Project Report by Cao Zhongting G2304787B

1. The intelligent function

Function 1: Display the shortest route by MRT from a Property_unit to NTU. In order to realize the function, we need to plot the route from one MRT station to another, and then sum up the number of MRT segments along the route. Then we need to sort them and get the shortest three from an HDB block to another block by the closest MRT station.

Function 2: Find the route from a landmark to a Property_unit or another landmark. In order to realize the function, we need to create nodes of the landmarks and create relationships between them to the closest MRT_station. Then we will be able to find the shortest route using queries.

2. The graph database design

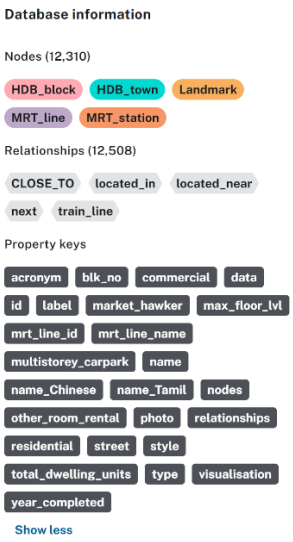
Create the main classes and indexes by using the following commands:

```
CREATE CONSTRAINT FOR (X:HDB_block) REQUIRE X.id IS UNIQUE
CREATE CONSTRAINT FOR (X:HDB_town) REQUIRE X.id IS UNIQUE
CREATE CONSTRAINT FOR (X:MRT_station) REQUIRE X.id IS UNIQUE
CREATE CONSTRAINT FOR (X:MRT_line) REQUIRE X.id IS UNIQUE
CREATE FULLTEXT INDEX nodeIndex FOR
(n:HDB_block|HDB_town|MRT_station|MRT_line) ON EACH [n.id, n.type, n.label,
n.name_Chinese, n.name_Tamil, n.acronym]
```

Then add typical landmarks instances:

```
MERGE (marina_bay_sands:Landmark {name: "Marina Bay Sands"})
MERGE (gardens_by_the_bay:Landmark {name: "Gardens by the Bay"})
MERGE (singapore_botanic_gardens:Landmark {name: "Singapore Botanic Gardens"})
MERGE (sentosa_island:Landmark {name: "Sentosa Island"})
MERGE (chinatown:Landmark {name: "Chinatown"})
MERGE (little_india:Landmark {name: "Little India"})
MERGE (east_coast_park:Landmark {name: "East Coast Park"})
```

We can check the graph of the classes and instances created:



3. The data preparation

Firstly, import the data of mrt_station, edge_MRT_link, hdb_town, hdb_block and edge_MRT-HDB_town from the excel links by using “LOAD CSV WITH HEADERS FROM ” and the following commands, in which r represents the link loaded:

```
MERGE (mrt:MRT_station {
   id: r.code,
   type: r.type,
   label: r.station_name_English,
   name_Chinese: r.station_name_Chinese,
   name_Tamil: r.station_name_Tamil,
   photo: r.photo })
ON CREATE SET mrt.mrt_line_name = [r.train_line],
   mrt.mrt_line_id = [r.train_line_code]
ON MATCH SET mrt.mrt_line_name = mrt.mrt_line_name + [r.train_line],
   mrt.mrt_line_id =    mrt.mrt_line_id + [r.train_line_code]
MERGE (line:MRT_line {id: r.train_line_code, type: 'MRT_line', label: r.train_line})
MERGE (mrt) -[:train_line {type: 'train_line'}]-> (line)
```

Secondly, link up the MRT stations:

```
MATCH (mrt1:MRT_station { id: r.Source })
MATCH (mrt2:MRT_station { id: r.Target })
MERGE (mrt1) -[:next {type: r.Relation}]-> (mrt2)
```

