Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

NAME : PRACHI PATEL
SAP ID: 60004200049
BRANCH : COMPUTER ENGINEERING(A2)
SUBJECT : Advanced DataBase Management System

EXPERIMENT LIST

EXPERIMENT - 1

Case Study on Professional & Commercial Database

29/9/22

NAME - PRACHI PATEL
SAP ID - 60004200049
BRANCH - Computer Engineering.

24/25

Experiment 1 - Case Study

AIM :- Case study on professional and commercial database

Database used - Scylla DB

ScyllaDB is an open source distributed NOSQL wide column data store. It supports the same protocol as Cassandra and the same file formats (SSTable), but it is completely rewritten implementation using C++ language. Scylla has a ring-type architecture.
It is distributed highly available, high performance, low maintenance, highly scalable NOSQL database. All nodes are created equal, there are no master/slave nodes. Data is automatically distributed and replicate on the cluster according to the replication strategy. Scylla supports multiple data centres.
Ola primarily chose ScyllaDB as it supported for high throughput, low latency and high availability. The most fascinating feature was stability of database. It provided both read and write feature which was an improved feature to modify the data model of Cassandra architecture. Two workloads are handled by Scylla - First being real-time reads at high throughput along with batch writes from ML pipelines. Second one is uniform mix of reads and writes.

Company - OLA

Features :-
- Data and metadata storage
- good scaling operations
- integrated article section for driver reviews
- multiple payment options (transactions)
- it provided native metrics monitoring support for Prometheus.
-

Proposed database :- MongoDB
After going through the options one option was to use cloud storage and replace scyllaDB but then even MongoDB looked a better option in managerial as well as cost capacity.
Thus MongoDB is an open source document database and leading NOSQL database. MongoDB is written in C++. It is a non-relational document database that provides support for JSON storage. It has a flexible datamodel that allows you to have objects in one collection with different sets of fields. It enables you to store unstructured data and provides full indexing support and replication with rich and intuitive APIs. One can adjust cluster to automatically scale when needed. This way the cost is minimized, while still having the flexibility to handle sudden traffic bursts.
Mongo DB cloud database Atlas is a powerful and compelling alternative to managing your own NOSQL or traditional database. It has fully managed database services through our choice of cloud provider. It also provides MongoDB query API that allows you to query deep into documents and even perform complex analytics pipelines.

Comparison :-

Given that we are a company that needs better transaction speed and unstructured data, shifting to MongoDB would be an excellent option for us.

1) On comparing the security features we found that both are at par with each other on it.

2) MongoDB has two roles primary and secondary as they use primary for read and write while secondary for read only over scylla DB that gives all operations to the nodes. This feature adds an additional security to ~~mog~~ Mongo DB.

3) On scaling point of view MongoDB clusters replica sets and uses smart routing instances whereas in scylla DB we need to keep adding on nodes

4) As mentioned earlier that it is a document based database so it has flexible collections as uses json and also has frictionless and rich queries while scylla has table schema and does has performance ~~gua~~ guaranteed queries but with many limitations.

5) MongoDB is a completely index based document whereas in scylla db the metadata is in rows and columns.

6) Scylla does have a few external data visualization MongoDB has internal tool called atlas which has made considerable strides in

helping the company visualize data.

7) Additionally with data visualization MongoDB (atlas clusters) is compatible with all languages python, go, dart, swift etc thus giving an edge over scylla db.

8) MongoDB has its own lightweight reactive object store designed to work seamlessly with mobile applications whereas scylla doesn't have any specific mobile oriented driver or sdk.

9) On transactional point of view mongodb has multi-document ACID Transactions with snapshot and isolation whereas scylla db doesn't have such features.

Recommendation :-

Scylla should provide custom map/reduce implementation for more flexibility and less query limitations. Both ways are good enough for the company but I would suggest a shift to MongoDB as they have many features that have outperformed those of scylla DB and mongo DB as a whole has been able to stay up with the current trends in market and provides best of technologies. Mongo DB Atlas can be used with any of leading clouds service providers.

Conclusion :-

We have ~~adopted~~ compared both the database and we have a clear view on choosing mongodb as our data is not needed to be structured and needs more flexibility and mongodb provides with better features for mobile integration as well as more ML model integration along with providing better visualization and multi-document ACID transactions.

EXPERIMENT - 2

Optimize using B and B+ trees

7/10/22

24/25

NAME - PRACHI PATEL
SAP ID - 60004200049
BRANCH - Computer Engineering A2

Experiment - 2 :-

AIM = Optimize using B and B+ trees.

THEORY -

1) B Tree :-

B tree is a self balancing search tree. B Tree is a specialized m-way tree that can be widely used for disk access. A B-Tree of order m can have at most m-1 keys and m children. One of the main reasons for using B tree is its capability to store a large number of keys in a single node and large key values by keeping the height of the tree relatively small.

A B-tree of order m contains all the following properties :-

1) Every node in a B-tree contains at most m children
2) Every node in a B-tree except the root node and leaf node contains at least m/2 children.
3) The root nodes must have atleast 2 nodes.
4) All leaf nodes must be at the same level.

It is not necessary that all the nodes contain the same number of children but, each other must have m/2 number of nodes.

	Root	Intermidate Node
Max children	P	P
Min children	2	[P/2]
Max key	P-1	P-1
Min key	1	[P/2] -1

Time Complexity:-
Search Time complexity - O(logn)
Insertion Time complexity - O(logn)
Deletion Time complexity - O(logn)

Application of B-tree :-
1) It is used in large databases to access data stored on the disk.
2) Searching for data in a dataset can be achieved in significantly less time using the B-tree

B+ tree :-
It is an extension of B tree which allows efficient insertion, deletion & search operations. In B tree keys and records both can be stored in internal as well as leaf nodes. Whereas, in B+ tree, records (data) can only be stored on the leaf node of a B+ tree while internal nodes can store key values. The leaf node of a B+ tree are linked together in form of singly linked lists to make the search queries more efficient. B+ tree are used to store a large amount of data which cannot be stored in main memory. Due to the fact that size of main memory is always limited, the internal nodes of B+ tree are stored in main memory whereas leaf nodes are stored in secondary memory.

Advantages of B+ tree :-
1) Records can be fetched in equal number of disk access.
2) Height of tree remains balanced and less as compared to B-tree
3) We can access the data stored in a B+ tree sequentially as well as directly.

	ROOT	Internal Node	Leaf Node
Max children	m	m	n
Min ~~Max~~ children	2	$\lceil m/2 \rceil$	$\lceil n/2 \rceil$
Max key	m-1	m-1	n
Min key	1	$\lceil m/2 \rceil - 1$	$\lceil n/2 \rceil - 1$

CODE :

1) B TREE

```
from datetime import  datetime
class BTreeNode:
  def __init__(self, leaf=False):
        self.leaf = leaf
        self.keys = []
        self.child = []

# Tree
class BTree:
  def __init__(self, t):
        self.root = BTreeNode(True)
        self.t = t

        # Insert node
  def insert(self, k):
        root = self.root
        if len(root.keys) == (2 * self.t) - 1:
        temp = BTreeNode()
        self.root = temp
        temp.child.insert(0, root)
        self.split_child(temp, 0)
        self.insert_non_full(temp, k)
        else:
        self.insert_non_full(root, k)

        # Insert nonfull
  def insert_non_full(self, x, k):
        i = len(x.keys) - 1
        if x.leaf:
        x.keys.append((None, None))
        while i >= 0 and k[0] < x.keys[i][0]:
```

```
        x.keys[i + 1] = x.keys[i]
        i -= 1
        x.keys[i + 1] = k
        else:
        while i >= 0 and k[0] < x.keys[i][0]:
        i -= 1
        i += 1
        if len(x.child[i].keys) == (2 * self.t) - 1:
        self.split_child(x, i)
        if k[0] > x.keys[i][0]:
        i += 1
        self.insert_non_full(x.child[i], k)

        # Split the child
def split_child(self, x, i):
        t = self.t
        y = x.child[i]
        z = BTreeNode(y.leaf)
        x.child.insert(i + 1, z)
        x.keys.insert(i, y.keys[t - 1])
        z.keys = y.keys[t: (2 * t) - 1]
        y.keys = y.keys[0: t - 1]
        if not y.leaf:
        z.child = y.child[t: 2 * t]
        y.child = y.child[0: t - 1]

# Print the tree
def print_tree(self, x, l=0):
        print("Level ", l, " ", len(x.keys), end=":")
        for i in x.keys:
        print(i, end=" ")
        print()
        l += 1
        if len(x.child) > 0:
        for i in x.child:
        self.print_tree(i, l)
```

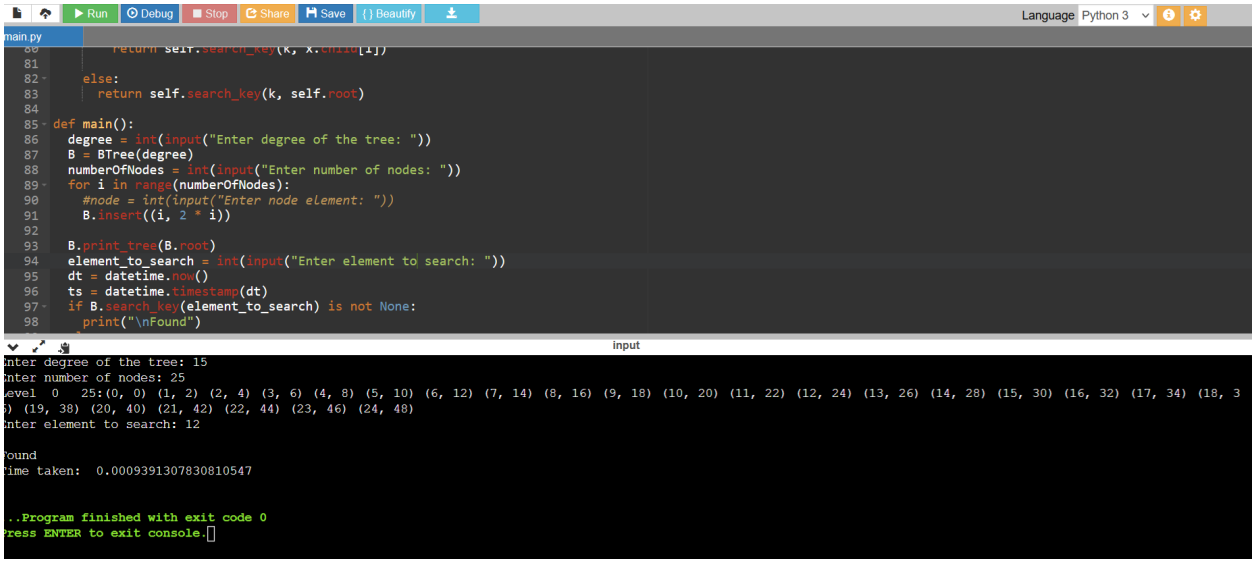
```
  # Search key in the tree
  def search_key(self, k, x=None):
        if x is not None:
        i = 0
        while i < len(x.keys) and k > x.keys[i][0]:
        i += 1
        if i < len(x.keys) and k == x.keys[i][0]:
        return (x, i)
        elif x.leaf:
        return None
        else:
        return self.search_key(k, x.child[i])

        else:
        return self.search_key(k, self.root)

def main():
  degree = int(input("Enter degree of the tree: "))
  B = BTree(degree)
  numberOfNodes = int(input("Enter number of nodes: "))
  for i in range(numberOfNodes):
        #node = int(input("Enter node element: "))
        B.insert((i, 2 * i))

  B.print_tree(B.root)
  element_to_search = int(input("Enter element to search: "))
  dt = datetime.now()
  ts = datetime.timestamp(dt)
  if B.search_key(element_to_search) is not None:
        print("\nFound")
  else:
        print("\nNot Found")
  dt2 = datetime.now()
  ts2 = datetime.timestamp(dt2)
  print("Time taken: ",(ts2-ts))
```

```
if __name__ == '__main__':
  main()
```



2) B+ TREE

```
# B+ tee in python
import math
from datetime import  datetime
# Node creation
class Node:
	def __init__(self, order):
	self.order = order
	self.values = []
	self.keys = []
	self.nextKey = None
	self.parent = None
	self.check_leaf = False

	# Insert at the leaf
	def insert_at_leaf(self, leaf, value, key):
	if (self.values):
	temp1 = self.values
	for i in range(len(temp1)):
		if (value == temp1[i]):
		self.keys[i].append(key)
		break
		elif (value < temp1[i]):
		self.values = self.values[:i] + [value] + self.values[i:]
		self.keys = self.keys[:i] + [[key]] + self.keys[i:]
		break
		elif (i + 1 == len(temp1)):
		self.values.append(value)
		self.keys.append([key])
		break
	else:
	self.values = [value]
	self.keys = [[key]]
```

```
# B plus tree
class BplusTree:
	def __init__(self, order):
	self.root = Node(order)
	self.root.check_leaf = True

	# Insert operation
	def insert(self, value, key):
	value = str(value)
	old_node = self.search(value)
	old_node.insert_at_leaf(old_node, value, key)

	if (len(old_node.values) == old_node.order):
	node1 = Node(old_node.order)
	node1.check_leaf = True
	node1.parent = old_node.parent
	mid = int(math.ceil(old_node.order / 2)) - 1
	node1.values = old_node.values[mid + 1:]
	node1.keys = old_node.keys[mid + 1:]
	node1.nextKey = old_node.nextKey
	old_node.values = old_node.values[:mid + 1]
	old_node.keys = old_node.keys[:mid + 1]
	old_node.nextKey = node1
	self.insert_in_parent(old_node, node1.values[0], node1)

	# Search operation for different operations
	def search(self, value):
	current_node = self.root
	while(current_node.check_leaf == False):
	temp2 = current_node.values
	for i in range(len(temp2)):
		if (value == temp2[i]):
		current_node = current_node.keys[i + 1]
		break
		elif (value < temp2[i]):
		current_node = current_node.keys[i]
```

```
            break
            elif (i + 1 == len(current_node.values)):
            current_node = current_node.keys[i + 1]
            break
    return current_node

    # Find the node
    def find(self, value, key):
    l = self.search(value)
    for i, item in enumerate(l.values):
    if item == value:
            if key in l.keys[i]:
            return True
            else:
            return False
    #return False

    # Inserting at the parent
    def insert_in_parent(self, n, value, ndash):
    if (self.root == n):
    rootNode = Node(n.order)
    rootNode.values = [value]
    rootNode.keys = [n, ndash]
    self.root = rootNode
    n.parent = rootNode
    ndash.parent = rootNode
    return

    parentNode = n.parent
    temp3 = parentNode.keys
    for i in range(len(temp3)):
    if (temp3[i] == n):
            parentNode.values = parentNode.values[:i] + \
            [value] + parentNode.values[i:]
            parentNode.keys = parentNode.keys[:i +
                                    1] + [ndash] + parentNode.keys[i + 1:]
```

```
        if (len(parentNode.keys) > parentNode.order):
        parentdash = Node(parentNode.order)
        parentdash.parent = parentNode.parent
        mid = int(math.ceil(parentNode.order / 2)) - 1
        parentdash.values = parentNode.values[mid + 1:]
        parentdash.keys = parentNode.keys[mid + 1:]
        value_ = parentNode.values[mid]
        if (mid == 0):
        parentNode.values = parentNode.values[:mid + 1]
        else:
        parentNode.values = parentNode.values[:mid]
        parentNode.keys = parentNode.keys[:mid + 1]
        for j in parentNode.keys:
        j.parent = parentNode
        for j in parentdash.keys:
        j.parent = parentdash
        self.insert_in_parent(parentNode, value_, parentdash)

# Delete a node
def delete(self, value, key):
node_ = self.search(value)

temp = 0
for i, item in enumerate(node_.values):
if item == value:
        temp = 1

        if key in node_.keys[i]:
        if len(node_.keys[i]) > 1:
        node_.keys[i].pop(node_.keys[i].index(key))
        elif node_ == self.root:
        node_.values.pop(i)
        node_.keys.pop(i)
        else:
        node_.keys[i].pop(node_.keys[i].index(key))
        del node_.keys[i]
```

```
                node_.values.pop(node_.values.index(value))
                self.deleteEntry(node_, value, key)
                else:
                print("Value not in Key")
                return
        if temp == 0:
        print("Value not in Tree")
        return

        # Delete an entry
        def deleteEntry(self, node_, value, key):

        if not node_.check_leaf:
        for i, item in enumerate(node_.keys):
                if item == key:
                node_.keys.pop(i)
                break
        for i, item in enumerate(node_.values):
                if item == value:
                node_.values.pop(i)
                break

        if self.root == node_ and len(node_.keys) == 1:
        self.root = node_.keys[0]
        node_.keys[0].parent = None
        del node_
        return
        elif (len(node_.keys) < int(math.ceil(node_.order / 2)) and node_.check_leaf
== False) or (len(node_.values) < int(math.ceil((node_.order - 1) / 2)) and
node_.check_leaf == True):

        is_predecessor = 0
        parentNode = node_.parent
        PrevNode = -1
        NextNode = -1
        PrevK = -1
```

```
PostK = -1
for i, item in enumerate(parentNode.keys):

        if item == node_:
        if i > 0:
        PrevNode = parentNode.keys[i - 1]
        PrevK = parentNode.values[i - 1]

        if i < len(parentNode.keys) - 1:
        NextNode = parentNode.keys[i + 1]
        PostK = parentNode.values[i]

if PrevNode == -1:
        ndash = NextNode
        value_ = PostK
elif NextNode == -1:
        is_predecessor = 1
        ndash = PrevNode
        value_ = PrevK
else:
        if len(node_.values) + len(NextNode.values) < node_.order:
        ndash = NextNode
        value_ = PostK
        else:
        is_predecessor = 1
        ndash = PrevNode
        value_ = PrevK

if len(node_.values) + len(ndash.values) < node_.order:
        if is_predecessor == 0:
        node_, ndash = ndash, node_
        ndash.keys += node_.keys
        if not node_.check_leaf:
        ndash.values.append(value_)
        else:
        ndash.nextKey = node_.nextKey
```

```
            ndash.values += node_.values

            if not ndash.check_leaf:
            for j in ndash.keys:
            j.parent = ndash

            self.deleteEntry(node_.parent, value_, node_)
            del node_
    else:
            if is_predecessor == 1:
            if not node_.check_leaf:
            ndashpm = ndash.keys.pop(-1)
            ndashkm_1 = ndash.values.pop(-1)
            node_.keys = [ndashpm] + node_.keys
            node_.values = [value_] + node_.values
            parentNode = node_.parent
            for i, item in enumerate(parentNode.values):
                    if item == value_:
                    p.values[i] = ndashkm_1
                    break
            else:
            ndashpm = ndash.keys.pop(-1)
            ndashkm = ndash.values.pop(-1)
            node_.keys = [ndashpm] + node_.keys
            node_.values = [ndashkm] + node_.values
            parentNode = node_.parent
            for i, item in enumerate(p.values):
                    if item == value_:
                    parentNode.values[i] = ndashkm
                    break
            else:
            if not node_.check_leaf:
            ndashp0 = ndash.keys.pop(0)
            ndashk0 = ndash.values.pop(0)
            node_.keys = node_.keys + [ndashp0]
            node_.values = node_.values + [value_]
```

```
                parentNode = node_.parent
                for i, item in enumerate(parentNode.values):
                        if item == value_:
                        parentNode.values[i] = ndashk0
                        break
                else:
                ndashp0 = ndash.keys.pop(0)
                ndashk0 = ndash.values.pop(0)
                node_.keys = node_.keys + [ndashp0]
                node_.values = node_.values + [ndashk0]
                parentNode = node_.parent
                for i, item in enumerate(parentNode.values):
                        if item == value_:
                        parentNode.values[i] = ndash.values[0]
                        break

                if not ndash.check_leaf:
                for j in ndash.keys:
                j.parent = ndash
                if not node_.check_leaf:
                for j in node_.keys:
                j.parent = node_
                if not parentNode.check_leaf:
                for j in parentNode.keys:
                j.parent = parentNode


# Print the tree
def printTree(tree):
        lst = [tree.root]
        level = [0]
        leaf = None
        flag = 0
        lev_leaf = 0

        node1 = Node(str(level[0]) + str(tree.root.values))
```

```
        while (len(lst) != 0):
        x = lst.pop(0)
        lev = level.pop(0)
        if (x.check_leaf == False):
        for i, item in enumerate(x.keys):
                print(item.values)
        else:
        for i, item in enumerate(x.keys):
                print(item.values)
        if (flag == 0):
                lev_leaf = lev
                leaf = x
                flag = 1


n = int(input("Enter number of nodes:"))
bplustree = BplusTree(n)
for i in range(n):
        bplustree.insert(str(i), str(2 * i))
printTree(bplustree)
print("Enter the key and value to search:")

key = input()
value = input()
dt = datetime.now()
ts = datetime.timestamp(dt)
# print(ts)
if(bplustree.find(key, value)):
        print("Found")
else:
        print("Not found")
dt2 = datetime.now()
ts2 = datetime.timestamp(dt2)
# print(ts2)
```

print("Time required:", ts2 - ts)

Run Debug Stop Share Save { } Beautify Language Python 3

main.py B+ tree

```
# B+ tee in python
import math
from datetime import  datetime
# Node creation
class Node:
    def __init__(self, order):
        self.order = order
        self.values = []
        self.keys = []
        self.nextKey = None
        self.parent = None
        self.check_leaf = False

    # Insert at the leaf
```

