DATABASE PROJECT

Members of the group :

BOULFRAD Imam
DJEROUA Mustapha Hacène

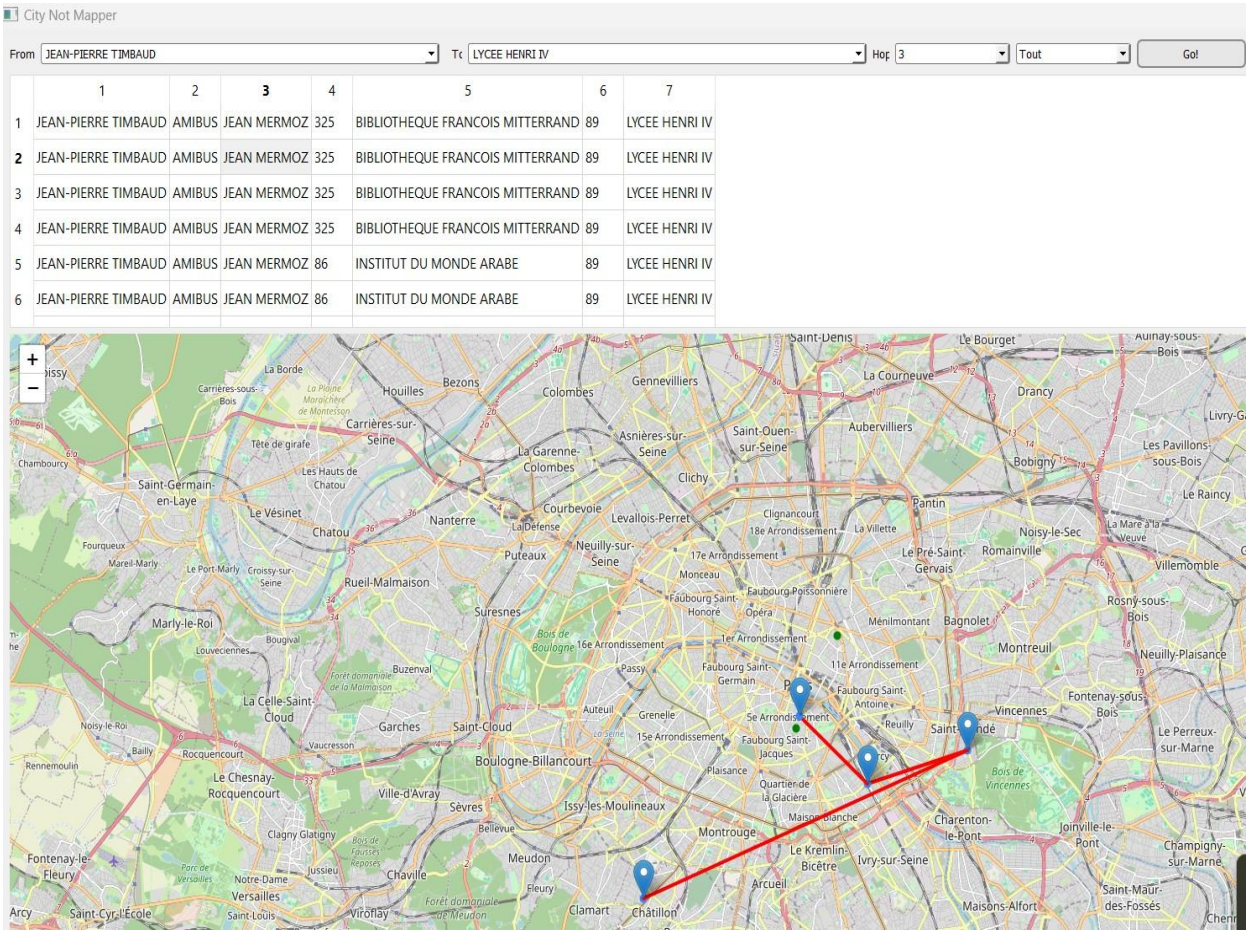
Under the supervision of :