Thirdly, create the hdb_town and hdb_block instances:

```
CREATE (town:HDB_town {
```

```
    id: r.Town,
    label: r.Label,
    type: r.type    })
MATCH (town:HDB_town { id: r.Target })
CREATE ( block:HDB_block {
    id: r.Source,
    label: r.Label,
    type: r.type,
    blk_no: r.blk_no,
    street: r.street,
    max_floor_lvl: toInteger(r.max_floor_lvl),
    year_completed: toInteger(r.year_completed),
    residential: toBoolean(r.residential),
    commercial: toBoolean(r.commercial),
    market_hawker: toBoolean(r.market_hawker),
    multistorey_carpark: toBoolean(r.multistorey_carpark),
    total_dwelling_units: toInteger(r.total_dwelling_units),
    oneRoom_rental: toInteger(r.oneRoom_rental),
    twoRoom_rental: toInteger(r.twoRoom_rental),
    threeRoom_rental: toInteger(r.threeRoom_rental),
    other_room_rental: toInteger(r.other_room_rental) })
CREATE (block) -[:located_in {type: r.Relation}]-> (town)
```

After that, link MRT_station to HDB_town:

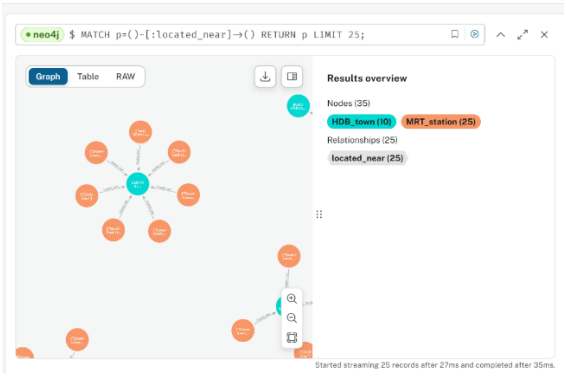
```
MATCH (mrt:MRT_station { id: r.Source })
MATCH (town:HDB_town { id: r.Target })
MERGE (mrt) -[:located_near {type: r.Relation}]-> (town)
```

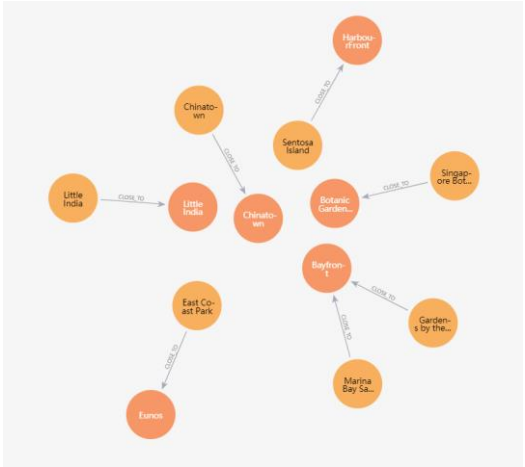
Finally, link Landmark to MRT_station with the relation of “CLOSE_TO”:

```
MATCH (landmark:Landmark {name: "Marina Bay Sands"}), (mrt:MRT_station {label:
"Bayfront"})
MERGE (landmark)-[:CLOSE_TO]->(mrt)
MATCH (landmark:Landmark {name: "Gardens by the Bay"}), (mrt:MRT_station {label:
"Bayfront"})
MERGE (landmark)-[:CLOSE_TO]->(mrt)
MATCH (landmark:Landmark {name: "Singapore Botanic Gardens"}), (mrt:MRT_station
{ id: "CC19_DT9"})
MERGE (landmark)-[:CLOSE_TO]->(mrt)
MATCH (landmark:Landmark {name: "Sentosa Island"}), (mrt:MRT_station {label:
"HarbourFront"})
MERGE (landmark)-[:CLOSE_TO]->(mrt)
MATCH (landmark:Landmark {name: "Chinatown"}), (mrt:MRT_station {label:
```

"Chinatown"})
MERGE (landmark)-[:CLOSE_TO]->(mrt)
MATCH (landmark:Landmark {name: "Little India"}), (mrt:MRT_station {label: "Little India"})
MERGE (landmark)-[:CLOSE_TO]->(mrt)
MATCH (landmark:Landmark {name: "East Coast Park"}), (mrt:MRT_station {label: "Eunos"})
MERGE (landmark)-[:CLOSE_TO]->(mrt)

The data has been imported into the project and the relations have been constructed, and we can match the useful entries as the following graph shows.





