

FIT5137 S2 2023 Assignment 4: PTV Answer Sheet (Weight = 30%)**Mengzhen Li, Yuxiang Zou, Zeyu Li****Due date: Wednesday, 25 October 2023, 11:55pm**

Version: 2.0 – 25/09/2023

Assignment Task list

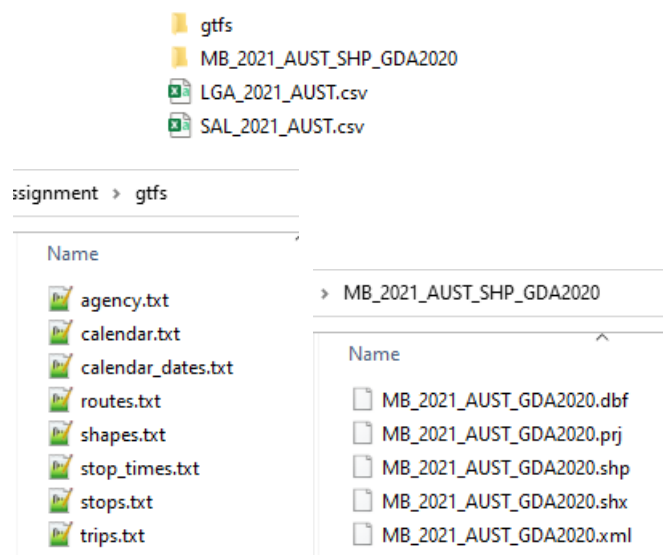
Your assignment consists of several parts. Always read the instruction one by one. Do not move to the step without completing the previous step:

- Task 1: Data Restoration - Restore the data to the database. Monitor the success indicator to ensure successful restoration of the data.
- Task 2: Data Preprocessing - Perform necessary structure maintenance and create result tables for further processing.
- Task 3: Data Analytics - Develop SQL queries to analyze the data and evaluate performance.
- Task 4: Data Visualization - Create visualizations to present the results of the data analytics.

For simplicity, **all the data required for this assignment is readily available in the PostGIS Docker container.** You can access these datasets within the container by navigating to the **/data/adata** folder. If you don't know how to do it, refer to the labs 10 activities. As a data analyst, it is your responsibility to understand and explore these publicly available data.

```
root@db94d38b7162:/home/student# ls /data/adata
gtfs  LGA_2021_AUST.csv  MB_2021_AUST_SHP_GDA2020  SAL_2021_AUST.csv
```

Verify your data before the restoration process.



As a data analyst, it is your responsibility to understand and explore these publicly available data.

Do not edit or remove any content on the answer sheet, including the questions. Please write your answers in the answer boxes provided. If necessary, you can adjust the size of the answer box to fit your answer.

Task 1: Data Restoration

Before you can start the data analytic processes, the first thing you have to do is to restore the external data to your database. Make sure you prepare a destination schema to restore your data. The destination schema for your assignment is “ptv”.

Note:

- Before initiating the data restoration process, **it is essential to thoroughly explore the dataset**. This exploration involves identifying appropriate data types, determining field lengths, and making other relevant considerations that will inform the creation of the table structure.
- **Ensure that you restore the data into the PTV schema using regular (local) tables. Do not utilize foreign tables, as the data must be stored directly within the PostgreSQL database – updated 27/09/2023**
- Make sure all 8 GTFS tables are restored successfully
- Index or constraints can be added to the table after the data has been restored completely
(Note: This index or constraint requirement is **NOT** mandatory in this Task 1 – updated 25/09/2023)
- **No data cleaning required for this assignment**
- For more information, see the FAQ for Assignment 4.

1.1 PTV schema

Write the SQL script to create the destination schema named “ptv”.

```
create schema ptv;
```

1.2 GTFS

Write the SQL script to restore **ALL** tables in GTFS files.

```
drop table ptv.agency;
CREATE TABLE ptv.agency
(
  agency_id      NUMERIC(3),
  agency_name    VARCHAR(15),
  agency_url     VARCHAR(50),
  agency_timezone VARCHAR(30),
  agency_lang    CHAR(2)
);

copy ptv.agency from '/data/adata/gtfs/agency.txt'
delimiter ','
csv header;

drop table calendar;
create table ptv.calendar
(
  service_id varchar(20),
  monday numeric (1),
  tuesday numeric (1),
  wednesday numeric (1),
  thursday numeric (1),
  friday numeric (1),
  saturday numeric (1),
  sunday numeric (1),

```

```
        start_date varchar(10),
        end_date varchar(10)
    )

copy ptv.calendar from '/data/adata/gtfs/calendar.txt'
delimiter ','
csv header;

drop table ptv.

create table ptv.calendar_dates
(
    service_id varchar(20),
    date varchar(10),
    exception_type numeric(1)
);

copy ptv.calendar_dates from '/data/adata/gtfs/calendar_dates.txt'
delimiter ','
csv header;

drop table ptv.routes;
create table ptv.routes
(
    route_id varchar(15),
    agency_id numeric(3),
    route_short_name varchar(20),
    route_long_name varchar(100),
    route_type numeric(3),
    route_color char(10),
    route_text_color char(10)
);

copy ptv.routes from '/data/adata/gtfs/routes.txt'
delimiter ','
csv header;

create table ptv.shapes
(
    shape_id varchar(20),
    shape_pt_lat numeric(15,13),
    shape_pt_long numeric(16,13),
    shape_pt_sequence numeric(5),
    shape_dist_traveled numeric(10,2)
);

copy ptv.shapes from '/data/adata/gtfs/shapes.txt'
delimiter ','
csv header;

drop table ptv.stop_times;
create table ptv.stop_times
(
    trip_id varchar(30),
    arrival_time char(10),
    departure_time char(10),
    stop_id numeric(10),
    stop_sequence numeric(10),
    stop_headsign varchar(10),
```

```

pickup_type numeric(1),
drop_off_type numeric(1),
shape_dist_traveled varchar(12)
)

copy ptv.stop_times from '/data/adata/gtfs/stop_times.txt'
delimiter ','
csv header;

create table ptv.stops
(
    stop_id numeric(5),
    stop_name varchar(100),
    stop_lat numeric(15,13),
    stop_lon numeric(16,13)
);

copy ptv.stops from '/data/adata/gtfs/stops.txt'
delimiter ','
csv header;

create table ptv.trips
(
    route_id varchar(20),
    service_id varchar(20),
    trip_id varchar(25),
    shape_id varchar(20),
    trip_headsign varchar(50),
    direction_id numeric(1)
);

copy ptv.trips from '/data/adata/gtfs/trips.txt'
delimiter ','
csv header;

