Transport for Cairo مواصلات للقاهرة

TfC Tools Plugin

User Guide

TfC Tools Plugin User Guide v1.1 Date 11/17/2025

File Name	V	Date	Changes	Authors	Contributors
TfC Tools Plugin - User Guide.docx	1.1	17-Nov-25 5:35:00 PM		SME	AME

Table of Contents

List of Figures

List of Tables

1. Introduction

TfC Tools is a QGIS plugin suite developed by **Transport for Cairo (TfC)** to streamline and standardize transport data processing workflows.
The suite provides user-friendly interfaces for common analytical tasks used in TfC's research and transport data management projects.

TfC Tools currently includes 3 main plugins:

1. RL2SDI (RouteLab to SDI Migration) - migrates field survey data from TfC's RouteLab database to a standardized PostGIS Spatial Data Infrastructure (SDI)
2. GIS2GTFS – converts GIS data into GTFS.
3. Vehicle and Passenger Flow – estimates vehicle flows and passenger flows on road segments based on GTFS data.

1.1 Who this guide is for

This guide is intended for:

- Transport analysts and GIS specialists working with RouteLab, GTFS, or PostGIS data.
- Researchers and planners aiming to integrate transport data workflows into QGIS.
- Developers or contributors who wish to understand or extend the TfC Tools plugin.

You can use the plugins independently or as part of a workflow:
RouteLab → RL2SDI → GIS2GTFS → Vehicle and Passenger Flow.

1.2 Tool requirements

The plugins are compatible with **QGIS 3.40 and later**.

2. TfC Tools

1.3 Installing the plugin

To install **TfC Tools**, you'll first need to download the plugin package and add it to QGIS.

1.3.1 Option 1: Manual installation from ZIP

1. Download the latest plugin ZIP from the GitHub repository
2. Download and open QGIS (version 3.40 or higher)
3. From the menu bar, go to **Plugins** → **Manage and Install Plugins** (Figure 1)

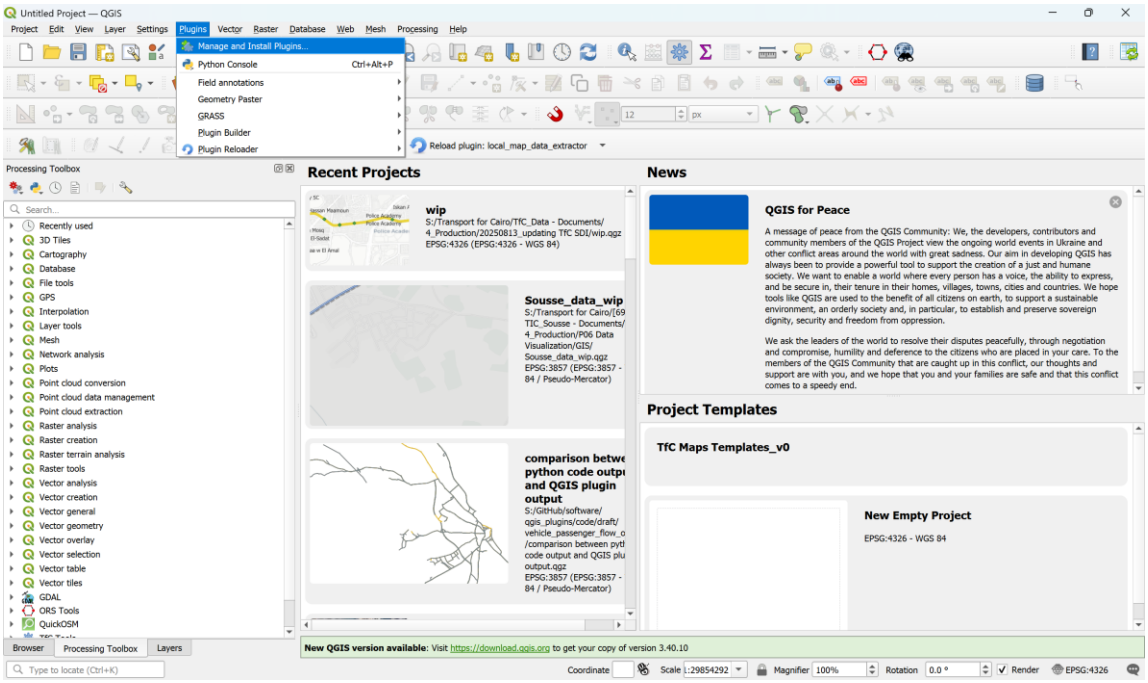


Figure 1 Installing the plugin – select the Plugin menu

4. In the dialog that appears, click the **Install from ZIP** → click **Browse** and select the downloaded ZIP file on your PC → click **Install Plugin** → click **Close** (Figure 2)

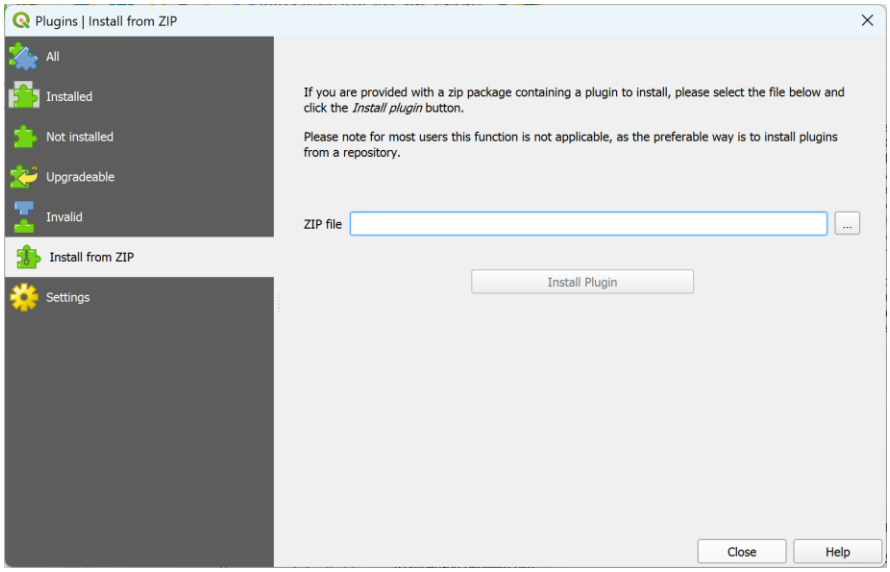


Figure 2 Installing the plugin - Insert from ZIP

1.3.2 Option 2: Install from QGIS Plugin Repository

TfC Tools are available directly through QGIS' Plugin Repository:

1. Open **Plugins** → **Manage and Install Plugins…**
2. Search for **TfC Tools**.
3. Click **Install Plugin**.

1.4 Where to find the plugin

After installation:

Open **Plugins** → **TfC Tools** and make sure it's checked to enable it as in Figure 3.

You can find TfC Tools under the **Processing Toolbox** panel as in Figure 4.