input

```
Enter number of nodes:15
['0', '1', '10', '11', '12', '13', '14', '2']
['3', '4', '5', '6', '7', '8', '9']
Enter the key and value to search:
12
5
Not found
Time required: 7.081031799316406e-05


...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion :-

B tree is a self balancing tree and aids sorting while allowing searching, insertion, deletion and sequential access. B+ tree aids in allowing inherent problems with B tree. For B tree we used and had search time of 0.00093 seconds while using B+ tree we got search time as 7.08 milliseconds

EXPERIMENT - 3

Simulate Query optimization

21/10/22

24
25

NAME - PRACHI PATEL
SAP ID - 60004200049
BRANCH - Computer Engineering

Experiment - 3 :-

AIM :- To simulate query optimisation by performing a SQL query on database

Theory :-

Query optimization is of great importance for the performance of a realtime / rational database. especially for the execution of complex SQL statements.

There are 2 ways :-

1) Heuristic Based
2) Cost Based.

Heuristic Based :-

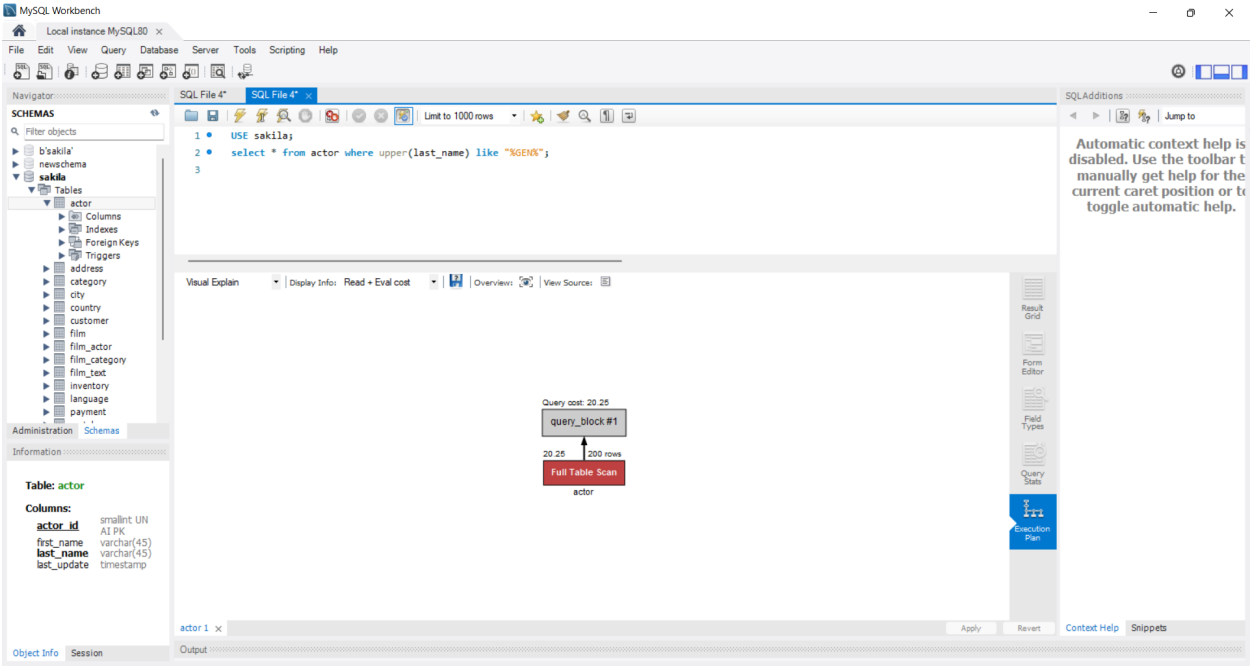
A query tree is a data structure that corresponds to a relational algebra expression. The same query could respond to many different relational algebra expressions & hence many different query trees. The task of heuristic optimization of query trees is to find a final query tree that is efficient to execute. The main heuristic is to apply first the operations that reduce the size of intermidate results. Performs the selection process foremost in the query

Cost based optimisation:-

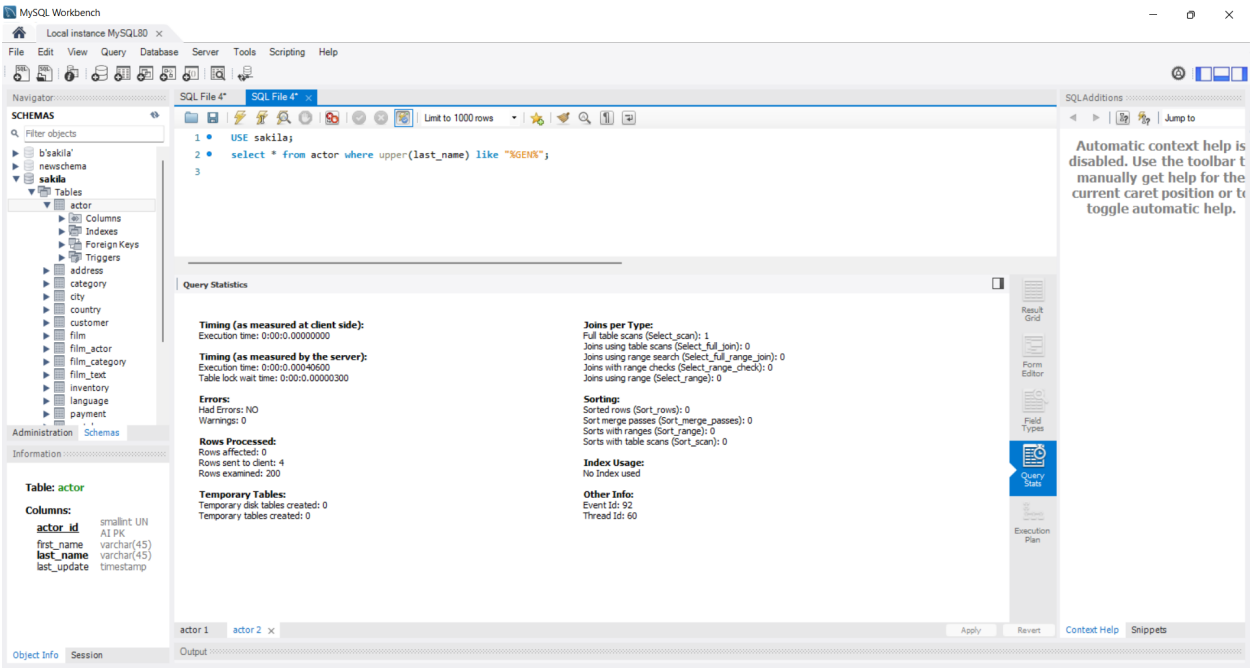
Estimate and compare the costs of executing a query using different execution strategies and chose the strategy with the lowest cost estimate. The cost of any query includes access cost to secondary storage, storage cost, computation cost, memory usage, cost, no of memory buffer at time of execution, communication cost. The optimizer allocates a cost is numerical form for each step of a feasible plan for a given query and environment and then discovers these values together to get a cost estimate for the plan or possible strategy.

TABLE LEVEL OPTIMIZATION

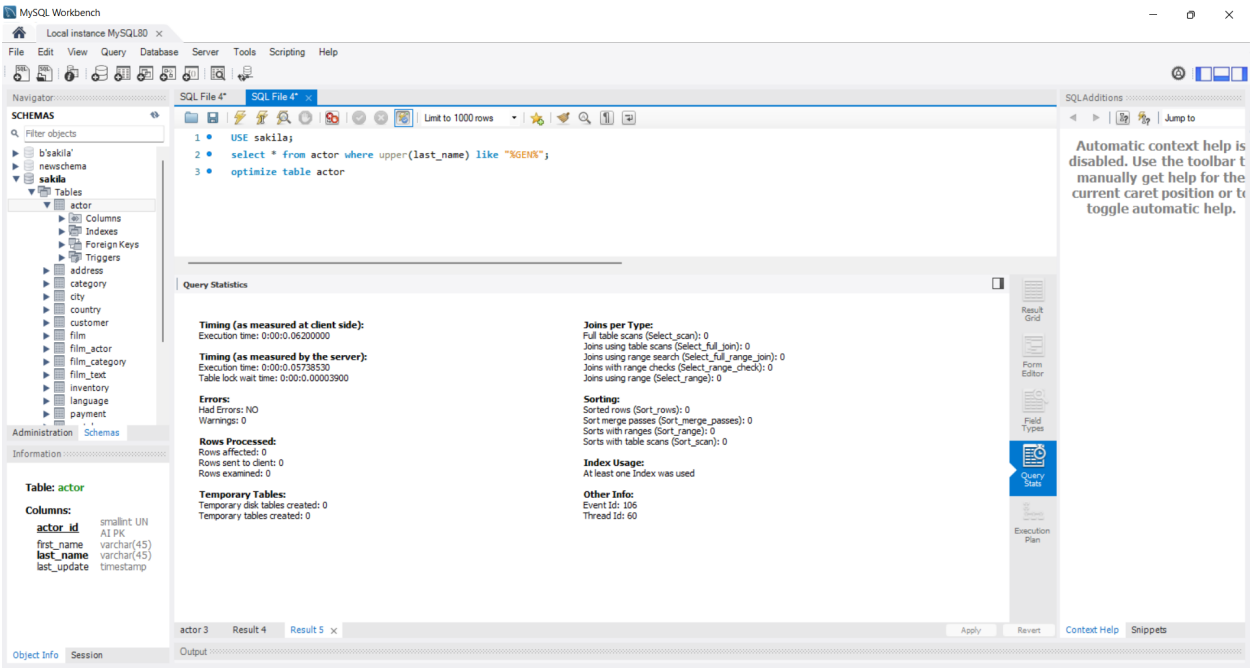
1) SELECT QUERY



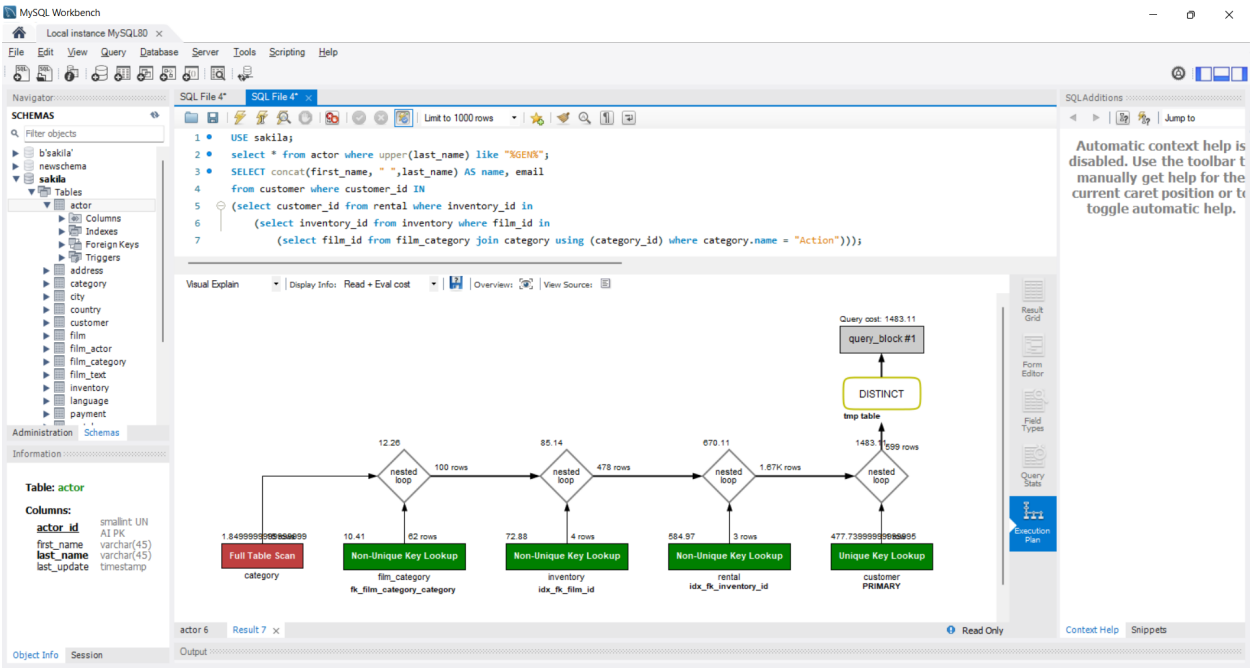
BEFORE OPTIMIZING



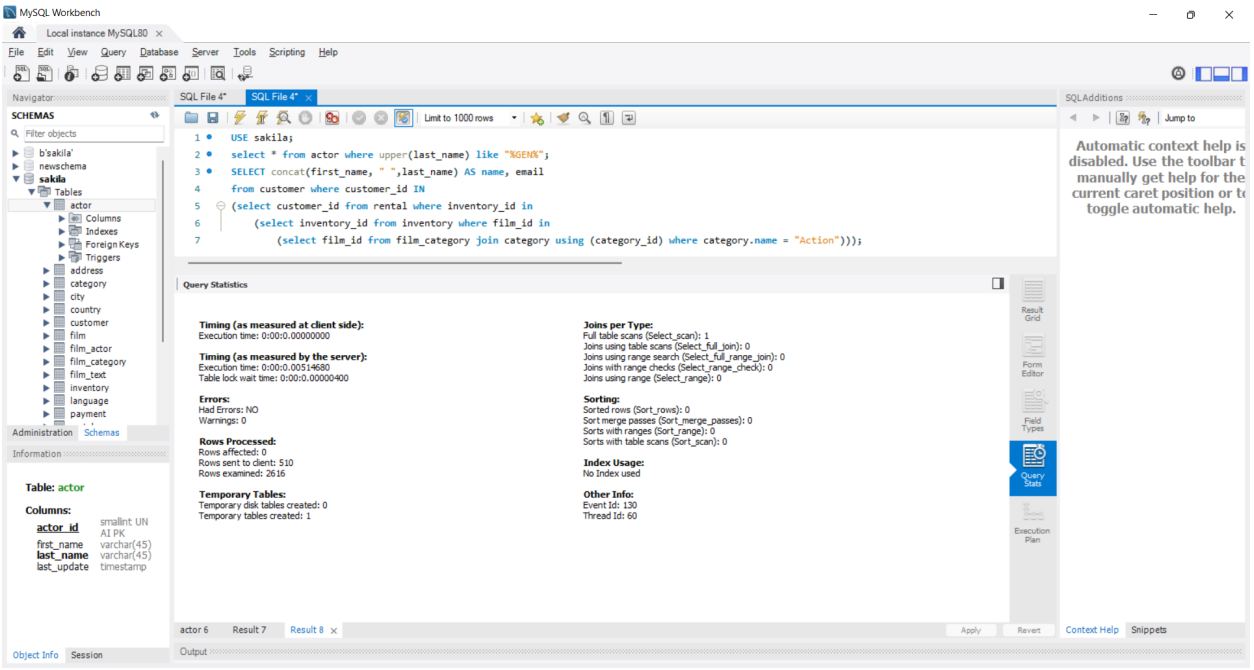
AFTER OPTIMIZING



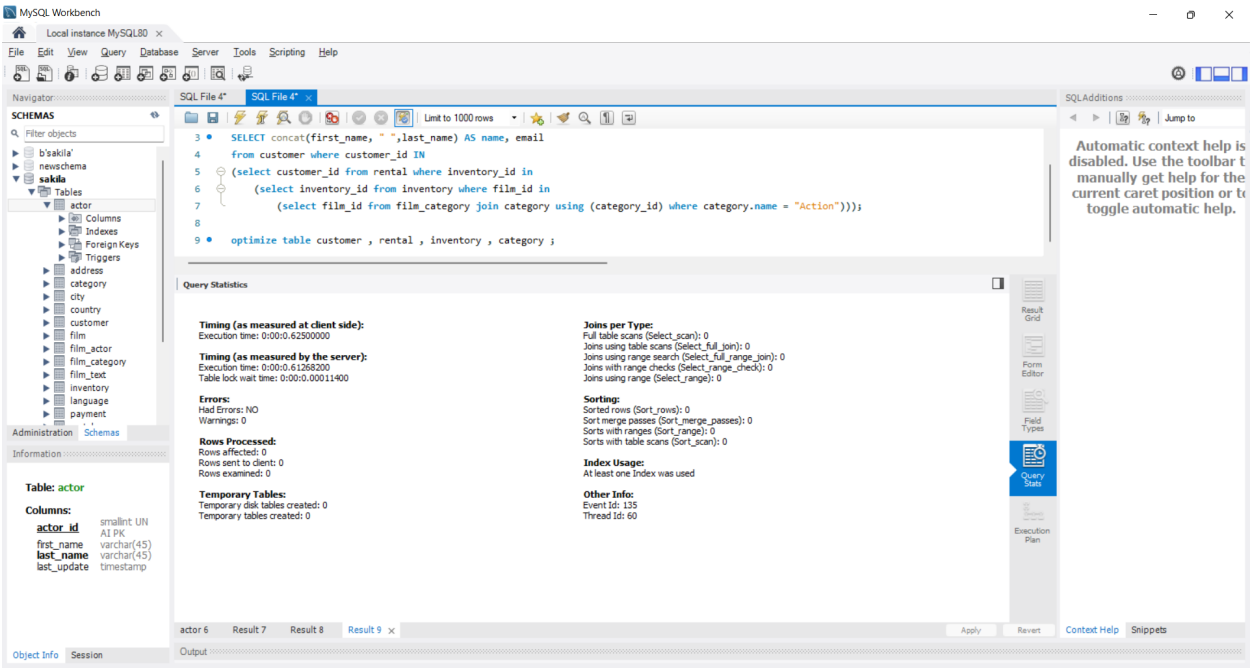
2) NESTED



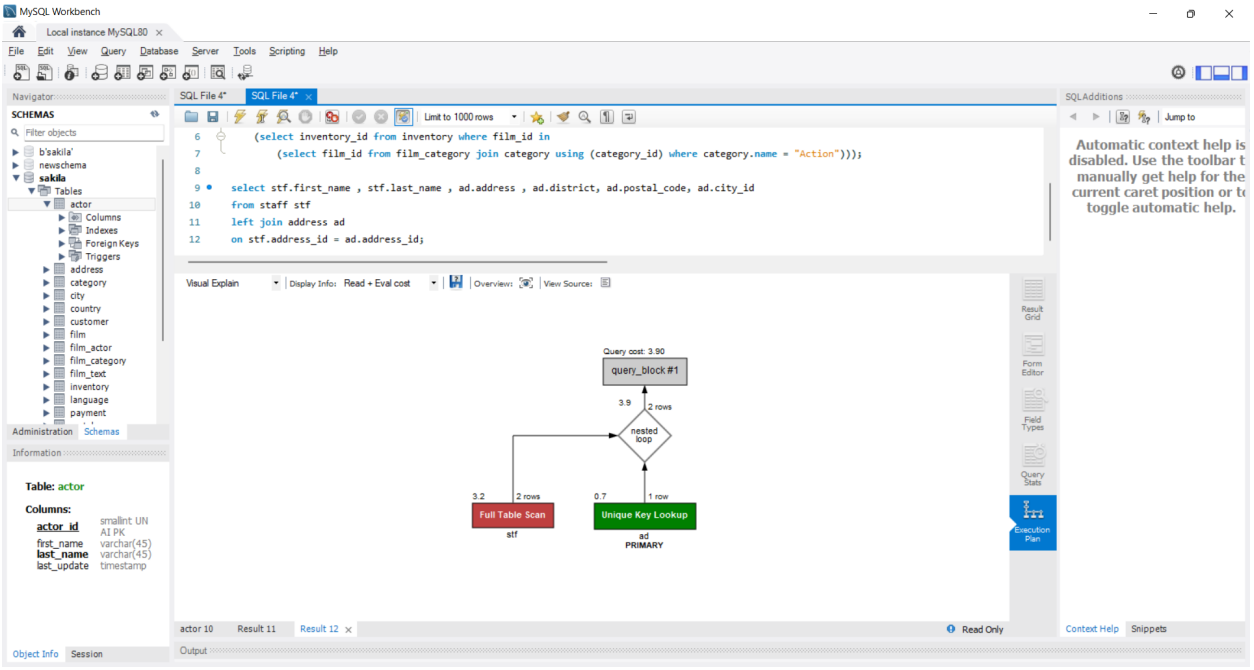
BEFORE OPTIMIZING



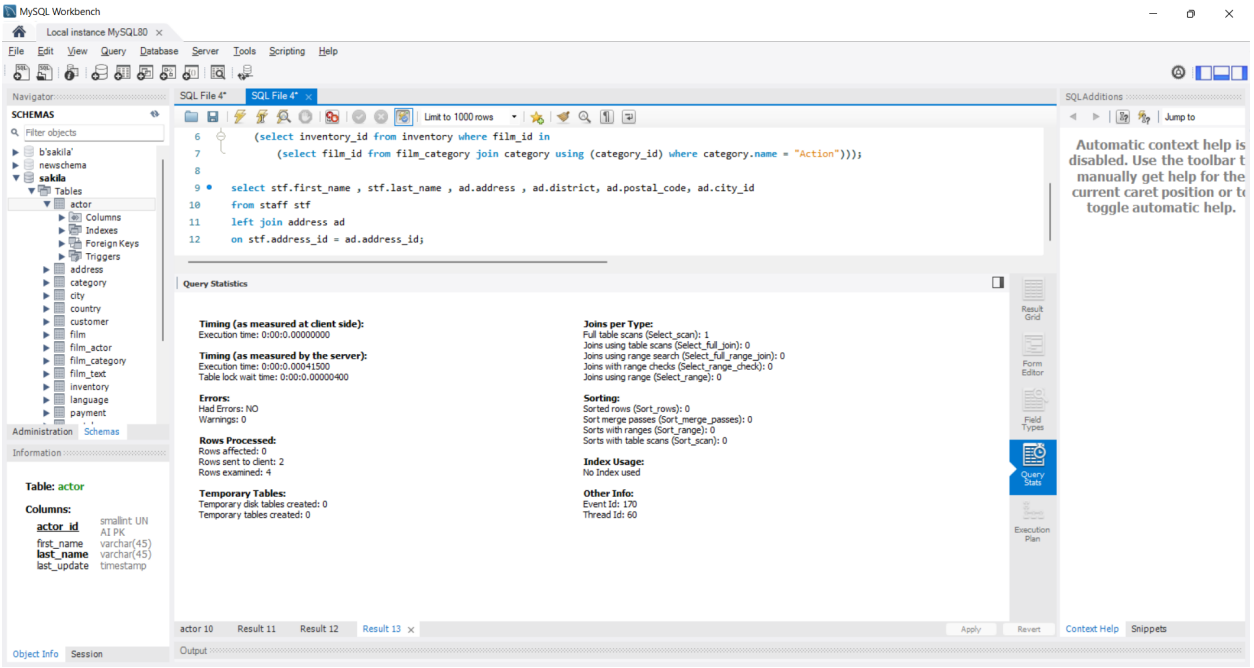
AFTER OPTIMIZING



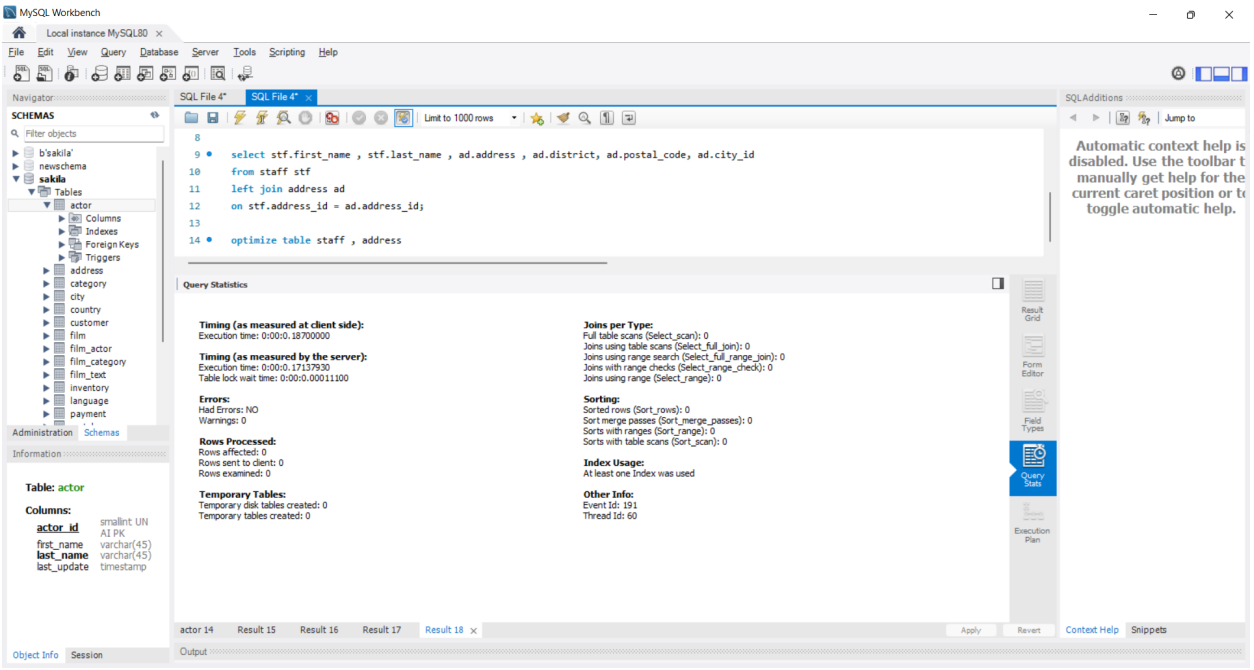
3) LEFT JOIN



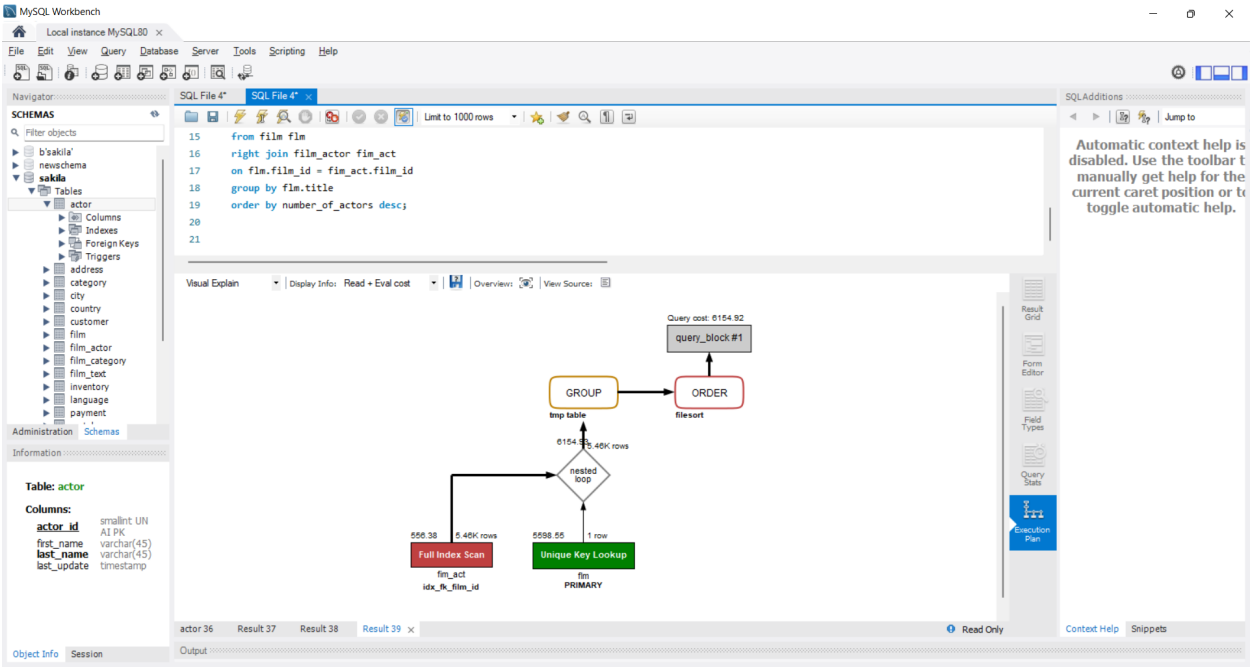
BEFORE OPTIMIZING



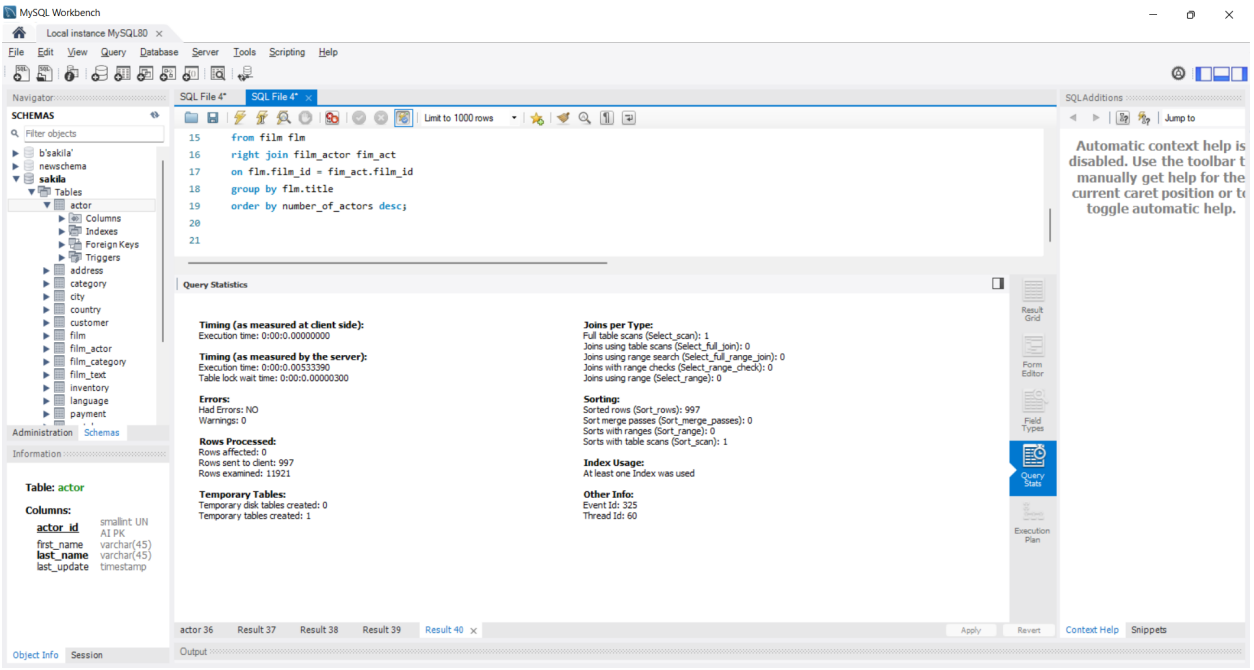
AFTER OPTIMIZING



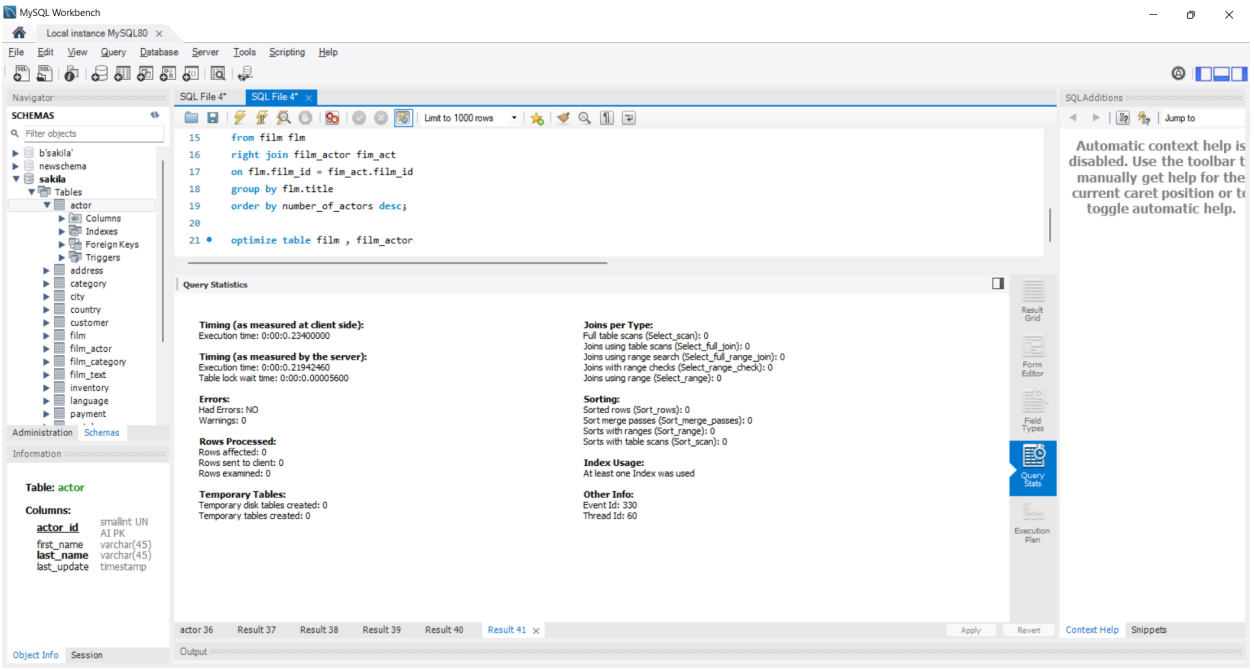
4) RIGHT JOIN



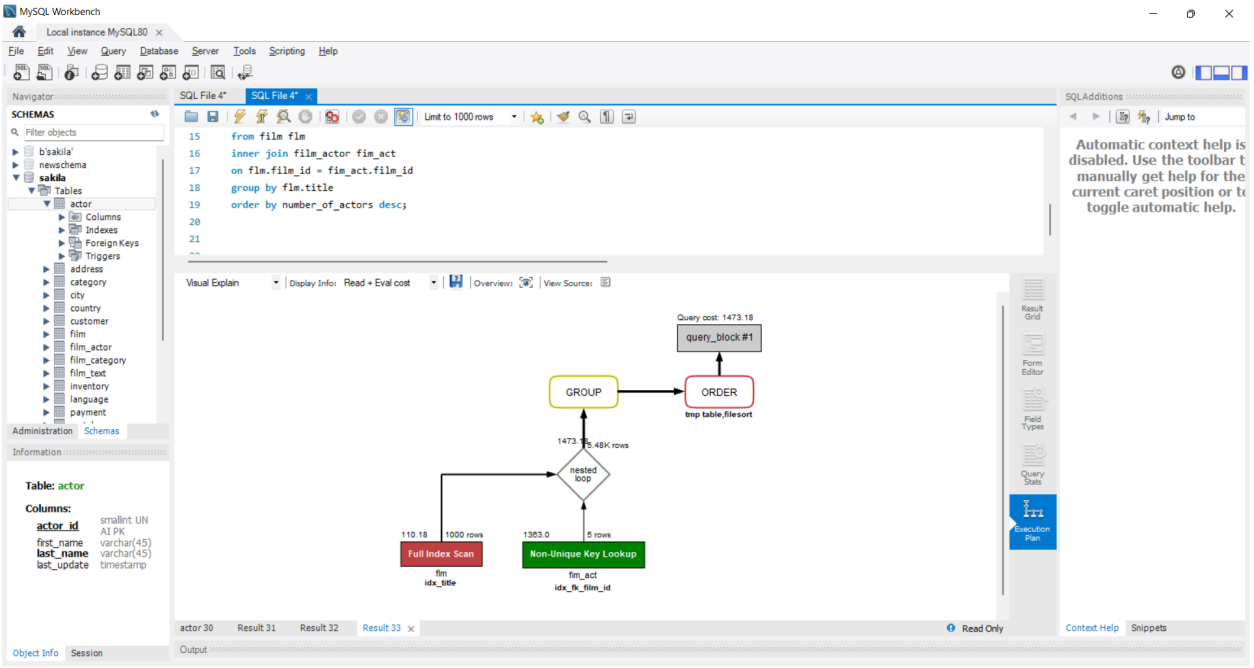
BEFORE OPTIMIZING



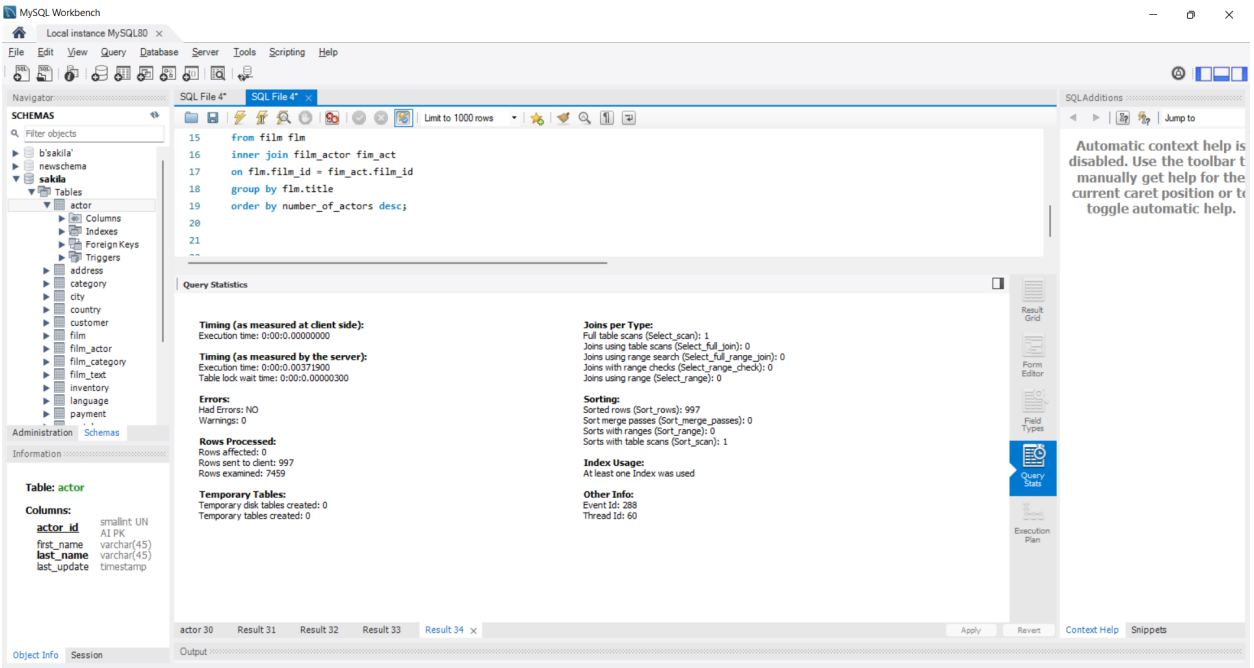
AFTER OPTIMIZING



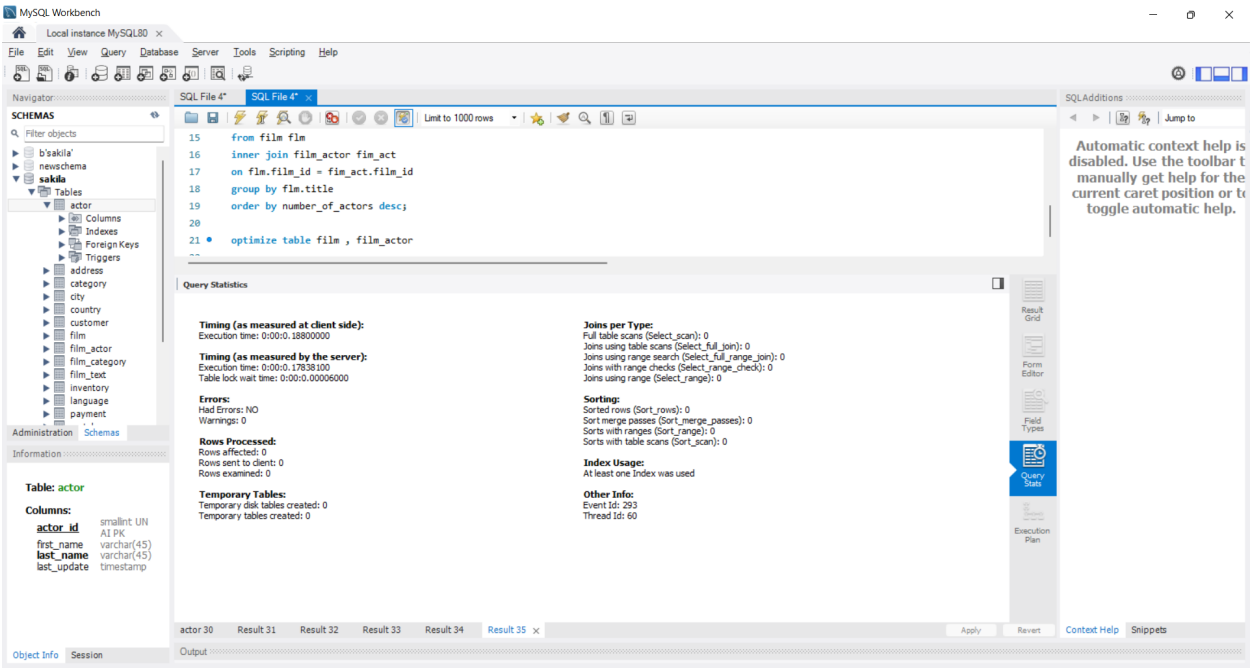
5) INNER JOIN



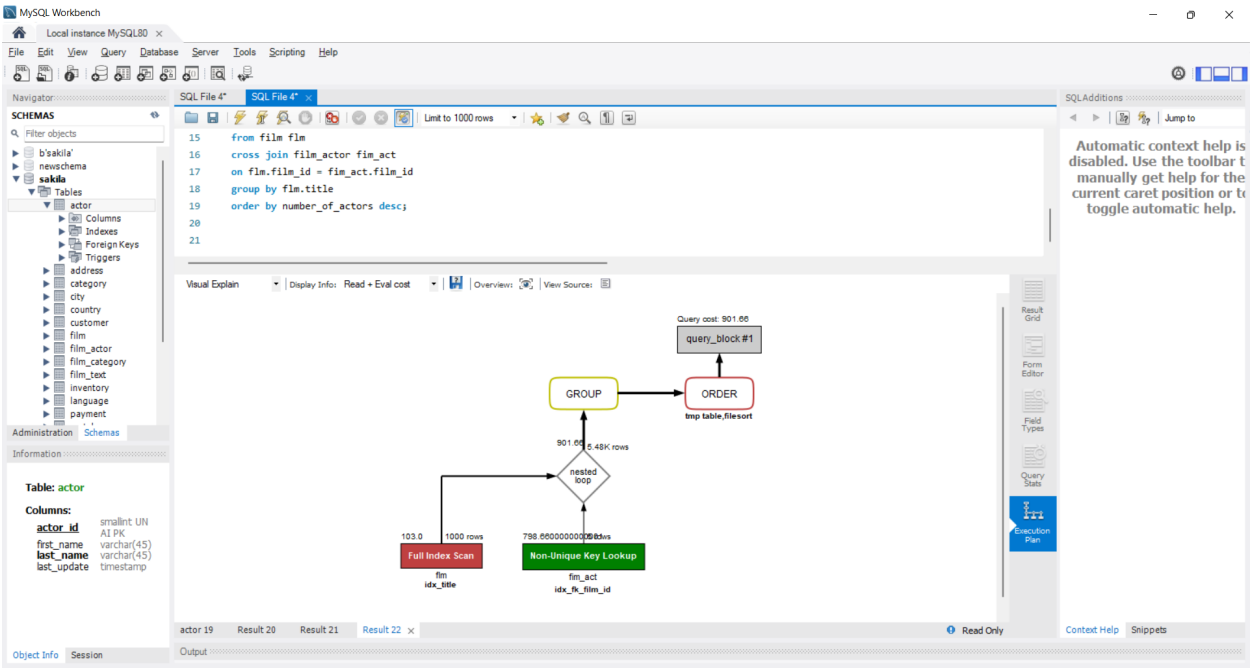
BEFORE OPTIMIZING



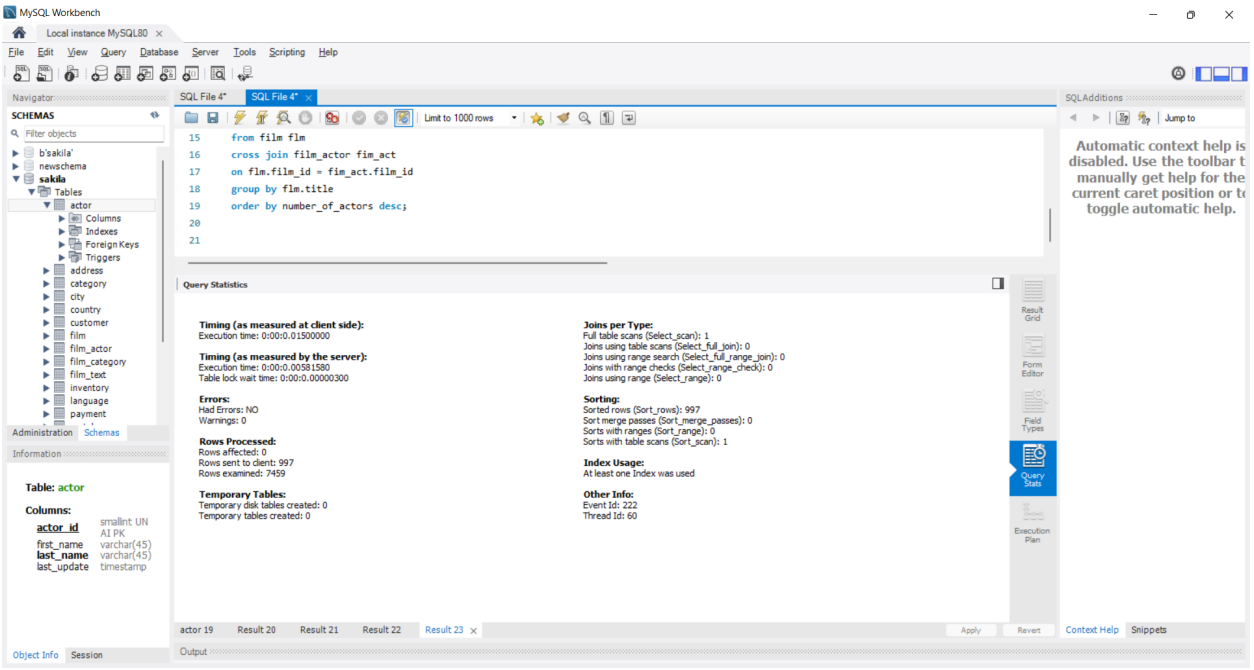
AFTER OPTIMIZING



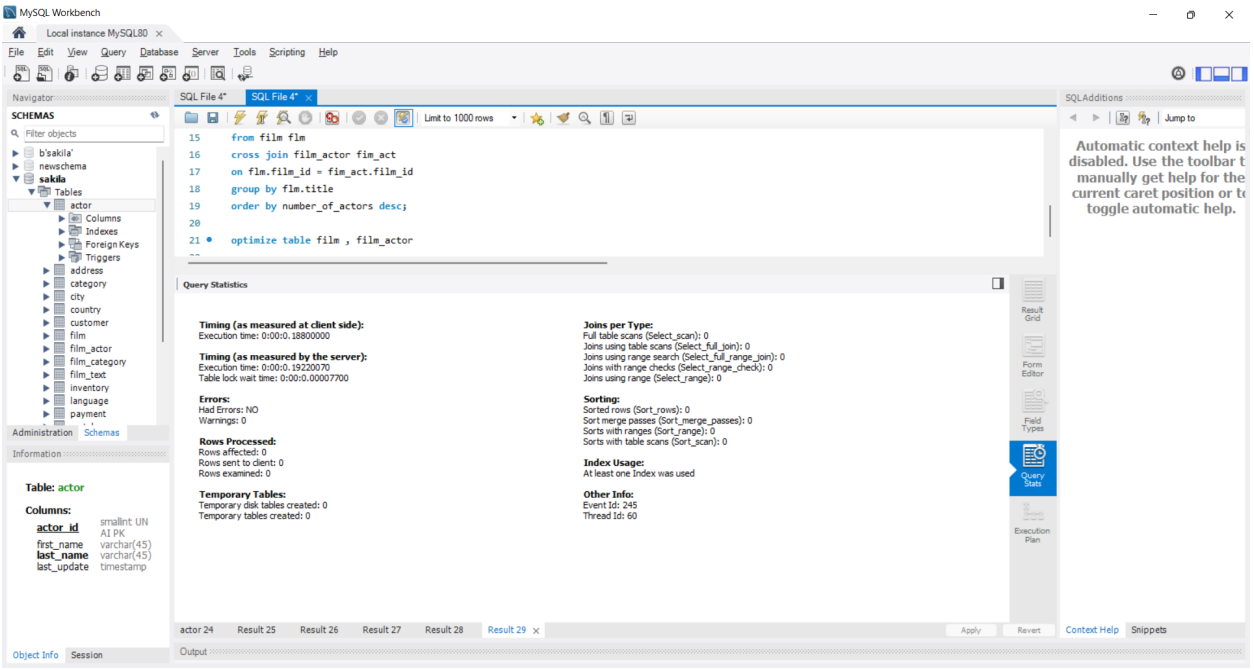
6) CROSS JOIN



BEFORE OPTIMIZING

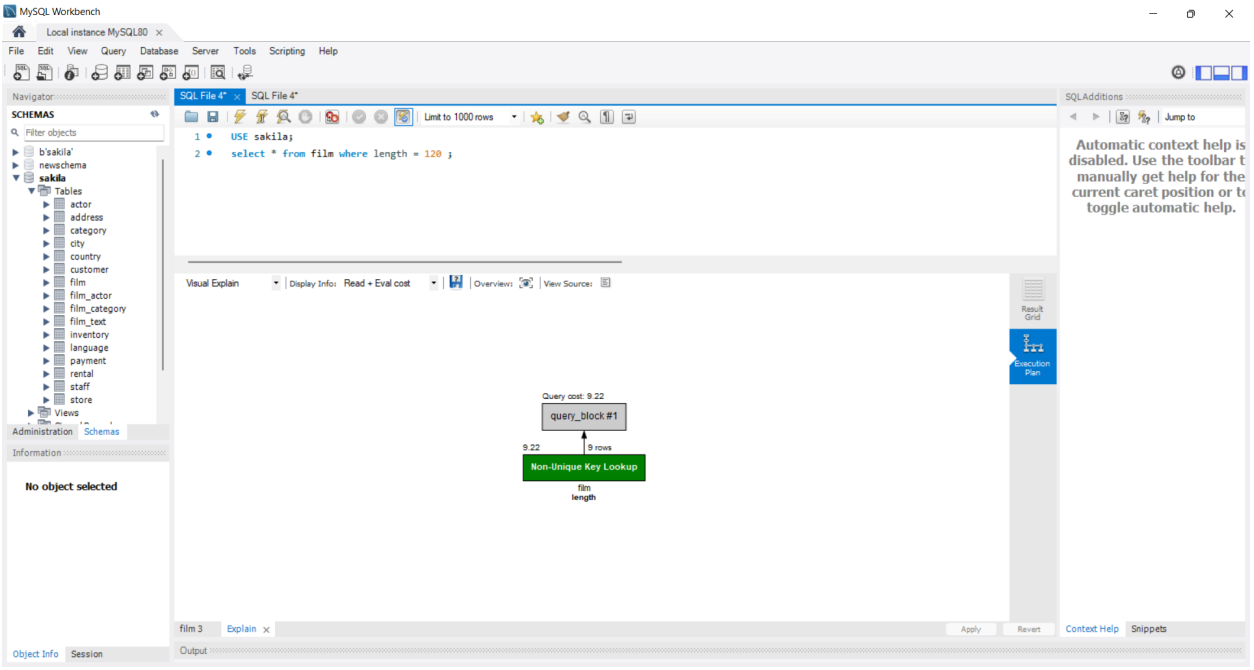


AFTER OPTIMIZING

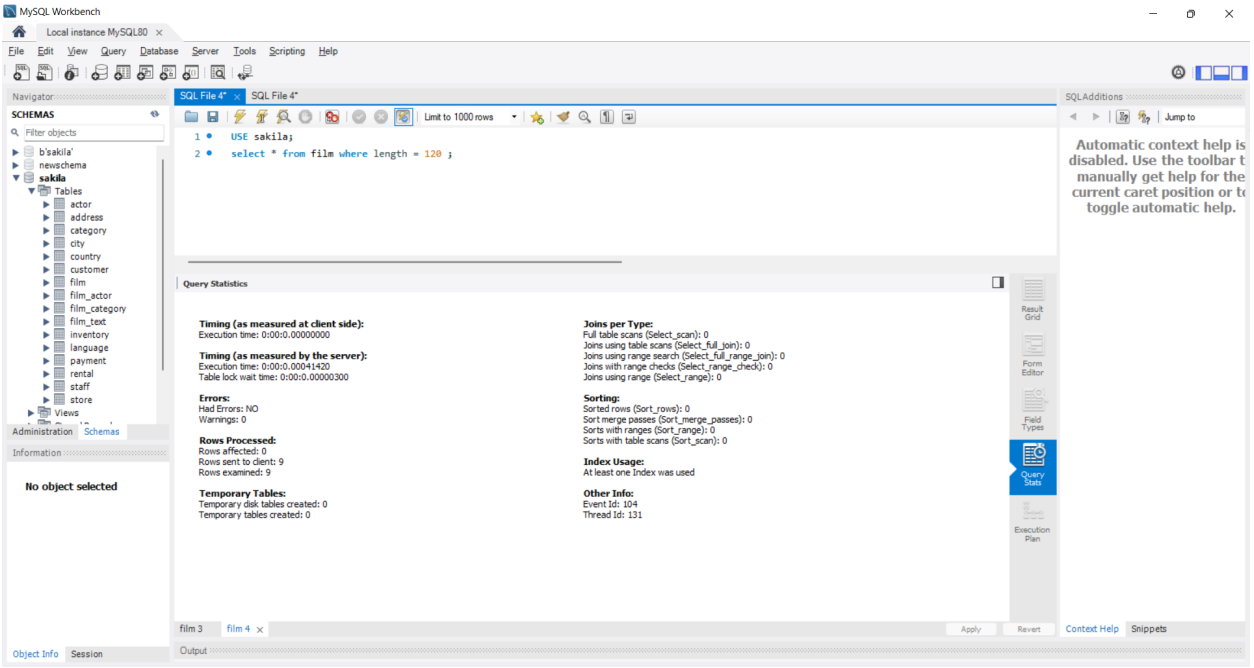


INDEX LEVEL OPTIMIZATION

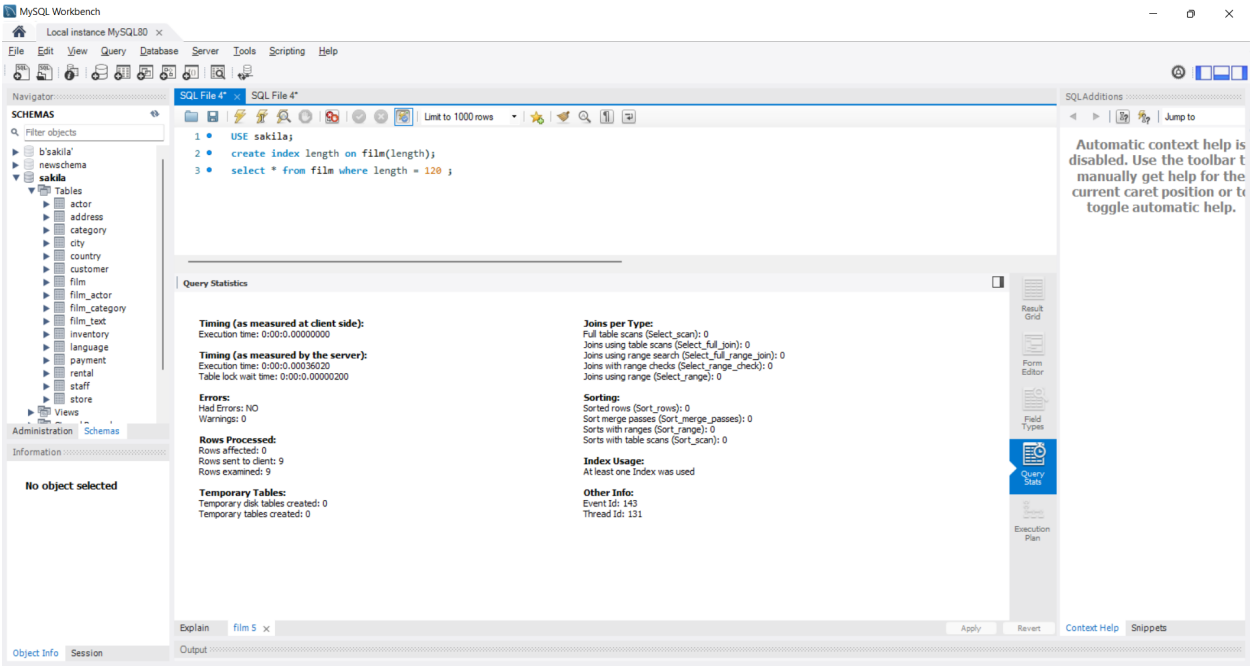
1)SIMPLE QUERY



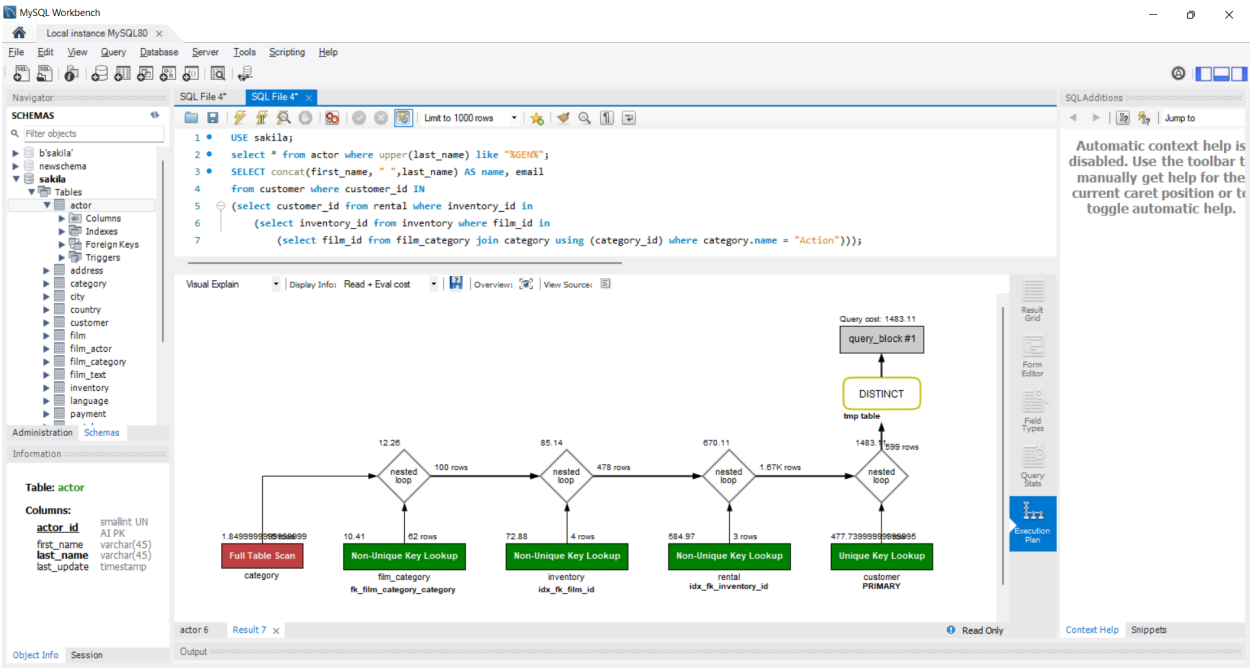
BEFORE INDEXING



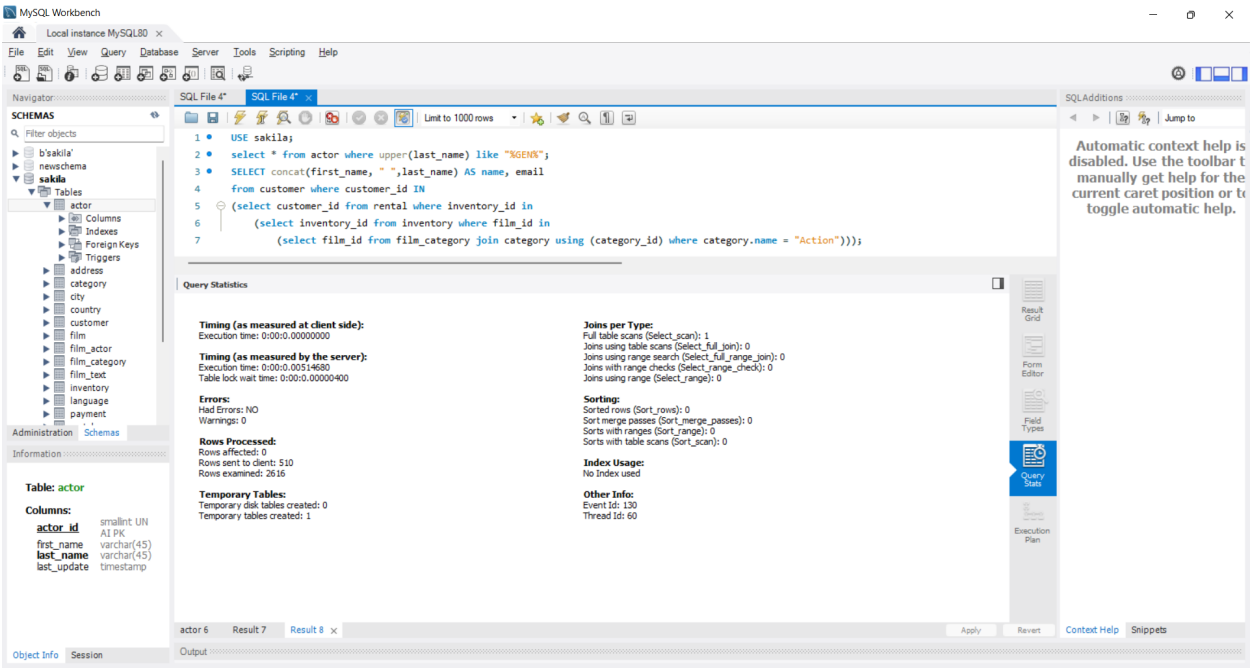
AFTER INDEXING


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SCHEMAS
sakila
Tables
actor
address
category
city
country
customer
film
film_actor
film_category
film_text
inventory
language
payment
rental
staff
store
Views
SQL File 4*
USE sakila;
create index length on film(length);
select * from film where length = 120 ;
Query Statistics
Timing (as measured at client side):
Execution time: 0:00:0.00000000
Timing (as measured by the server):
Execution time: 0:00:0.00036020
Table lock wait time: 0:00:0.00000200
Errors:
Had Errors: NO
Warnings: 0
Rows Processed:
Rows affected: 0
Rows sent to client: 9
Rows examined: 9
Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0
Joins per Type:
Full table scans (Select_scan): 0
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0
Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0
Index Usage:
At least one Index was used
Other Info:
Event Id: 143
Thread Id: 131
No object selected

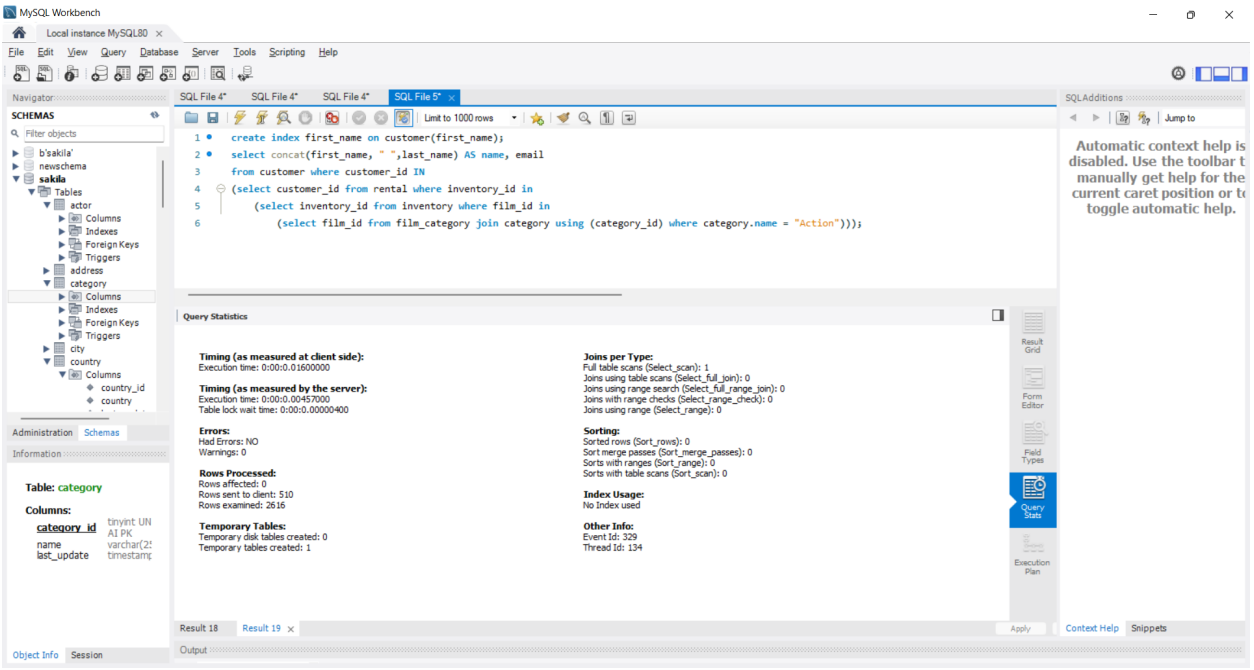
2) NESTED


MySQL Workbench
Local instance MySQL80
SQL File 4*
USE sakila;
select * from actor where upper(last_name) like "%GEN%";
SELECT concat(first_name, " ",last_name) AS name, email
from customer where customer_id IN
(select customer_id from rental where inventory_id in
(select inventory_id from inventory where film_id in
(select film_id from film_category join category using (category_id) where category.name = "Action")));
Visual Explain
Query cost: 1483.11
query_block #1
DISTINCT
tmp table
nested loop
100 rows
478 rows
1.67K rows
599 rows
Full Table Scan
category
Non-Unique Key Lookup
film_category
fk_film_category_category
Non-Unique Key Lookup
inventory
idx_fk_film_id
Non-Unique Key Lookup
rental
idx_fk_inventory_id
Unique Key Lookup
