table) with LocationID(stg.taxi_zone_lookup table).

The screenshot shows the Power Query interface with the 'Merge' dialog open. The 'Left table for merge' is set to 'stg.nyctaxi_yellow' and the 'Right table for merge' is set to 'stg.taxi_zone_lookup'. The 'Merge' dropdown is set to 'Merge'. The preview pane shows the merged table with columns: trip_distance, pu_location_id, do_location_id, payment_method, and total_amount. The right pane shows the 'taxi_zone_lookup' table with columns: LocationID, Borough, Zone, and service_zone. A note at the bottom says 'The selection matches 2,964,606 of 2,964,606 rows from the first table'. The 'OK' button is highlighted.

Again, join merged table with stg.taxi_zone_lookup table using left outer join. do_location_id(merged table) with LocationID(stg.taxi_zone_lookup).

The screenshot shows the Power Query interface with the 'Merge' dialog open. The 'Left table for merge' is set to 'Merge' (the result of the previous merge), and the 'Right table for merge' is set to 'stg.taxi_zone_lookup'. The 'Merge' dropdown is set to 'Merge'. The preview pane shows the merged table with columns: trip_distance, pu_location_id, do_location_id, payment_method, total_amount, Borough, Zone, and service_zone. The right pane shows the 'taxi_zone_lookup' table with columns: LocationID, Borough, Zone, and service_zone. A note at the bottom says 'Estimating matches based on data previews'. The 'OK' button is highlighted.

Final look like this,

df_pres_processing_nytaxi

Power Query df_pres_processing_nytaxi Dataflow saved

Home Transform Add column View Help

Queries [4]

ABC 123

moved columns Inserted condition... Reordered column... Removed column... Inserted condition... Reordered column... Removed column... Renamed columns Calculated date Changed column...

ABC 123 ...

Source 2

Merge

Expanded stg tax...

Source 2

Merge (2)

Renamed column

Query settings

df_pres_processing_nytaxi

ProjectWares house

ProjectLakehouse

df_pres_processing_nytaxi

Table.ReorderColumns("#"Removed columns 1", "vendor", "tpep_pickup_datetime", "tpep_dropoff_datetime", "pu_borough", "pu_zone", "do_borough", "do_zone", "payment_method", "passenger_count", "trip_distance", "total_amount")

	vendor	tpep_pickup_datetime	tpep_dropoff_datetime	pu_borough	pu_zone	do_borough	do_zone	payment_method	passenger_count	trip_distance	total_amount
1	Verifone	1/2/2024	1/2/2024	Queens	JFK Airport	Manhattan	Clinton East	Cash	1	17.99	-87.69
2	Verifone	1/3/2024	1/3/2024	Queens	JFK Airport	Manhattan	Clinton East	Dispute	1	17.76	-87.69
3	Verifone	1/3/2024	1/3/2024	Queens	JFK Airport	Manhattan	Clinton East	Dispute	1	23.35	-87.69
4	Verifone	1/10/2024	1/10/2024	Queens	JFK Airport	Manhattan	Clinton East	Dispute	1	17.62	-87.69
5	Verifone	1/10/2024	1/10/2024	Queens	JFK Airport	Manhattan	Clinton East	Dispute	1	18.75	-87.69
6	Verifone	1/16/2024	1/16/2024	Queens	JFK Airport	Manhattan	Clinton East	Dispute	1	18.5	-87.69
7	Verifone	1/16/2024	1/16/2024	Queens	JFK Airport	Manhattan	Clinton East	Dispute	1	19.37	-87.69
8	Verifone	1/16/2024	1/16/2024	Queens	JFK Airport	Manhattan	Clinton East	No Charge	1	19.15	-87.69

Columns: 11 Rows: 99+ Add default destination...

Step

Publish

Back to data pipeline folder and create new pipeline named as ‘pl_pres_processing_nytaxi’. Create Dataflow activity and provide relevant details to the setting pane.

The screenshot shows the Azure Data Factory interface. At the top, there's a navigation bar with 'Home', 'Activities', 'Run', and 'View' tabs. Below the navigation bar is a toolbar with various icons for actions like Validate, Run, Schedule, Add trigger, View run history, Copy data, Dataflow, Notebook, Lookup, and more. On the left side, there's a sidebar with 'Workspaces' and a list of projects: 'Omlake catalog', 'Monitor', 'NYC Yellow Taxi Data ...', 'pl_pres_pro ...', 'ProjectWare house', and 'ProjectLakehouse'. The main area displays a Dataflow card titled 'Dataflow' with the sub-card 'Process to Presentation'. This Dataflow card is highlighted with a red box. At the bottom, the 'Settings' tab is active, showing the 'Workspace' dropdown set to 'NYC Yellow Taxi Data Project' and the 'Dataflow' dropdown set to 'df_pres_processing_nyctaxi', both also highlighted with red boxes.