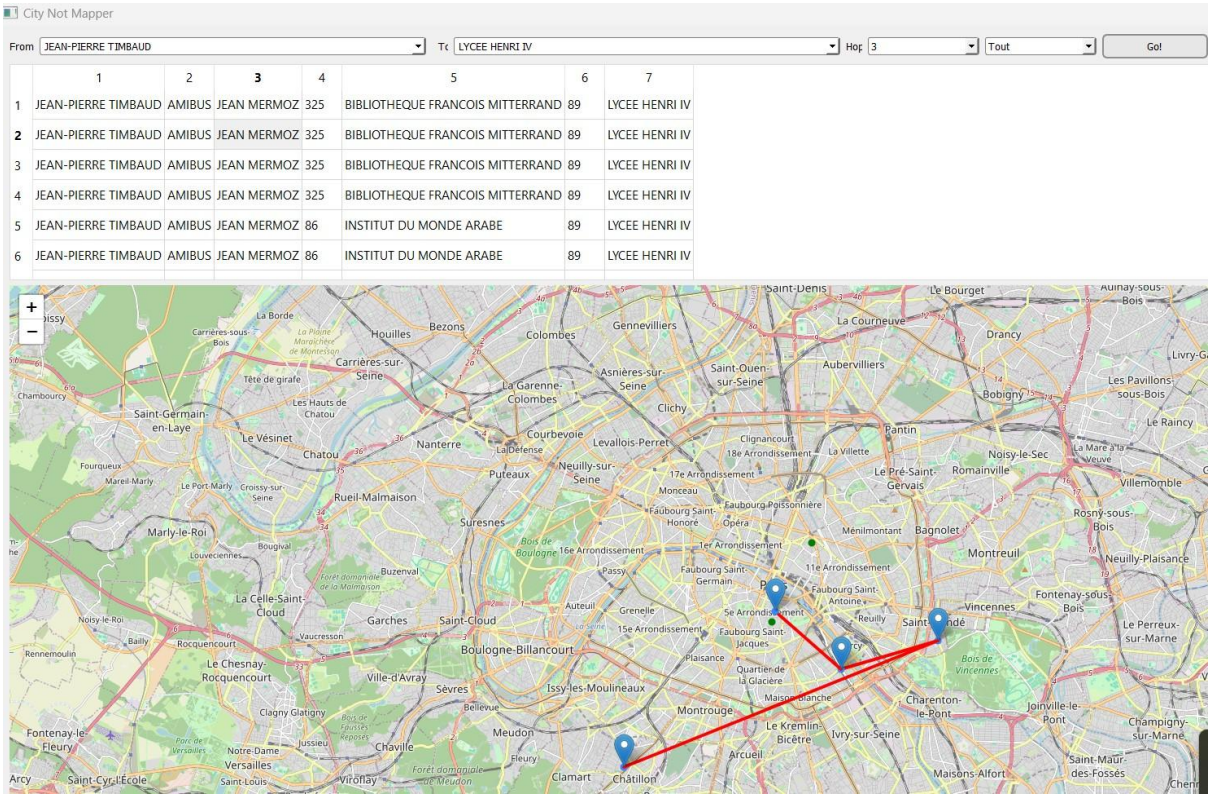
Pr Nabil H.Mustafa

PART 1 : The functionalities:

In our City Mapper type application you can do the following :

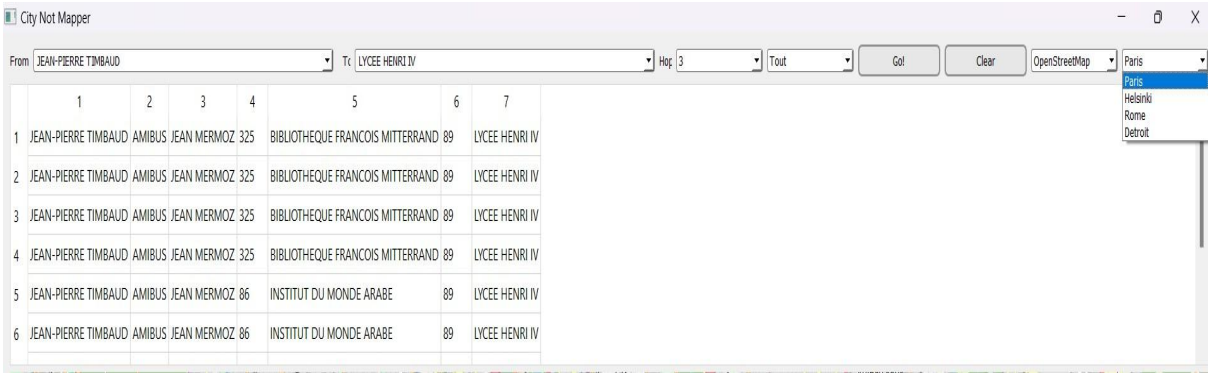
1- Find a list of the possible routes that one might take to go from point A to B, and once they click on the table that shows them all of these possible routes segments will be drawn to make it easier to visualize. You can also choose the number of hops (which basically means how many times you want to commute to arrive at your destination)