customer
PRIMARY
Table: actor
Columns:
actor_id
first_name
last_name
last_update

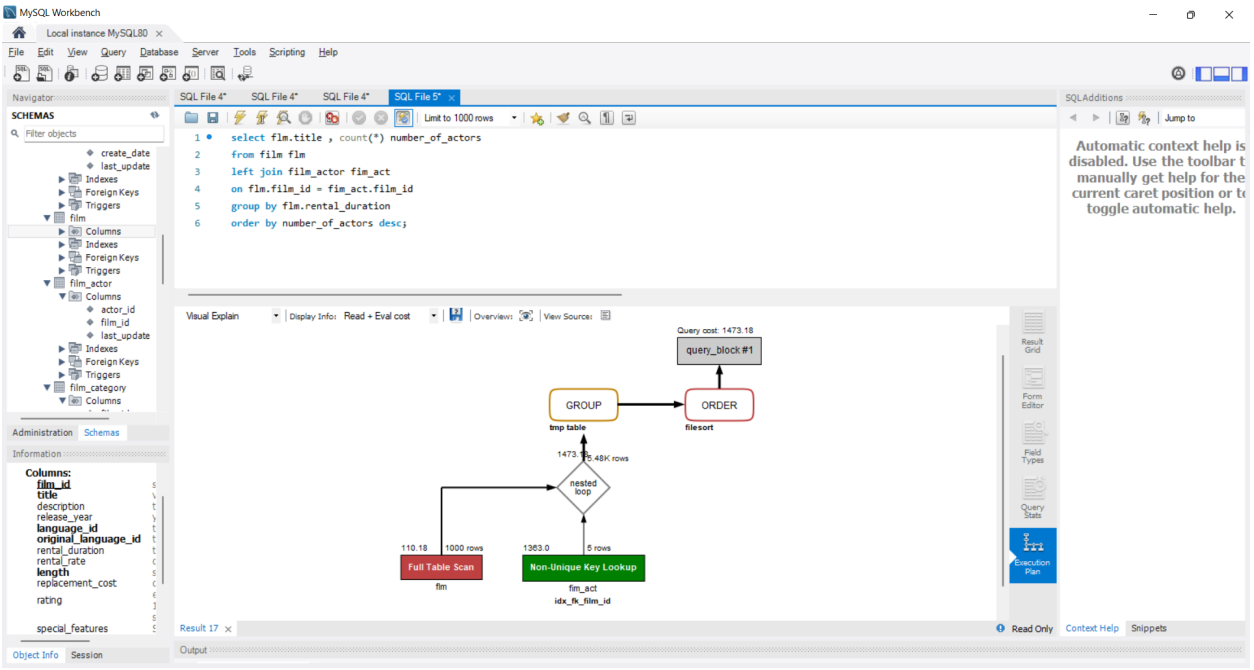
BEFORE INDEXING



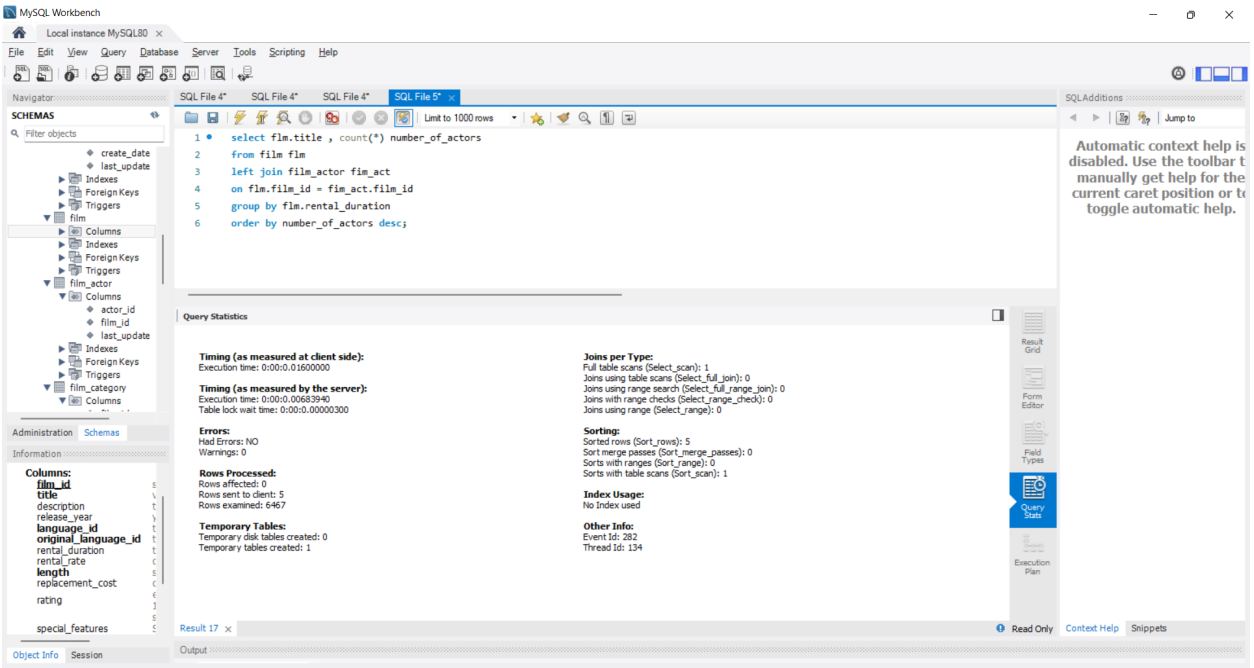
AFTER INDEXING



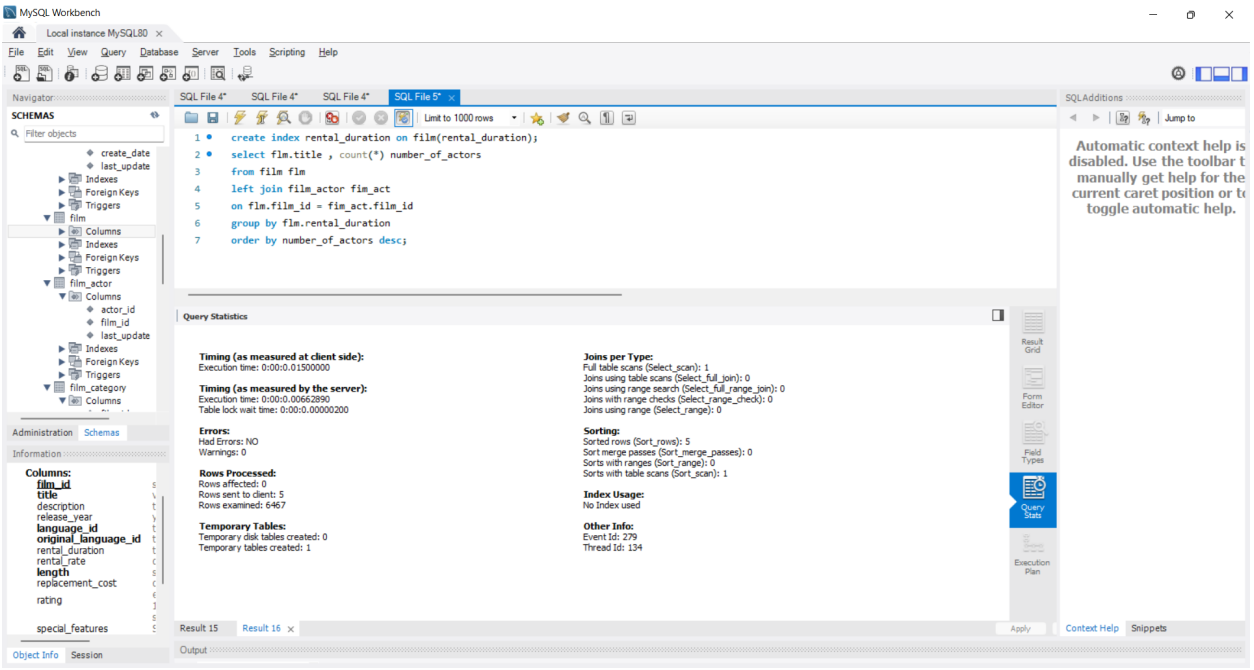
3) LEFT JOIN



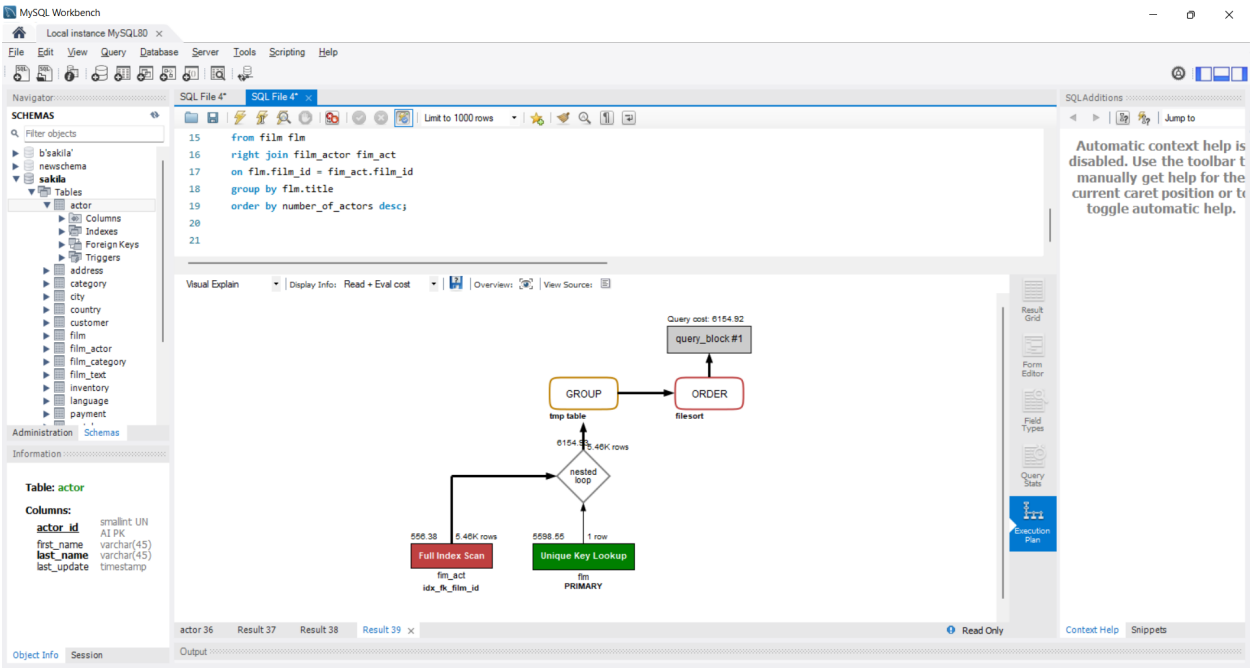
BEFORE INDEXING



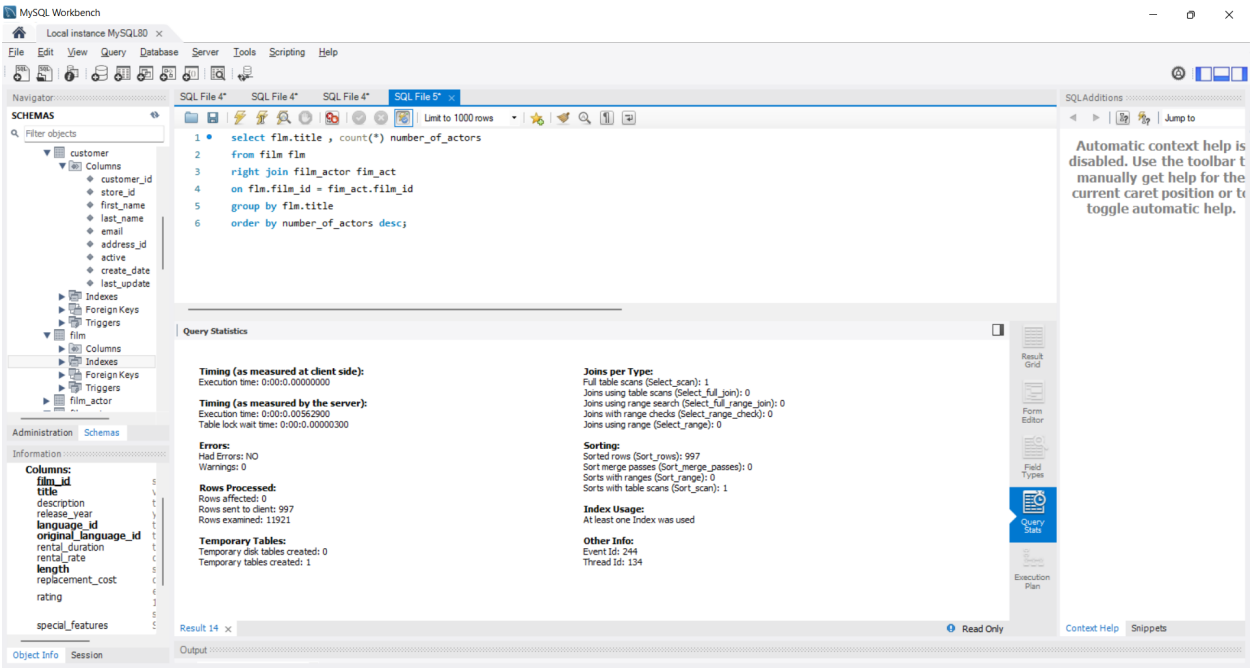
AFTER INDEXING



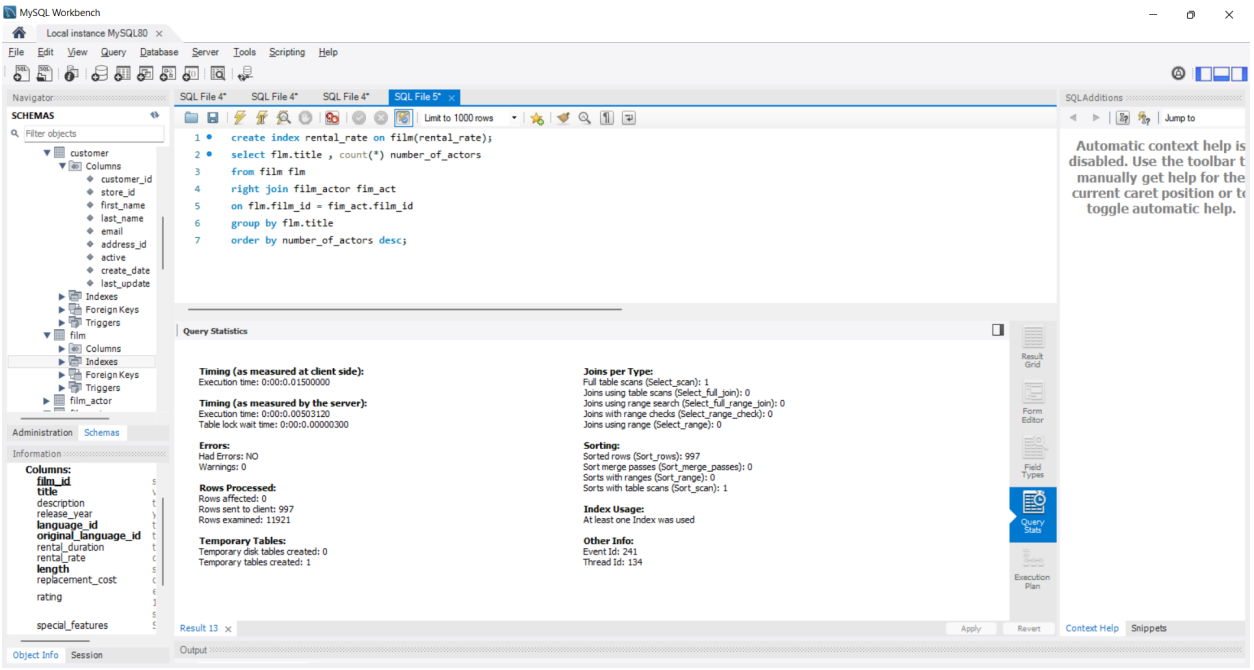
4) RIGHT JOIN



BEFORE INDEXING



AFTER INDEXING



Conclusion :-

In this experiment we performed table level and index level optimization and compared the results of no optimization. During a table scan the database reads multiple blocks from disk in single I/O. The cost of the scan depends on the number of blocks to be scanned and the multiblock read count value. The cost of index scan depends on the number of index blocks to be scanned and the number of rows to be fetched using the rowid in the index keys. It depends on index clustering factor.

EXPERIMENT - 4

Query Monitoring

4/11/22

NAME - PRACHI PATEL
SAP ID - 60004200049
BRANCH - Computer Engineering

24
25

Experiment - 4 :-

Aim - Implementation of Query monitor in MySQL.

Theory -

In SQL, explain keyword provides a description of how SQL queries are executed by the databases. These descriptions include optimizer logs, how tables are joined and in which order. Hence it would be beneficial tool in which order optimization. 'Explain' (EXPLAIN) also helps in taking care of the fact that user who doesn't know or have access of database will be provided with details about how it executes the queries. The main thing to note about EXPLAIN is that it will be used at begining of query i.e before SELECT, INSERT, UPDATE etc

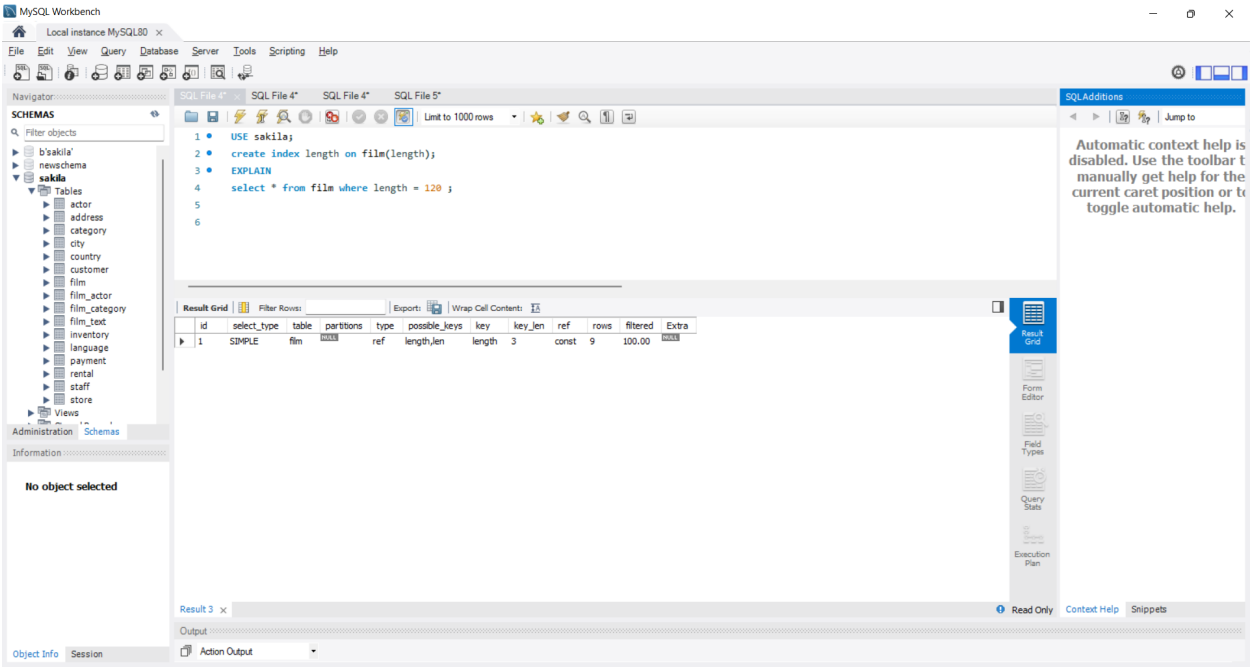
1) id - It represents id of query to be explained
2) select_type - The complexity in select clause is showed.
3) table - The name of table used is displayed here.
4) partitions - This shows the number of partitions of table joined in query
5) type - It specifies the join type.
6) Possible_keys - which keys should have been refrenced
7) key_len - length of key used
8) used_ref - Mentions any sort of refrences used in query while comparing columns or not.

The above mentioned words are some column names of EXPLAIN keyword results.

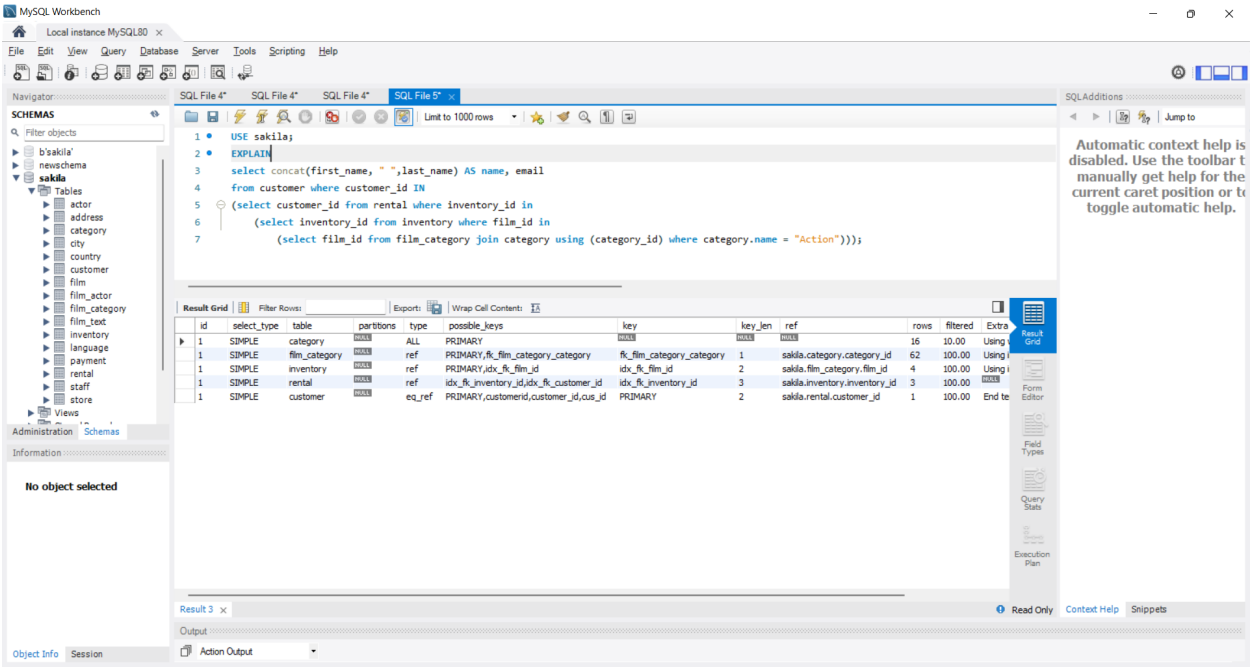
The EXPLAIN has some limitations though. They are,

1) It doesnot provide any information about how triggers, store function or UDFs will affect our query.
2) It cannot work for stored procedures.
3) It doesn't tell you about optimization
4) It produces estimated statistics that can be very inaccurate.
5) It doesn't produce every information regarding query execution plan
6) It provides information about how MySQL executes statements.
7) With the help of explain you can see where you can add indexes to tables so that the statement executes faster by using indexes to find rows.

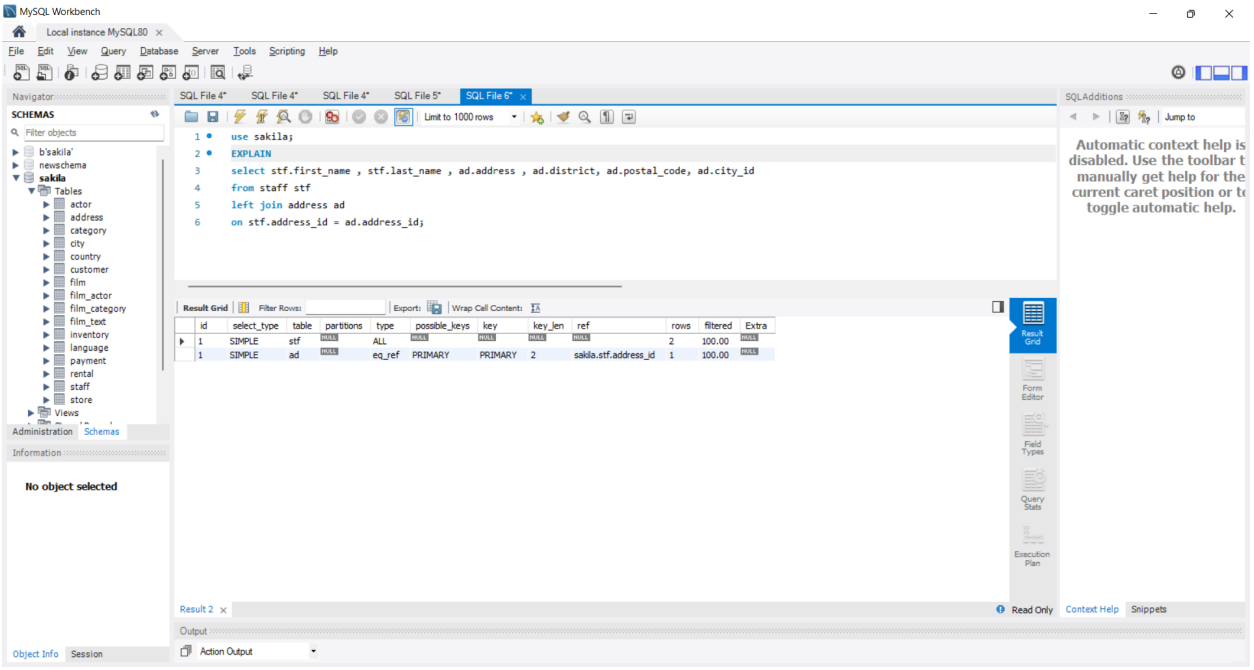
1) SELECT QUERY



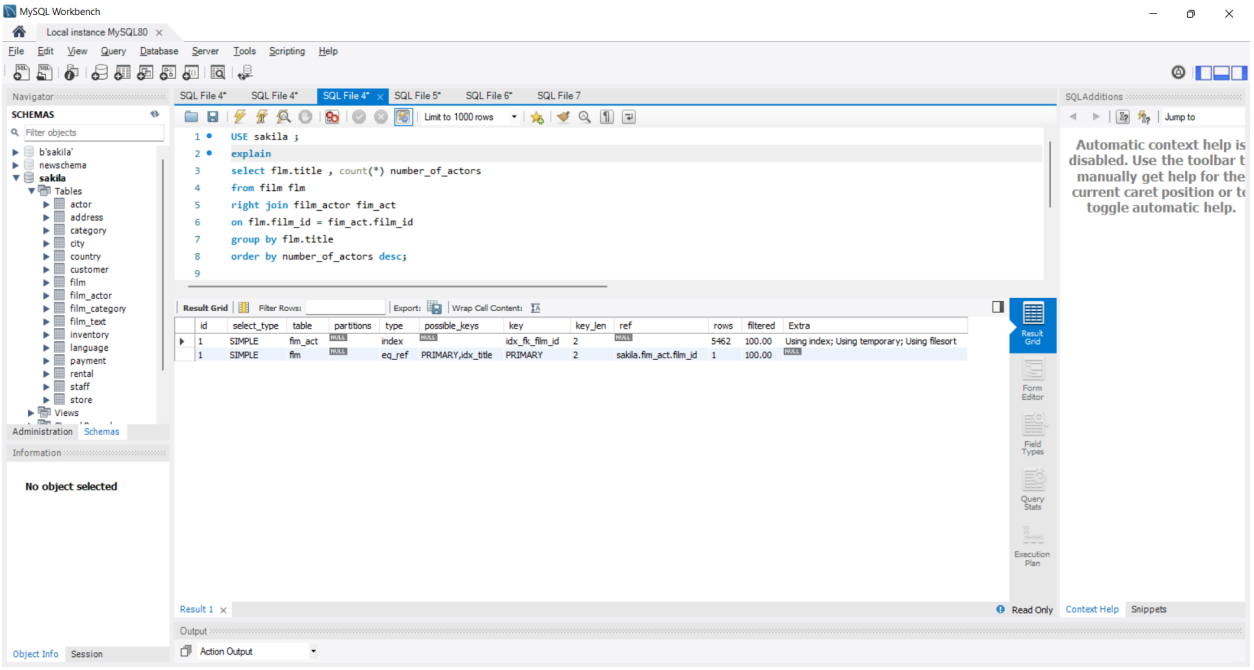
2) NESTED



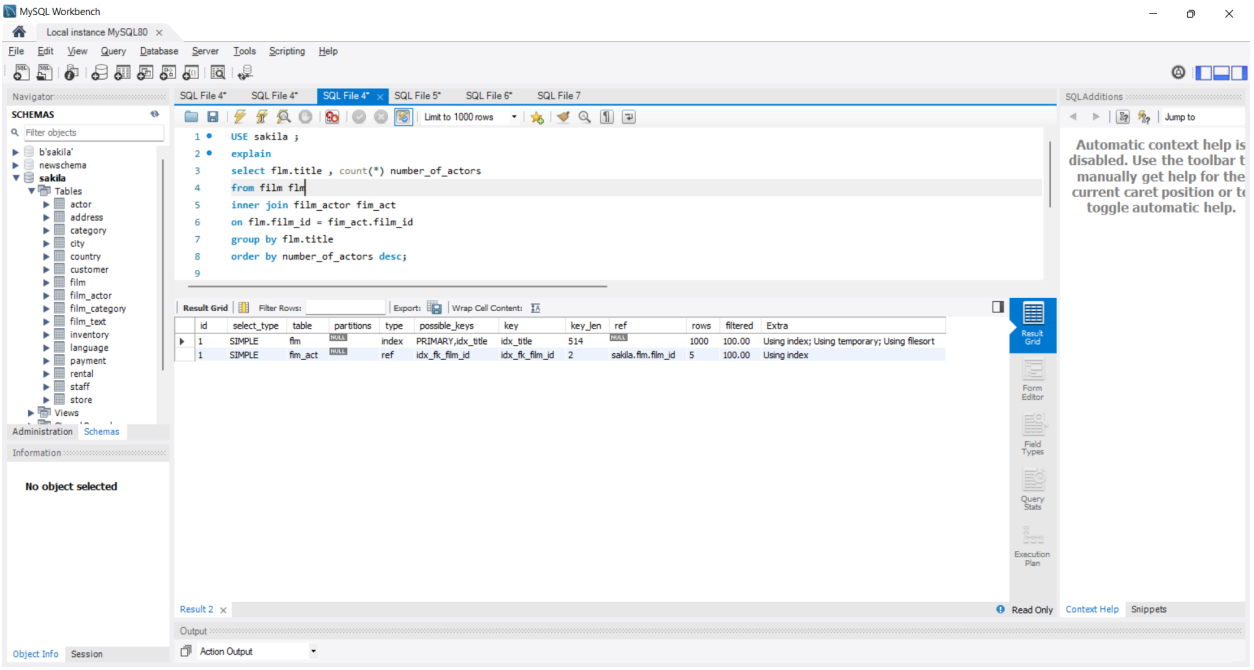
3) LEFT JOIN



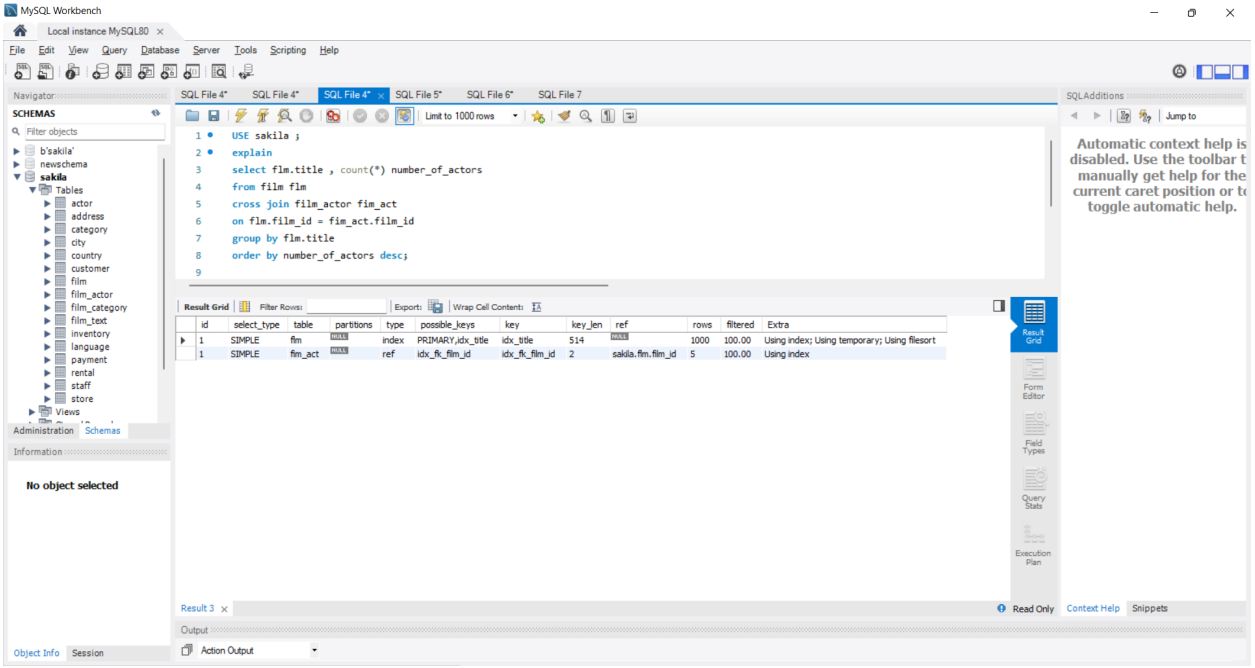
4) RIGHT JOIN



5) INNER JOIN



6) CROSS JOIN



Conclusion :-

We implemented query monitoring using EXPLAIN statement basic as well as nested queries and on different types of join. EXPLAIN keyword provides us the description of how the SQL queries are executed by databases along with optimizer logs, order of joining etc.

EXPERIMENT - 5

Fragmentation (Range , List , Hash & Key)

4/11/22

24/25

NAME - PRACHI PATEL
SAP ID - 60004200049
BRANCH - Computer Engineering

Experiment - 5 :-

AIM - To implement various types of fragmentation like Range, list, key and Hash.

Theory - Partioning in MySQL is used to split the rows of table into seperate tables in different locations but still is treated as single table.

Range Partitioning :-
This partitioning allows us to partition the rows of table based on column values that fall with in a specific range. The given range is always in a contiguous form but should not overlap each other and also use the values less than operator to define the ranges.

List Partitioning :-
It is same as Range Partioning. Here the partition is defined and selected based on columns matching one of a set of discrete value list rather than set of a contiguous range of values. It is performed by partition by list clause. The exp is an column value that returns integer value. List rather, the values IN (value = list) statement will be used to define each partition.

Hash Partitioning :-
This partitioning is used to distribute data based on predefined number of partition. In other words, it splits table as of value returned by

user defined expression. It is mainly used to distribute data evenly into partition. It is performed with partition by Hash (exp) clause

Key Partioning:-

It is similar to Hash Partioning where the hash partioning uses user-specified expression & MySQL server supplied hashing function for key. If we use other storage engines, MySQL server employs its own internal hashing function that is performed by using Partition by key clause.

1) CREATE TABLE