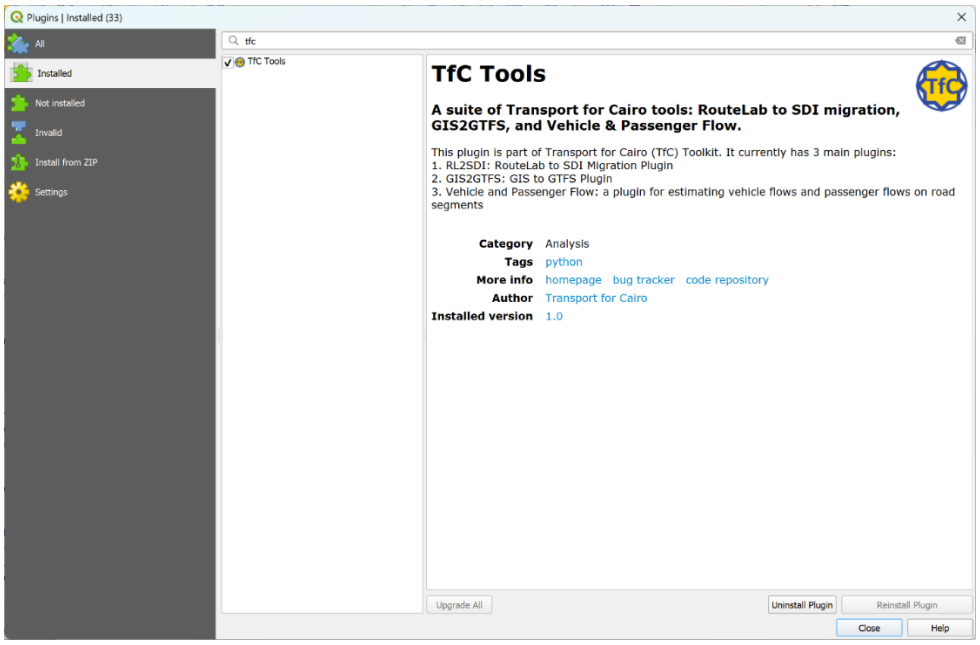


Figure 3 TfC Tools Plugin in the Plugins menu

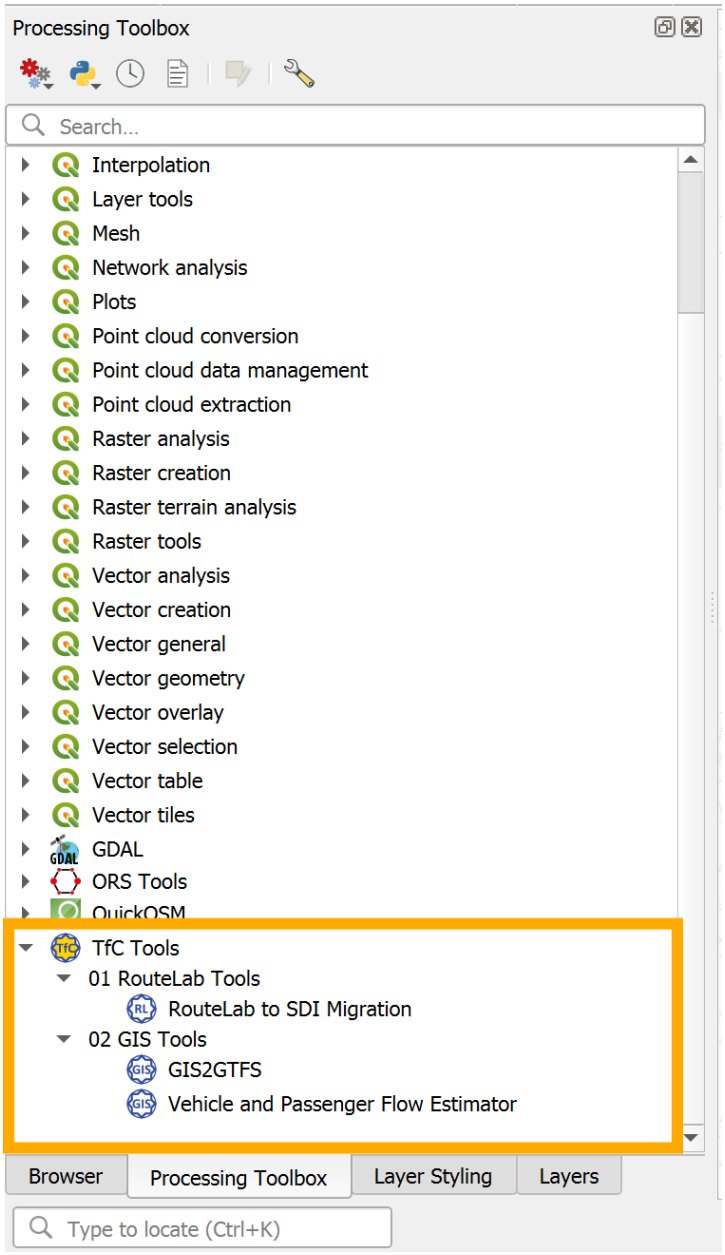


Figure 4 TfC Tools in the Processing Toolbox panel

3. RL2SDI Migration Plugin

1.5 Purpose of the Tool

The plugin migrates field surveys data from the Transport for Cairo TfC's suite RouteLab database to a PostgreSQL database in a standard format that can later be used for analysis.

The plugin automates the migration of field survey data from **Transport for Cairo's (TfC) RouteLab** database into a **Postegres Database**, following a standardized schema that is suitable for analysis and interoperability, particularly with the other **TfC Tools** plugins, **GIS2GTFS** and **Vehicle and Passenger Flow**.

1.6 How to use the Plugin

1.6.1 Plugin User Interface

The plugin requires the following 3 inputs:

1. **RouteLab database connection** – The source database containing field survey data (Database credentials are provided by TfC)
2. **PostgreSQL database connection** – The target database where the data will be migrated and structured.
3. **Project ID** – A unique identifier assigned to each project within TfC's RouteLab.