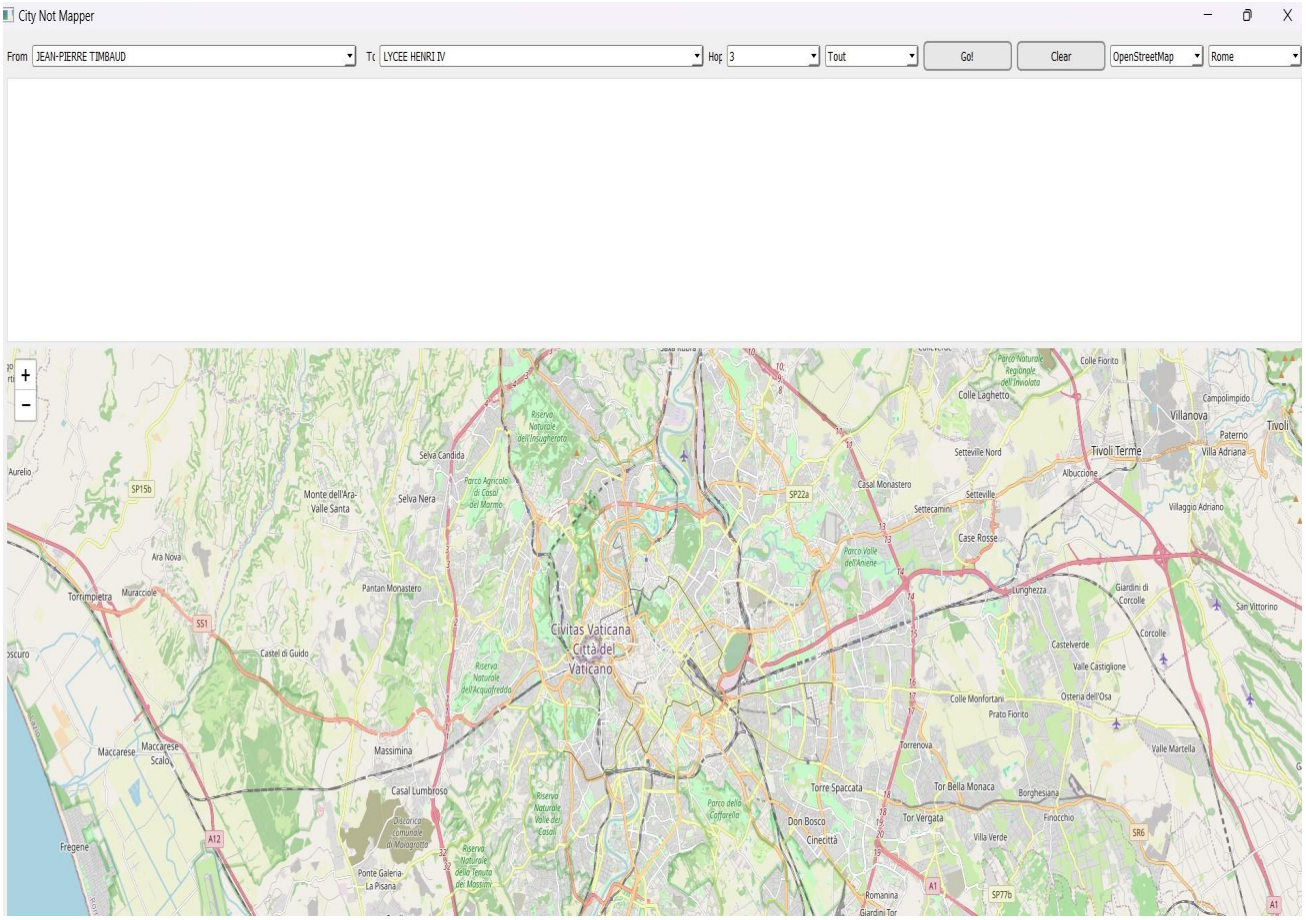
2- Choose the means of transport that you want to use (Tout = anything , Bus, Metro, Train or Tram are the ones available)