```sql
-- create database expadbms;
-- use expadbms;
CREATE TABLE Sales ( cust_id INT NOT NULL, name VARCHAR(40),
store_id VARCHAR(20) NOT NULL, bill_no INT NOT NULL,
bill_date DATE PRIMARY KEY NOT NULL, amount DECIMAL(8,2) NOT NULL)
PARTITION BY RANGE (year(bill_date))(
PARTITION p0 VALUES LESS THAN (2016),
PARTITION p1 VALUES LESS THAN (2017),
PARTITION p2 VALUES LESS THAN (2018),
PARTITION p3 VALUES LESS THAN (2020));
INSERT INTO Sales VALUES
(1, 'Mike', 'S001', 101, '2015-01-02', 125.56),
(2, 'Robert', 'S003', 103, '2015-01-25', 476.50),
(3, 'Peter', 'S012', 122, '2016-02-15', 335.00),
(4, 'Joseph', 'S345', 121, '2016-03-26', 787.00),
(5, 'Harry', 'S234', 132, '2017-04-19', 678.00),
(6, 'Stephen', 'S743', 111, '2017-05-31', 864.00),
(7, 'Jacson', 'S234', 115, '2018-06-11', 762.00),
(8, 'Smith', 'S012', 125, '2019-07-24', 300.00),
(9, 'Adam', 'S456', 119, '2019-08-02', 492.20);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