```

1.3 ABS Mesh Blocks

Scripts to restore the Mesh Blocks files by using correct dataset file. Restore the file using ogr2ogr into table “mb2021”

```

ogr2ogr PG:"dbname=gisdb user=postgres"
"/data/adata/MB_2021_AUST_SHP_GDA2020/MB_2021_AUST_GDA2020.shp" -nln ptv.mb2021 -
overwrite -nlt MULTIPOLYGON

```

1.4 ABS Allocation Files

Write the SQL script to restore the LGA2021 Allocation file.

```

drop table ptv.lga2021;
create table ptv.lga2021
(
    mb_code_2021 char(11),
    lga_code_2021 char(5),
    lga_name_2021 char(60),

```

```
state_code_2021 char(1),
state_name_2021 varchar(50),
aus_code_2021 char(3),
aus_name_2021 varchar(20),
area_albers_sqkm numeric(10,4),
asgs_loci_uri_2021 varchar(60)
);

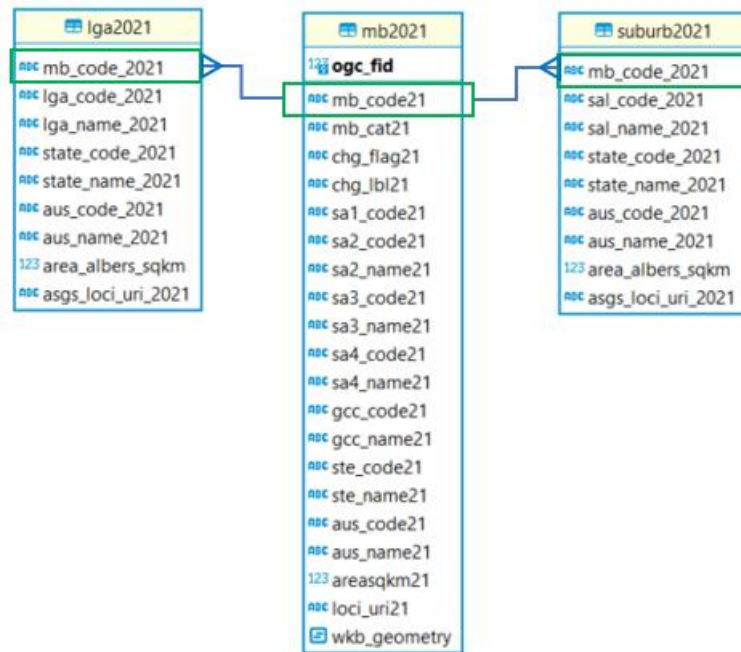
copy ptv.lga2021 from '/data/adata/LGA_2021_AUST.csv'
delimiter ','
csv header;
```

Write the SQL script to restore the SAL 2021 Allocation file for suburb2021.

```
create table ptv.sal2021
(
    mb_code_2021 char(11),
    sal_code_2021 char(5),
    sal_name_2021 char(60),
    state_code_2021 char(1),
    state_name_2021 varchar(50),
    aus_code_2021 char(3),
    aus_name_2021 varchar(20),
    area_albers_sqkm numeric(10,4),
    asgs_loci_uri_2021 varchar(60)
);

copy ptv.sal2021 from '/data/adata/SAL_2021_AUST.csv'
delimiter ','
csv header;
```

The allocation tables have 1-N relationship with the mb2021 in mb_code21 – mb_code_2021. Although there are no PK – FK defined in the table, the relationship rule still apply.



1.5 Data Verification

Verify your restoration by running this script. 1. Do not modify the verification script. 2. Make sure that the table name is consistent with the table we provided.

Output: Attach a screenshot of the results to include all tables you have restored in Task 1.

```
with tbl as
(select table_schema, TABLE_NAME
 from information_schema.tables
 where table_schema in ('ptv'))
select table_schema, TABLE_NAME,
(xpath('/row/c/text()', query_to_xml(format('select count(*) as c from %I.%I', table_schema, TABLE_NAME),
FALSE, TRUE, '')))[1]::text::int AS rows_n
from tbl
order by table_name;
```

Screenshot:

The screenshot shows a SQL query execution interface. The query is the same as the one provided in the previous block. The results are displayed in a table with three columns: table_schema, table_name, and rows_n. The results are sorted by table_name.

	table_schema	table_name	rows_n
1	ptv	agency	10
2	ptv	calendar	380
3	ptv	calendar_dates	15
4	ptv	lga2021	368,286
5	ptv	mb2021	368,286
6	ptv	routes	3,300
7	ptv	sal2021	368,286
8	ptv	shapes	9,757,418
9	ptv	stop_times	8,122,810
10	ptv	stops	27,821
11	ptv	trips	236,613

Task 2: Data Preprocessing

2.1 Filter Melbourne Metropolitan area

The mb2021 table contains whole mesh blocks in Australia. To minimise the query cost, we want to ensure that you only use the mesh blocks in Melbourne Metropolitan. The Melbourne Metropolitan's mesh blocks can be identified from the gcc_name21. If the column contains "Greater Melbourne", this mesh block is located in Melbourne Metropolitan.

Create a table named "mb2021_mel" that contains ONLY the mesh blocks in Melbourne Metropolitan.

	mbc_sa1_code21	mbc_sa2_code21	mbc_sa2_name21	mbc_sa3_code21	mbc_sa3_name21	mbc_sa4_code21	mbc_sa4_name21	mbc_gcc_code21	mbc_gcc_name21	mbc_ste_code21	mbc_ste_name21	mbc
22	10901117322	109011173	Albury - North	10901	Albury	109	Murray	1RNSW	Rest of NSW	1	New South Wales	AU
23	10901117322	109011173	Albury - North	10901	Albury	109	Murray	1RNSW	Rest of NSW	1	New South Wales	AU
24	21401137143	214011371	Frankston	21401	Frankston	214	Mornington Peninsula	2GMEL	Greater Melbourne	2	Victoria	AU
25	10901117325	109011173	Albury - North	10901	Albury	109	Murray	1RNSW	Rest of NSW	1	New South Wales	AU
26	10901117301	109011173	Albury - North	10901	Albury	109	Murray	1RNSW	Rest of NSW	1	New South Wales	AU
27	10901117323	109011173	Albury - North	10901	Albury	109	Murray	1RNSW	Rest of NSW	1	New South Wales	AU