3- You can also switch between different cities (each with its own database) just by choosing a city from the ones available in the dropdown box :



	1	2	3	4	5	6	7
1	JEAN-PIERRE TIMBAUD	AMIBUS	JEAN MERMOZ	325	BIBLIOTHEQUE FRANCOIS MITTERRAND	89	LYCEE HENRI IV
2	JEAN-PIERRE TIMBAUD	AMIBUS	JEAN MERMOZ	325	BIBLIOTHEQUE FRANCOIS MITTERRAND	89	LYCEE HENRI IV
3	JEAN-PIERRE TIMBAUD	AMIBUS	JEAN MERMOZ	325	BIBLIOTHEQUE FRANCOIS MITTERRAND	89	LYCEE HENRI IV
4	JEAN-PIERRE TIMBAUD	AMIBUS	JEAN MERMOZ	325	BIBLIOTHEQUE FRANCOIS MITTERRAND	89	LYCEE HENRI IV
5	JEAN-PIERRE TIMBAUD	AMIBUS	JEAN MERMOZ	86	INSTITUT DU MONDE ARABE	89	LYCEE HENRI IV
6	JEAN-PIERRE TIMBAUD	AMIBUS	JEAN MERMOZ	86	INSTITUT DU MONDE ARABE	89	LYCEE HENRI IV

- Note that you'll have to wait for a few seconds for the data to be loaded, and then press on the "GO" button for the map to show up :



PART 2 : The tables used :

Because we have implemented the “different cities functionality”, we needed different tables in order to get the routes proper to each city.
So the total number of tables used are 4 :
-stop_route_info_paris.
-stop_route_info_helsinki.
-stop_route_info_rome.
-stop_route_info_detroit.
- I will provide in the “sql queries section” the ones used in order to get the following tables.
-A file named config.JSON has also been created to store the different types of information needed to access each different set of data separately (like the name of the database,which tables should be used.. etc)
-Here are snippets of each one (taken inside of the PGAdmin 4 program) :

	stop_id text	stop_stop_i text	stop_name text	route_type bigint	route_name text	stop_coordinates json	coordinates_text text
1	4867	7942	SEVRES - BABYLONE	3	39	{"type":"Point","coordinates":[2.325910342,48.85109205...	2.325910342,48.85109205
2	4867	7942	SEVRES - BABYLONE	3	70	{"type":"Point","coordinates":[2.325910342,48.85109205...	2.325910342,48.85109205
3	4868	7943	BAC - SAINT-PLACIDE	3	39	{"type":"Point","coordinates":[2.323649903,48.85011218...	2.323649903,48.85011218
4	4868	7943	BAC - SAINT-PLACIDE	3	70	{"type":"Point","coordinates":[2.323649903,48.85011218...	2.323649903,48.85011218
5	4869	7944	VANEAU - SAINT-ROMAIN	3	39	{"type":"Point","coordinates":[2.319905478,48.84821533...	2.319905478,48.84821533
6	4869	7944	VANEAU - SAINT-ROMAIN	3	70	{"type":"Point","coordinates":[2.319905478,48.84821533...	2.319905478,48.84821533

coordinates_point point	coordinates_text2 text	coordinates_point2 point	lat double precision	lon double precision
(2.325910342,48.851092056)	[null]	[null]	48.851092056	2.325910342
(2.325910342,48.851092056)	[null]	[null]	48.851092056	2.325910342
(2.323649903,48.850112185)	[null]	[null]	48.850112185	2.323649903
(2.323649903,48.850112185)	[null]	[null]	48.850112185	2.323649903
(2.319905478,48.848215338)	[null]	[null]	48.848215338	2.319905478
(2.319905478,48.848215338)	[null]	[null]	48.848215338	2.319905478

