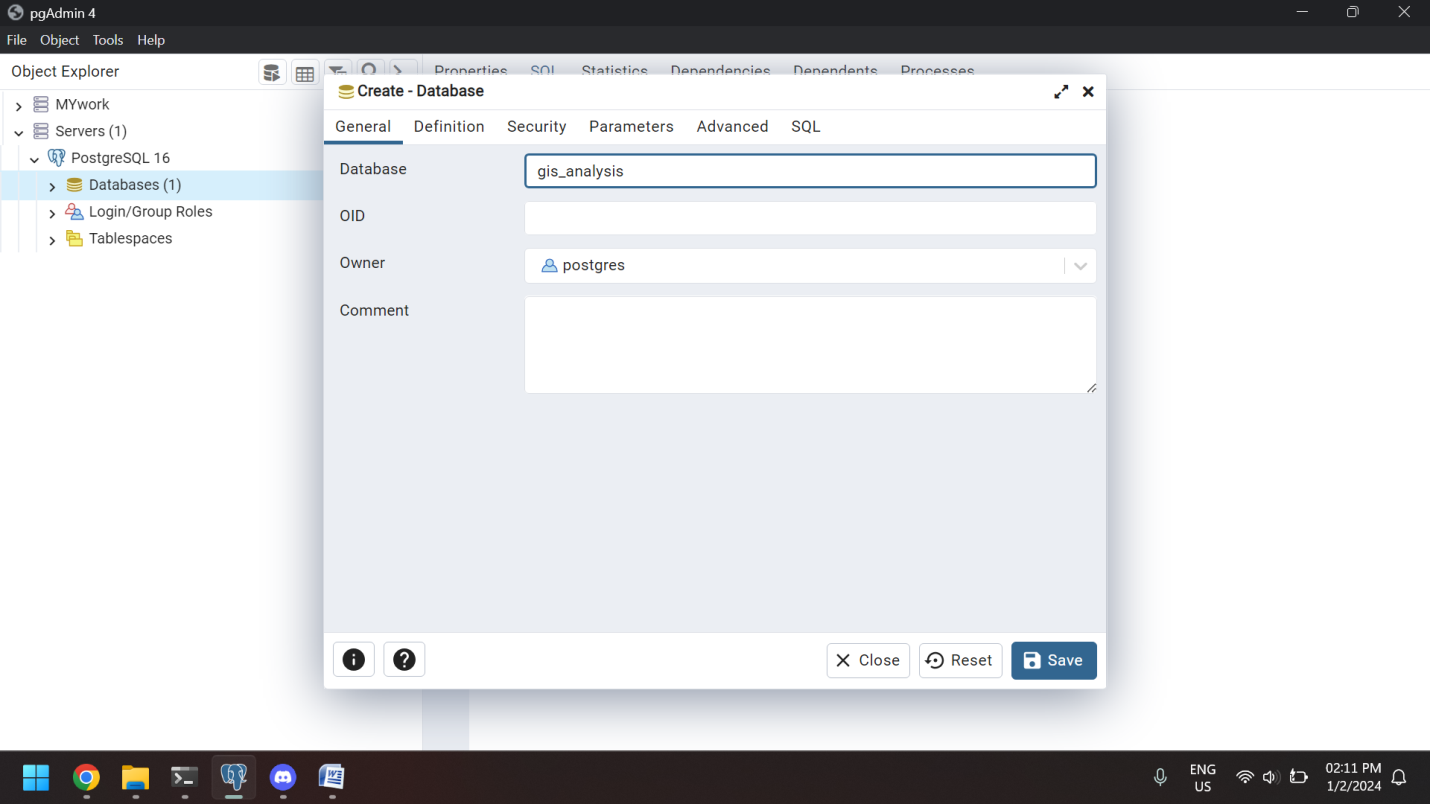
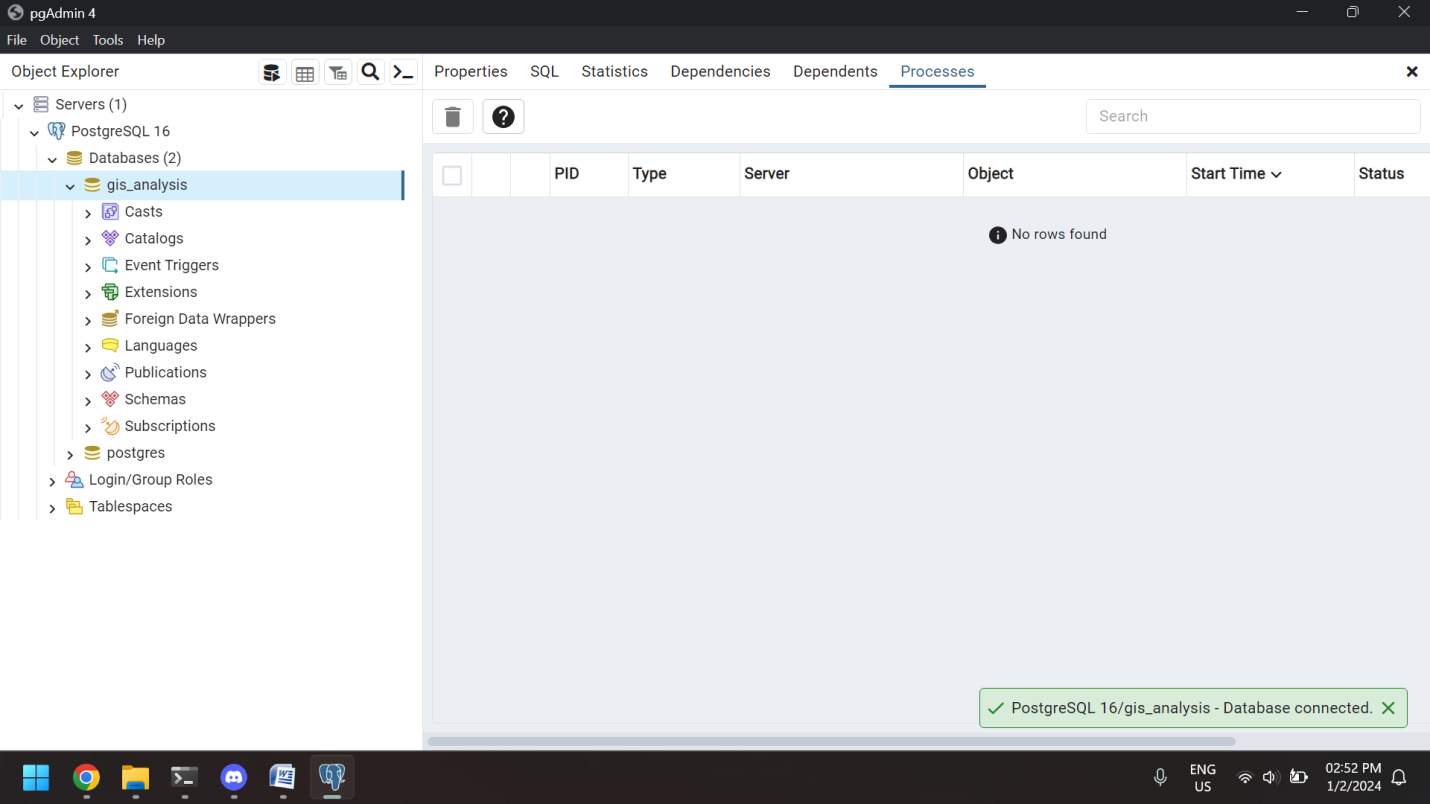