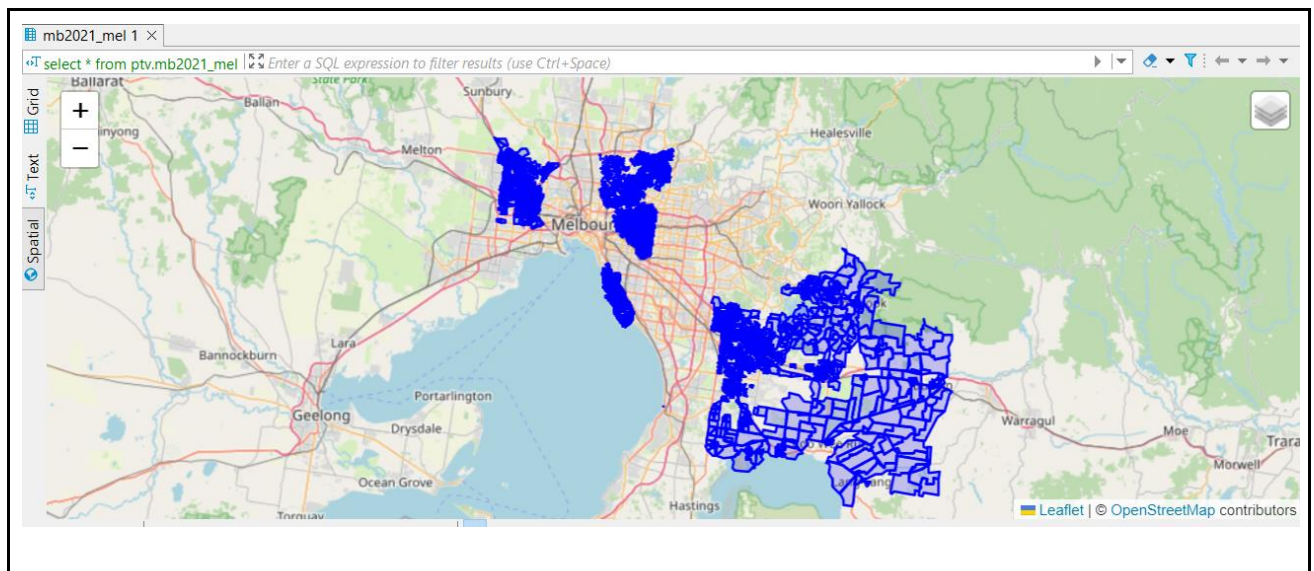
Write the SQL script to do this.

```
CREATE TABLE IF NOT EXISTS ptv.mb2021_mel AS
SELECT * FROM ptv.mb2021
WHERE gcc_name21 ILIKE '%greater melbourne%';

SELECT * FROM ptv.mb2021_mel;
```

Attach a screenshot of the Spatial Map results.

Screenshot:



2.2 Melbourne Metropolitan Boundary

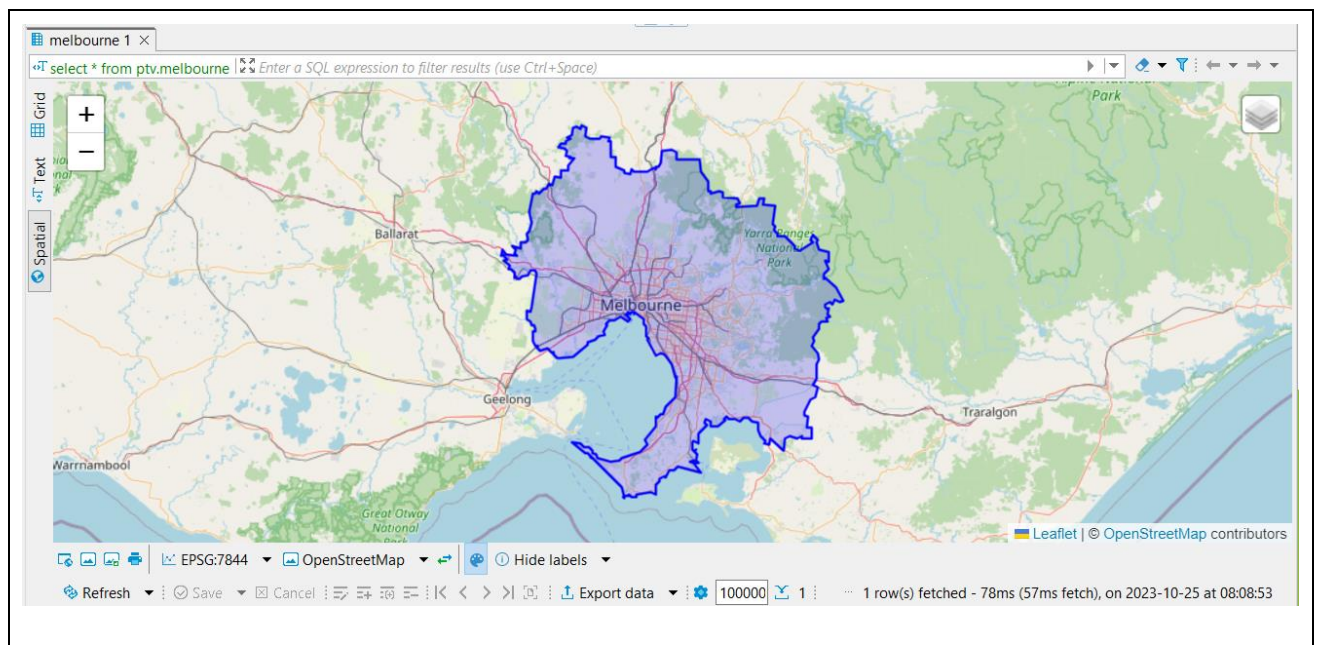
Since the working area will be Melbourne Metropolitan, it is important to have a polygon for the boundary of our working area. Create a table, named “**melbourne**” for Melbourne Metropolitan boundary. Hint: aggregate all mesh blocks polygon to create one large polygon for Melbourne Metropolitan boundary.

Write the SQL script to do this.

```
CREATE TABLE IF NOT EXISTS ptv.melbourne AS
SELECT ST_Union(mm.wkb_geometry) AS geom
FROM ptv.mb2021_mel mm;

SELECT * FROM ptv.melbourne;
```

Attach a screenshot of the Spatial Map results.