The Paris table

	stop_id text	stop_stop_i text	stop_name text	route_type bigint	route_name text	stop_coordinates json	lat numeric	lon numeric	coordinates_text text
1	239	280	Hakaniemi	3	731	{"type":"Point","coordinates":[24.95073,60.17932]}	60.17932	24.95073	24.95073,60.1793
2	1816	1895	Terrintie	3	76A	{"type":"Point","coordinates":[25.04124,60.27618]}	60.27618	25.04124	25.04124,60.2761
3	2929	3027	Logen	3	217	{"type":"Point","coordinates":[24.74963,60.22465]}	60.22465	24.74963	24.74963,60.2246
4	3104	3204	Riistatie	3	565	{"type":"Point","coordinates":[24.71583,60.23013]}	60.23013	24.71583	24.71583,60.2301
5	3994	4097	Espoo	2	U	{"type":"Point","coordinates":[24.656446,60.20509...	60.205095	24.656446	24.656446,60.20509
6	4809	4928	Louhelantie	3	311A	{"type":"Point","coordinates":[24.85892,60.27658]}	60.27658	24.85892	24.85892,60.2765
7	4854	4989	Hirsikuja	3	335B	{"type":"Point","coordinates":[24.78564,60.28542]}	60.28542	24.78564	24.78564,60.2854

The Helsinki table

	stop_id text	stop_stop_i text	stop_name text	route_type bigint	route_name text	stop_coordinates json	lat numeric	lon numeric	coordinates_text text	coord point
1	1750	1751	PAUL & MINOCK	3	046	{"type":"Point","coordinates":[-83.231489,42.33569]}	42.33569	-83.231489	-83.231489,42.3356	(-83.
2	5631	5640	E. Grand Blvd and Jefferson	3	012	{"type":"Point","coordinates":[-83.002176,42.34734...	42.347344	-83.002176	-83.002176,42.34734	(-83.
3	5632	5641	Maryland and Jefferson	3	009	{"type":"Point","coordinates":[-82.937911,42.37579...	42.375798	-82.937911	-82.937911,42.37579	(-82.
4	5632	5641	Maryland and Jefferson	3	025	{"type":"Point","coordinates":[-82.937911,42.37579...	42.375798	-82.937911	-82.937911,42.37579	(-82.
5	5633	5642	SOMERSET MALL	3	498	{"type":"Point","coordinates":[-83.181507,42.56139...	42.561396	-83.181507	-83.181507,42.56139	(-83.
6	5638	5647	Warren & Telegraph	3	014	{"type":"Point","coordinates":[-83.272974,42.34123...	42.341236	-83.272974	-83.272974,42.34123	(-83.

The Detroit table

stop_stop_i text	stop_name text	route_type bigint	route_name text	stop_coordinates json	lat numeric	lon numeric	coordinates_text text	coordinates_ point
4858	IGEA	3	990	{"type":"Point","coordinates":[12.440385,41.93423...	41.934234	12.440385	12.440385,41.93423	(12.440385,
7713	ANAGNINA (MA)	3	500	{"type":"Point","coordinates":[12.5873,41.842361]}	41.842361	12.5873	12.5873,41.84236	(12.5873,41
8514	CIPRO	1	MEA	{"type":"Point","coordinates":[12.447356,41.90758...	41.907586	12.447356	12.447356,41.90758	(12.447356,
8615	GARDENIE	1	MEC	{"type":"Point","coordinates":[12.562108,41.88611...	41.886115	12.562108	12.562108,41.88611	(12.562108,
8616	TEANO	1	MEC	{"type":"Point","coordinates":[12.551361,41.88948...	41.889483	12.551361	12.551361,41.88948	(12.551361,
8617	MALATESTA	1	MEC	{"type":"Point","coordinates":[12.540216,41.88732...	41.887325	12.540216	12.540216,41.88732	(12.540216,

The Rome table

Description :

-The structure of all the tables is similar except for the Paris one, the reason is different is because it was the first city we started with so we just did some experimentation on different ways that we could be able to find the route (that's why it contains more columns)