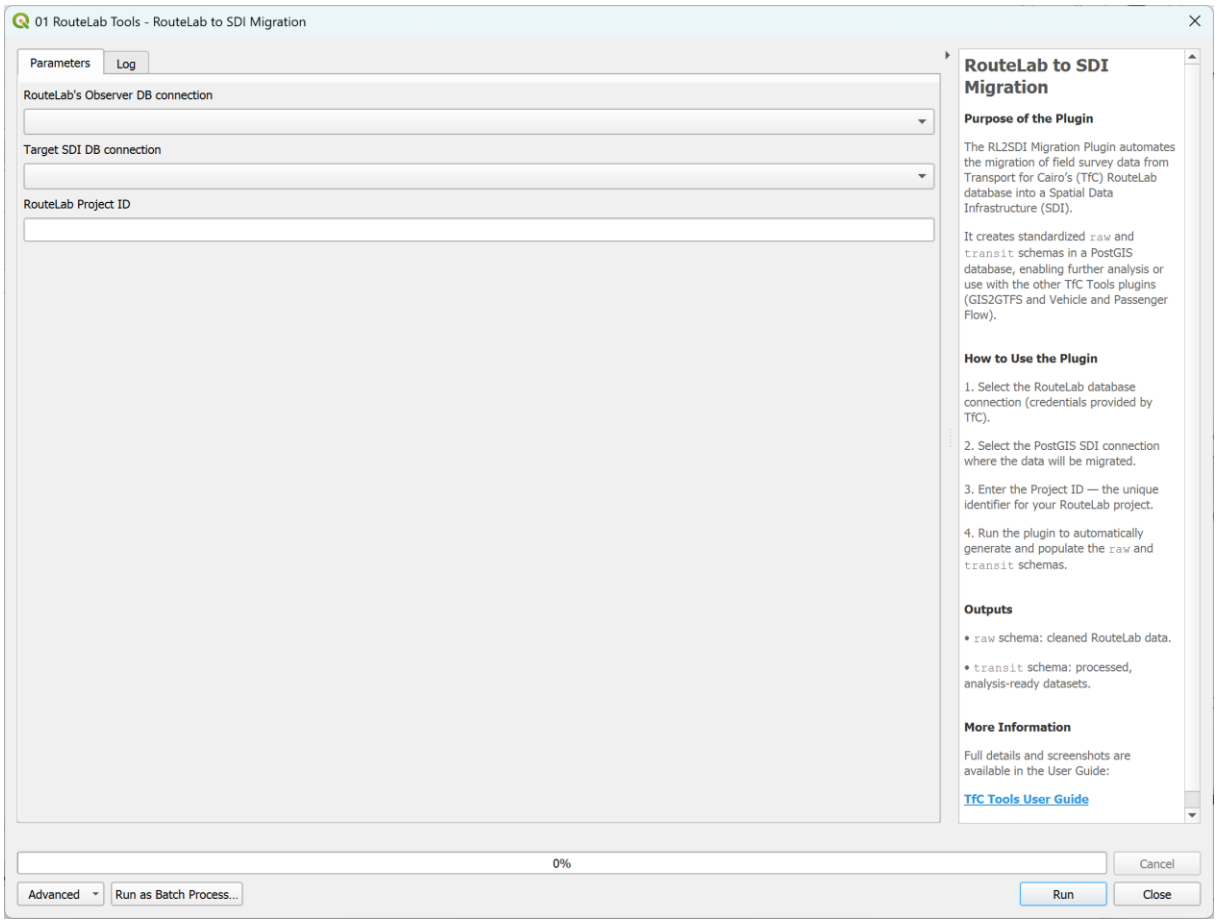


Figure 5 RL2SDI Plugin User Interface

1.6.2 Plugin output

Once the plugin completes its process, two new schemas will be created within your PostgreSQL database: **raw** and **transit**.

- **raw schema** – contains the cleaned data imported from the RouteLab field surveys.
- **transit schema** – contains the processed and analysis-ready datasets generated from the raw data.

A full description of the structure, including all tables, fields, and data types within these two schemas, is provided in Section 1.13 in the Appendix.

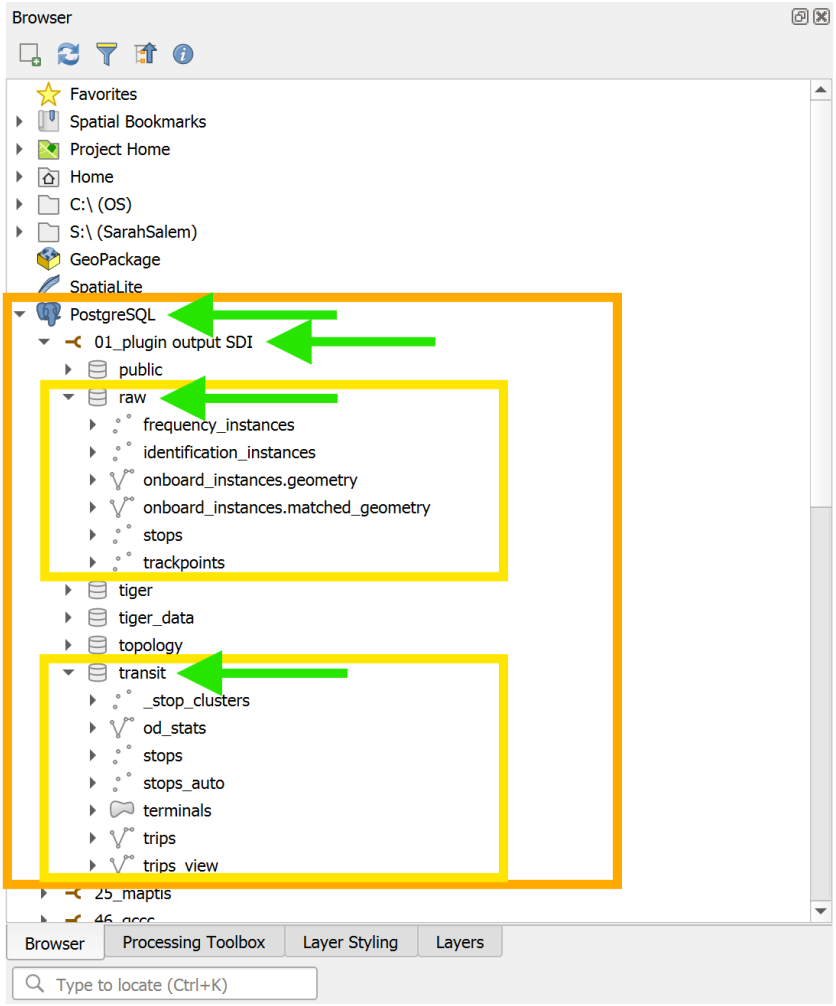


Figure 6 the plugin output PostgreSQL database

1.6.3 Next steps

Once the RouteLab-to-SDI migration is complete, the data can be used directly for analysis and visualization in QGIS or other GIS tools.
Alternatively, you can continue the workflow using the other **TfC Tools** plugins—**GIS2GTFS** and **Vehicle and Passenger Flow**—which rely on the same standardized PostgreSQL database schema generated by this plugin.

4. GIS2GTFS Plugin

1.7 Purpose of the Plugin

The plugin automates the creation of a **General Transit Feed Specification (GTFS)** dataset from existing transport data stored in a **PostgreSQL database**. It relies on the standardized PostgreSQL database schema used by **Transport for Cairo (TfC)**—the same schema generated by the **RL2SDI** plugin—ensuring full compatibility with other tools and workflows.

1.8 How to use the plugin

1.8.1 Plugin User Interface

The plugin requires the following inputs:

1. **PostgreSQL database connection** – must follow TfC's standard schema, either by:
   - using the PostgreSQL database output from the **RL2SDI Migration Plugin**, or
   - Structuring your data according to the schema described in **Appendix Table 4**.
2. **Feed version** – e.g. 1.0
3. **Start date** – e.g. 20250101
4. **End date** – e.g. 20251231
5. **Service ID** – defines the service configuration for the GTFS feed.
6. **Use continuous drop-off/pick-up** – an optional GTFS parameter enabled by default; you may disable it if not applicable.
7. **Output folder 1** – temporary folder for intermediate raw files used by the plugin. *(These files are not needed after processing; an empty folder is recommended.)*
8. **Output folder 2** – destination folder for the final GTFS .txt files.

The Plugin interface is shown in Figure 5 below.