2.3 Add Geometry column to Stops table

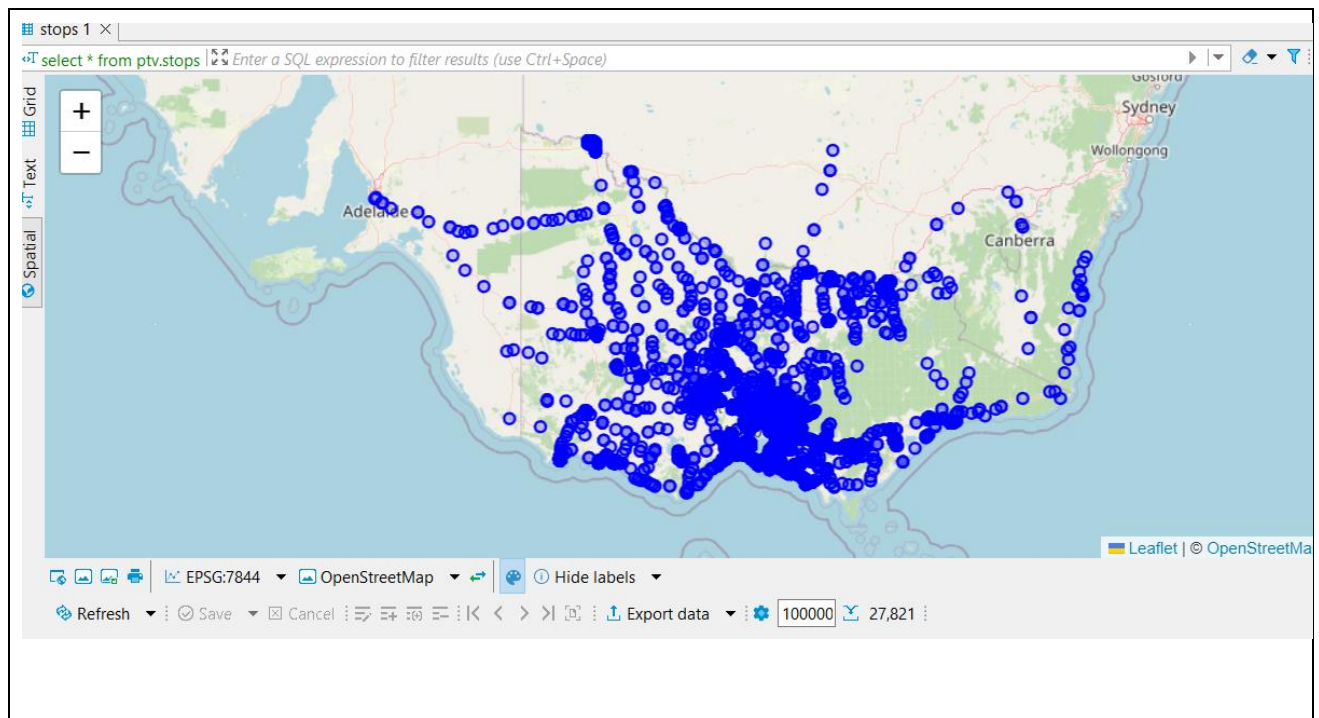
Stops table does not have any geometry column. Add a geometry column by using the latitude and longitude value that are available in the table. Make sure you use GDA2020 (SRID:7844) for this column.

Write the SQL script to do this.

```
ALTER TABLE ptv.stops
ADD COLUMN geom geometry;
UPDATE ptv.stops
SET geom = ST_SetSRID(ST_Point(stop_lon, stop_lat), 7844);

SELECT * FROM ptv.stops;
```

Attach a screenshot of the Spatial Map results.



2.4 Denormalise GTFS structure

The ptv.stops table does not show direct information regarding the vehicle types, routes_short_name and routes_long_name. These information are stored in the routes table.

Create a table called "stops_routes_mel" to encompass the following attributes: stop_id, stop_name, coordinates, route number (derived from routes_short_name), route name (derived from routes_long_name), and vehicle type. This data set should encompass all stops within the Melbourne Metropolitan area.

The vehicle type is determined by the corresponding route type, where:

- 0 corresponds to tram
- 2 corresponds to train
- 3 corresponds to bus
- Any other route type is labeled as 'Unknown'.

Use this figure as an example of expected result. (Note: Data value is for demonstration purposes only.)

	stop_id	stop_name	geom	route_number	route_name	vehicle
1	1000	Dole Ave/Cheddar Rd (Reservoir)	POINT (145.018951051008 -37.7007748061772)	556	Northland SC - Epping Plaza SC	Bus
2	10001	Rex St/Taylor's Rd (Kings Park)	POINT (144.776152425766 -37.7269752097248)	418	St Albans Station - Caroline Springs	Bus
3	10002	Yuille St/Centenary Ave (Melton)	POINT (144.595789405033 -37.6761595024019)	458	Melton Station - Kurunjang	Bus
4	10002	Yuille St/Centenary Ave (Melton)	POINT (144.595789405033 -37.6761595024019)	943	Watergardens Station - Melton	Bus
5	10009	Gum Rd/Main Rd West (Albanvale)	POINT (144.775899388911 -37.7414971143014)	424	St Albans Station - Brimbank Central SC	Bus
6	1001	Lloyd Ave/Cheddar Rd (Reservoir)	POINT (145.019685286526 -37.6991830099504)	556	Northland SC - Epping Plaza SC	Bus
7	10010	Kings Rd/Main Rd West (St Albans)	POINT (144.78008474429 -37.7419455261211)	424	St Albans Station - Brimbank Central SC	Bus
8	10011	Moffat St/Main Rd West (St Albans)	POINT (144.783466504334 -37.7423246041254)	424	St Albans Station - Brimbank Central SC	Bus
9	10012	Washington St/Main Rd West (St Albans)	POINT (144.787912291551 -37.7427956612577)	424	St Albans Station - Brimbank Central SC	Bus
10	10013	Kate St/Main Rd West (St Albans)	POINT (144.79457341719 -37.7435693788456)	424	St Albans Station - Brimbank Central SC	Bus
11	10013	Kate St/Main Rd West (St Albans)	POINT (144.79457341719 -37.7435693788456)	425	St Albans Station - Watergardens Station	Bus
12	10014	Raleighs Rd/Centenary Ave (Melton)	POINT (144.588776428553 -37.6753043554238)	458	Melton Station - Kurunjang	Bus

Make sure you remove any duplications in your result.