cust_id	name	store_id	bill_no	bill_date	amount
3	Peter	S012	122	2016-02-15	335.00
4	Joseph	S345	121	2016-0	
5	Harry	S234	132	2017-04-19	678.00
6	Stephen	S743	111	2017-05-31	864.00
7	Jacson	S234	115	2018-06-11	762.00
8	Smith	S012	125	2019-07-24	300.00
9	Adam	S456	119	2019-08-02	492.20
NULL	NULL	NULL	NULL	NULL	NULL

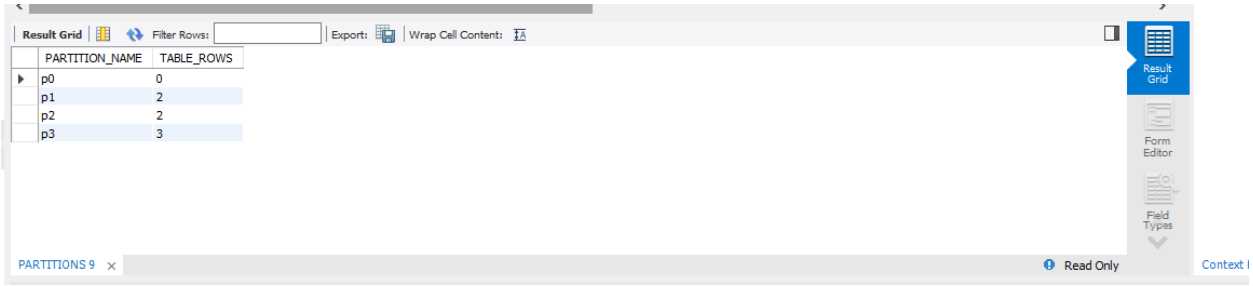
2016-02-15

```sql
SELECT * FROM Sales;
SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS, AVG_ROW_LENGTH, DATA_LENGTH
FROM INFORMATION_SCHEMA.PARTITIONS
WHERE TABLE_SCHEMA = 'expadbms' AND TABLE_NAME = 'Sales';
ALTER TABLE Sales TRUNCATE PARTITION p0;
SELECT PARTITION_NAME, TABLE_ROWS
FROM INFORMATION_SCHEMA.PARTITIONS
WHERE TABLE_SCHEMA = 'expadbms' AND TABLE_NAME = 'Sales';
CREATE TABLE Stores (
    cust_name VARCHAR(40),
    bill_no VARCHAR(20) NOT NULL,
    store_id INT PRIMARY KEY NOT NULL,
    bill_date DATE NOT NULL,
    amount DECIMAL(8,2) NOT NULL
)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

TABLE_NAME	PARTITION_NAME	TABLE_ROWS	AVG_ROW_LENGTH	DATA_LENGTH
sales	p0	0	0	16384
sales	p1	2	8192	16384
sales	p2	2	8192	16384
sales	p3	3	5461	16384

2) AFTER ALTER TABLE



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PARTITION_NAME	TABLE_ROWS
p0	0
p1	2
p2	2
p3	3

PARTITIONS 9 | Read Only

3) LIST PARTITIONING

```
28    WHERE TABLE_SCHEMA = 'expadbms' AND TABLE_NAME = 'Sales';
29    CREATE TABLE StoresList (
30        cust_name VARCHAR(40),
31        bill_no VARCHAR(20) NOT NULL,
32        store_id INT PRIMARY KEY NOT NULL,
33        bill_date DATE NOT NULL,
34        amount DECIMAL(8,2) NOT NULL
35    )
36    PARTITION BY LIST(store_id) (
37    PARTITION pEast VALUES IN (101, 103, 105),
38    PARTITION pWest VALUES IN (102, 104, 106),
39    PARTITION pNorth VALUES IN (107, 109, 111),
40    PARTITION pSouth VALUES IN (108, 110, 112));
41    INSERT INTO StoresList VALUES
42    ( 'Mike', 'S001', 101, '2015-01-02', 125.56),
43    ( 'Robert', 'S003', 103, '2015-01-25', 476.50),
44    ( 'Peter', 'S012', 104, '2016-02-15', 335.00),
45    ( 'Joseph', 'S345', 105, '2016-03-26', 787.00),
46    ( 'Harry', 'S234', 106, '2017-04-19', 678.00),
47    ( 'Stephen', 'S743', 107, '2017-05-31', 864.00),
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

cust_name	bill_no	store_id	bill_date	amount
Mike	S001	101	2015-01-02	125.56
Robert	S003	103	2015-01-25	476.50
Joseph	S345	105	2016-03-26	787.00
Peter	S012	104	2016-02-15	335.00
Harry	S234	106	2017-04-19	678.00
Stephen	S743	107	2017-05-31	864.00
Jacson	S234	108	2018-06-11	762.00

4) HASH PARTITIONING

```
CREATE TABLE StoresHashNew (
    cust_name VARCHAR(40),
    bill_no VARCHAR(20) NOT NULL,
    store_id INT PRIMARY KEY NOT NULL,
    bill_date DATE NOT NULL,
    amount DECIMAL(8,2) NOT NULL
)
PARTITION BY HASH(store_id)
PARTITIONS 4;
INSERT INTO StoresHashNew VALUES
( 'Mike', 'S001', 101, '2015-01-02', 125.56),
( 'Robert', 'S003', 103, '2015-01-25', 476.50),
( 'Peter', 'S012', 104, '2016-02-15', 335.00),
( 'Joseph', 'S345', 105, '2016-03-26', 787.00),
( 'Harry', 'S234', 106, '2017-04-19', 678.00),
( 'Stephen', 'S743', 107, '2017-05-31', 864.00),
( 'Jacson', 'S234', 108, '2018-06-11', 762.00);
select * from StoresHashNew
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

cust_name	bill_no	store_id	bill_date	amount
Peter	S012	104	2016-02-15	335.00
Jacson	S234	108	2018-06-11	762.00
Mike	S001	101	2015-01-02	125.56
Joseph	S345	105	2016-03-26	787.00
Harry	S234	106	2017-04-19	678.00
Robert	S003	103	2015-01-25	476.50
Stephen	S743	107	2017-05-31	864.00
NULL	NULL	NULL	NULL	NULL

5) RANGE COLUMN PARTITIONING

```
CREATE TABLE test_part (A INT, B CHAR(5), C INT, D INT)
PARTITION BY RANGE COLUMNS(A, B, C)
 (PARTITION p0 VALUES LESS THAN (50, 'test1', 100),
 PARTITION p1 VALUES LESS THAN (100, 'test2', 200),
 PARTITION p2 VALUES LESS THAN (150, 'test3', 300),
 PARTITION p3 VALUES LESS THAN (MAXVALUE, MAXVALUE, MAXVALUE));
 INSERT INTO test_part VALUES
( 45,'M', 101,  125.56),
( 47,'R',  103,  476.50),
( 74,'P',  104,  335.00),
( 190,'J', 105,  787.00);
select * from test_part;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

A	B	C	D
45	M	101	126
47	R	103	477
74	P	104	335
190	J	105	787

test_part 13 | Read Only

6) LIST COLUMN PARTITIONING

```
CREATE TABLE AgentDetail (
agent_id VARCHAR(10),
agent_name VARCHAR(40),
city VARCHAR(10))
PARTITION BY LIST COLUMNS(agent_id) (
PARTITION pNewyork VALUES IN('A1', 'A2', 'A3'),
PARTITION pTexas VALUES IN('B1', 'B2', 'B3'),
PARTITION pCalifornia VALUES IN ('C1', 'C2', 'C3'));

 INSERT INTO AgentDetail VALUES
( 'A1','Mike', 101  ),
( 'A2','Robert',  103),
( 'B1','Peterson',  104),
( 'C1','Jack', 105);
select * from AgentDetail
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

agent_id	agent_name	city
A1	Mike	101
A2	Robert	103
B1	Peterson	104
C1	Jack	105

entDetail 14 | Read Only

7) KEY PARTITIONING

```
CREATE TABLE AgentDetailKey (
     agent_name VARCHAR(40) ,
    agent_id INT NOT NULL PRIMARY KEY

)
PARTITION BY KEY()
PARTITIONS 2;
INSERT INTO AgentDetailKey VALUES
( 'Mike', 101  ),
( 'Robert',  103),
( 'Peterson',  104),
( 'Jack', 105);
select * from AgentDetailKey
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

agent_name	agent_id
Mike	101
Robert	103
Jack	105
Peterson	104
NULL	NULL

entDetailKey 15 | Apply | Revert

Conclusion :-

Various types of fragmentation were studied using Range, List, Key & Hash partitioning & change in table i.e the partitions formed were observed; Range partitioning assigns rows to partitions based on column values falling within a range. List partitioning is selected based on columns matching of one of discrete values. Hash partioning is selected based on value returned by a user-defined expression that operates on column value in rows to be inserted into table.