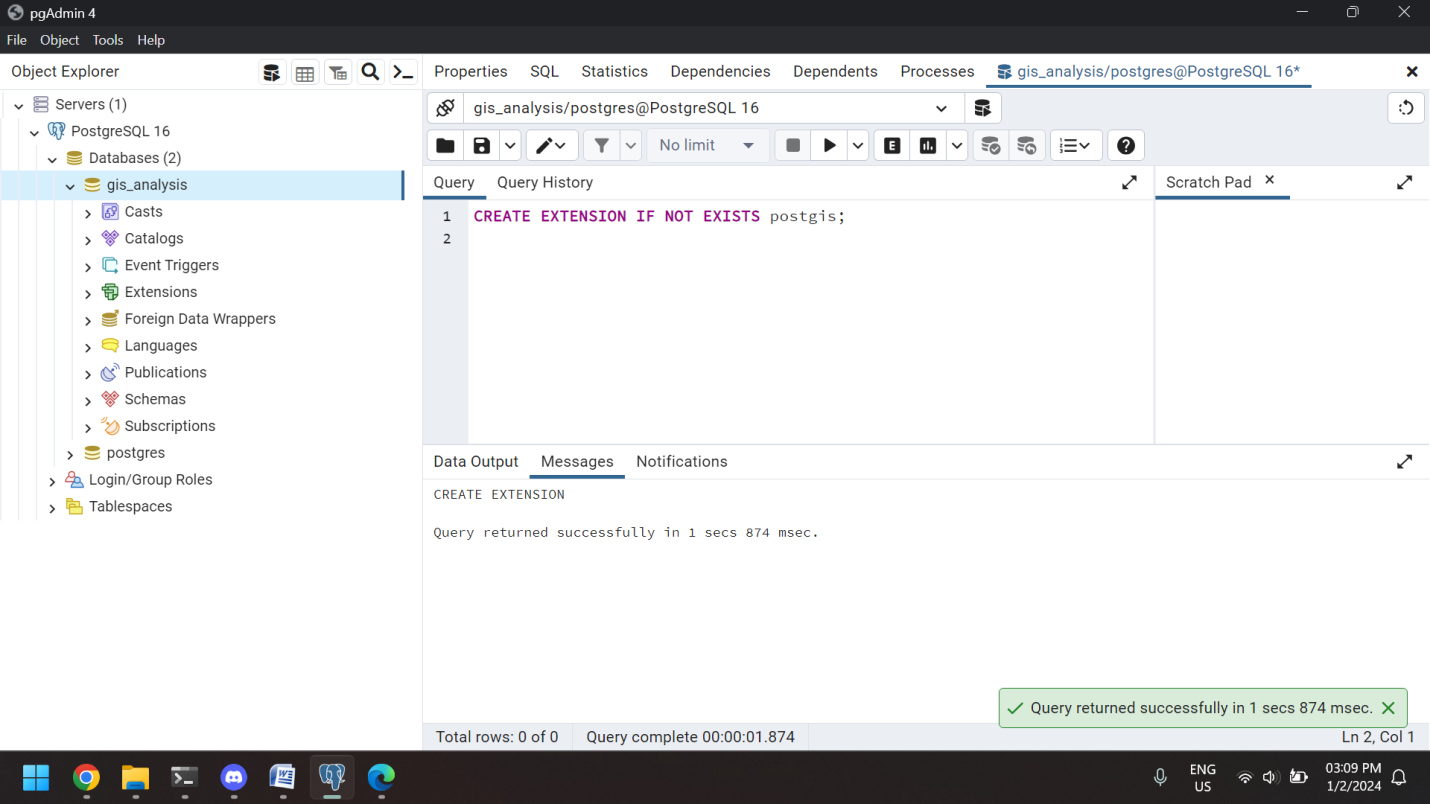