Write your SQL query here

```
CREATE TABLE ptv.stops_routes_mel AS
SELECT DISTINCT
  s.stop_id,
  s.stop_name,
  s.geom,
  r.route_short_name AS route_number,
  r.route_long_name AS route_name,
  CASE r.route_type
    WHEN 0 THEN 'Tram'
    WHEN 2 THEN 'Train'
    WHEN 3 THEN 'Bus'
    ELSE 'Unknown'
  END AS vehicle
FROM ptv.stops s
JOIN ptv.stop_times st ON s.stop_id = st.stop_id
JOIN ptv.trips t ON st.trip_id = t.trip_id
JOIN ptv.routes r ON t.route_id = r.route_id
JOIN ptv.melbourne m ON ST_Within(s.geom, m.geom);
```

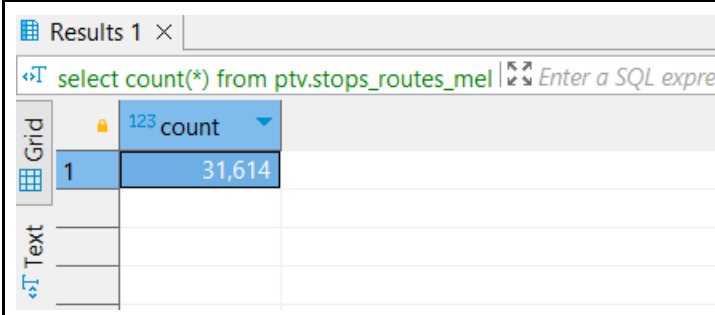
Please complete the following statistics from **stops_routes_mel** table:

Question 2.4.1:How many rows do you have in the stops_routes_mel table?

Write the SQL script to do this and attach a screenshot of the query

```
select count(*) from ptv.stops_routes_mel;
```

Screenshot



The screenshot shows a SQL query results window with the title 'Results 1 x'. The query entered is 'select count(*) from ptv.stops_routes_mel'. The results are displayed in a grid format. The first row shows a count of 123 for the 'count' column. The second row shows a count of 31,614 for the '1' column.

	count
1	31,614

Question 2.4.2:How many unique stop_ids do you have in the stops_routes_mel table?

Write the SQL script to do this and attach a screenshot of the query

```
select count(distinct stop_id)
from ptv.stops_routes_mel;
```

Screenshot

The screenshot shows a SQL query results window titled 'Results 1'. The query is `select count(distinct stop_id) from ptv.sto`. The results are displayed in a table with two columns: 'count' and '1'. The value '20,644' is shown in the 'count' column.

count
20,644

Task 3: Data Analytics

3.1 Suburbs Accessibility

Create an SQL query to identify the **number of bus stops** in each Suburb. Your result should have the suburb name and the total bus stops in it.

Hint :

- identify the mesh block location of a bus stop. Then, aggregate the number in Suburb level.
- One suburb consists of multiple mesh blocks

Write your SQL query here

```
WITH subAccess AS (
    SELECT
        stop_id,
        mb_code21
    FROM
        ptv.stops_routes_mel,
        ptv.mb2021_mel
    WHERE
        vehicle = 'Bus' AND
        ST_Within(ptv.stops_routes_mel.geom, ptv.mb2021_mel.wkb_geometry)
)

SELECT
    s.sal_name_2021 AS suburb_name,
    COUNT(sa.stop_id) AS no_stops
FROM
    subAccess sa
JOIN
    ptv.sal2021 s ON sa.mb_code21 = s.mb_code_2021
GROUP BY
    s.sal_name_2021;
```

Provide the screenshot of your **execution plan** and the real **execution time** for this query. You can get the real execution time under your result set.

Note:

- Execution plan can be found in SQL Editor -> Explain Execution Plan
- The execution time is shown in the screenshot below.(Note: the screenshot is for demonstration purposes only)

Refresh Save Cancel Export data 200 1 1 row(s) fetched - 4ms on 2023-03-24 at 07:37:08

Execution plan :

Node Type	Entity	Cost	Rows	Time	Condition
Aggregate		17833464069....	7497		
Gather M		17833464069....	14994		
Aggre		17833463069....	7497		
Sort		17833463069....	692542		
		6757.66 - 178...	692542		
		6757.66 - 233...	24785		(s.mb_code_...
	sal2021	0.00 - 10668.52	153452		
		5091.85 - 509...	24785		
	mb2021_mel	0.00 - 5091.85	24785		
	stops_routes_mel	0.00 - 908.17	28733		(vehicle = 'B...

Execution time :

456 row(s) fetched - 6m 5s,

You are now tasked with devising an approach to enhance your query execution and minimize execution time. Provide a comprehensive explanation of your strategy, accompanied by the SQL script outlining the measures you've taken to optimize query performance. Additionally, include a screenshot showcasing the execution plan and execution time, effectively visualizing the enhancements achieved following the optimization.

Strategy and SQL script:

Strategy:

Use Explicit Joins: Instead of using a comma-separated list of tables in the FROM clause, use explicit JOIN clauses. This makes the query more readable and can sometimes provide performance benefits.

Indexes: Ensure that there are indexes on columns involved in joins and filters. For this query, consider adding indexes on:

ptv.stops_routes_mel.stop_id
 ptv.stops_routes_mel.geom
 ptv.mb2021_mel.wkb_geometry
 ptv.mb2021_mel.mb_code21
 ptv.sal2021.mb_code_2021

Spatial Indexes: Since we're using spatial functions like st_within, consider using spatial indexes on the geometry columns, if not already indexed.

Filter Early: Reduce the number of rows as early as possible. In the WITH clause (CTE), we're filtering by vehicle = 'Bus'. This filter can be applied even before the spatial check to reduce the number of rows processed by the st_within function.