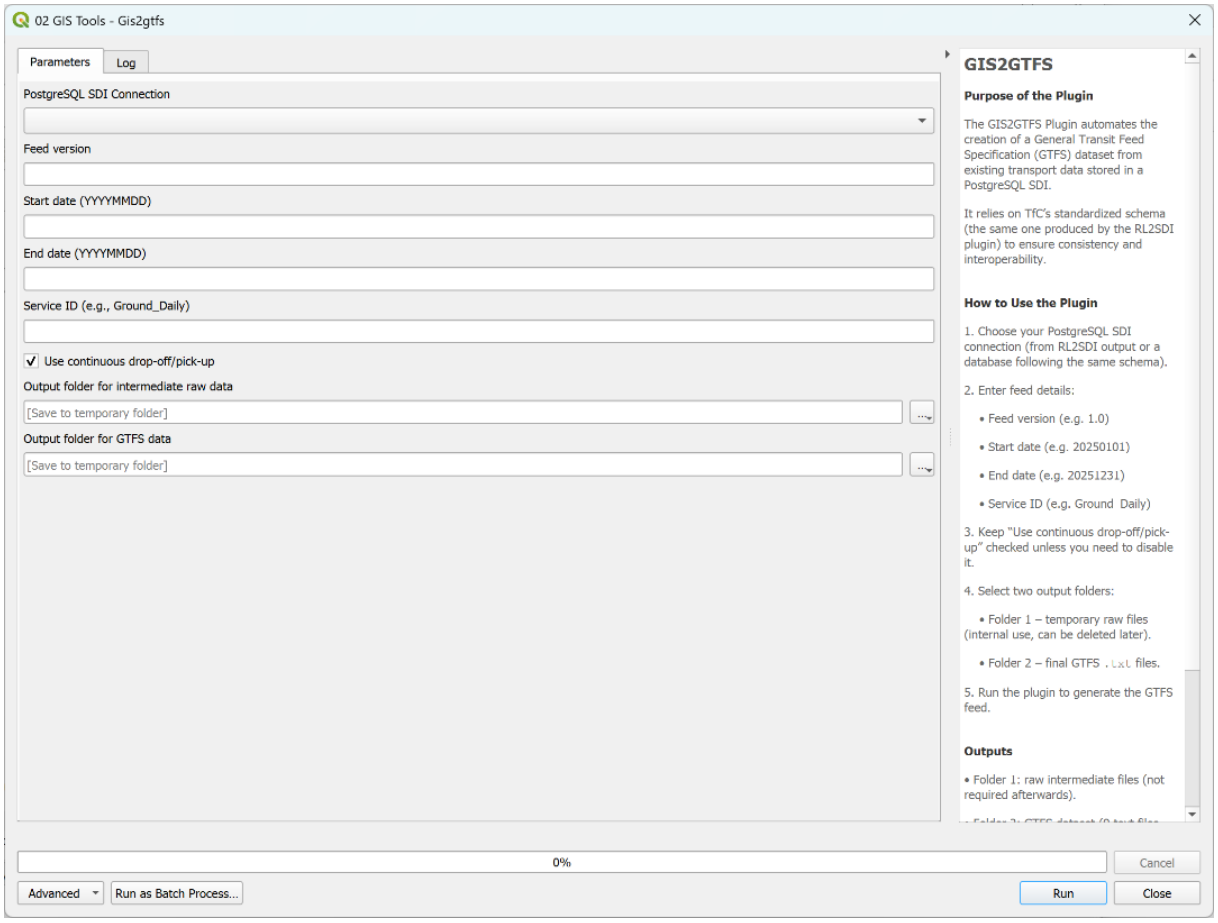


Figure 7 GIS2GTFS Plugin User Interface

1.8.2 Plugin output

Upon execution, the plugin generates two sets of outputs corresponding to the two folders selected in the UI:

- **Raw output (Folder 1)** – intermediate processing files not required after the GTFS is created (see Figure 8).
- **GTFS feed (Folder 2)** – the main output, consisting of **9 text files** that make up the GTFS dataset (see Figure 9).

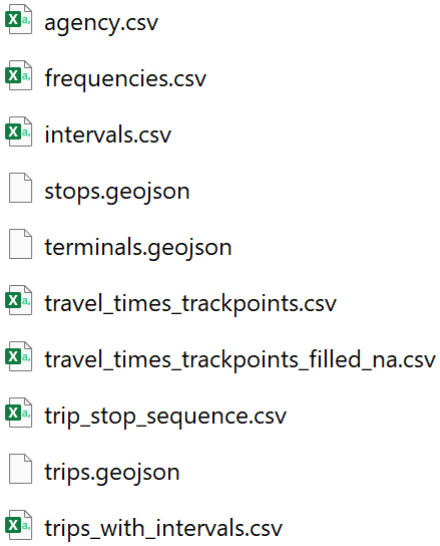


Figure 8 Raw output in Folder 1

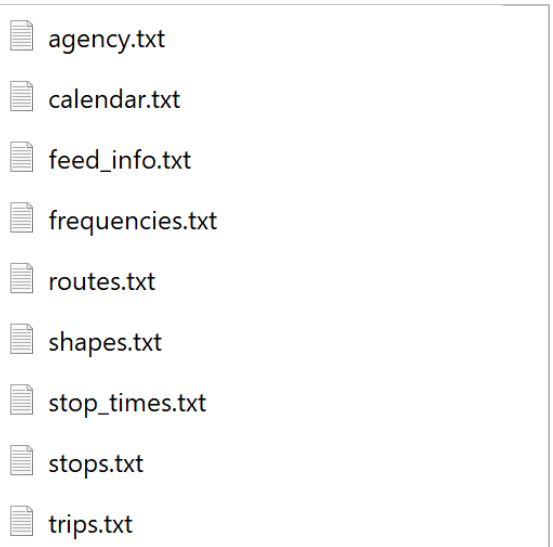


Figure 9 Main GTFS output in Folder 2

1.8.3 GTFS validation

To ensure the GTFS feed meets specification standards, it is recommended to validate it using **MobilityData's GTFS Validator** by following these steps:

1. Compress the generated .txt files into a single .zip file
2. Download the open [MobilityData GTFS Validator]
3. Open the validator and select the location of the GTFS .zip file
4. Click **Validate** To run the checks. A webpage will open displaying the validation report.
5. Review the results:
   - **Errors** must be corrected before the GTFS feed can be considered valid.
   - **Warnings** are non-critical, and the GTFS will still function, but it's recommended to review and resolve them when possible.

5. Vehicle and Passenger Flow Plugin

1.9 Purpose of the Tool

The plugin estimates **vehicle** and **passenger flows** within a defined study area using **GTFS data**. It processes vehicle appearances and passenger load information, calculates flows for selected time intervals (e.g., morning and afternoon peaks), and outputs spatial layers ready for analysis and visualization in **QGIS**.

The tool can be used with **any valid GTFS feed**, whether produced by the **GIS2GTFS** plugin or obtained from another source. It is designed to integrate seamlessly with data structured under the standard PostgreSQL database schema generated by **RL2SDI**.

1.10 How to use the Plugin

1.10.1 Plugin User Interface

The plugin requires the following 3 inputs:

1. **GTFS file (.zip)** – The input GTFS dataset, either created using the **GIS2GTFS** plugin or any valid GTFS feed.
2. **PostgreSQL database connection** – must follow TfC's standard schema, either by:
   - using the PostgreSQL database output from the **RL2SDI Migration Plugin**, or
   - Structuring your data according to the schema described in **Appendix Table 5.**
3. **Output folder path** – The location on your computer where the generated files will be saved.

The plugin interface is shown in Figure 10 below.