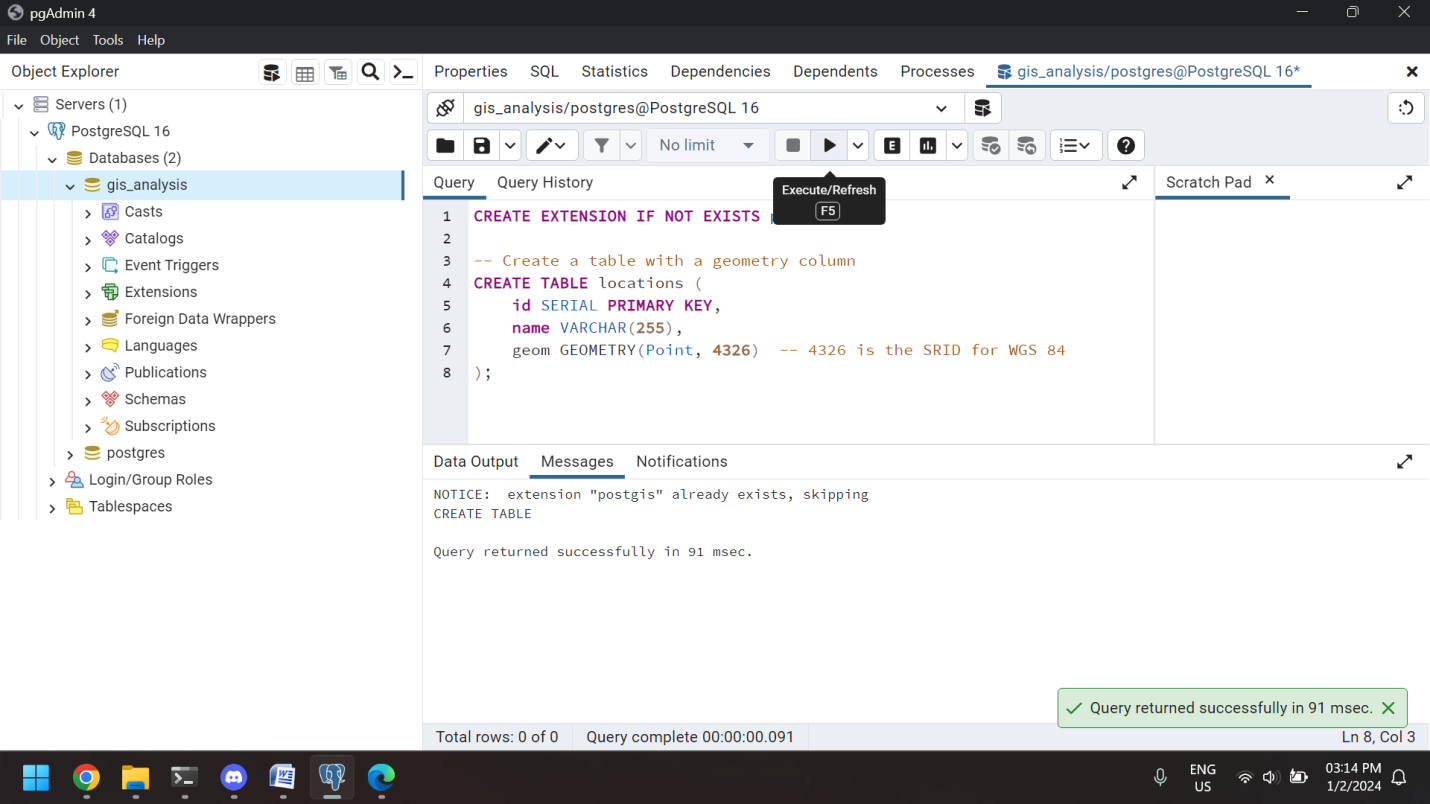
**Creating the database**