Aggregation: Use the DISTINCT keyword within the CTE if there are possible duplicates.

```
WITH subAccess AS
(
  SELECT DISTINCT sr.stop_id, mb.mb_code21
  FROM ptv.stops_routes_mel sr
  JOIN ptv.mb2021_mel mb ON st_within(sr.geom, mb.wkb_geometry)
  WHERE sr.vehicle = 'Bus'
)
SELECT s.sal_name_2021 AS suburb_name,
       COUNT(sa.stop_id) AS no_stops
FROM subAccess sa
JOIN ptv.sal2021 s ON sa.mb_code21 = s.mb_code_2021
GROUP BY s.sal_name_2021;
```

Improved Execution plan :

Statistics 1		Execution plan - 1		Execution plan - 2		Execution plan - 3 ×	
Node Type	Entity	Cost	Rows	Time	Condition		
▼ Aggregate		17833998887....	7497				
▼ Sort		17833998887....	1662101				
▼ Hash J		17833425289....	1662101		((mb.mb_co...		
▼ Uni		17833403193....	1662101				
▼		17833403193....	1662101				
		17833402193....	692542				
		0.00 - 178333...	692542				
	mb2021_mel	0.00 - 5091.85	24785				
	stops_routes_mel	0.00 - 908.17	28733		(vehicle = 'B...		
▼ Ha:		12816.86 - 12...	368286				
	sal2021	0.00 - 12816.86	368286				

Improved Execution Time :

456 row(s) fetched - 5m 43s, on 2023-10-25 at 08:41:37

Question 3.1.1: Provide a list of the five suburbs with the lowest count of stops. In case multiple suburbs share the same minimum number of stops in your findings, arrange them in ascending order based on their suburb names.

Write the SQL script to do this and attach a screenshot of the query

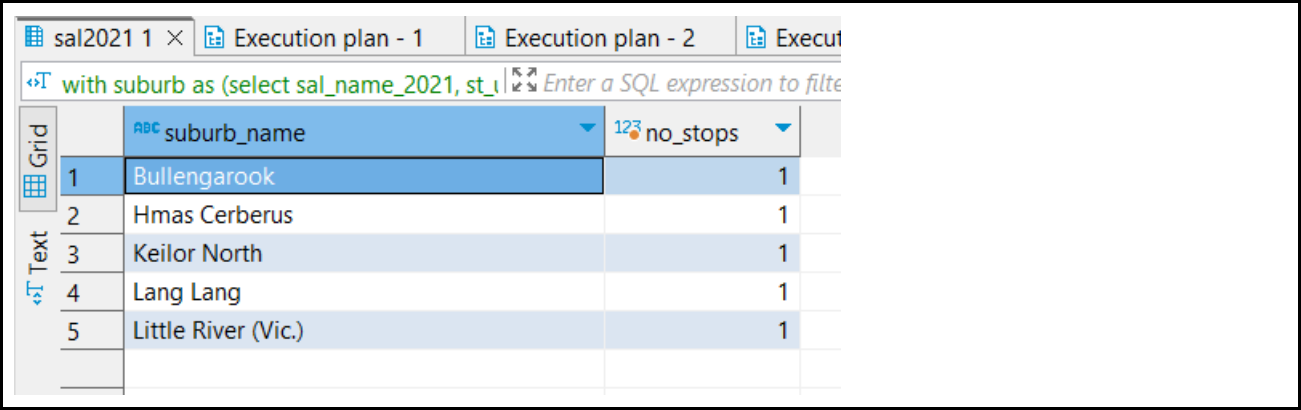
```
WITH subAccess AS
(
  SELECT DISTINCT sr.stop_id, mb.mb_code21
  FROM ptv.stops_routes_mel sr
  JOIN ptv.mb2021_mel mb ON st_within(sr.geom, mb.wkb_geometry)
  WHERE sr.vehicle = 'Bus'
)
SELECT s.sal_name_2021 AS suburb_name,
```



```

COUNT(sa.stop_id) AS no_stops
FROM subAccess sa
JOIN ptv.sal2021 s ON sa.mb_code21 = s.mb_code_2021
GROUP BY s.sal_name_2021
order by no_stops asc, suburb_name asc
Limit 5;
```

Screenshot



The screenshot shows a SQL query execution interface. At the top, there are tabs for 'sal2021 1', 'Execution plan - 1', 'Execution plan - 2', and 'Execute'. Below the tabs is a text input field containing the query: 'with suburb as (select sal_name_2021, st_u'. To the right of the input field is a dropdown menu labeled 'Enter a SQL expression to filter'. Below the input field is a table with 5 rows. The first row is highlighted in blue. The table has two columns: 'suburb_name' and 'no_stops'. The data is as follows:

	suburb_name	no_stops
1	Bullengarook	1
2	Hmas Cerberus	1
3	Keilor North	1
4	Lang Lang	1
5	Little River (Vic.)	1

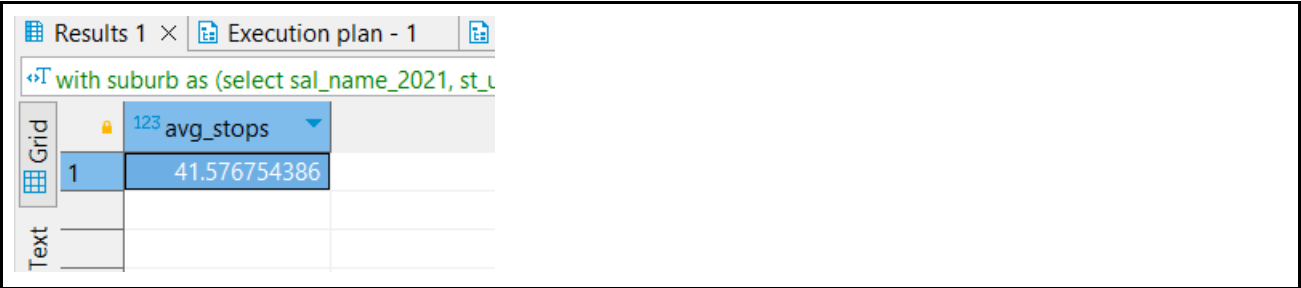
Question 3.1.2: Average number of distinct stops in suburb

Write the SQL script to do this and attach a screenshot of the query

```

WITH subAccess AS
(
  SELECT DISTINCT sr.stop_id, mb.mb_code21
  FROM ptv.stops_routes_mel sr
  JOIN ptv.mb2021_mel mb ON st_within(sr.geom, mb.wkb_geometry)
  WHERE sr.vehicle = 'Bus'
),
suburbStops AS
(
  SELECT s.sal_name_2021 AS suburb_name,
    COUNT(sa.stop_id) AS no_stops
  FROM subAccess sa
  JOIN ptv.sal2021 s ON sa.mb_code21 = s.mb_code_2021
  GROUP BY s.sal_name_2021
)
SELECT AVG(no_stops) AS avg_stops
FROM suburbStops;
```