Select destination from data warehouse as nyctaxi_yellow table to store cleansed data.

The screenshot shows the 'Choose destination target' dialog in Power Query. The 'Existing table' radio button is selected. The 'nyctaxi_yellow' table is listed under the 'ProjectWarehouse' connection. The table details are shown as follows:

Column	Type	Notes
vendor	String	
tpep_pickup_datetime	DateTime	
tpep_dropoff_datetime	DateTime	
pu_borough	String	
pu_zone	String	
do_borough	String	
do_zone	String	
payment_method	String	1,2
passenger_count	Number	1,2
trip_distance	Number	1,2

The table has no rows.

Create Stored procedure activity as shown in the figure.

The screenshot shows the 'Stored procedure' activity configuration screen in the Azure Data Factory interface. The 'Connection' dropdown is highlighted with a red box. The 'General' tab is selected. The 'Settings' tab is also visible. The 'Stored procedure name' field contains the value '[metadata].[insert_presentation_met...]'.

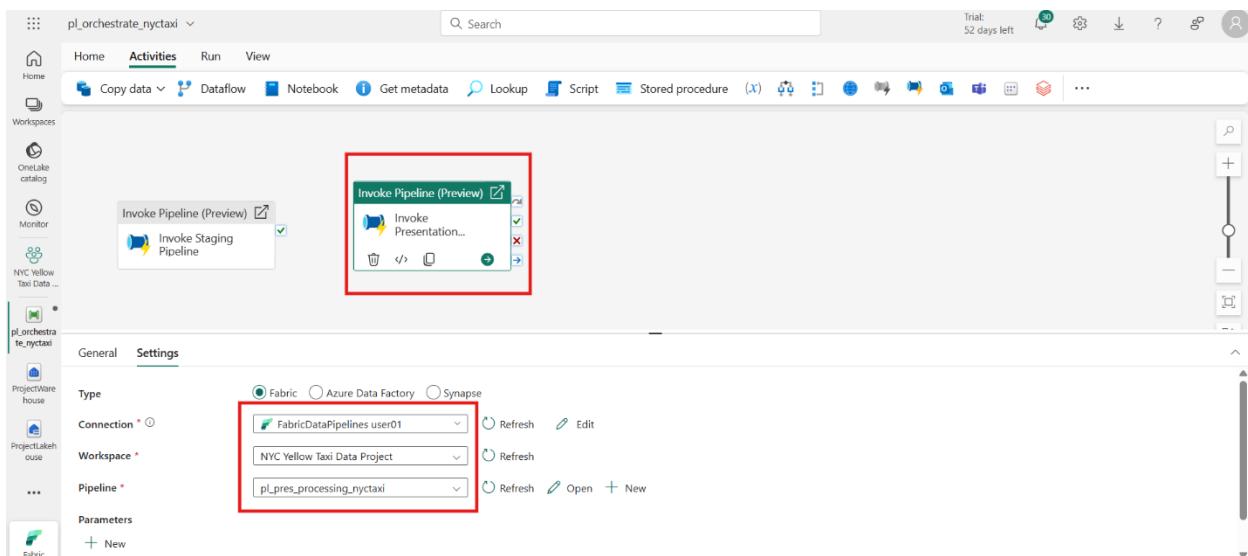
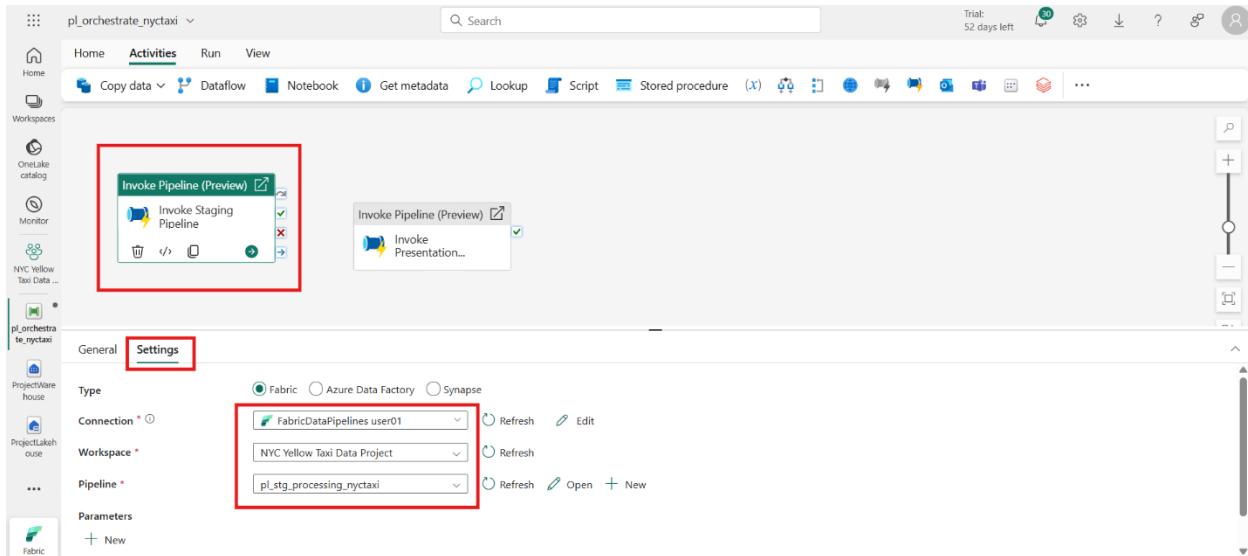
Provide further details to the setting tab as shown in the below figure.