-If we want to talk about each attribute then we can say the following :

-stop_stop_I : is the id of the stop that we got from the stops (table)

-stop_name : is the stop name of that id
-route_type : is the type of transport that passes by that stop (metro/tram.. etc)
-route_name : is the name of the bus/train etc…..
-stop_coordinates : are the coordinates of the stop in JSON form
-coordinates_text : are just the stop coordinates in text form (made our lives easier for the table_click)
-coordinates_point : are just the stop coordinates in point form
- lan and lon : are just the latitude and longitude of the specific stop

- the reason we put our emphasis on the coordinates is because we have found that they differ from the network_nodes table (lat and lon columns) and the stops table (coordinates in geometry form column).

PART 3 : Dependencies

From the tables i've provided we can see that all of the attributes depend on either the stop_id or the stop_stop_id. Because provided that we got the vast majority of information from the stops and routes table, in these tables we can find that :

- The stop_stop_id and stop_id are the primary key of this table (and thus all the tables)
- knowing the primary key lets you know : name of the stop, route_name/type and it's coordinates (and thus knowing every single type of coordinates whether it being text or point format)
- the coordinates_point column lets you extract coordinates_point and coordinates_text (they depend on it)

PART 4 : BCNF or NOT ?

-From the tables we can clearly see that they are not in BCNF form because we have a partial dependency, since knowing the stop_coordinates lets you determine the coordinates_text and coordinates_point.
-they are also not in 3NF form since knowing the lat and lon will let you determine the coordinates_text and from that the coordinates_point.

PART 5 : The SQL queries :

I will provide the sql queries used to run the hops 1,2,3 and also the sql queries used to transform the date from GeoJSON to JSON format, and also the ones used to get the tables:

```
SELECT DISTINCT A.coordinates_text, A.stop_name, A.route_name, B.coordinates_text, B.stop_name
FROM {table_name} as A, {table_name} as B
WHERE A.stop_name = $${_fromstation}$$ AND B.stop_name = $${_tostation}$$ AND A.route_name = B.route_name
AND A.route_type IN ($${_transport}$$,$${_transport1}$$,$${_transport2}$$,$${_transport3}$$)
```

Hops 1

```
SELECT DISTINCT A.coordinates_text, A.stop_name, A.route_name, B.coordinates_text, B.stop_name, C.route_name, D.coordinates_text, D.stop_name
FROM {table_name} as A, {table_name} as B, {table_name} as C, {table_name} as D
WHERE A.stop_name = $${_fromstation}$$ AND D.stop_name = $${_tostation}$$ AND A.route_name = B.route_name
AND B.stop_name = C.stop_name AND C.route_name = D.route_name
AND A.route_type IN ($${_transport}$$,$${_transport1}$$,$${_transport2}$$,$${_transport3}$$) AND A.route_type = C.route_type
AND A.route_name <> C.route_name AND A.stop_name <> B.stop_name AND B.stop_name <> D.stop_name LIMIT 10
```

Hops 2

Query | Query History

```
SELECT DISTINCT A.coordinates_text, A.stop_name, A.route_name, B2.coordinates_text,
B2.stop_name, B2.route_name, C2.coordinates_text, C2.stop_name, C2.route_name, D.coordinates_text, D.stop_name
FROM {table_name} as A, {table_name} as B1, {table_name} as B2, {table_name} as C1, {table_name} as C2, {table_name} as D
WHERE A.stop_name = $${_fromstation}$$ AND A.route_name = B1.route_name AND B1.stop_name = B2.stop_name
AND B2.route_name = C1.route_name AND C1.stop_name = C2.stop_name
AND C2.route_name = D.route_name AND D.stop_name = $${_tostation}$$
AND A.route_type IN ($${_transport}$$,$${_transport1}$$,$${_transport2}$$,$${_transport3}$$)
AND A.route_type = C1.route_type AND A.route_type = D.route_type AND A.route_name <> B2.route_name
AND B2.route_name <> C2.route_name AND A.route_name <> C2.route_name AND A.stop_name <> B1.stop_name
AND B2.stop_name <> C1.stop_name AND C2.stop_name <> D.stop_name LIMIT 10
```