4. The Cypher queries to support the intelligent function

The first challenge is to locate the starter and the end. In this case, we can choose Pioneer as the MRT_station close to NTU, and Bugis as the Property_unit. And the solution of getting the shortest path is to use “next” to connect this two MRT_stations:

MATCH (mrt1:MRT_station { label: "Pioneer" }),
(mrt2:MRT_station { label: "Bugis" }),

```
                path = shortestPath (    (mrt1)-[:next*1..50]-(mrt2)    )
RETURN path, length(path)
```

Secondly, because the label of Botanic Garden is "Botanic Gardens • Kebun Bunga", I failed to create the connection at the first time. After noticing this reason, I use its “id” instead.

For example, if we want to travel from East Coast Park to Marina Bay Sands, this is the query of the shortest path:

```
MATCH (eastCoast:Landmark {name: "East Coast Park"}),
      (marina:Landmark {name: "Marina Bay Sands"}),
      (eastCoast)-[:CLOSE_TO]->(mrt1:MRT_station),
      (marina)-[:CLOSE_TO]->(mrt2:MRT_station),
      path = shortestPath((mrt1)-[:next*1..50]-(mrt2))
RETURN path, length(path) AS path_length;
```

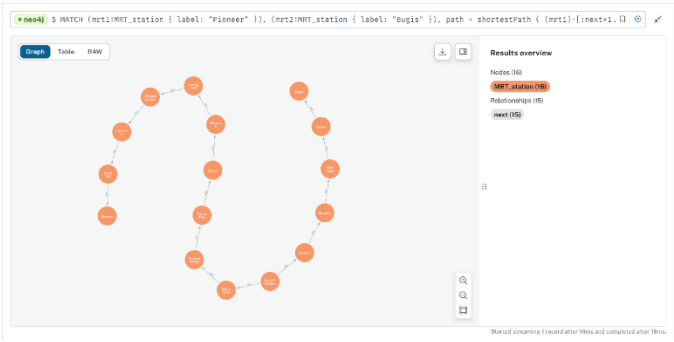
And if we want to depart from Marina Bay Sands to Sentosa Island, this is the query of the shortest path:

```
MATCH (marina:Landmark {name: "Marina Bay Sands"}),
      (sentosa:Landmark {name: "Sentosa Island"}),
      (marina)-[:CLOSE_TO]->(mrt1:MRT_station),
      (sentosa)-[:CLOSE_TO]->(mrt2:MRT_station),
      path = shortestPath((mrt1)-[:next*1..50]-(mrt2))
RETURN path, length(path) AS path_length;
```

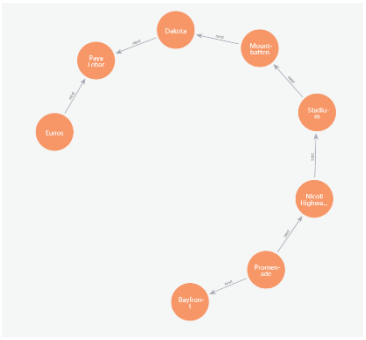
5. The design of the result output

In order to display the result clearly, we can use the label of the MRT_stations on each node and show the complete route as the following graph shows.

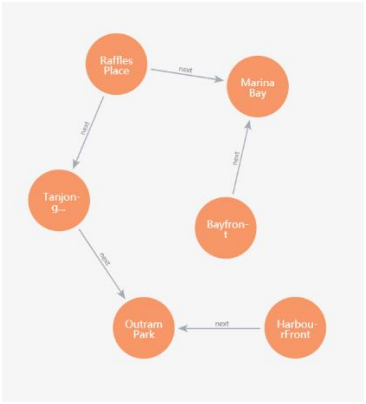
The following path is the shortest route from a home near Bugis to NTU:



The following path is the shortest route from East Coast Park to Marina Bay Sands:



The following path is the shortest route from Marina Bay Sands to Sentosa Island:



6. Possible future extensions

In order to make the application more intelligent, there are several possible methods. Firstly, bus stops can be added into the database so that combined route of MRT and bus can be searched and suggested. Secondly, the transition indication can be completed. Transition stops, estimated transiting time, boarding platform and direction will be restored in it. As a result, users will be more convenient and well-informed to take the public transportation. And for the landmarks, the rating point and operating hours can be added for the tourists to better plan their itinerary.