****

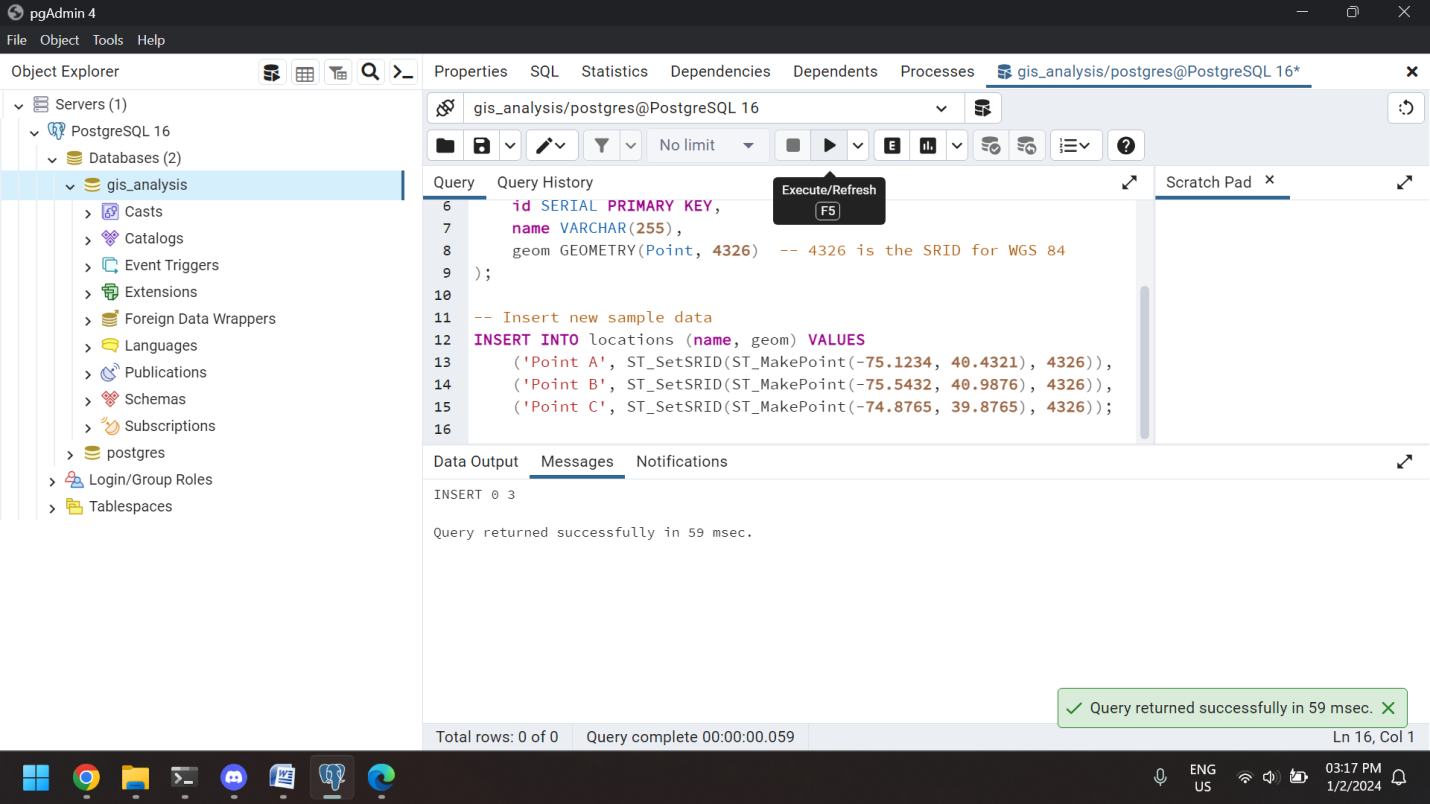
****

****

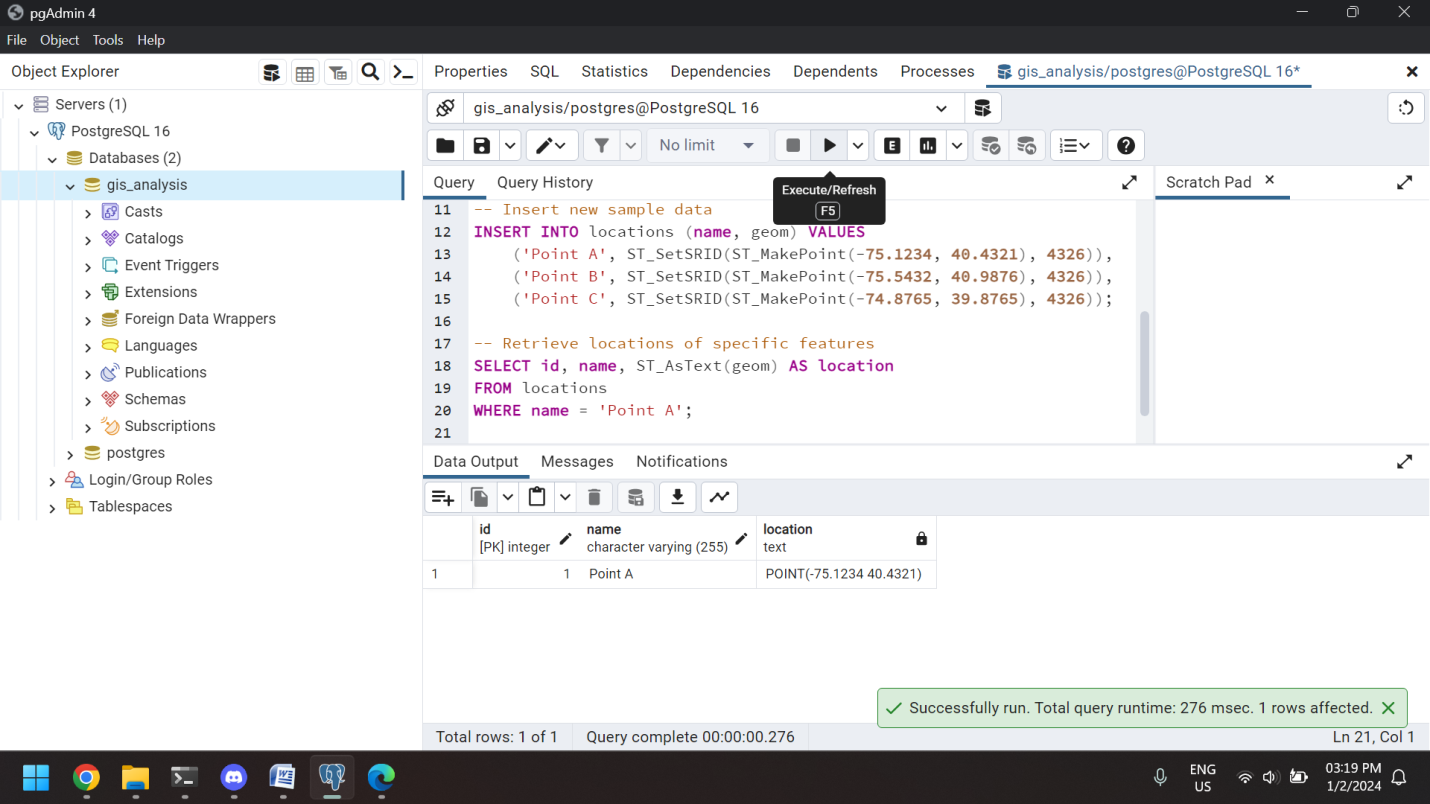
**Creating the table**

****

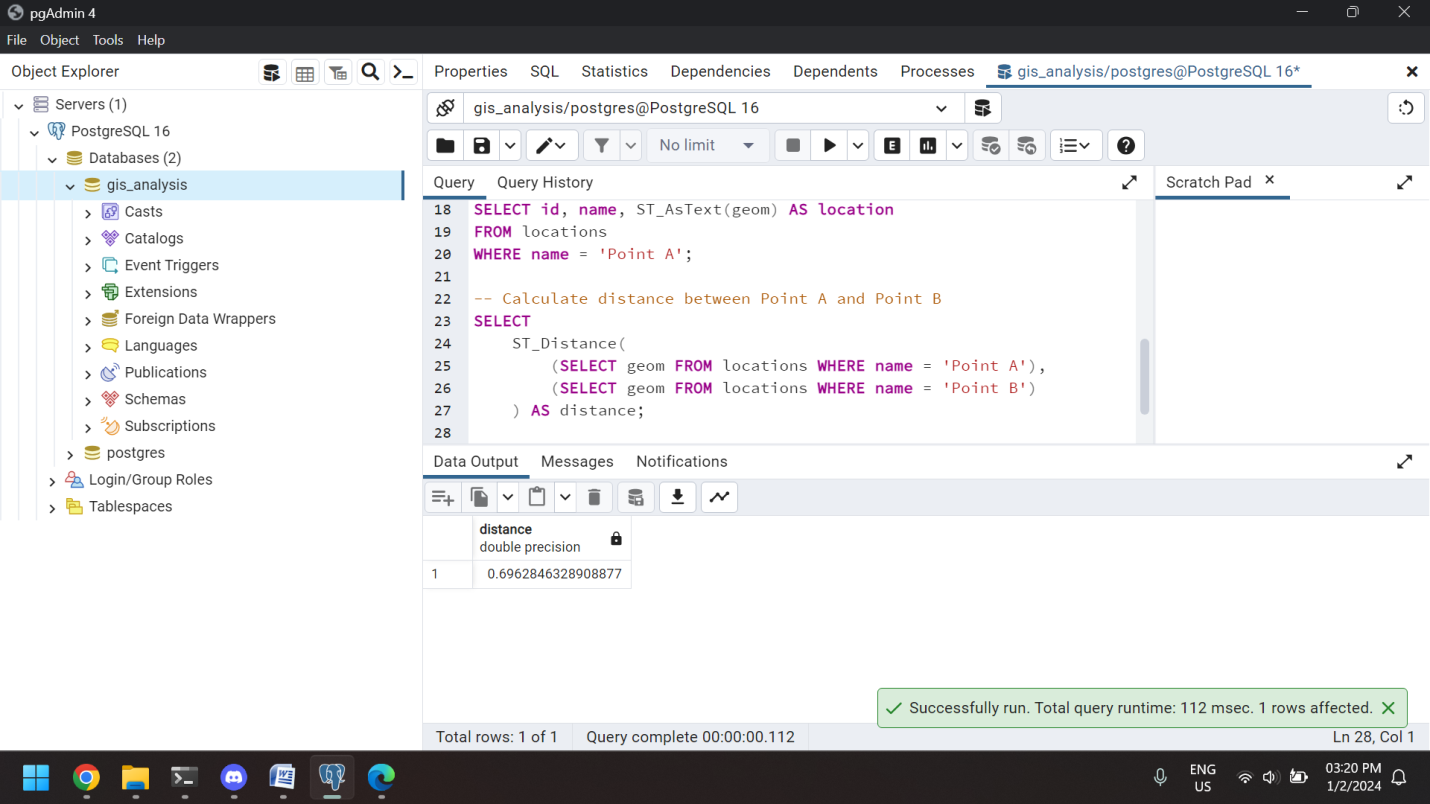
**Inserting the values into table**

****

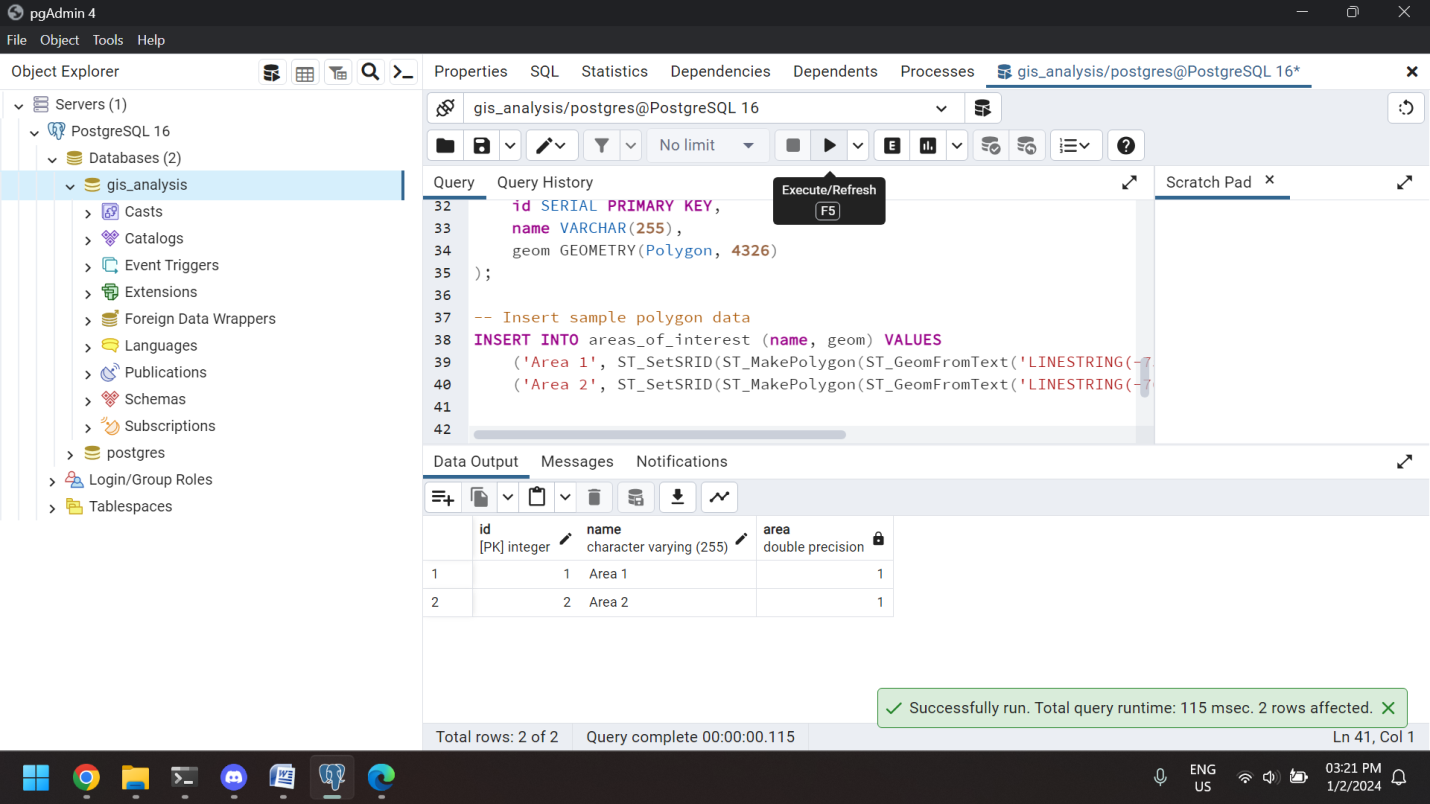
**Question-1:**

****

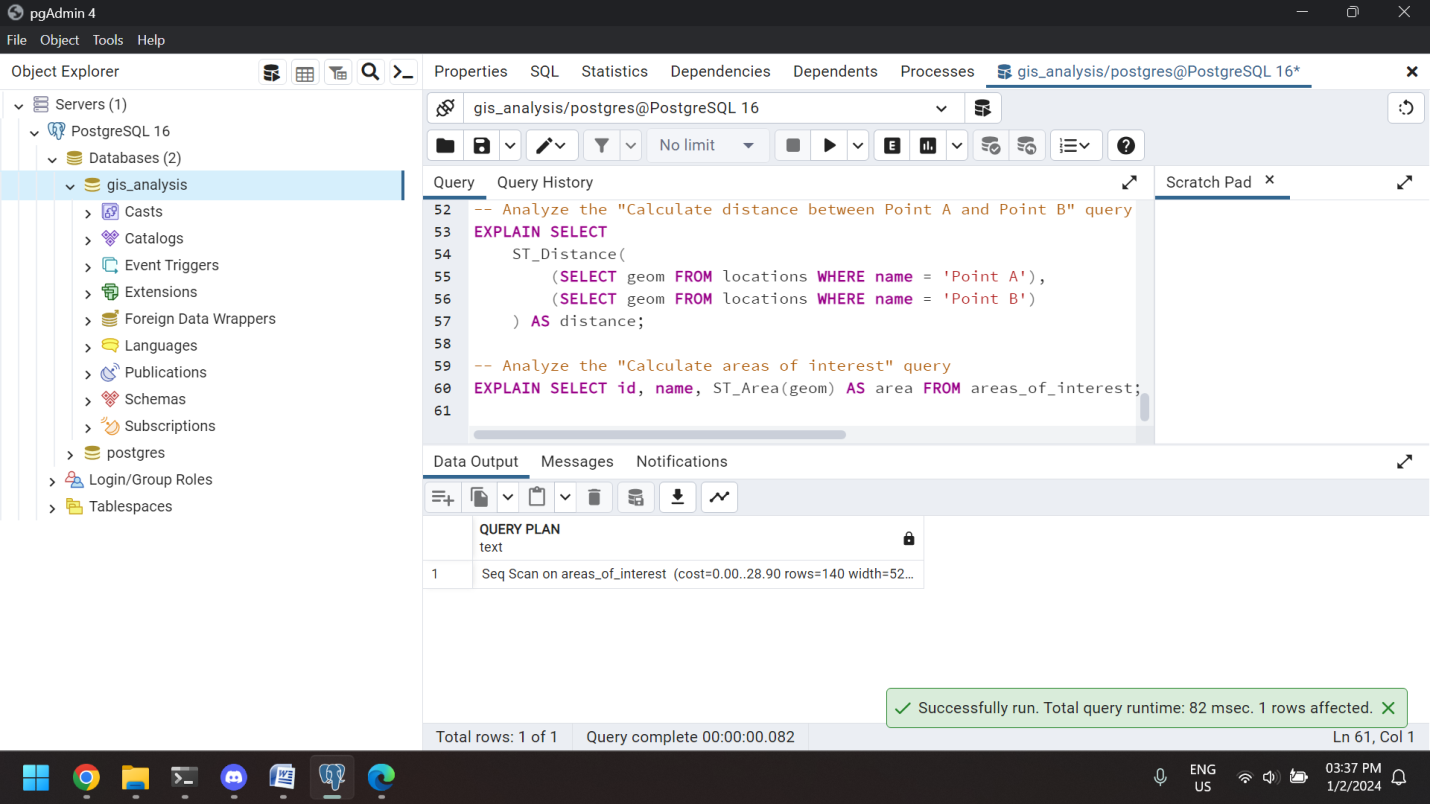
**Question-2:**

****

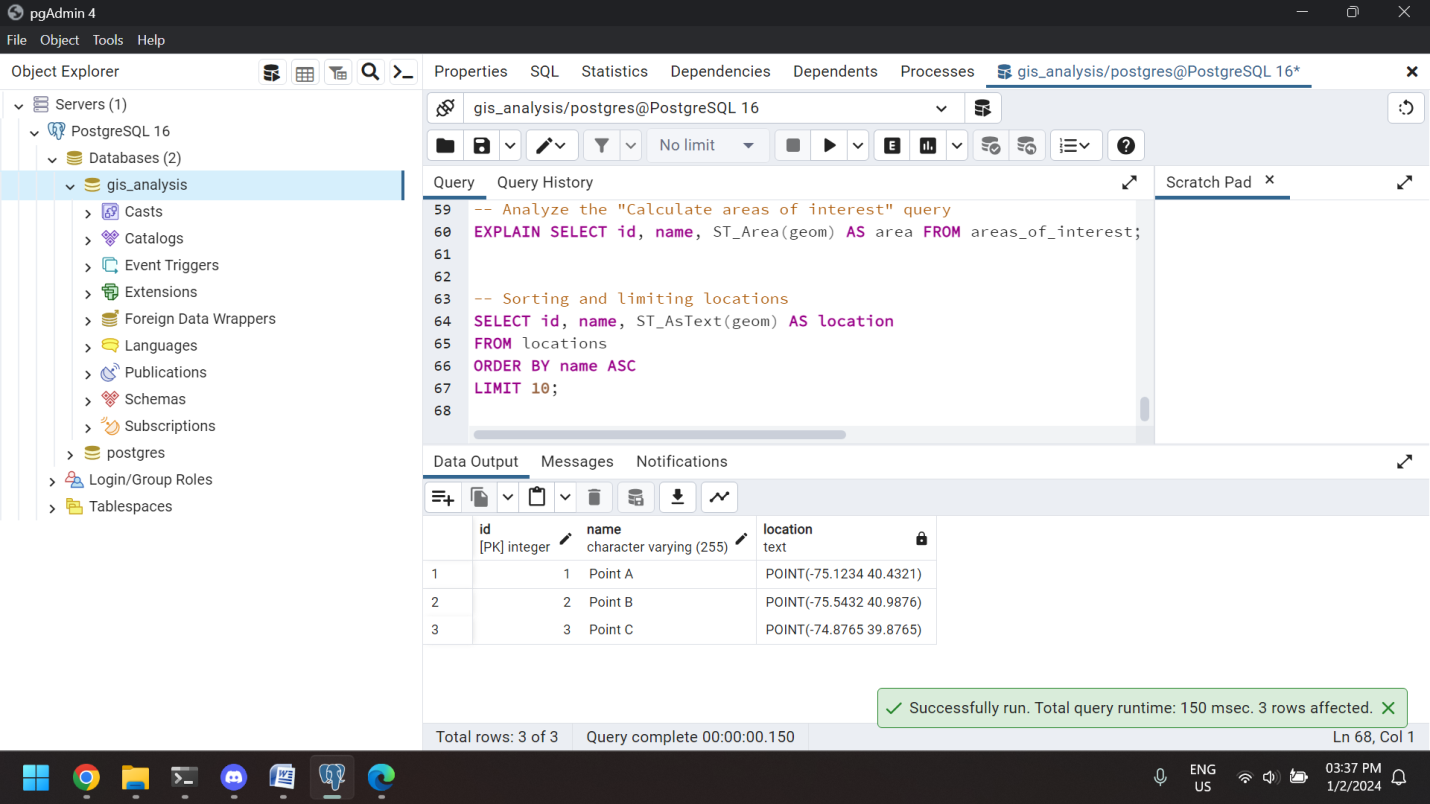
**Question-3:**

****

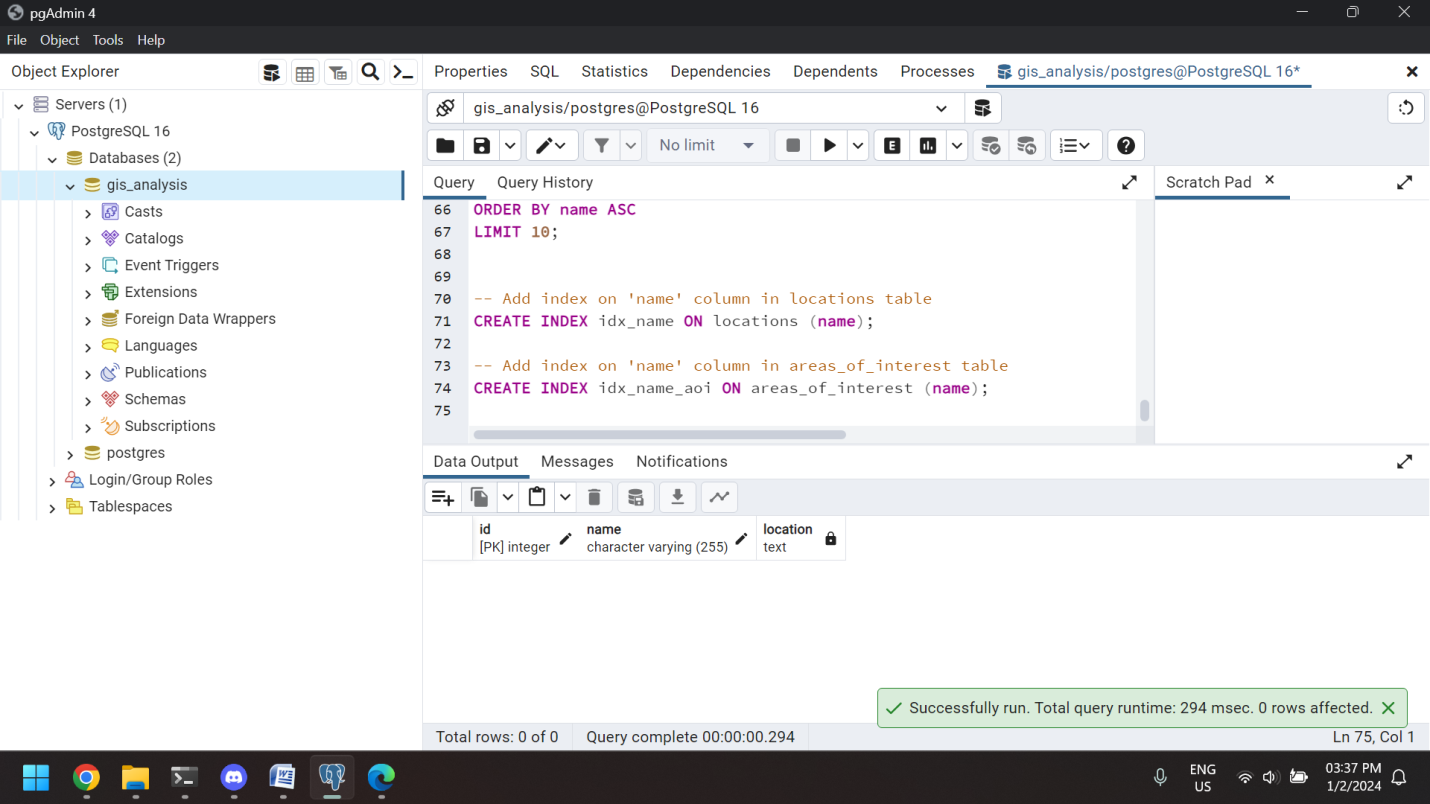
**Question-4:**

****

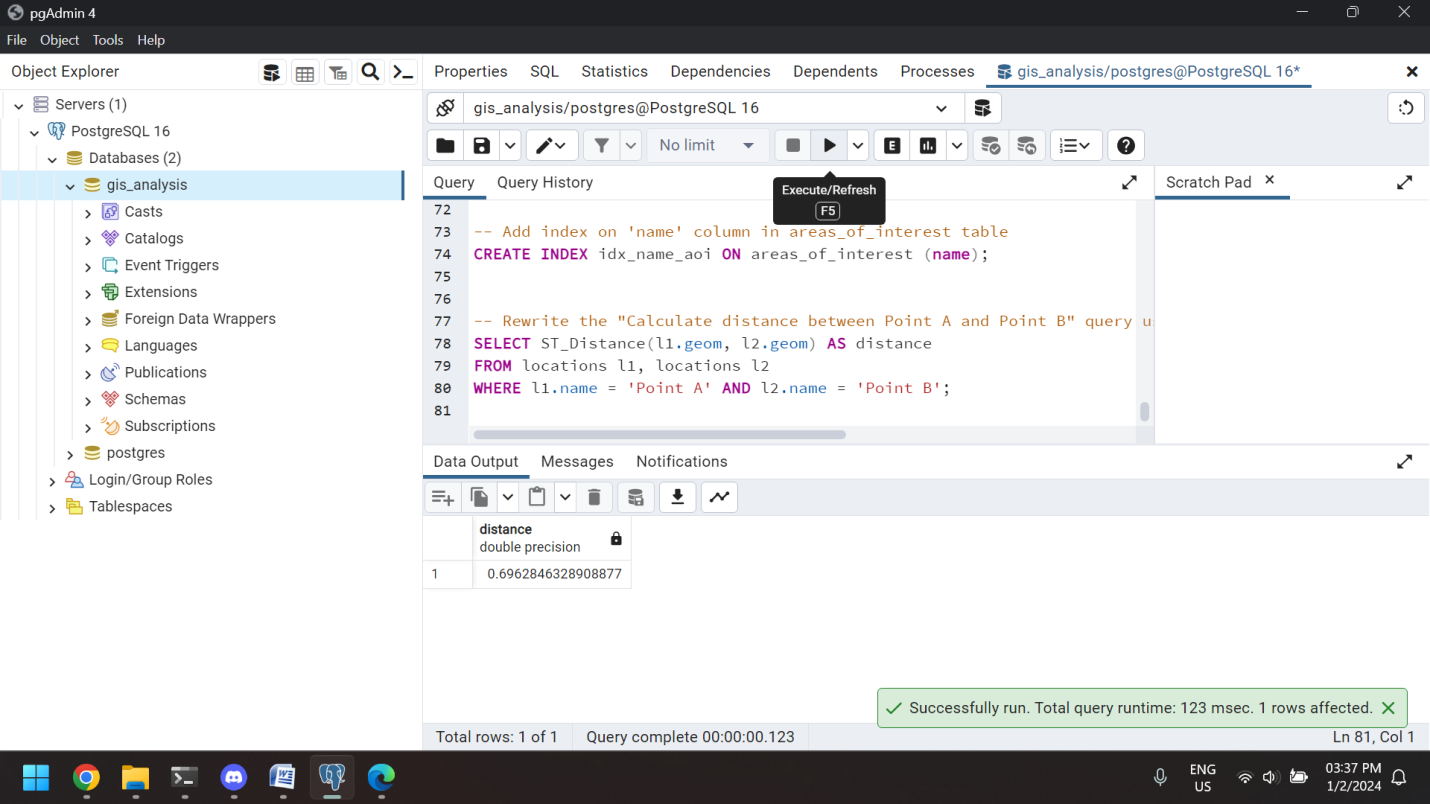