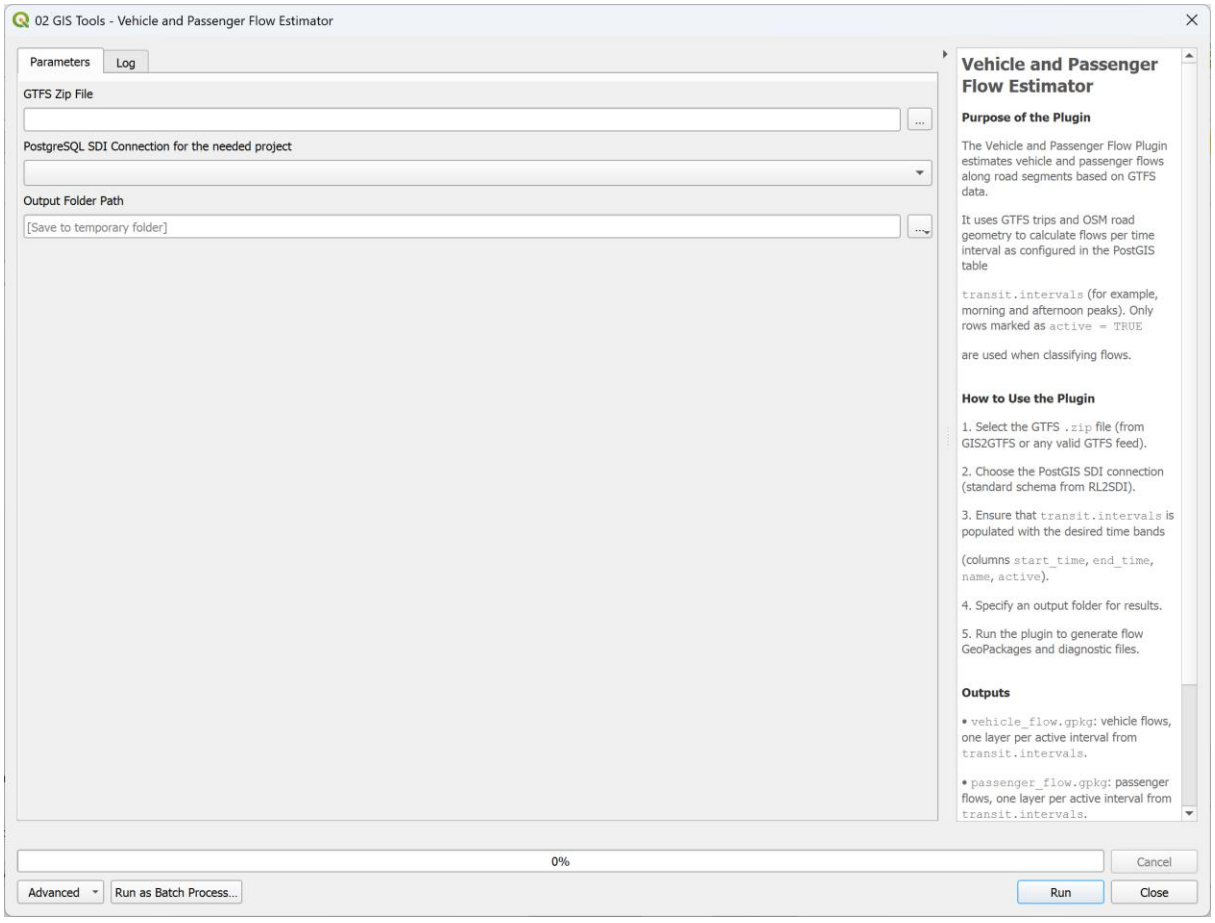


Figure 10 Vehicle and Passenger Flow Plugin User Interface

1.10.2 Plugin output

The plugin produces **two GeoPackage (.gpkg)** files, each containing two layers—one for the **morning peak** and one for the **afternoon peak**:

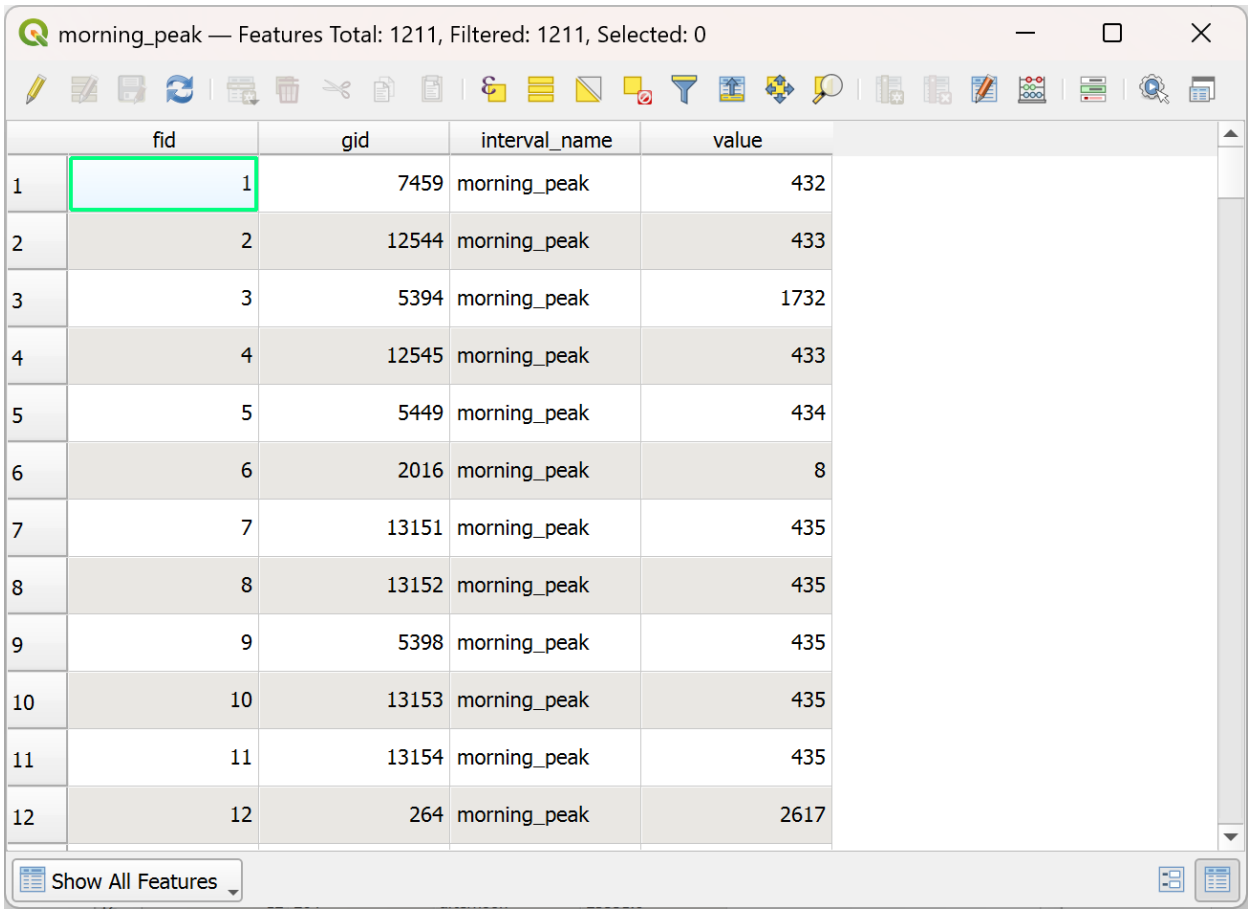
- **vehicle_flow.gpkg** – Vehicle flow data by road segment.
- **passenger_flow.gpkg** – Passenger flow data by road segment.

Each layer includes line geometries and associated flow attributes, ready for visualization and further analysis in QGIS.

Table 1 and Figure 11 illustrate sample outputs produced by the plugin.

Table 1 Main fields from the Vehicle and Passenger Flow plugin output

Field name	Description
interval_name	Interval name.
value	The main output of the plugin. This is the number of passengers or vehicles estimated on this road segment per time interval.



	fid	gid	interval_name	value
1	1	7459	morning_peak	432
2	2	12544	morning_peak	433
3	3	5394	morning_peak	1732
4	4	12545	morning_peak	433
5	5	5449	morning_peak	434
6	6	2016	morning_peak	8
7	7	13151	morning_peak	435
8	8	13152	morning_peak	435
9	9	5398	morning_peak	435
10	10	13153	morning_peak	435
11	11	13154	morning_peak	435
12	12	264	morning_peak	2617

Figure 11 example for passenger flow output in a time interval (morning peak)

1.11 What the Plugin Does

Here's a quick overview of what the plugin does once you run it:

1. **Reads the GTFS data** from the provided .zip file.
2. Defines the **study area** and extracts the **OSM road network** that covers all transit routes in the GTFS.
3. Matches **transit routes to road segments** using the *Fréchet distance* method to determine which roads each trip follows.
4. **Estimates and assigns vehicle and passenger flows** to each road segment for defined time periods (including morning and afternoon peaks).
5. **Saves the results** as GeoPackage (.gpkg) files.
6. Performs **data quality and consistency checks**, validating geometries and attributes, and handling missing or incomplete data using fallback methods.

6. Appendix

1.12 Setting up a PostgreSQL Database Connection on QGIS

For the TfC Tools plugin to function properly, a **PostgreSQL database connection** with **saved credentials** is required.

This section explains how to create the connection and ensure that the credentials are correctly stored.