The screenshot shows the 'Settings' tab for a pipeline named 'pl_pres_processing_nyctaxi'. A red box highlights the 'Stored procedure' section, which contains a 'SP Loading Presentation...' activity. Another red box highlights the 'Stored procedure parameters' section, showing three parameters: 'pipeline_run_id' (String type, value: '@pipeline().RunId'), 'processed_date' (Datetime type, value: '@pipeline().TriggerTime'), and 'table_name' (String type, value: 'presentation_nyctaxi.y...').

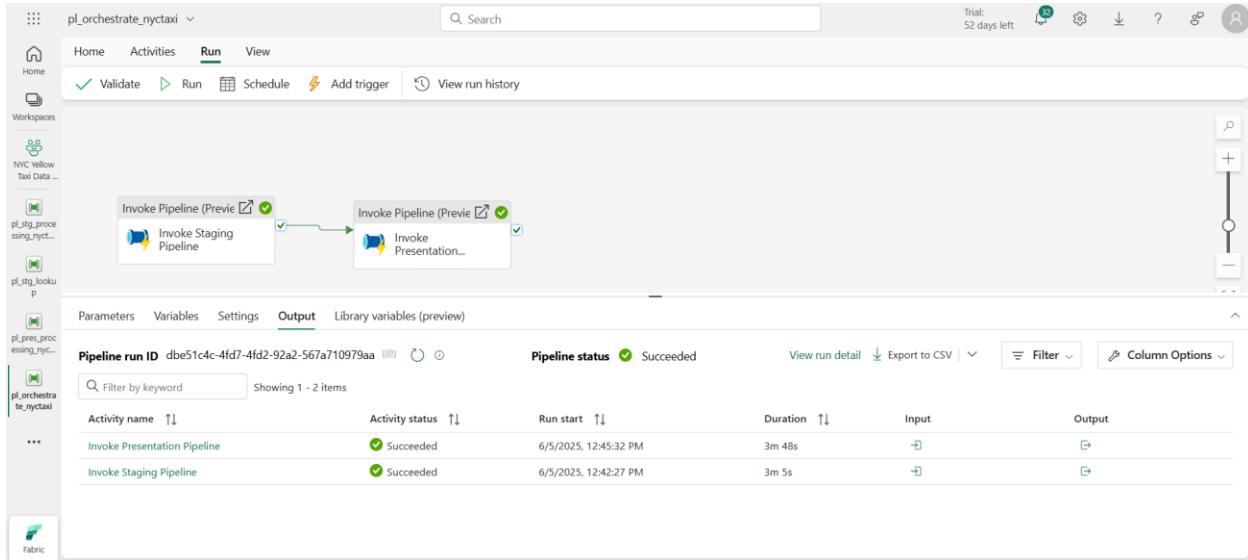
Run this pipeline and get successfully run message.