Screenshot



The screenshot shows a SQL query execution interface. At the top, there are tabs for 'Results 1', 'Execution plan - 1', and 'Execute'. Below the tabs is a text input field containing the query: 'with suburb as (select sal_name_2021, st_u'. To the right of the input field is a dropdown menu labeled 'Enter a SQL expression to filter'. Below the input field is a table with 1 row. The first row is highlighted in blue. The table has two columns: 'avg_stops' and 'no_stops'. The data is as follows:

	avg_stops	no_stops
1	41.576754386	


```

lga_stop AS (
  SELECT DISTINCT
    la.lga_name_2021,
    s.geom
  FROM
    ptv.stops_routes_mel s
  JOIN
    lga la ON ST_Within(s.geom, la.geom)
  WHERE
    vehicle = 'Bus'
),
lga_residential AS (
  SELECT
    la.lga_name_2021,
    mb.mb_code21,
    mb.wkb_geometry geom
  FROM
    ptv.mb2021_mel mb
  JOIN
    ptv.lga2021 la ON mb.mb_code21 = la.mb_code_2021
  WHERE
    initcap(mb.mb_cat21) LIKE '%Residential%'
),
ns AS (
  SELECT
    s.lga_name_2021,
    COUNT(DISTINCT r.mb_code21) non_blankspot
  FROM
    lga_stop s
  JOIN
    lga_residential r ON s.lga_name_2021 = r.lga_name_2021
    AND ST_Within(s.geom, r.geom)
  GROUP BY
    s.lga_name_2021
),
ra AS (
  SELECT
    la.lga_name_2021,
    COUNT(DISTINCT mb.mb_code21) count_residential
  FROM
    lga_residential mb
  GROUP BY
    la.lga_name_2021
)
SELECT
  ns.lga_name_2021 AS LGA_name,
  count_residential,
  count_residential - non_blankspot AS count_blankspot,
  ROUND((count_residential - non_blankspot)::NUMERIC / count_residential::NUMERIC *
100, 2) || '%' AS blankspot_percentage
FROM
  ns
JOIN
  ra ON ns.lga_name_2021 = ra.lga_name_2021
ORDER BY
  blankspot_percentage ASC;

```

Screenshot

Iga2021 1 × Execution plan - 1 Execution plan - 2 Execution plan - 3				
with lga as (select lga_name_2021, st_uni Enter a SQL expression to filter results (use Ctrl+Space)				
Grid	ABC lga_name	123 count_residential	123 count_blankspot	ABC blankspot_pe
1	Murrindindi	27	22	81.48%
2	Mitchell	187	162	86.63%
3	Moorabool	214	164	76.64%
4	Macedon Ranges	249	208	83.53%
5	Nillumbik	502	386	76.89%
6	Melbourne	852	774	90.85%
7	Maribyrnong	884	673	76.13%
8	Hobsons Bay	906	648	71.52%
9	Yarra	914	855	93.54%
10	Cardinia	945	796	84.23%
11	Bayside (Vic.)	1,019	702	68.89%
12	Maroondah	1,156	826	71.45%
13	Manningham	1,167	766	65.64%
14	Moonee Valley	1,210	913	75.45%
15	Port Phillip	1,252	1,114	88.98%
16	Stonnington	1,272	1,143	89.86%
17	Yarra Ranges	1,298	907	69.88%
18	Banyule	1,302	936	71.89%
19	Knox	1,414	968	68.46%
20	Greater Dandenong	1,447	1,022	70.63%

Question 3.2.1: Complete the following statistical data based on the result.

Note:

- The query and screenshot are not required for this section. You can write down your results directly
- If more than one LGA has the same percentage of blankspots, sort them by LGA name in ascending order.(Updated 22/10/23)

Criteria	Answer
Top 5 LGAs with the highest % of blankspots	Yarra/Melbourne/Stonnington/Port Philip/Mitchell
Top 5 LGAs with the lowest % of blankspots	Manningham/Whitehorse/Knox/Bayside/Yarra Ranges
Average % of blankspots	76.8%

Task 4: Data Visualisation

In this task, you are required to incorporate a heatmap visualization.

4.1 LGA Blankspot Analysis

In this task, you will put the blankspot percentage in the heatmap. Provide the segmentation as follow:

- $X \leq 20\%$
- $20\% < X \leq 40\%$
- $40\% < X \leq 60\%$
- $60\% < X \leq 80\%$
- $X > 80\%$

Write an SQL query to create a table named 'lga_blankspot' containing the blankspot percentages categorised by the LGA from the previous task 3.2 and sufficient spatial data. And attach a screenshot of the table contents.