Note: This setup only needs to be done once on each computer.

1.12.1 Create a new connection

1. Open the **Browser** panel. If you can't find it go to **View** → **Panels** → check **Browser**
2. Right click on **PostgreSQL** and choose **New Connection**
3. In the dialog, fill in the required fields — **Name**, **Host**, **Port**, and **Database**.
4. Under the **Basic** tab, enter your **User name** and **Password**, check **Store**, then click **OK**.
5. Check **Also list tables with no geometry** to be able to see them in the Brower menu

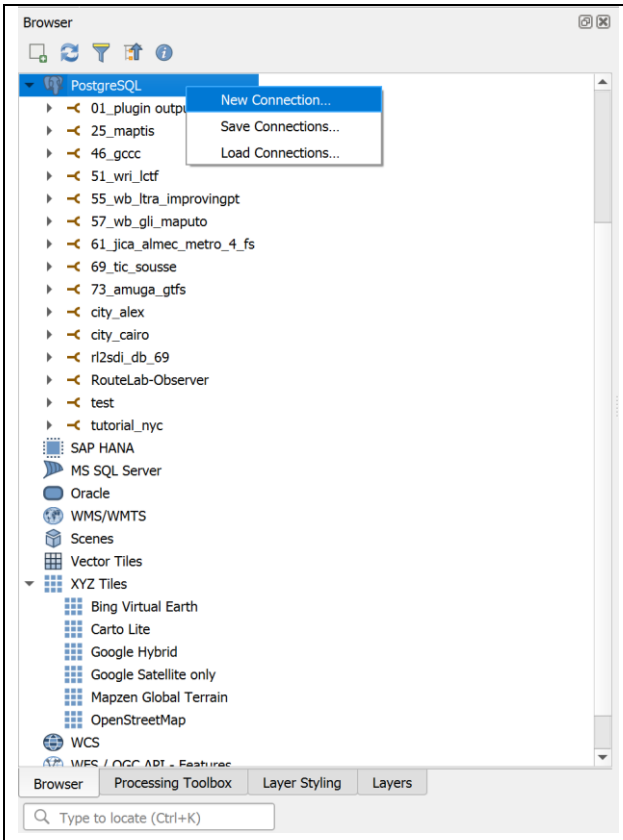


Figure 12 adding a new PostgreSQL connection: step 1

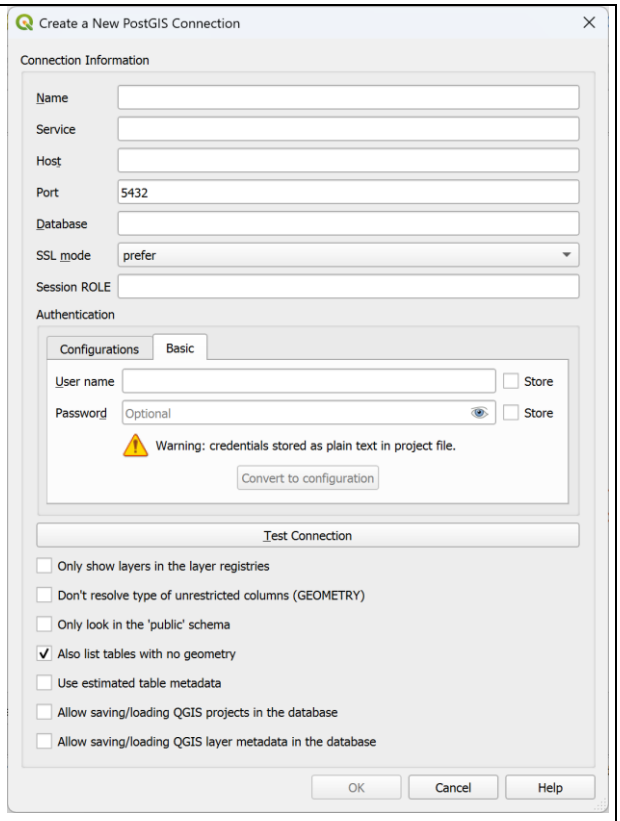


Figure 13 adding a new PostgreSQL connection: step 2

1.12.2 Verify that credentials are saved

Sometimes QGIS allows a PostgreSQL connection with unsaved credentials.
Follow these steps to confirm that your credentials are stored correctly:

1. Open the **Browser** panel
2. Right-click the PostgreSQL database connection you're going to use, and select **Edit Connection**

3. Open the **Basic** tab and verify that your **User name** and **Password** are entered and that **Store** is checked.
4. If the fields are empty, re-enter your credentials, check **Store**, and click **OK**.
5. To confirm that the credentials are saved, close and reopen the **Edit Connection** dialog. If the fields are blank again, repeat the process until the credentials appear after reopening.

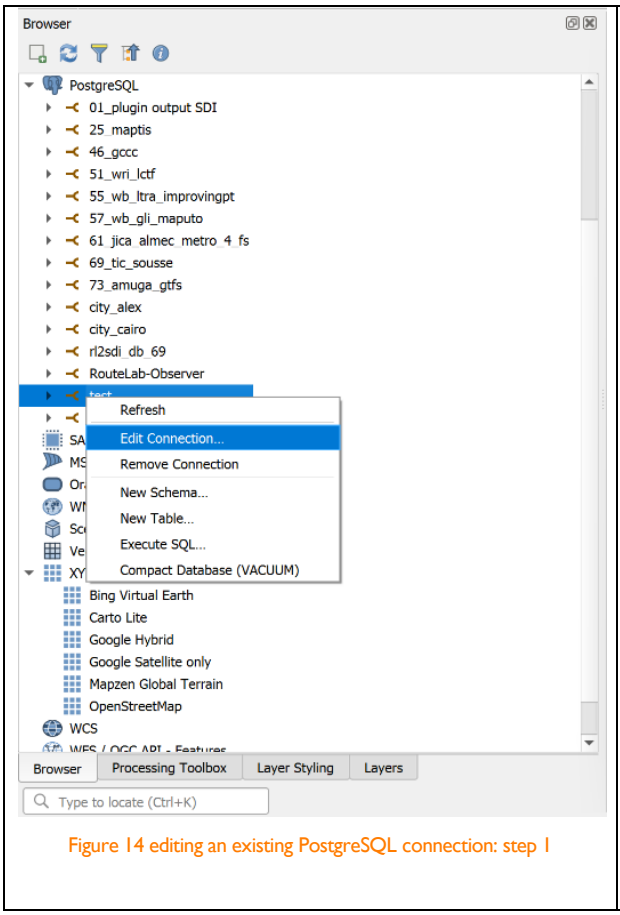


Figure 14 editing an existing PostgreSQL connection: step 1

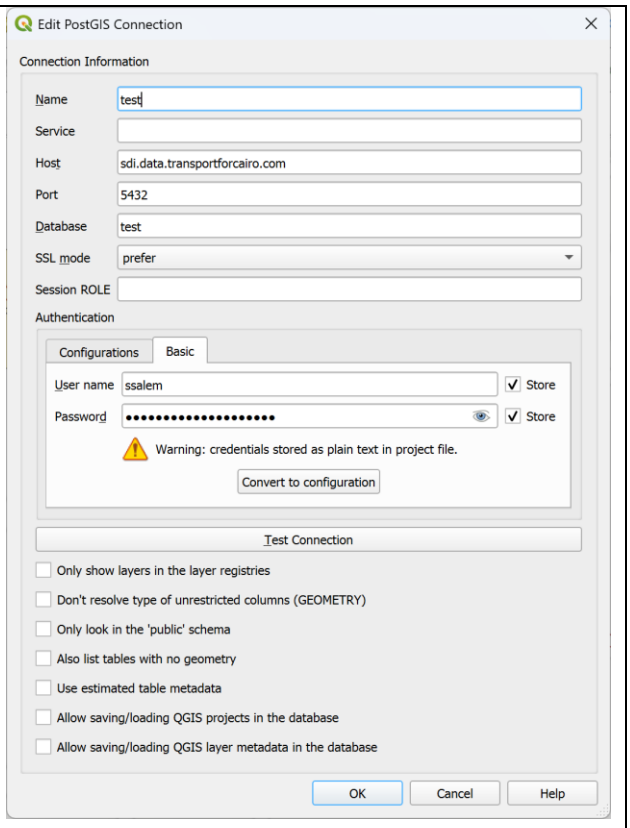


Figure 15 editing an existing PostgreSQL connection: step 2

1.13 RL2SDI Plugin – Generated PostgreSQL Database Schema

The RL2SDI plugin automatically creates two database schemas — **raw** and **transit** — as part of the data migration and standardization process.