The screenshot shows the 'Run' tab for the same pipeline. It displays a successful run with Pipeline run ID '726e737f-2410-4e94-a138-9b82e3407b0b'. The Pipeline status is 'Succeeded'. A success message box is visible in the top right corner. Below the pipeline activities, a table shows activity runs with columns: Activity name, Activity status, Run start, Duration, Input, and Output. A progress bar indicates 'Loading activity runs'.

Create final orchestration pipeline to invoke pl_stg_processing_nyctaxi pipeline and pl_pres_processing_nyctaxi pipeline.



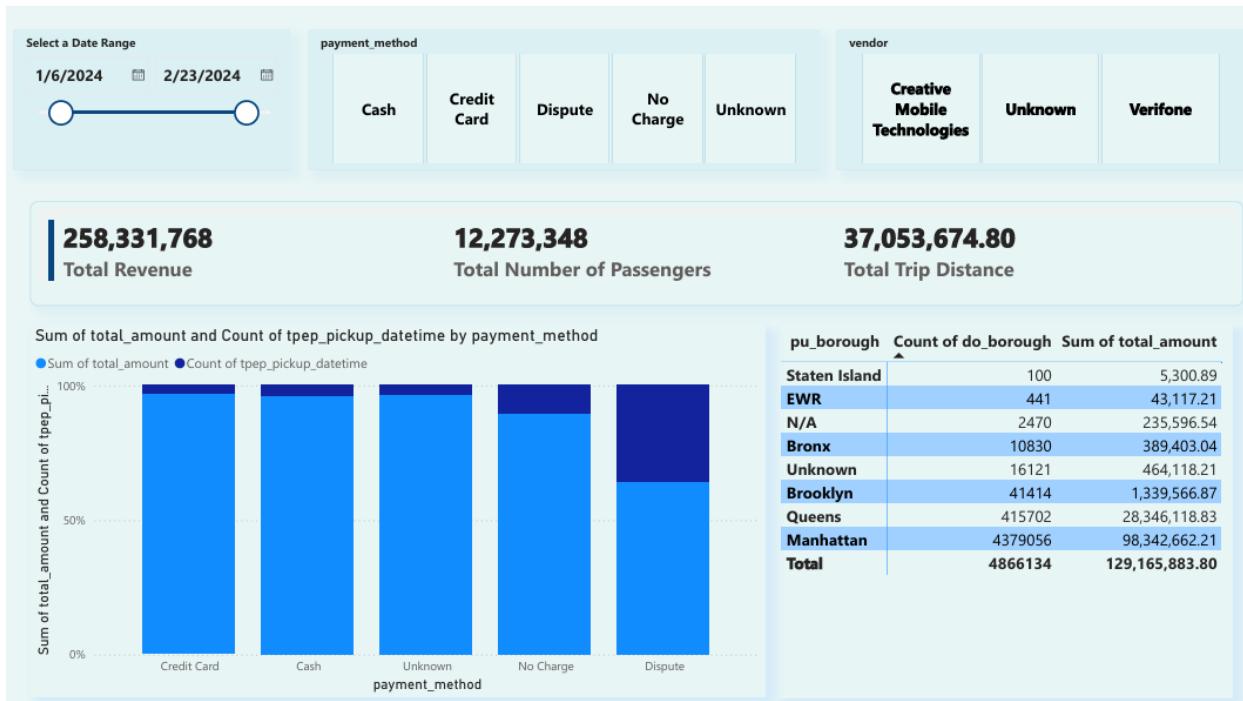
Final orchestration pipeline ran successfully.



Create New semantic model and create power BI report using dbo.nyctaxi_yellow table.

The screenshot shows the ProjectWarehouse reporting interface. A modal dialog titled "Manage default semantic model" is open, prompting the user to select or deselect tables for the semantic model. The "Tables" section is expanded, showing the "nyctaxi_yellow" table selected. The "Confirm" button is visible at the bottom right of the dialog.

Created simple dashboard looks like this,



Final fabric workspace looks like this,

The Fabric workspace interface shows the following structure for the "NYC Yellow Taxi Data Project":

- NYC Taxi Data Pipelines:** Folder containing:
 - df_pres_processing_nyctaxi: Dataflow Gen2
 - NYC Semantic Model
 - NYC Taxi Data Dashboard: Report
 - ProjectLakehouse: Lakehouse
 - ProjectLakehouse: Semantic model
 - ProjectLakehouse: SQL analytics
 - ProjectWarehouse: Warehouse
 - ProjectWarehouse: Semantic model

End