EXPERIMENT - 6

2 Phase & 3 Phase Protocol

18/11/2022

NAME - PRACHI PATEL
SAP ID - 60004200049
BRANCH - Computer Engineering

24
25

Experiment - 6 - 2 PC & 3PC

AIM :- To implement two phase / three phase commit protocol

Two-phase commit Protocol :-
The 2 phase commit is most popular technique to implement the concurrency control. This protocol comprises a co-ordinator, which resides on the site that initiates the transaction and subordinate sites responsible for executing the sub-transactions. The protocol works in 2 phases; In the first phase, the co-ordinator tries to get decision of whether to commit or abort. In second phase, the coordinator relays the decision back to its subordinate.

1) Phase 1 or decision making phase -
Step 1 - when a site first initiates a transaction. It selects a coordinate and writes a "PREPARE" log in its stable storage and sends them and writes "PREPARE" message to all its subordinates.
Step 2 - After recieving the prepare message, the subordinates decide whether they are capable of commiting the transaction. If it can commit the transcation, it will send a "READY" message back to Coordinator and also logs "READY" in its local storage.
Step 3 - Coordinator waits for the ready message from all its subordinate sites. If all reply messages contain Ready message, then the coordinator sends an acknowledgement revealing to abort all the sub transactions.

This ends the phase 1 cycle.

2) Phase 2 or decision recording phase :-

Step 1 - After recieving the READY message from all its subordinates, the coordinator writes a COMMIT log in its stable storage; otherwise, it will write an abort log in its storage.

Step 2 - When all the subordinates recieve acknowledgement back to the coordinator stating that all the sub transactions and write a COMMIT log in their local database and send an acknowledgment back to the coordinator of successful completion of executing transaction.

Step 3 - Upon recieving acknowledgement from all subordinates, the coordinator writes a log "COMPLETE" in its storage and the transaction ultimately completes.

If any of the subordinates fail to send the acknowledgement in between the defined time and the coordinator, the coordinator then sends a message to abort all the sub-transactions.

Three Phase Commit Protocol :-

The main drawback of 2PC protocol is that it leads to blockage of the resources for indefinite time in case of a failure. Let's elaborate for this case. When any site fails in executing this protocol, the other sites involved in the activity need to wait until the site recovers from failure. The recovery can take few minutes, or few hours, or even few days, leading to indefinite starvation. If the site failing is coordinator, then all the sites participating in transaction need to wait; and these sites block the resources that may be requested by the other sites executing other transactions. To provide a transaction that should go on uninterruptedly a new protocol has been formulated, which is known as 3PC. This protocol includes an extra phase in between the decision making phase

and decision recording phase, which reassures the sites regarding the failure.

1) Phase 2 or reassuring phase in 3PC protocol

It involves the step that if the coordinator recieves an ABORT reply from one of the subordinates, then the coordinator will send "PREPARE ABORT" message to all the subordinates. Now, if all the subordinates send a "READY" message to commit the sub-transactions, then after recieving the "READY" message from all the subordinate, the coordinator will send the "PREPARE COMMIT" message and write this message to its storage.

CODE :

A) 2 Phase Commit

1. Server

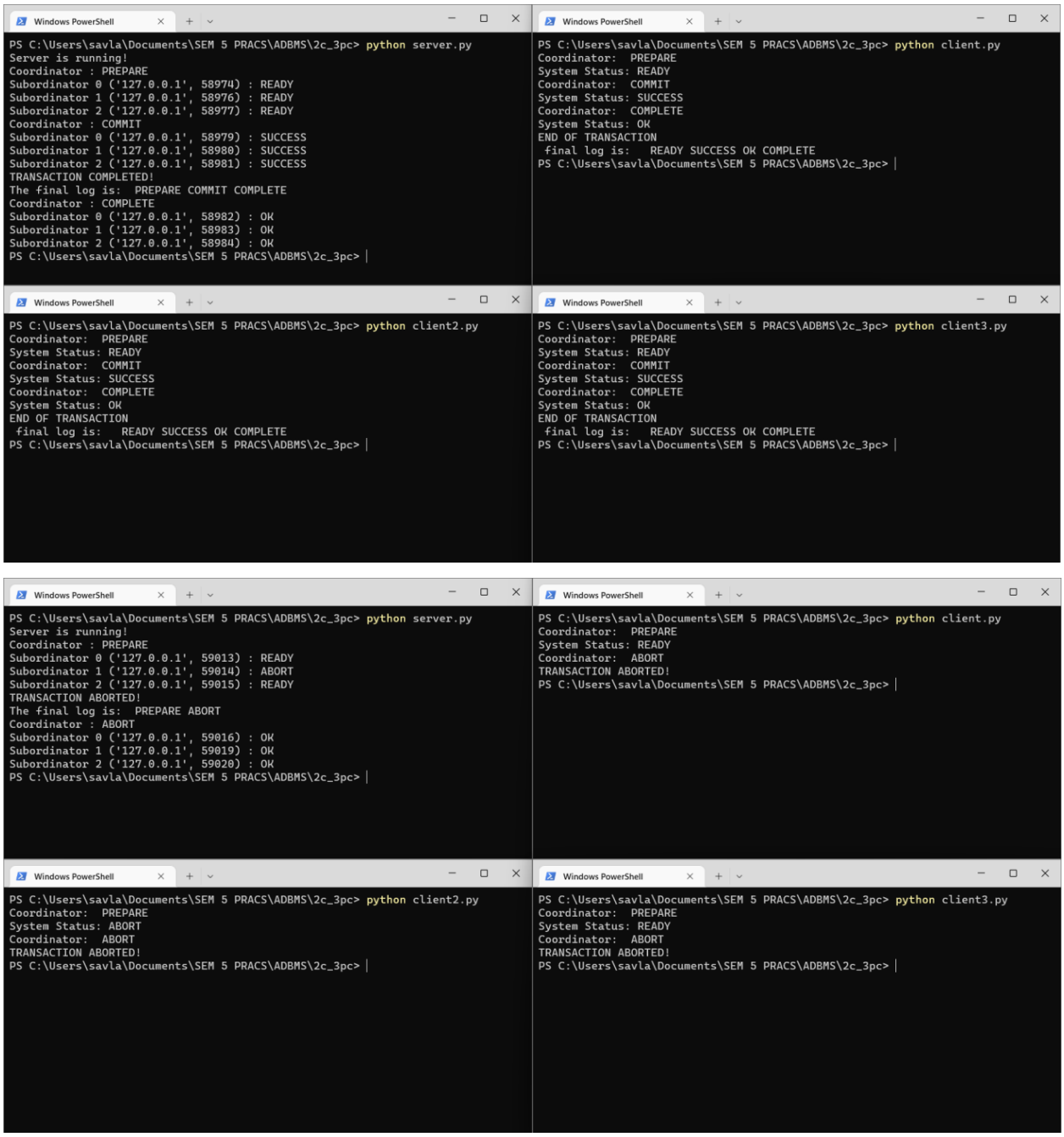
```
import socket
def ServerSoc():
host = "127.0.0.1"
port = 8000
print("Server is running!")
msg = "PREPARE"
log = msg
over = 0
s_soc = socket.socket()
s_soc.bind((host,port))
s_soc.listen(2)
while(1):
replies = []
print(f"Coordinator : {msg.upper()}")
for i in range(3):
conn,add = s_soc.accept()
conn.send(msg.encode())
data = conn.recv(1024).decode()
replies.append(data.upper())
print(f"Subordinator {i} {add} : {data.upper()}")
if over == 1:
break
if ("ABORT" in replies) or (len(replies)<3) :
msg = "ABORT"
print("TRANSACTION ABORTED!\nThe final log is: ", log+" "+msg)
over = 1
elif "SUCCESS" in replies:
msg = "COMPLETE"
print("TRANSACTION COMPLETED!\nThe final log is: ", log+" "+msg)
over = 1
else:
msg = "COMMIT"
```

```
log += " "+msg
ServerSoc()
```

2. Client

```
import socket
def ClientSoc():
host = "127.0.0.1"
port = 8000
log = ""
over = 0
while(1):
try:
s_soc = socket.socket()
s_soc.connect((host,port))
rec_data = s_soc.recv(1024).decode()
print("Coordinator: ", rec_data.upper())
if rec_data.upper() == "ABORT":
msg = "OK"
print("TRANSACTION ABORTED!")
over = 1
elif rec_data.upper() == "SUCCESS":
msg = "OK"
print("TRANSACTION COMPLETED!")
over = 1
else:
msg = input("System Status: ").upper()
log += " "+msg
s_soc.send(msg.encode())
if over == 1:
break
s_soc.close()
except:
print("END OF TRANSACTION\n final log is: ",log+" "+rec_data.upper())
break
ClientSoc()
```

OUTPUT :



B) 3 Phase Commit

1. Server

```
import socket
def ServerSoc():
```

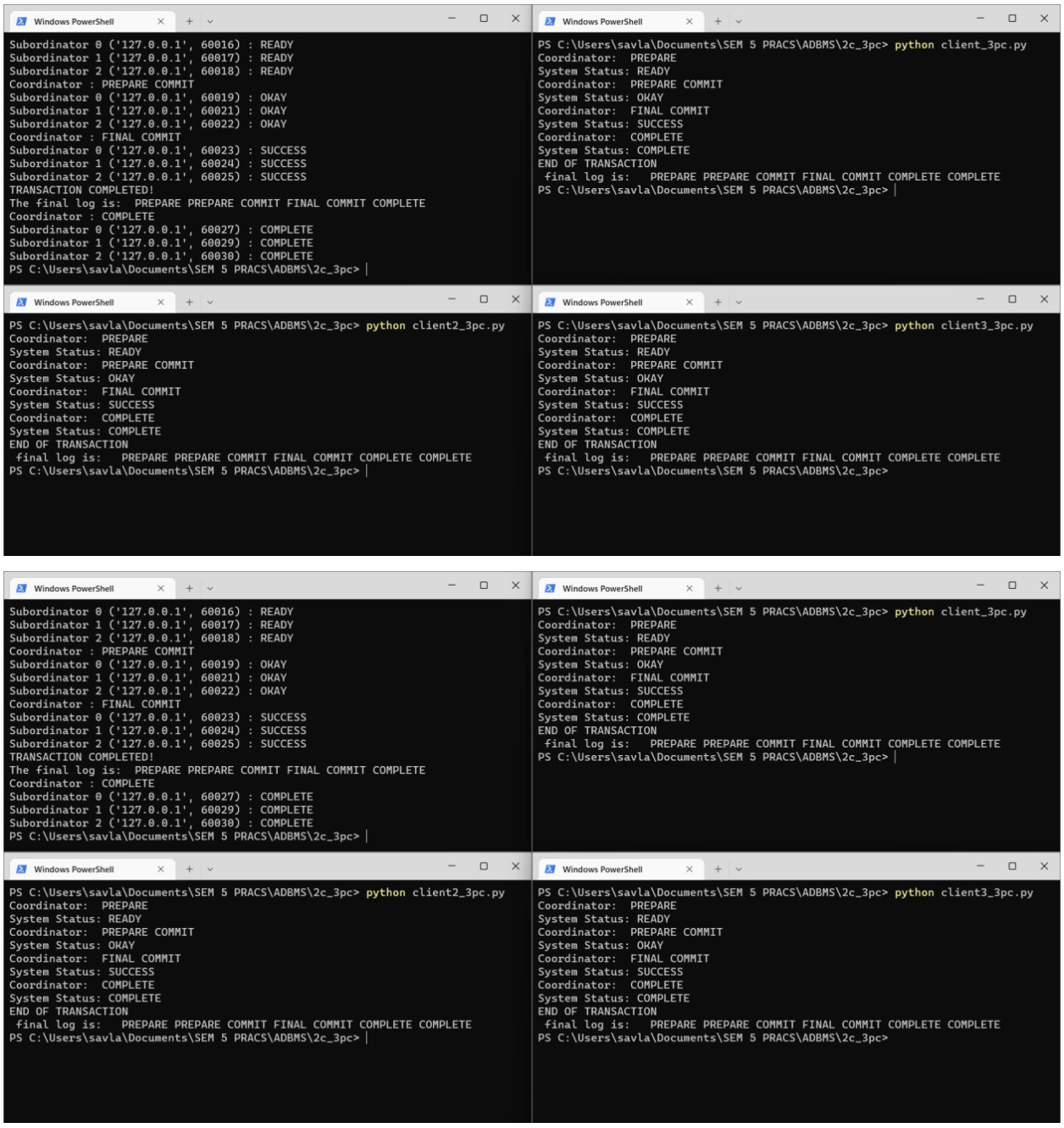
```
host = "127.0.0.1"
port = 8000
print("Server is running!")
msg = "PREPARE"
log = msg
over = 0
s_soc = socket.socket()
s_soc.bind((host,port))
s_soc.listen(2)
while(1):
replies = []
print(f"Coordinator : {msg.upper()}")
for i in range(3):
conn,add = s_soc.accept()
conn.send(msg.encode())
data = conn.recv(1024).decode()
replies.append(data.upper())
print(f"Subordinator {i} {add} : {data.upper()}")
if over == 2:
break
if ("ABORT" in replies) or (len(replies)<3) :
msg = "PREPARE ABORT"
over = 1
elif "SUCCESS" in replies:
msg = "COMPLETE"
print("TRANSACTION COMPLETED!\nThe final log is: ", log+" "+msg)
over = 2
elif "READY" in replies:
msg = "PREPARE COMMIT"
else:
if over == 1:
msg = "FINAL ABORT"
over = 2
else:
msg ="FINAL COMMIT"
print("TRANSACTION ABORTED!\nThe final log is: ", log+" "+msg)
```

```
log += " "+msg
ServerSoc()
```

2. Client

```
import socket
def ClientSoc():
host = "127.0.0.1"
port = 8000
log = ""
over = 0
while(1):
try:
s_soc = socket.socket()
s_soc.connect((host,port))
rec_data = s_soc.recv(1024).decode()
print("Coordinator: ", rec_data.upper())
if rec_data.upper() == "FINAL ABORT":
msg = "OK"
print("TRANSACTION ABORTED!")
over = 1
elif rec_data.upper() == "SUCCESS":
msg = "OK"
print("TRANSACTION COMPLETED!")
over = 1
else:
msg = input("System Status: ").upper()
log += " "+rec_data.upper()
s_soc.send(msg.encode())
if over == 1:
break
s_soc.close()
except:
print("END OF TRANSACTION\n final log is: ",log+" "+rec_data.upper())
break
ClientSoc()
```

OUTPUT :


Windows PowerShell
Subordinator 0 ('127.0.0.1', 60016) : READY
Subordinator 1 ('127.0.0.1', 60017) : READY
Subordinator 2 ('127.0.0.1', 60018) : READY
Coordinator : PREPARE COMMIT
Subordinator 0 ('127.0.0.1', 60019) : OKAY
Subordinator 1 ('127.0.0.1', 60021) : OKAY
Subordinator 2 ('127.0.0.1', 60022) : OKAY
Coordinator : FINAL COMMIT
Subordinator 0 ('127.0.0.1', 60023) : SUCCESS
Subordinator 1 ('127.0.0.1', 60024) : SUCCESS
Subordinator 2 ('127.0.0.1', 60025) : SUCCESS
TRANSACTION COMPLETED!
The final log is: PREPARE PREPARE COMMIT FINAL COMMIT COMPLETE
Coordinator : COMPLETE
Subordinator 0 ('127.0.0.1', 60027) : COMPLETE
Subordinator 1 ('127.0.0.1', 60029) : COMPLETE
Subordinator 2 ('127.0.0.1', 60030) : COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc>
Windows PowerShell
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client_3pc.py
Coordinator: PREPARE
System Status: READY
Coordinator: PREPARE COMMIT
System Status: OKAY
Coordinator: FINAL COMMIT
System Status: SUCCESS
Coordinator: COMPLETE
System Status: COMPLETE
END OF TRANSACTION
final log is: PREPARE PREPARE COMMIT FINAL COMMIT COMPLETE COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc>
Windows PowerShell
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client2_3pc.py
Coordinator: PREPARE
System Status: READY
Coordinator: PREPARE COMMIT
System Status: OKAY
Coordinator: FINAL COMMIT
System Status: SUCCESS
Coordinator: COMPLETE
System Status: COMPLETE
END OF TRANSACTION
final log is: PREPARE PREPARE COMMIT FINAL COMMIT COMPLETE COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc>
Windows PowerShell
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client3_3pc.py
Coordinator: PREPARE
System Status: READY
Coordinator: PREPARE COMMIT
System Status: OKAY
Coordinator: FINAL COMMIT
System Status: SUCCESS
Coordinator: COMPLETE
System Status: COMPLETE
END OF TRANSACTION
final log is: PREPARE PREPARE COMMIT FINAL COMMIT COMPLETE COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc>

Windows PowerShell
Subordinator 0 ('127.0.0.1', 60016) : READY
Subordinator 1 ('127.0.0.1', 60017) : READY
Subordinator 2 ('127.0.0.1', 60018) : READY
Coordinator : PREPARE COMMIT
Subordinator 0 ('127.0.0.1', 60019) : OKAY
Subordinator 1 ('127.0.0.1', 60021) : OKAY
Subordinator 2 ('127.0.0.1', 60022) : OKAY
Coordinator : FINAL COMMIT
Subordinator 0 ('127.0.0.1', 60023) : SUCCESS
Subordinator 1 ('127.0.0.1', 60024) : SUCCESS
Subordinator 2 ('127.0.0.1', 60025) : SUCCESS
TRANSACTION COMPLETED!
The final log is: PREPARE PREPARE COMMIT FINAL COMMIT COMPLETE
Coordinator : COMPLETE
Subordinator 0 ('127.0.0.1', 60027) : COMPLETE
Subordinator 1 ('127.0.0.1', 60029) : COMPLETE
Subordinator 2 ('127.0.0.1', 60030) : COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc>
Windows PowerShell
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client_3pc.py
Coordinator: PREPARE
System Status: READY
Coordinator: PREPARE COMMIT
System Status: OKAY
Coordinator: FINAL COMMIT
System Status: SUCCESS
Coordinator: COMPLETE
System Status: COMPLETE
END OF TRANSACTION
final log is: PREPARE PREPARE COMMIT FINAL COMMIT COMPLETE COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc>
Windows PowerShell
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client2_3pc.py
Coordinator: PREPARE
System Status: READY
Coordinator: PREPARE COMMIT
System Status: OKAY
Coordinator: FINAL COMMIT
System Status: SUCCESS
Coordinator: COMPLETE
System Status: COMPLETE
END OF TRANSACTION
final log is: PREPARE PREPARE COMMIT FINAL COMMIT COMPLETE COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc>
Windows PowerShell
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client3_3pc.py
Coordinator: PREPARE
System Status: READY
Coordinator: PREPARE COMMIT
System Status: OKAY
Coordinator: FINAL COMMIT
System Status: SUCCESS
Coordinator: COMPLETE
System Status: COMPLETE
END OF TRANSACTION
final log is: PREPARE PREPARE COMMIT FINAL COMMIT COMPLETE COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc>

CONCLUSION :-

After implementing programs for two phase and three phase commit protocols, it can be concluded that these protocols avoid a deadlock situation in distributed database environment; making 3 phase commit protocol more resilient to system failures than 2 phase commit protocol.

EXPERIMENT - 7

Query execution of XML Database

18/11/22

24/25

NAME - PRACHI PATEL
SAP ID - 60004200049
BRANCH - Computer Engineering

Experiment -7 - XML database

AIM :- Query Execution on XML database.

XML :-
The Extensible Markup Language (XML) was not designed for database Applications. In fact, like the Hyper-Text Markup Language (HTML) on which the World Wide Web (WWW) is based, XML has its roots in document Management, and is derived from a language for structuring large documents known as the standard Generalised Markup Language (SGML). It is particularly useful as a data format when an application or integrate information from several other applications. When XML is used in these contexts, many database issue arise, including how to organize, manipulate and query the XML data.

Querying XML data :-
Given the increasing number of applications that use XML to exchange, mediate, and store data, tools for effective management of XML data are becoming increasingly important. In particular, tools for querying and transformation of XML are essential to extract information from large bodies of XML data and to convert data between different representations (schemas) in XML. Just as the output of a relational

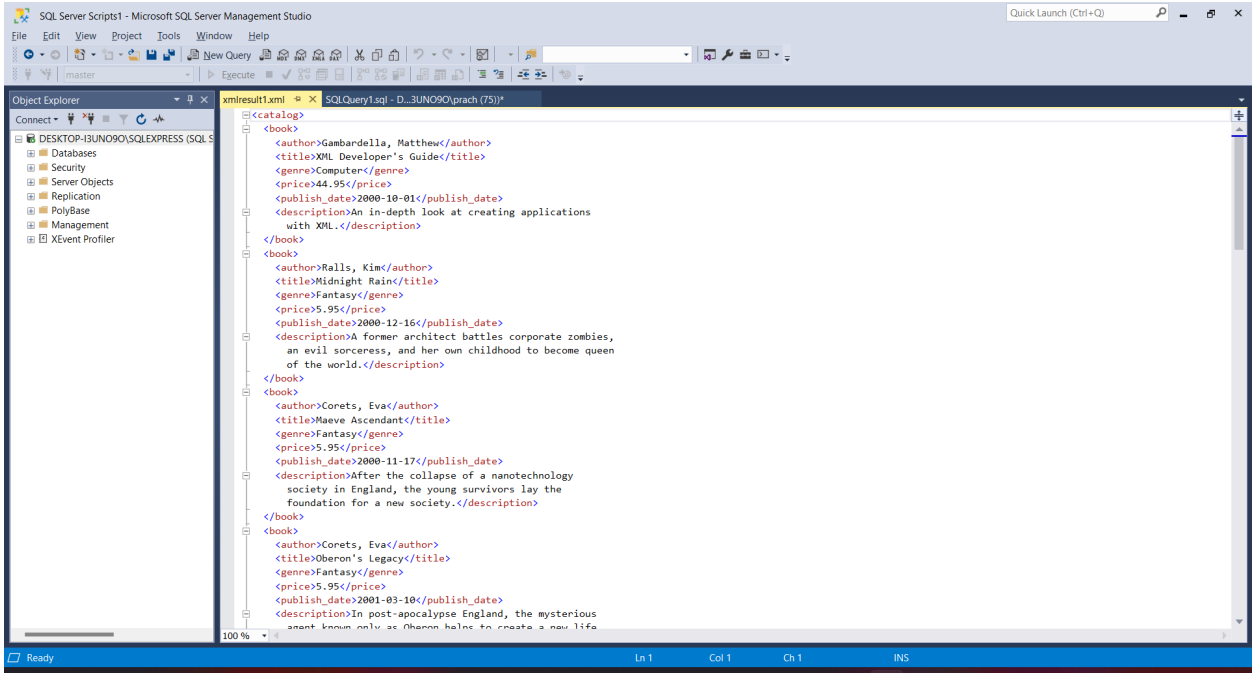
query is a relation, the output of an XML query can be an XML document.

1) XPath is a language for path expressions and is actually a building block for XQuery.