Each schema contains a set of structured tables (and, in some cases, materialized views) that store survey, stop, trip, and route data in a format compatible with TfC's PostgreSQL database structure.

The following tables summarize the output schema of the RL2SDI plugin, including the **database name**, **schema name**, **table/view name**, **field name**, **data type**, and **object type**.

1.13.1 Schema: raw

Table 2 Raw schema outputs generated by the RL2SDI plugin

Table name	Field name	Data type	Object type
frequency_instances	agency	text	TABLE
	assignment_id	text	TABLE
	avg_headway_sec	double precision	TABLE
	canceled_at	timestamp with time zone	TABLE
	created_at	timestamp with time zone	TABLE
	deleted_at	text	TABLE
	destination	text	TABLE
	finished_at	timestamp with time zone	TABLE
	fr_code	text	TABLE
	fr_id	text	TABLE
	fr_name	text	TABLE
	geometry	geometry(Point,4326)	TABLE
	geometry_properties	text	TABLE
	gid	integer	TABLE
	id	text	TABLE
	interval	text	TABLE
	interval_end	text	TABLE
	interval_id	text	TABLE
	interval_start	text	TABLE
	notes	text	TABLE
	observation_duration	bigint	TABLE
	observations_count	bigint	TABLE
	origin	text	TABLE
	project_id	text	TABLE

	setting_id	text	TABLE
	status	text	TABLE
	survey	text	TABLE
	trip_id	text	TABLE
	updated_at	timestamp with time zone	TABLE
identification_instances	accepted_at	timestamp with time zone	TABLE
	accepted_by	text	TABLE
	agency	text	TABLE
	agency_id	text	TABLE
	bus_number	text	TABLE
	created_at	timestamp with time zone	TABLE
	deleted_at	text	TABLE
	destination	text	TABLE
	destination_name	text	TABLE
	destination_name_local	text	TABLE
	fr_code	text	TABLE
	fr_id	text	TABLE
	fr_name	text	TABLE
	geometry	geometry(Point,4326)	TABLE
	geometry_properties	text	TABLE
	gid	integer	TABLE
	id	text	TABLE
	logs	text	TABLE
	m_destination_name	text	TABLE
	m_destination_name_local	text	TABLE
	m_origin_name	text	TABLE
	m_origin_name_local	text	TABLE
	metadata	text	TABLE
	notes	text	TABLE
	op_from	text	TABLE
	op_to	text	TABLE
	origin	text	TABLE
	origin_name	text	TABLE

	origin_name_local	text	TABLE
	project_id	text	TABLE
	route_id	text	TABLE
	setting_id	text	TABLE
	status	text	TABLE
	trip_id	text	TABLE
	updated_at	timestamp with time zone	TABLE
	user_id	text	TABLE
onboard_instances	agency	text	TABLE
	arrived_at	timestamp with time zone	TABLE
	assignment_id	text	TABLE
	canceled_at	timestamp with time zone	TABLE
	created_at	timestamp with time zone	TABLE
	deleted_at	text	TABLE
	departed_at	timestamp with time zone	TABLE
	destination	text	TABLE
	finished_at	timestamp with time zone	TABLE
	fr_code	text	TABLE
	fr_id	text	TABLE
	fr_name	text	TABLE
	geometry	geometry(LineString,4326)	TABLE
	geometry_properties	text	TABLE
	gid	integer	TABLE
	id	text	TABLE
	interval	text	TABLE
	interval_end	text	TABLE
	interval_id	text	TABLE
	interval_start	text	TABLE
	matched_geometry	geometry(LineString,4326)	TABLE
	notes	text	TABLE
	onboarding_at	timestamp with time zone	TABLE
	origin	text	TABLE
	project_id	text	TABLE

	status	text	TABLE
	status_updated_at	timestamp with time zone	TABLE
	survey	text	TABLE
	trip_id	text	TABLE
	updated_at	timestamp with time zone	TABLE
	valid	boolean	TABLE
	validated_at	timestamp with time zone	TABLE
	validated_by	text	TABLE
	vehicle_id	bigint	TABLE
stops	alight	integer	TABLE
	alight_female	integer	TABLE
	alight_male	integer	TABLE
	board	integer	TABLE
	board_female	integer	TABLE
	board_male	integer	TABLE
	geom	geometry(Geometry,4326)	TABLE
	gid	serial/int	TABLE
	h3_index	text	TABLE
	name	text	TABLE
	observer_id	text	TABLE
	onboard_instance_observer_id	**text**	TABLE
	parent_onboard_instance_status	**text**	TABLE
	parent_onboard_instance_valid	**boolean**	TABLE
stops_created_at	**created_at**	**timestamp with time zone**	TABLE
	observer_id	**text**	TABLE
trackpoints	**geom**	**geometry(Geometry,4326)**	TABLE
	gid	**serial/int**	TABLE
	onboard_instance_id	**text**	TABLE
	onboard_instance_status	**text**	TABLE
	onboard_instance_valid	**boolean**	TABLE

	timestamp	**timestamp(0) with time zone**	TABLE

1.13.2 Schema: transit

Table 3 Transit schema outputs generated by the RL2SDI plugin

Table name	Field name	Data type	Object type
_stop_clusters	centroid	geometry	MATERIALIZED VIEW
	cluster_id	integer	MATERIALIZED VIEW
	mode_name	text	MATERIALIZED VIEW
	n_points	bigint	MATERIALIZED VIEW
agencies	agency_id	text	TABLE
	agency_name	text	TABLE
	agency_timezone	text	TABLE
	agency_url	text	TABLE
	common_name	text	TABLE
	gid	serial/int	TABLE
	has_serial	boolean	TABLE
	vehicle_id	integer	TABLE
intervals	active	boolean	TABLE
	end_time	time without time zone	TABLE
	gid	serial/int	TABLE
	name	character varying	TABLE
	observer_id	text	TABLE
	start_time	time without time zone	TABLE
od_stats	d_id	text	MATERIALIZED VIEW
	dist	double precision	MATERIALIZED VIEW
	duration	integer	MATERIALIZED VIEW
	geom	geometry(LineString,4326)	MATERIALIZED VIEW

	gid	bigint	MATERIALIZED VIEW
	interval_id	integer	MATERIALIZED VIEW
	interval_start	time without time zone	MATERIALIZED VIEW
	o_id	text	MATERIALIZED VIEW
	speed	double precision	MATERIALIZED VIEW
	vehicle_name	text	MATERIALIZED VIEW
stops	double	integer	TABLE
	geom	geometry(Point,4326)	TABLE
	gid	serial/int	TABLE
	location_type	integer	TABLE
	stop_desc	text	TABLE
	stop_id	text	TABLE
	stop_lat	double precision	TABLE
	stop_lon	double precision	TABLE
	stop_name	text	TABLE
stops_auto	cluster_id	integer	MATERIALIZED VIEW
	double	integer	MATERIALIZED VIEW
	geom	geometry	MATERIALIZED VIEW
	location_type	integer	MATERIALIZED VIEW
	stop_desc	text	MATERIALIZED VIEW
	stop_lat	double precision	MATERIALIZED VIEW
	stop_lon	double precision	MATERIALIZED VIEW
	stop_name	text	MATERIALIZED VIEW