Hops 3

Query | Query History

```
ALTER TABLE 'table_name' ADD COLUMN geojson_geometry JSON;

UPDATE  SET geojson_geometry = ST_AsGeoJSON(geometry_column)::JSON;

ALTER TABLE 'table_name' DROP COLUMN geometry_column;

ALTER TABLE 'table_name' RENAME COLUMN geojson_geometry TO geometry_column;
```

The query used to transform from GeoJSON to JSON (that way it’s much easier to manipulate)

Query Query History

```sql
CREATE TABLE stop_route_info AS
SELECT
    s.id AS stop_id,
    s."stop_I" AS stop_stop_I,
    s.name AS stop_name,
    r.route_type,
    r.route_name,
    s.geometry AS stop_coordinates
FROM
    stops s
JOIN
    routes r ON ST_Intersects(s.geometry_2, r.geometry_2);
```

Query Query History

```sql

ALTER TABLE stop_route_info
ADD COLUMN lat numeric,
ADD COLUMN lon numeric;

UPDATE stop_route_info
SET
    lat = (stop_coordinates->>'lat')::numeric,
    lon = (stop_coordinates->>'lon')::numeric;

```

```sql

ALTER TABLE stop_route_info
ADD COLUMN coordinates_text text;

UPDATE stop_route_info
SET coordinates_text = '[' || (stop_coordinates->'coordinates'->>0) || ', ' || (stop_coordinates->'coordinates'->>1) || ']';

ALTER TABLE stop_route_info
ADD COLUMN coordinates_point point;

UPDATE stop_route_info
SET coordinates_point = point(
    CAST(REGEXP_REPLACE(SPLIT_PART(coordinates_text, ', ', 1), '[^0-9.-]', '', 'g') AS double precision),
    CAST(REGEXP_REPLACE(SPLIT_PART(coordinates_text, ', ', 2), '[^0-9.-]', '', 'g') AS double precision));
```

```
UPDATE stop_route_info
SET coordinates_text =
    CONCAT(
        SUBSTRING(coordinates_text FROM 2 FOR POSITION(',' IN coordinates_text) - 2),
        SUBSTRING(coordinates_text FROM POSITION(',' IN coordinates_text) + 1 FOR LENGTH(coordinates_text) - POSITION(',' IN coordinates_text) - 2)
    )
WHERE coordinates_text LIKE '[%]';

UPDATE stop_route_info
SET coordinates_text = REPLACE(coordinates_text, ' ', ',');

UPDATE stop_route_info
SET lat = ST_Y(coordinates_point::geometry),
    lon = ST_X(coordinates_point::geometry);
```

Queries used to create the stop_route_info tables

PART 6 : The difficulties :

We faced a lot of difficulties doing this project, and to mention a few :

-BOULFRAD had a lot of issues installing python3, psql and PGAdmin4 on his personal computer since we have started working on this project using Windows systems.

-We haven't agreed at first on whether we should be creating new tables (like stop_route_info) or just merge the ones we have without needing to create new ones (doing a lot of subqueries).

-We've faced some issues at first when converting from csv/geoJSON to sql so we had to use another library called sqlalchemy and a PSQLl extension called postGIS to put the geometry type data in our tables

- We've tried to implement the time it takes for each route, but due to the complexity of our idea and the time constraint we couldn't manage to implement it.

- DJEROUA tried to implement the log system (where the queries are stored inside of a file and later can be retrieved for faster use the next time a user opens the app) but he encountered many issues when retrieving the queries from a queries.JSON file that follows a specific structure (and mostly due to time constraints).

- Managing between the exams and working on the project and it being during the holidays made it difficult to organize and work efficiently.

- The fact that the latitude and longitude of the tables network_nodes and stops being different put us to a stop and made us think about other alternatives for the hops 1,2,3 wich took several hours to overcome .

PART 7 : The contributions:

- Hops 1,2,3 : BOULFRAD + DJEROUA
- Conversion from csv/geoJSON to sql : DJEROUA
- Different types of transport : BOULFRAD
- Different cities : DJEROUA
- Writing the report : DJEROUA
- Adapting the professor’s code to match our ideas : DJEROUA + BOULFRAD