**Question-5:**

****

**Question-6:**

****

**Question-7:**

****

**1. External Merge Sort:**

**Using the number of passes formula in external merge sort**

**\( \lceil \log\_{B1} N \rceil \), where \( N = 1,000,000 \) and \( B = 6 \) (number of buffers).**

**Substitute the values and find the number of passes required.**

**2. B+Tree:**

**To discover all keys between \( 9^\* \) and \( 19^\* \), chase one pointer (sibling-to-sibling) in the internal node where the range begins, and another for each leaf node in the range.**

**3. Hash Table:**

**Determine the largest key less than 25 whose entry will cause a split in the hash table using the hash functions provided and the bucket split condition.**

**4. Sparse B+Tree:**

**Calculate the number of nodes in a sparse B+ tree with order \(d = 2\) and keys 1 through 20 using the order and number of keys. For a sparse B+ tree, each node can only have one key, and the total number of nodes is \(2 \times 20 = 40\).**

**5. SQL Query Plans:**

**Analyze the given relational algebraic expressions and decide which of the two logical plans is more likely to be efficient for the current SQL query.**

**6. Vectorized Processing Model:**

**When receiving input from numerous children, determine if each operator in the vectorized processing model requires multithreaded execution. Give a True/False response with a concise explanation.**

**7. Hash Join Optimization:**

**Use a solid hash algorithm, evaluate hash table size, and reduce the number of collisions to optimize hash join. Techniques include selecting the appropriate hash algorithm and modifying the size of the hash table.**

**8. Query Cost Calculation:**

**Count the number of page I/Os to determine the cost of the query plan. This entails predicting the number of page reads and writes for each plan operation.**

**9. Join I/O Costs:**

**Calculate the cost of I/O for a block nested loop join. The cost includes reading blocks from the outer and inner relations and writing the results.**

**10. Full Binary Tree:**

**A full binary tree with \(2n\) internal nodes has \(n + 1\) leaf nodes.**

**12. Cuckoo Hashing Sequence:**

**Analyze the Cuckoo hashing schema and forecast the sequence produced based on the hashing functions and insertion order provided.**