	stop_type	text	MATERIALIZED VIEW
terminals	geom	geometry(Geometry,4326)	TABLE
	gid	serial/int	TABLE
	name	text	TABLE
	name_ar	text	TABLE
	observer_id	text	TABLE
trip_stops_sequence	distance	double precision	MATERIALIZED VIEW
	distance_frac	double precision	MATERIALIZED VIEW
	distance_from_prev	double precision	MATERIALIZED VIEW
	gid	bigint	MATERIALIZED VIEW
	observer_trip_id	text	MATERIALIZED VIEW
	stop_id	text	MATERIALIZED VIEW
	stop_name	text	MATERIALIZED VIEW
	stop_sequence	bigint	MATERIALIZED VIEW
	t_id	integer	MATERIALIZED VIEW
	vehicle_name	text	MATERIALIZED VIEW
trips	agency_id	integer	TABLE
	agency_serial	text	TABLE
	d_id	integer	TABLE
	direction_id	integer	TABLE
	fare	real	TABLE
	geom	geometry(LineString,4326)	TABLE
	gid	serial/int	TABLE
	o_id	integer	TABLE

	observer_id	text	TABLE
	observer_route_id	text	TABLE
	route_type	integer	TABLE
	service_id	text	TABLE
trips_intervals	gid	serial/int	TABLE
	headway_estimation_method	text	TABLE
	headway_secs	integer	TABLE
	interval_id	integer	TABLE
	trip_id	integer	TABLE
trips_view	agency_id	text	MATERIALIZED VIEW
	d_id	integer	MATERIALIZED VIEW
	destination	text	MATERIALIZED VIEW
	direction_id	integer	MATERIALIZED VIEW
	fare	real	MATERIALIZED VIEW
	geom	geometry(LineString,4326)	MATERIALIZED VIEW
	gid	integer	MATERIALIZED VIEW
	len_km	double precision	MATERIALIZED VIEW
	o_id	integer	MATERIALIZED VIEW
	observer_id	text	MATERIALIZED VIEW
	origin	text	MATERIALIZED VIEW
	passenger_capacity	integer	MATERIALIZED VIEW
	route_id	text	MATERIALIZED VIEW
	route_long	text	MATERIALIZED VIEW

	route_short	text	MATERIALIZED VIEW
	route_type	integer	MATERIALIZED VIEW
	service_id	text	MATERIALIZED VIEW
	trip_short	text	MATERIALIZED VIEW
	vehicle_name	character varying	MATERIALIZED VIEW
vehicles	gid	serial/int	TABLE
	name	character varying	TABLE
	passenger_capacity	integer	TABLE

1.14 GIS2GTFS Plugin – Required PostgreSQL Database Schema

The following table lists everything the GIS2GTFS plugin expects under the schema "transit": which tables/views must exist, what columns they need, and why. Use it as a checklist to prepare your database before running the plugin.

Table 4 GIS2GTFS Plugin's PostgreSQL database schema requirements

Table Name	Table Description	Column	Type	Notes / Description
transit.agencies	Operator/agency definitions; joined with vehicles and used to set pickup/dropoff behavior.	agency_id	text	Unique agency key; joins from trips_view.agency_id.
		agency_name	text	GTFS agency_name.
		agency_url	text (http/https)	GTFS agency_url.
		agency_timezone	text (IANA TZ)	GTFS agency_timezone (e.g., Africa/Cairo).
		vehicle_id	integer (FK)	FK → transit.vehicles.id.
		has_serial	boolean	When true, plugin sets continuous_pickup & continuous_drop_off = 1.
transit.vehicles	Vehicle types/properties referenced by agencies.	gid	serial / integer (PK)	Primary key; referenced by agencies.vehicle_id.
		name	text	Exported as vehicle_name; used in downstream joins.
transit.stops	Public transport stops with stable IDs and map locations (for stops.txt).	gid	serial / integer (PK)	Stable stop identifier.
		stop_name	text	Human-readable name.
		geom	geometry(Point, 4326)	Source for stop_lat/stop_lon; must be valid points in WGS84.
transit.trips_view	Trips view used for routes, trips, and shapes. Geometry must	gid	serial / integer (unique)	Stable numeric key; used by frequencies.trip_id.

	be non-null for exported rows.	observer_id	text	
		route_id	text	GTFS route reference.
		service_id	text	GTFS service reference.
		direction_id	smallint (0/1)	GTFS direction flag.
		destination	text	Mapped to GTFS trip_headsign.
		route_short	text	Mapped to GTFS route_short_name.
		route_long	text	Mapped to GTFS route_long_name.
		route_type	integer	GTFS route_type.
		agency_id	text (FK)	Joins to agencies.agency_id.
		geom	geometry(LineString, 4326)	Used to generate shapes.txt (point sequences).
transit.intervals	Time windows used across stop_times and frequencies (e.g., morning peak).	gid	serial / integer (PK)	Primary key; joins from frequencies and od_stats.
		start_time	Time (HH:MM:SS)	Interval start (time of day).
		end_time	Time (HH:MM:SS)	Interval end (time of day).
transit.trip_stops_sequence	Ordered stop lists per trip; backbone of stop_times.txt.	gid	serial / integer	Ignored by builder if present.
		Observer_trip_id	text	
		stop_id	text (FK)	FK → stops.stop_id.
		stop_sequence	integer (1-based)	Visit order of stops along the trip.
transit.od_stats	Observed/estimated travel times between stop pairs per interval (used to compute arrival/departure).	o_id	text (FK)	Origin stop_id; must exist in transit.stops.
		d_id	text (FK)	Destination stop_id; must exist in transit.stops.
		interval_id	integer (FK)	FK → intervals.gid.
		interval_start	time or timestamp	Start time associated with the

				measurement window.
		duration	integer (seconds) or numeric	Travel time from o_id to d_id.
		vehicle_name	text	Joined from vehicles via agencies.
transit.trips_intervals	Headway values for trips at given time intervals (e.g. trip from x to y departs every 300 seconds during the morning peak interval)	trip_id	integer	Trip reference aligns with trips_view.gid
		interval_id	integer (FK)	FK → intervals.gid.
		headway_secs	integer	Freeform; usage varies.

1.15 Vehicle and Passenger Flow Plugin – Required PostgreSQL Database Schema

Table 5 Vehicle and Passenger Flow Plugin's PostgreSQL database schema requirements

schema	table	field	type	notes
raw	onboard_instances	id	text	Primary identifier of onboard survey instance.
		trip_id	text	Survey 'trip' grouping; used to relate stops to an instance.
		status	text	Expected values like 'finished' used to filter valid runs.
		valid	boolean	Used to exclude invalid runs.
		departed_at	timestamptz	Start timestamp for time-binning and headway logic.
		geometry	geometry(LineString,4326)	GPS path of the onboard run (WGS84).
		matched_geometry	geometry(LineString,4326)	Path matched to network if available.
	stops	board	integer	Boardings counted at this observed stop.
		alight	integer	Alightings counted at this observed stop.
		observer_id	text	Join key to stops_created_at for timestamp.
		geom	geometry(Geometry,4326)	Observed stop location (WGS84).
	stops_created_at	observer_id	text	Join to raw.stops.observer_id.
		created_at	timestamptz	Timestamp of stop record creation.