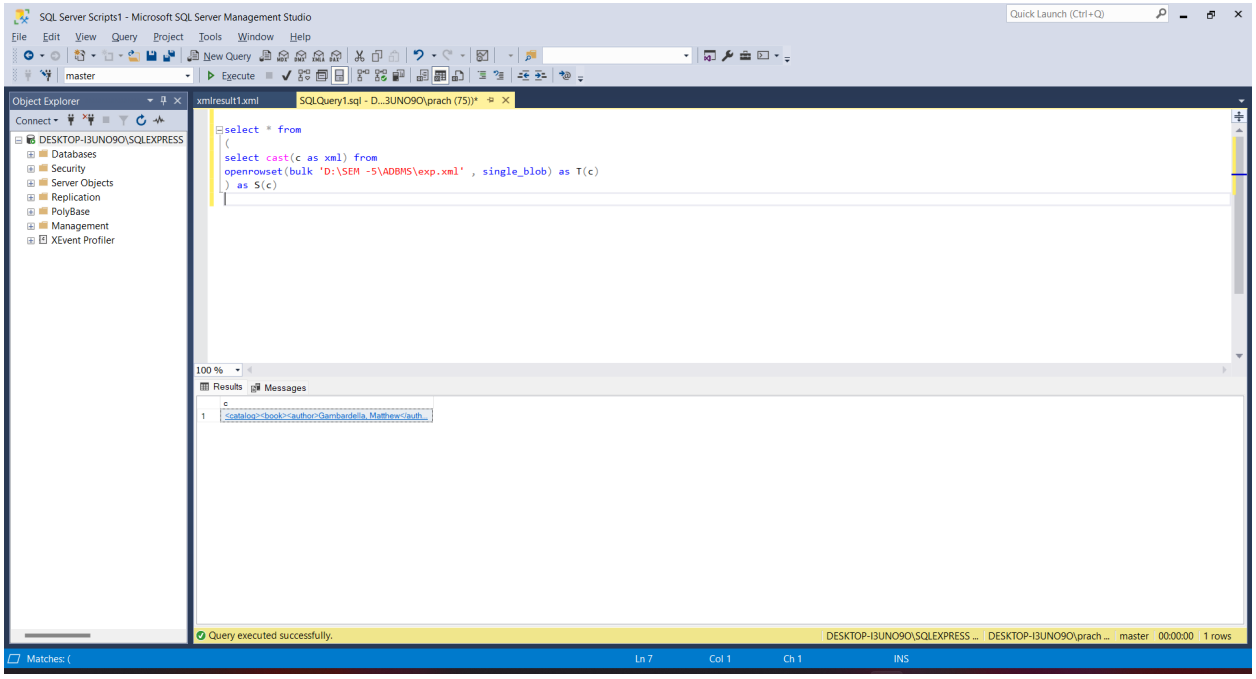
2) XQuery is the standard language for querying XML data. It is modeled after SQL but is significantly different, since it has to deal with nested XML data. XQuery also incorporates XPath expression.

CODE :

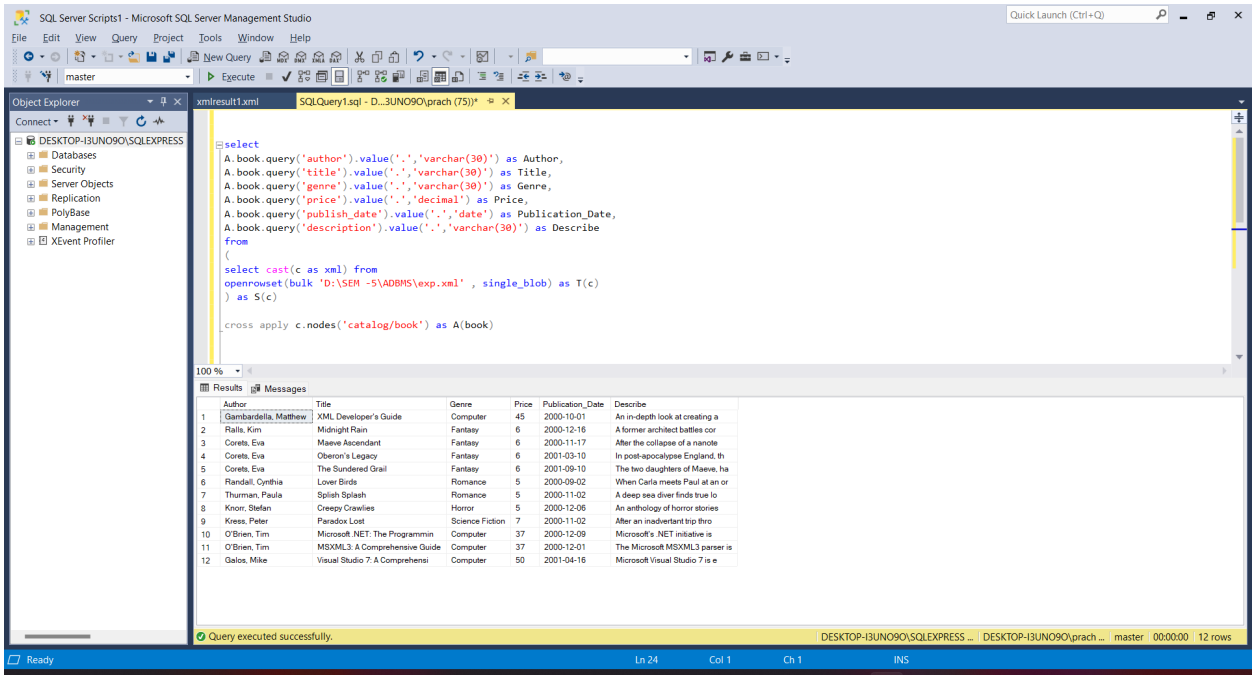
DATA



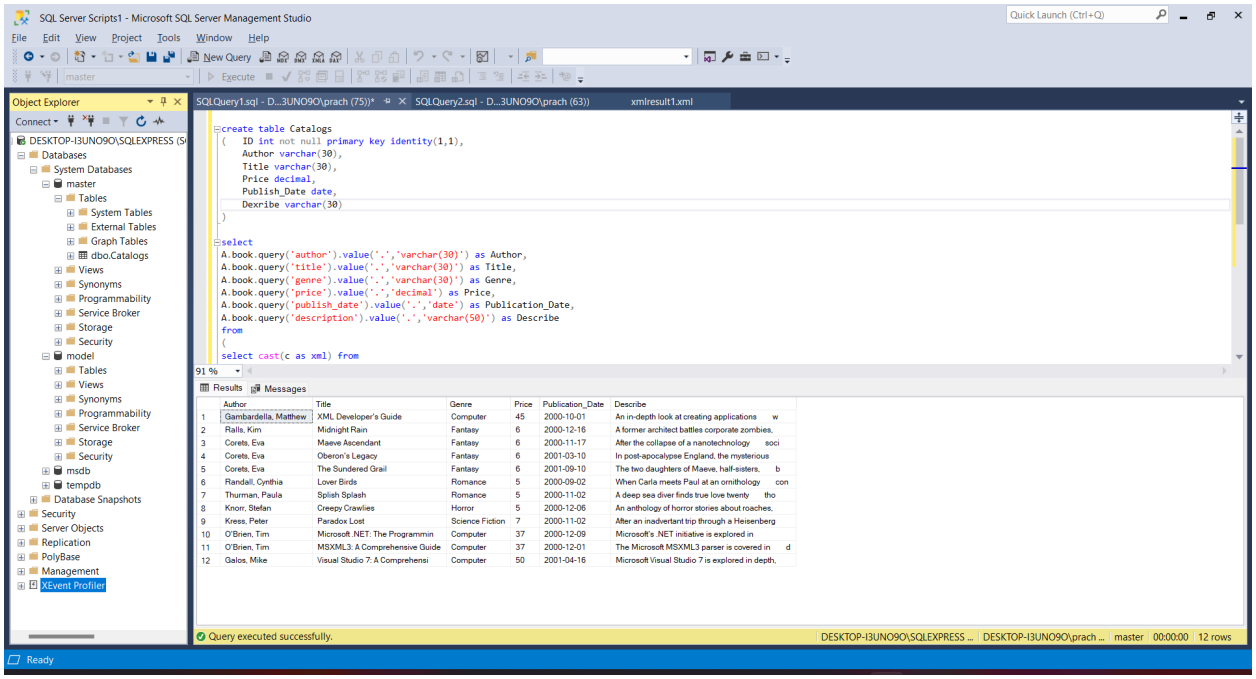
1) SELECT



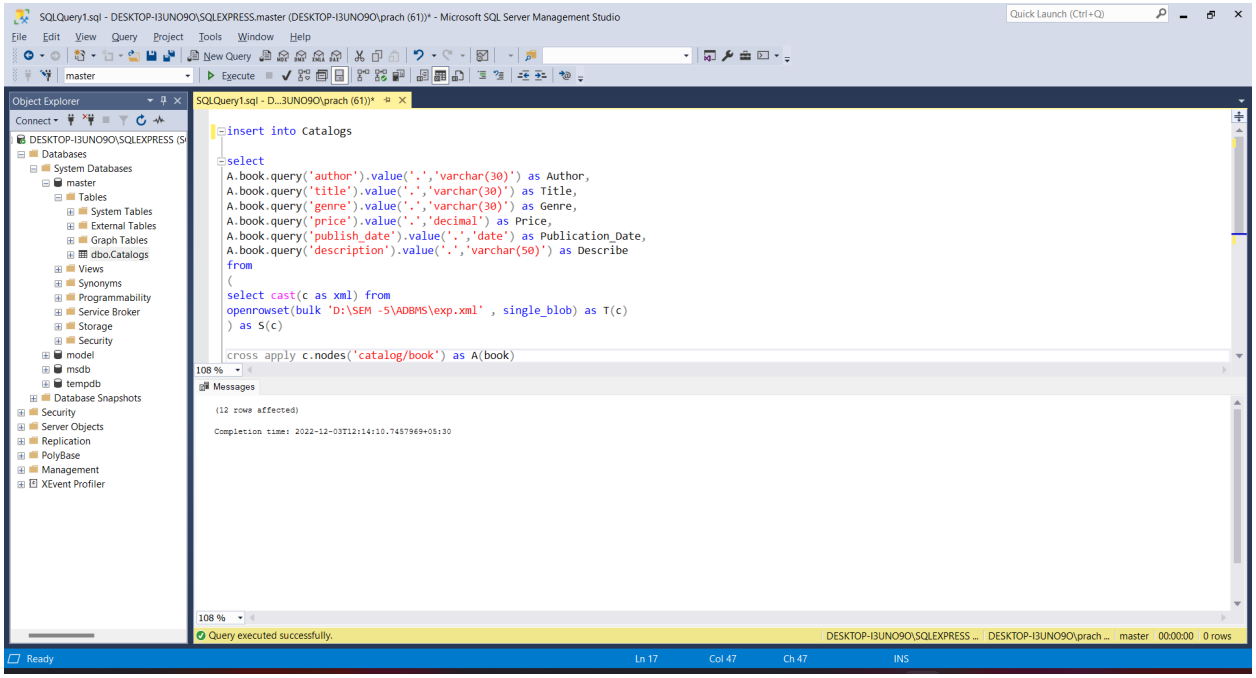
2) DISPLAY DATA IN FORM OF TABLE

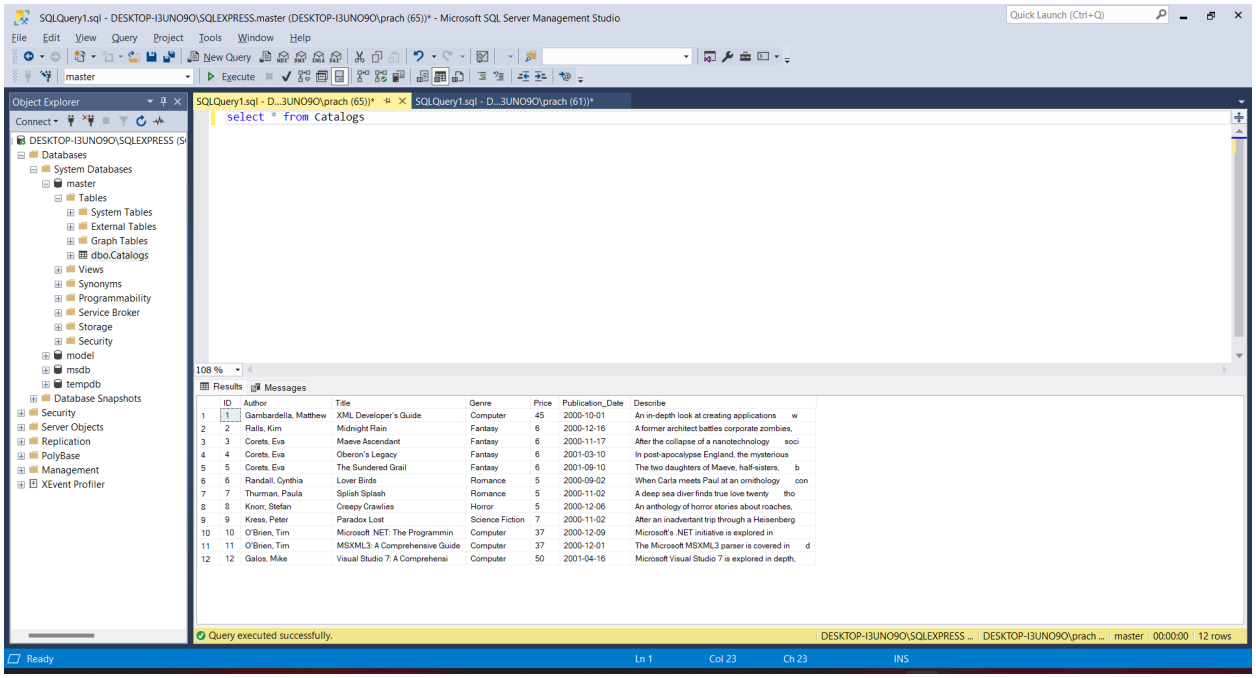


3) CREATE TABLE

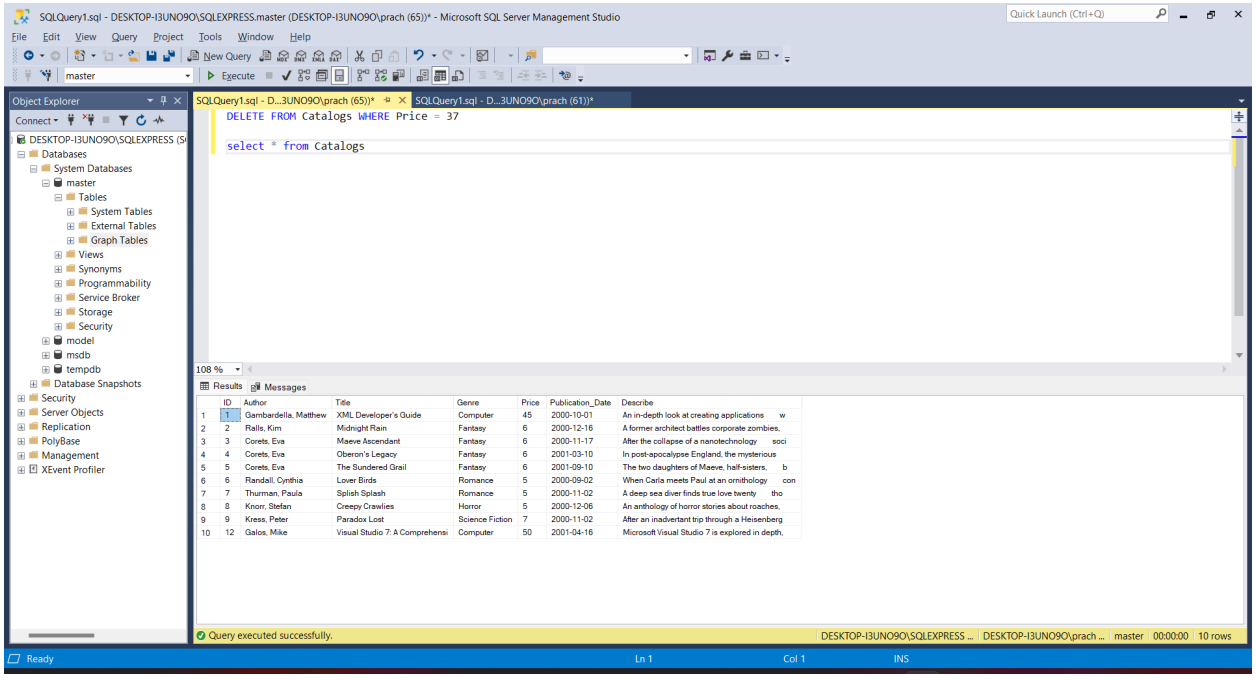


4)INSERT DATA IN TABLE





5) DELETE DATA FROM TABLE



CONCLUSION :-

After ~~implenting~~ implementing various queries on an XML databases, the results were observed and XML format (schema) was studied, along with XQuery which can be used to standardized language for combining documents, databases.

EXPERIMENT - 8

Data Handling using JSON

18/11/22

NAME - PRACHI PATEL
SAP ID - 60004200049
BRANCH - Computer Engineering

24/25

Experiment -8 - Data Handling using JSON

AIM :- Data handling using JSON (eg - Display user information from JSON file downloaded from Mobile)

JSON :-

JSON or Javascript Object Notation, is a human readable data interchange format specified in early 2000s. Even though JSON is based on a subset of the Javascript programming language standard, its completely language independent. JSON objects are associative containers, wherein a string key is mapped to a value (which can be a number, string, boolean, array, an empty value - null, or even another object) Almost any programming language has an implementation for this abstract data structure - Objects in Javascript, dictionaries in Python, hashtable in Java and C#, associative arrays in C++ and so on. JSON objects are easy for humans to understand and for machines to parse and generate.

Handling JSON Data :-

It is often observed that two terms - serialize and deserialize are associative with JSON objects. With the context of working with Javascript and JSON, in a nutshell, to get JSON value from Javascript is serialization and when it's another way (JSON to Javascript) is deserialization. The Javascript JSON object comprises methods using

which you can convert Javascript values to JSON format and JSON notation to Javascript values

For this purpose we need two methods.

1) JSON.stringify – JSON.stringify is used to convert Javascript values to JSON.

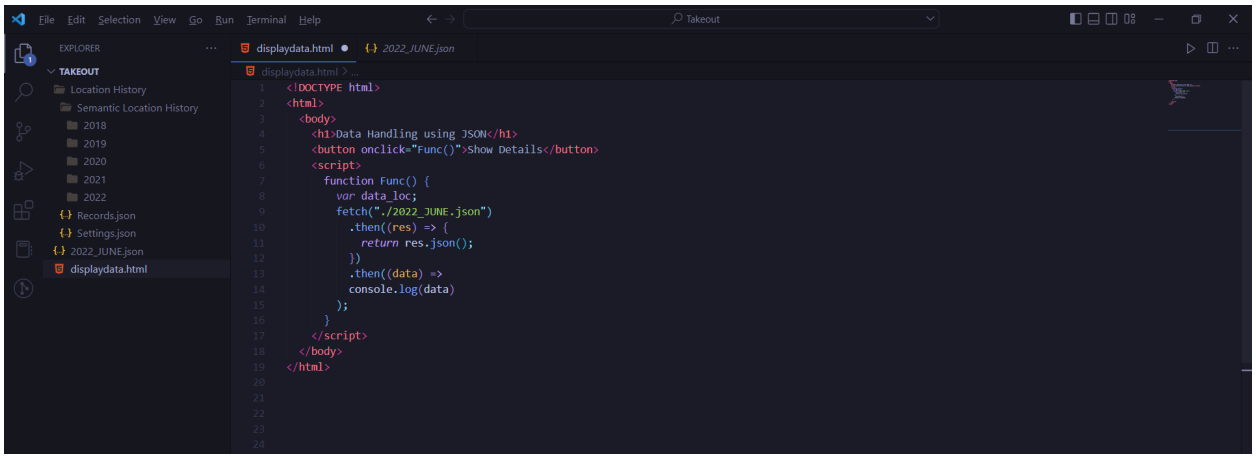
2) JSON.parse – JSON.parse is used to convert JSON notation to Javascript values.

3

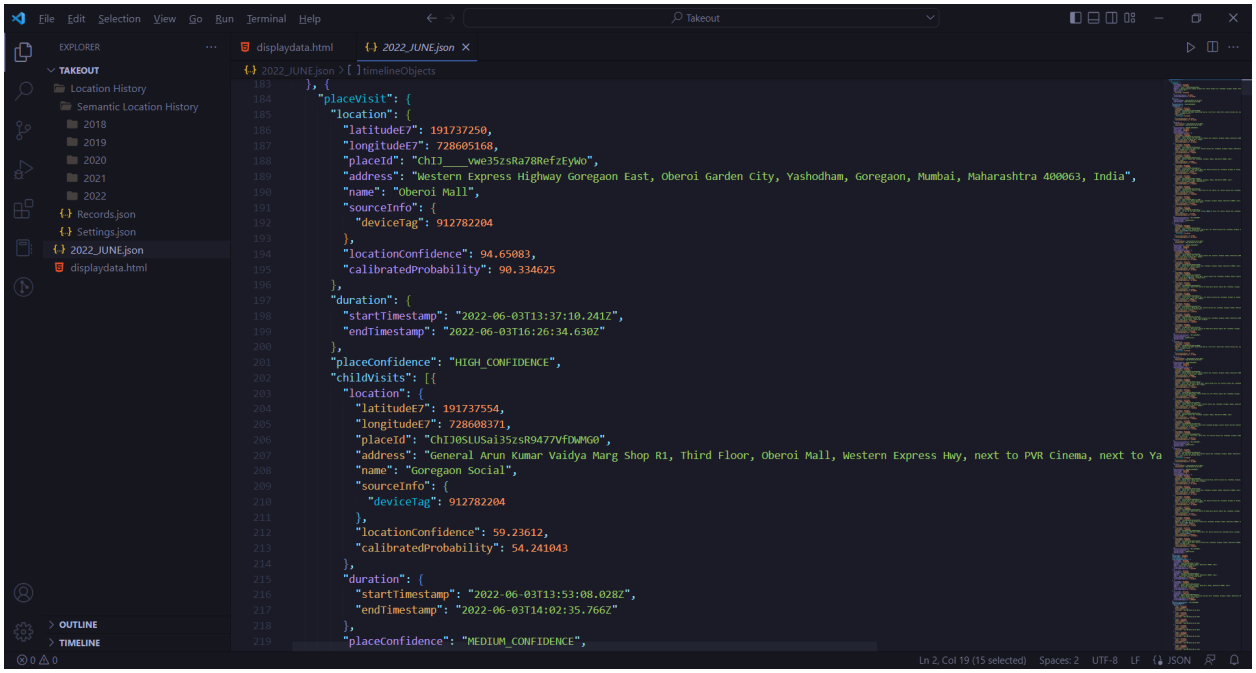
CODE :

Json data about various locations is collected from Google Takeout .
Using Google Takeout export Location history of Google Maps from your mobile in .json format.

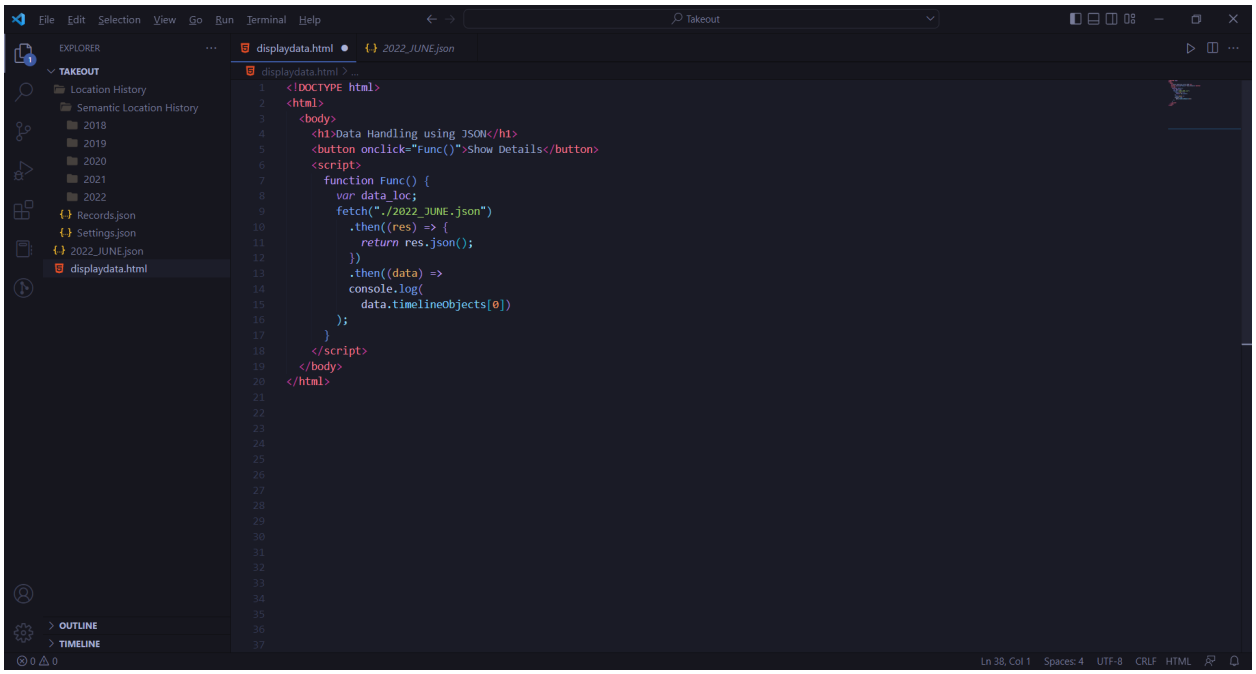
1)IMPORT JSON DATA IN HTML FILE



2) DATA :

