AIMOTIVE – PYTHON DEVELOPER TEST TASK

1. **Design a database to store the following information**

1. Road Segments

1. They have a unique ID
2. They may have a name, but it's not mandatory

2. Lanes

1. Every lane belongs to one and only one road segment
2. The road centerline is identified as lane zero
3. Right side lanes are numbered as -1, -2, -3, etc.
4. Left side lanes are numbered as 1, 2, 3, etc.
5. One-sided roads (e.g. positive lanes only) are possible
6. The width of the lanes is not constant along their length. The width of a lane is given at arbitrary points along its length, starting from the beginning of the road segment (e.g. at 0 m the width is 3.0 m, at 100 m it is 3.5 m, at 150 m it is 3.2 m.) The width of the zero lane is always 0.0 m.

3. Lane connectivity

1. Previous lanes / next lanes
1. More than one connecting lane is possible at both ends
2. They are obviously on another road segment

2. **Using the above database and your favorite Python tools to access it, provide Python functions for the following jobs. Please take care of handling bad input and cases when there is nothing to return**

1. Given a Road Segment ID, return the number of lanes in both directions

2. Given a Road Segment ID, a lane number and a distance from the start of the lane, return the lane width at that position (you may need to interpolate)

3. Given a Road Segment ID and a lane number, return the possible next lanes (Road Segment ID and lane number), where the simulated vehicle can continue its route.

DATABASE

The required database which specified above deployed on PostgreSQL Server. It is hosted by ElephantSQL web service with the following connection details:

Host: *tai.db.elephantsql.com*

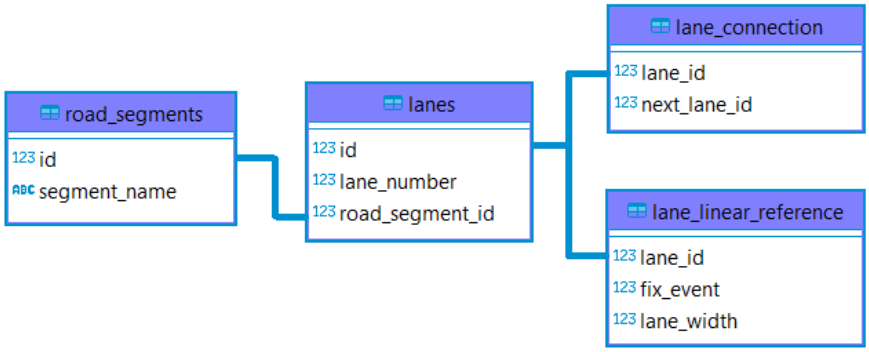
Database: *bxfpfzgy*

Port: *5432*

User: *bxfpfzgy*

Password*: H-8_3SARM_wknvE7W8A4eEcHAzIJ-n2F*

Database Structure



Database table samples are explained below. Test data is fictional. The names and measures are idealized, may not fit with the real world.

id	segment_name
1	M1 BAH csomópont - Budaörs
2	M1 Budaörs - Biatorbágy
3	M1 Biatorbágy - Bicske

Table: **road_segments**
Columns: **id** [unique ID]
segment_name [name]

id	lane_number	road_segment_id
1	-3	1
2	-2	1
3	-1	1
4	0	1
5	1	1
6	2	1
7	3	1
8	-3	2
9	-2	2
10	-1	2

Table: **lanes**
Columns: **id** [unique ID]
lane_number [constrained]
road_segment_id [r.s.unique ID]

lane_id	fix_event	lane_width
1	0	3.23
1	50	3.54
1	100	3.33
1	150	3.73
2	0	3.15
2	50	3.23
2	100	3.43
2	150	3.65
3	0	3.24
3	50	3.52
3	100	3.21
3	150	3.64
4	0	0
4	50	0
4	100	0

Table: **lane_linear_reference**
Columns: **lane_id** [lane unique ID]
fix event [fix points along its length, starting from the beginning of the road segment (meter)]
lane width [width of the lane (meter) in the fixed points]

lane_id	next_lane_id
1	8
2	9
3	10
4	11
5	12
6	13
7	14
8	15
9	16
10	17

Table: **lane_connection**
Columns: **lane_id** [lane unique ID]
next_lane_id [the next lane unique ID, where the simulated vehicle can continue its route.]

The whole database was created with the attached SQL file on GitHub:

- **sql\aimotive_db.sql**

PYTHON FUNCTIONS

Environment (Python 3.9.4)

The result functions based on python virtual environment which contains the favourite python libraries used in the task. It is in **python3** folder on GitHub.

Additional modules

Connecting to the PostgreSQL Server - an own module controls the connection, named **postgr_conn.py.** Result functions use this module at every turn when they need data access to the database server

Main modules

1) Task: Given a Road Segment ID, return the number of lanes in both directions
    - **lane_numbers.py**
        - Input parameters:
            - *Enter Road Segment ID:, e.g.* ***1***
        - Output parameters:
            - *A list of number of lanes, e.g.* ***-3,-2,-1,0,1,2,3***

2) Task: Given a Road Segment ID, a lane number and a distance from the start of the lane, return the lane width at that position (you may need to interpolate)
    - **lane_width.py**
        - Input parameters:
            - *Enter Road Segment ID: e.g.* ***1***
            - *Enter Lane Number: e.g.* ***-2***
            - *Enter arbitary point to calculate lane-width (m): e.g.* ***121***
        - Output parameters:
            - *A float value of the lane width: e.g.* ***3.5224***

3) Task: Given a Road Segment ID and a lane number, return the possible next lanes (Road Segment ID and lane number), where the simulated vehicle can continue its route.
    - **lane_continue.py**
        - Input parameters:
            - *Enter Road Segment ID: e.g.* ***1***
            - *Enter Lane Number: e.g.* ***3***
        - Output parameters:
            - *Possible next lane data, e.g.* ***Vehicle can can continue its route on 2. Road segment, 3. numbered lane.***