Note: Ensure that this table is structured to facilitate the creation of a visual heat map specifically for the Melbourne region.

```
CREATE TABLE ptv.lga_blankspot AS
WITH
lga AS (
  SELECT
    lga_name_2021,
    st_union(wkb_geometry) geom
  FROM
    ptv.mb2021_mel mb,
    ptv.lga2021 l
  WHERE
    mb.mb_code21 = l.mb_code_2021
  GROUP BY
    l.lga_name_2021
),
lga_stop AS (
  SELECT DISTINCT
    lga_name_2021,
    s.geom
  FROM
    ptv.stops_routes_mel s,
    lga
  WHERE
    vehicle = 'Bus'
    AND
    st_within(s.geom, lga.geom)
),
lga_residential AS (
  SELECT
    lga_name_2021,
    mb.mb_code21,
    mb.wkb_geometry AS geom
  FROM
    ptv.mb2021_mel mb,
    ptv.lga2021 l
  WHERE
```

```

        mb.mb_code21 = l.mb_code_2021
        AND
        INITCAP(mb.mb_cat21) LIKE '%Residential%'
    ),
    ns AS (
        SELECT
            s.lga_name_2021,
            COUNT(DISTINCT r.mb_code21) AS non_blankspot
        FROM
            lga_stop s,
            lga_residential r
        WHERE
            s.lga_name_2021 = r.lga_name_2021
            AND
            st_within(s.geom, r.geom)
        GROUP BY
            s.lga_name_2021
    ),
    ra AS (
        SELECT
            lga_name_2021,
            COUNT(DISTINCT mb_code21) AS count_residential
        FROM
            lga_residential
        GROUP BY
            lga_name_2021
    )

    SELECT
        ns.lga_name_2021 AS LGA_name,
        ROUND((count_residential - non_blankspot)::numeric / count_residential *
100, 2) AS blankspot_percentage,
        lga.geom
    FROM
        ns,
        ra,
        lga
    WHERE
        ns.lga_name_2021 = ra.lga_name_2021
        AND
        ns.lga_name_2021 = lga.lga_name_2021;

    ALTER TABLE ptv.lga_blankspot ADD
    COLUMN percentage_range VARCHAR(10);

    UPDATE ptv.lga_blankspot
    SET percentage_range = CASE
        WHEN blankspot_percentage <= 20 THEN 'X<=20%'
        WHEN blankspot_percentage > 20 AND blankspot_percentage <= 40 THEN '20%<X<=40%'
        WHEN blankspot_percentage > 40 AND blankspot_percentage <= 60 THEN '40%<X<=60%'
        WHEN blankspot_percentage > 60 AND blankspot_percentage <= 80 THEN '60%<X<=80%'
        ELSE 'X>=80%'
    END;

    SELECT * FROM ptv.lga_blankspot;

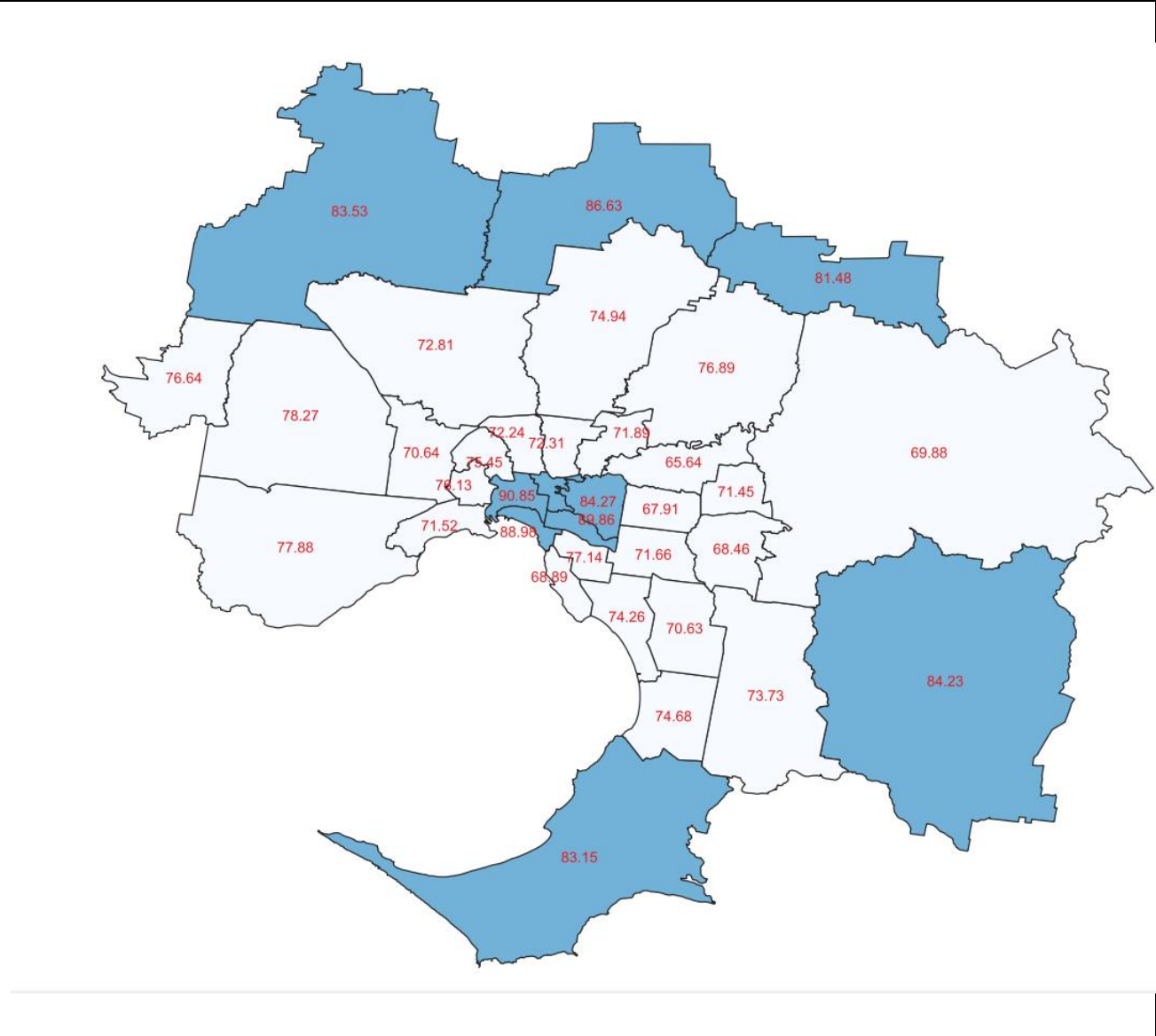
```

Screenshot

Statistics 1 × Execution plan - 1 Execution plan - 2 Execution plan - 3	
Name	Value
Updated Rows	35
Query	<pre> create table ptv.lga_blankspot as with lga as (select lga_name_2021, st_union(wkb_geometry) geom from ptv.mb2021_mel mm, ptv.lga2021 l where mm.mb_code21 = l.mb_code2021 group by l.lga_name_2021), lga_stop as (select distinct lga_name_2021, s.geom from ptv.stops_routes_mel s, lga l where vehicle = 'Bus' and st_within(s.geom, l.geom)), lga_residential as (select lga_name_2021, mm.mb_code21, mm.wkb_geometry geom from ptv.mb2021_mel mm, ptv.lga2021 l where mm.mb_code21 = l.mb_code2021 and initcap(mm.mb_cat21) like '%Residential%'), a as (select s.lga_name_2021, count(distinct r.mb_code21) non_blankspot from lga_stop s, lga_residential r where s.lga_name_2021 = r.lga_name_2021 and st_within(s.geom, r.geom) group by s.lga_name_2021), b as (select lga_name_2021, count(distinct mb_code21) count_residential from lga_residential group by lga_name_2021) select a.lga_name_2021 LGA_name, round((count_residential - non_blankspot)::numeric / count_residential::numeric*100,2) blankspot_percentage, lga.geom from a,b,lga where a.lga_name_2021 = b.lga_name_2021 and a.lga_name_2021 = lga.lga_name_2021 </pre>

Provide the screenshot for the heatmap by using QGIS. Please note that the heatmap is map-based and visually represents the distribution of blankspot percentages across different LGAs in the Melbourne region.

Remember to include appropriate labels, titles, and legends in your visualizations to make them easy to understand (Updated 05/10/2023)



End