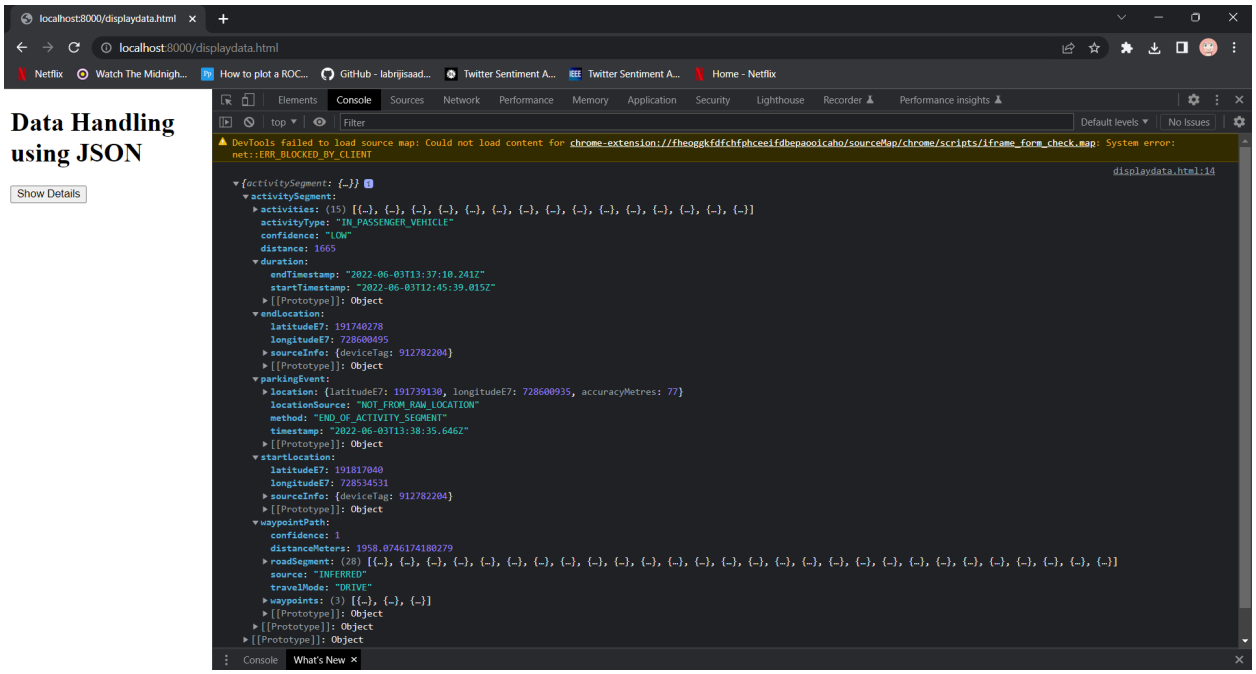


3) QUERY TO DISPLAY 1ST ACTIVITY SEGMENT OBJECT IN JSON DATA

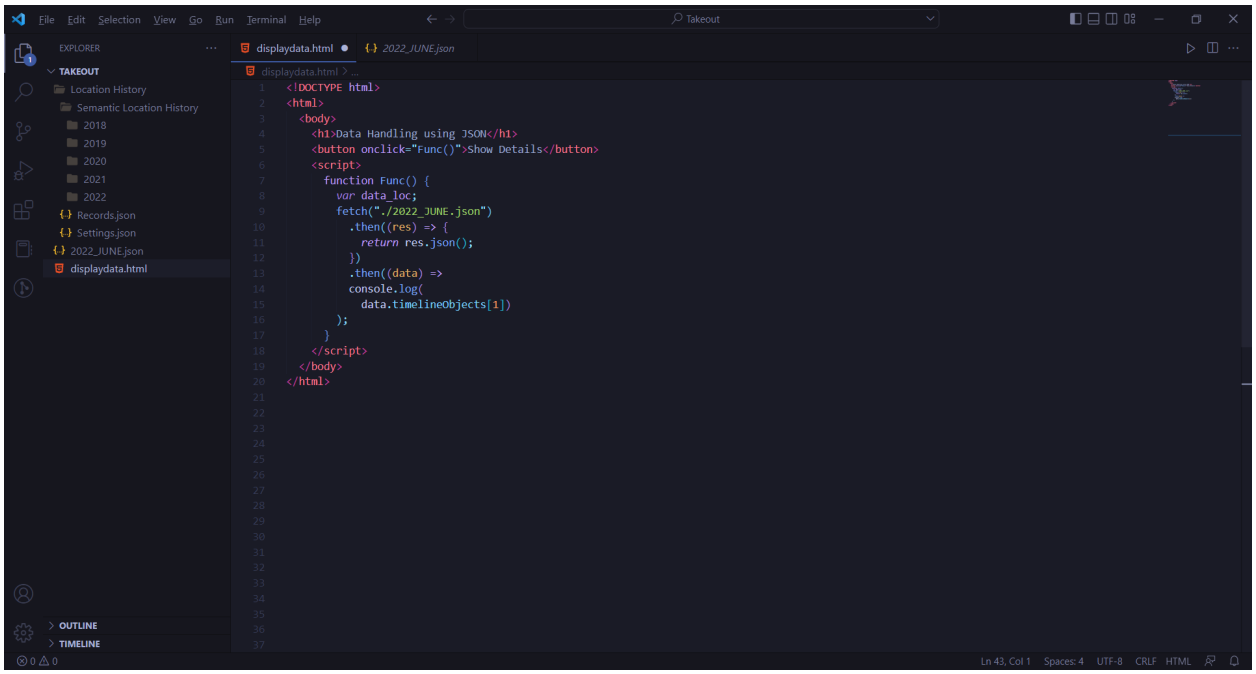


```
<!DOCTYPE html>
<html>
  <body>
    <h1>Data Handling using JSON</h1>
    <button onclick="Func()">Show Details</button>
    <script>
      function Func() {
        var data_loc;
        fetch("./2022_JUNE.json")
          .then((res) => {
            return res.json();
          })
          .then((data) =>
          console.log(
            data.timelineObjects[0])
          );
      }
    </script>
  </body>
</html>
```

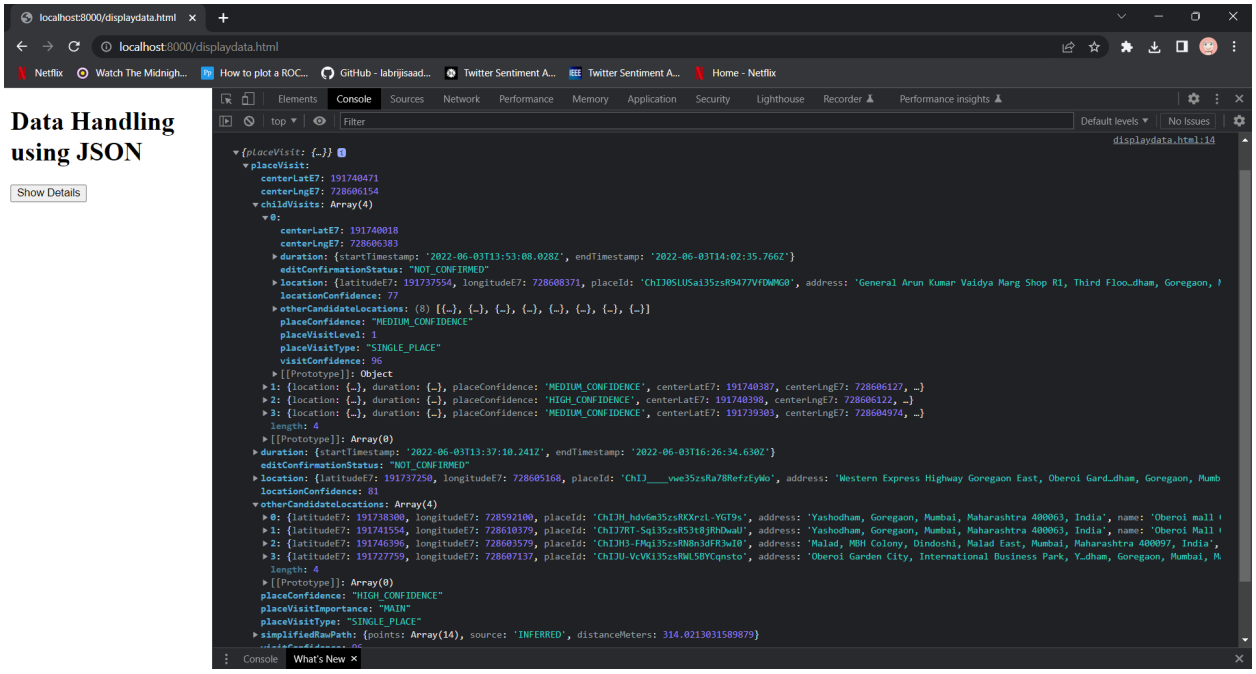
OUTPUT :



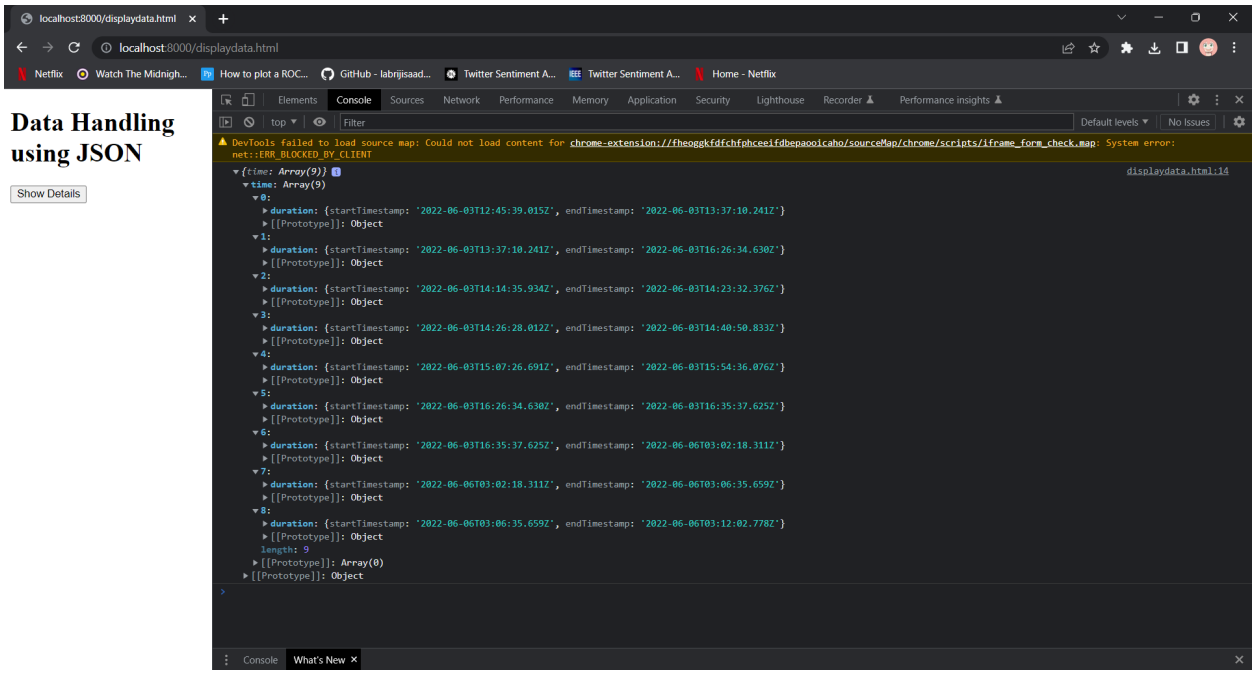
4) QUERY TO DISPLAY 1ST ACTIVITY SEGMENT OBJECT IN JSON DATA



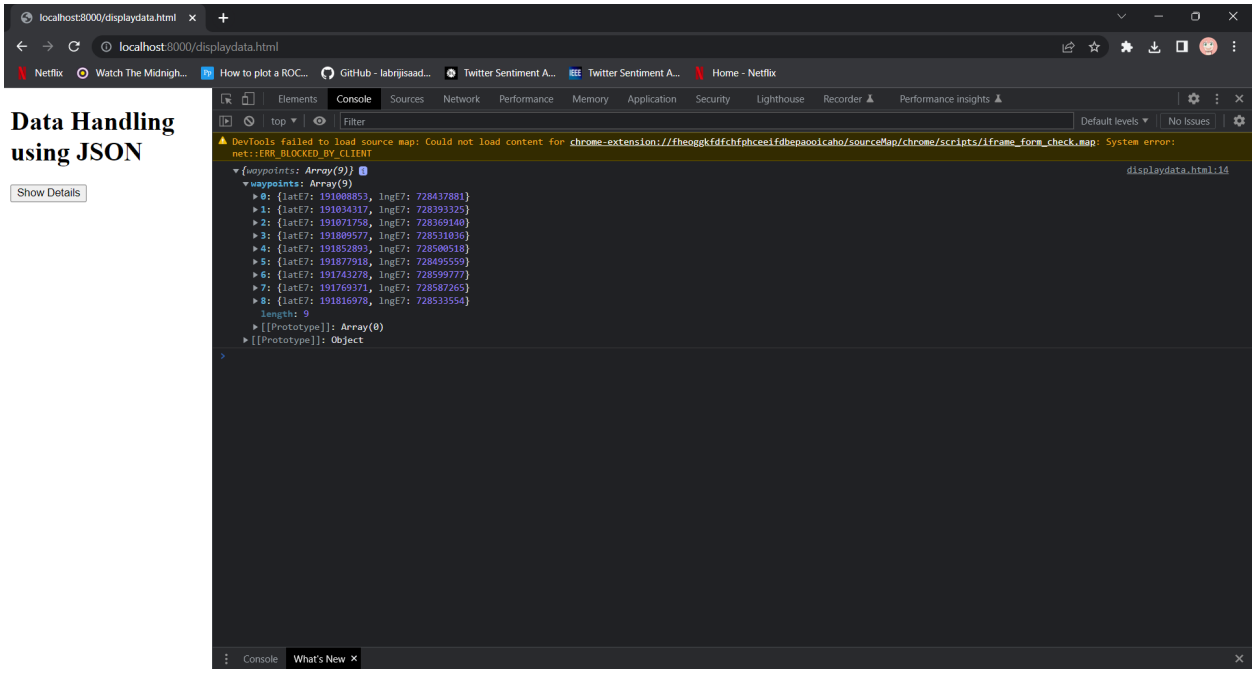
OUTPUT :



5) TEMPORAL QUERY



6)SPATIAL QUERY ON COORDINATES



CONCLUSION :-

After serializing and deserializing, JSON data obtained from google site we can conclude that Javascript methods like .stringify and .parse can be used to handle JSON data

EXPERIMENT - 9

Processing Spatial and Temporal data

18/11/22

24/25

NAME - PRACHI PATEL
SAP ID - 60004200049
BRANCH - Computer Engineering

Experiment - 9:

AIM :- Processing of spatial and temporal data

Spatial data :-
The introduction of ubiquitous networks and smart phones had led to the advent of location-based services. Customer expect information delivered based on, among other things, where they are. For example, a person touring the historic German city of Regensburg could recieve information about its buildings and parks via their mobile phone and language of choice.

Temporal data :-
some aspects of time is an important fact to remember for many applications. Banks, for example, want to remember what dates customers made payment on their loans. Airlines need to recall for the current and future days who will occupy seats on each flight. Thus, the management of time-varying, or temporal, data is availed when a database management system has built in temporal support.

Managing Spacial & Temporal data :-
A spatial database is a data management system for the collection, storage, manipulation and output of spacially related

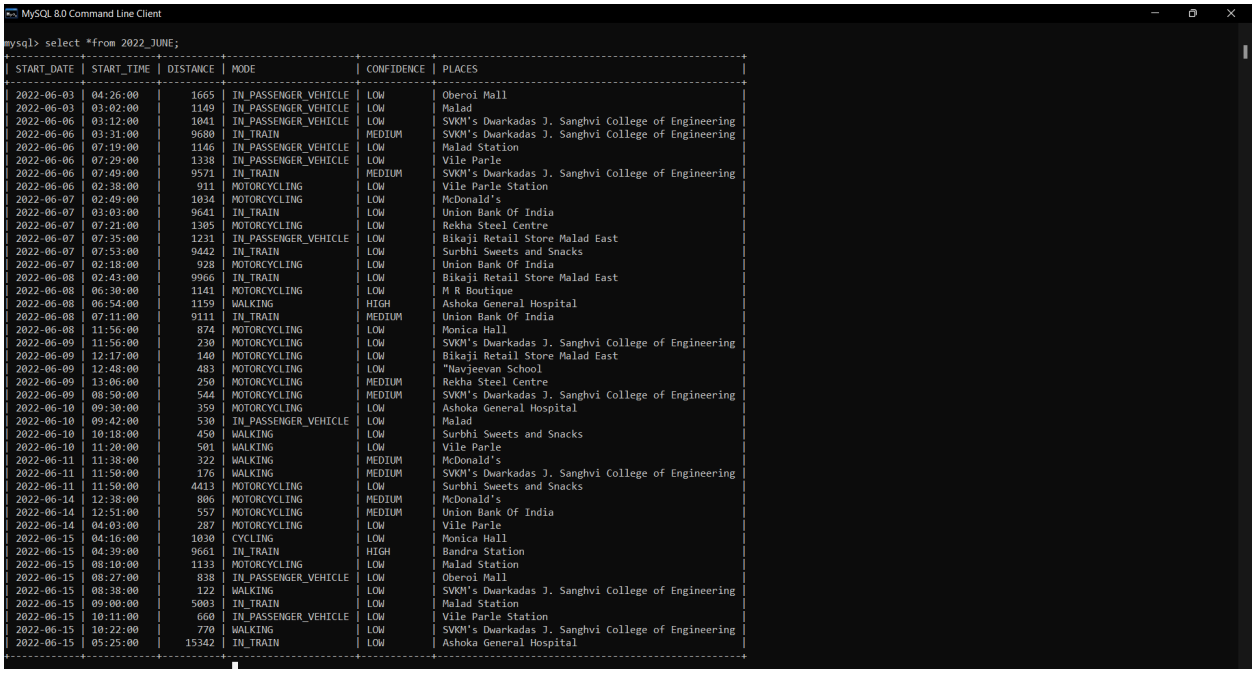
information. A theme refers to data describing a particular topic (eg - scenic lookouts, rivers, cities) and is the spatial counterpart of an entity. When a theme is presented on a screen or paper, it is commonly seen in conjunction with a map. Color may be used to indicate different themes.

A geographical object is an instance of a theme. Like an instance of an entity, it has set of attributes. In addition, it has spatial components that can describe both geometry and topology. Management of such data requires some data types to represent a point, line and region. SQL/MM also known as ISO 13249 is an extension of SQL to handle spatial data.

With temporal database, stored data have an associated time period indicating when the item was valid or stored in the database. Following are important data items for managing temporal data - Transaction time and valid time and bitemporal data.

CODE :

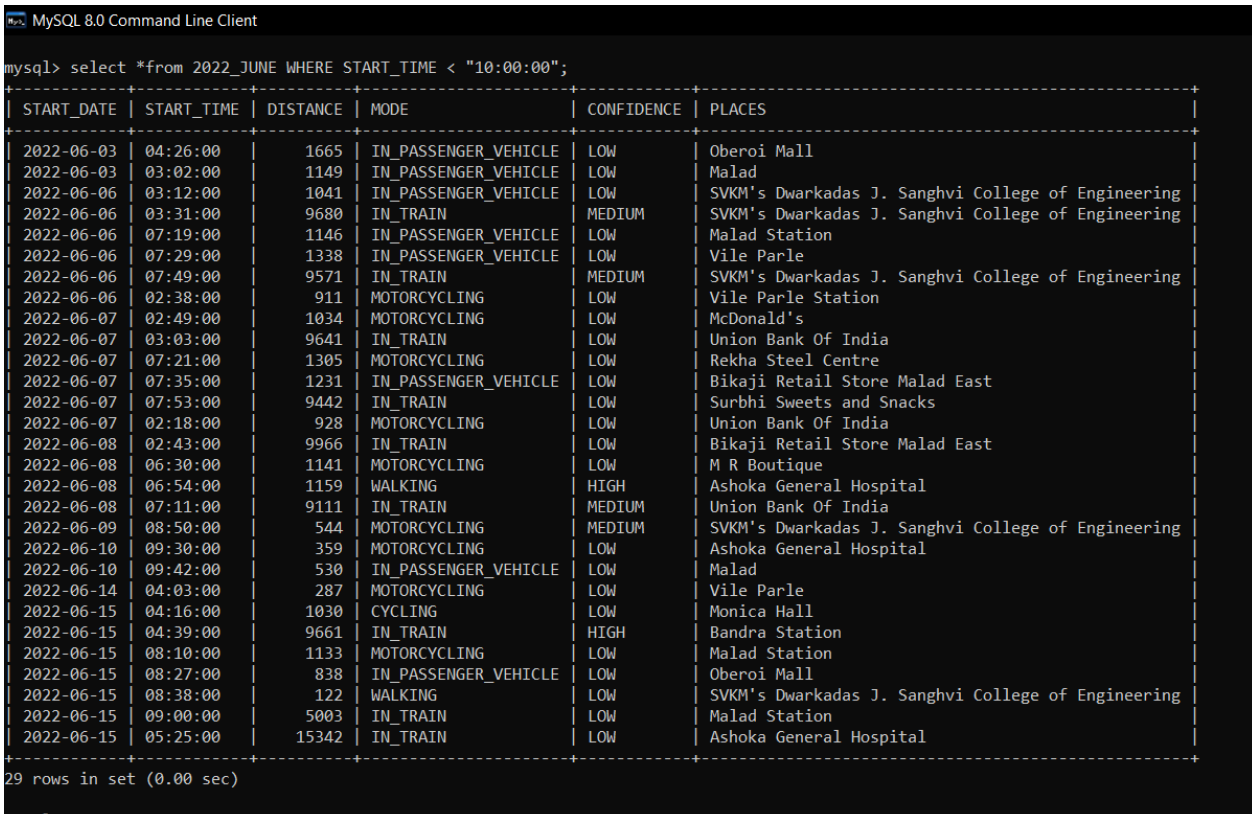
DATABASE:



```
MySQL 8.0 Command Line Client

mysql> select *from 2022_JUNE;
+------------+------------+----------+----------------------+------------+----------------------------------------------------+
| START_DATE | START_TIME | DISTANCE | MODE                 | CONFIDENCE | PLACES                                             |
+------------+------------+----------+----------------------+------------+----------------------------------------------------+
| 2022-06-03 | 04:26:00   |     1665 | IN_PASSENGER_VEHICLE | LOW        | Oberoi Mall                                        |
| 2022-06-03 | 03:02:00   |     1149 | IN_PASSENGER_VEHICLE | LOW        | Malad                                              |
| 2022-06-06 | 03:12:00   |     1041 | IN_PASSENGER_VEHICLE | LOW        | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-06 | 03:31:00   |     9680 | IN_TRAIN             | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-06 | 07:19:00   |     1146 | IN_PASSENGER_VEHICLE | LOW        | Malad Station                                      |
| 2022-06-06 | 07:29:00   |     1338 | IN_PASSENGER_VEHICLE | LOW        | Vile Parle                                         |
| 2022-06-06 | 07:49:00   |     9571 | IN_TRAIN             | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-06 | 02:38:00   |      911 | MOTORCYCLING         | LOW        | Vile Parle Station                                 |
| 2022-06-07 | 02:49:00   |     1034 | MOTORCYCLING         | LOW        | McDonald's                                         |
| 2022-06-07 | 03:03:00   |     9641 | IN_TRAIN             | LOW        | Union Bank Of India                                |
| 2022-06-07 | 07:21:00   |     1305 | MOTORCYCLING         | LOW        | Rekha Steel Centre                                 |
| 2022-06-07 | 07:35:00   |     1231 | IN_PASSENGER_VEHICLE | LOW        | Bikaji Retail Store Malad East                     |
| 2022-06-07 | 07:53:00   |     9442 | IN_TRAIN             | LOW        | Surbhi Sweets and Snacks                           |
| 2022-06-07 | 02:18:00   |      928 | MOTORCYCLING         | LOW        | Union Bank Of India                                |
| 2022-06-08 | 02:43:00   |     9966 | IN_TRAIN             | LOW        | Bikaji Retail Store Malad East                     |
| 2022-06-08 | 06:30:00   |     1141 | MOTORCYCLING         | LOW        | M R Boutique                                       |
| 2022-06-08 | 06:54:00   |     1159 | WALKING              | HIGH       | Ashoka General Hospital                            |
| 2022-06-08 | 07:11:00   |     9111 | IN_TRAIN             | MEDIUM     | Union Bank Of India                                |
| 2022-06-08 | 11:56:00   |      874 | MOTORCYCLING         | LOW        | Monica Hall                                        |
| 2022-06-09 | 11:56:00   |      230 | MOTORCYCLING         | LOW        | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-09 | 12:17:00   |      140 | MOTORCYCLING         | LOW        | Bikaji Retail Store Malad East                     |
| 2022-06-09 | 12:48:00   |      483 | MOTORCYCLING         | LOW        | "Navjeevan School                                  |
| 2022-06-09 | 13:06:00   |      250 | MOTORCYCLING         | MEDIUM     | Rekha Steel Centre                                 |
| 2022-06-09 | 08:50:00   |      544 | MOTORCYCLING         | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-10 | 09:30:00   |      359 | MOTORCYCLING         | LOW        | Ashoka General Hospital                            |
| 2022-06-10 | 09:42:00   |      530 | IN_PASSENGER_VEHICLE | LOW        | Malad                                              |
| 2022-06-10 | 10:18:00   |      450 | WALKING              | LOW        | Surbhi Sweets and Snacks                           |
| 2022-06-10 | 11:20:00   |      501 | WALKING              | LOW        | Vile Parle                                         |
| 2022-06-11 | 11:38:00   |      322 | WALKING              | MEDIUM     | McDonald's                                         |
| 2022-06-11 | 11:50:00   |      176 | WALKING              | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-11 | 11:50:00   |     4413 | MOTORCYCLING         | LOW        | Surbhi Sweets and Snacks                           |
| 2022-06-14 | 12:38:00   |      806 | MOTORCYCLING         | MEDIUM     | McDonald's                                         |
| 2022-06-14 | 12:51:00   |      557 | MOTORCYCLING         | MEDIUM     | Union Bank Of India                                |
| 2022-06-14 | 04:03:00   |      287 | MOTORCYCLING         | LOW        | Vile Parle                                         |
| 2022-06-15 | 04:16:00   |     1030 | CYCLING              | LOW        | Monica Hall                                        |
| 2022-06-15 | 04:39:00   |     9661 | IN_TRAIN             | HIGH       | Bandra Station                                     |
| 2022-06-15 | 08:10:00   |     1133 | MOTORCYCLING         | LOW        | Malad Station                                      |
| 2022-06-15 | 08:27:00   |      838 | IN_PASSENGER_VEHICLE | LOW        | Oberoi Mall                                        |
| 2022-06-15 | 08:38:00   |      122 | WALKING              | LOW        | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-15 | 09:00:00   |     5003 | IN_TRAIN             | LOW        | Malad Station                                      |
| 2022-06-15 | 10:11:00   |      660 | IN_PASSENGER_VEHICLE | LOW        | Vile Parle Station                                 |
| 2022-06-15 | 10:22:00   |      770 | WALKING              | LOW        | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-15 | 05:25:00   |    15342 | IN_TRAIN             | LOW        | Ashoka General Hospital                            |
+------------+------------+----------+----------------------+------------+----------------------------------------------------+
```

A) TEMPORAL QUERY



```
MySQL 8.0 Command Line Client

mysql> select *from 2022_JUNE WHERE START_TIME < "10:00:00";
+------------+------------+----------+----------------------+------------+----------------------------------------------------+
| START_DATE | START_TIME | DISTANCE | MODE                 | CONFIDENCE | PLACES                                             |
+------------+------------+----------+----------------------+------------+----------------------------------------------------+
| 2022-06-03 | 04:26:00   |     1665 | IN_PASSENGER_VEHICLE | LOW        | Oberoi Mall                                        |
| 2022-06-03 | 03:02:00   |     1149 | IN_PASSENGER_VEHICLE | LOW        | Malad                                              |
| 2022-06-06 | 03:12:00   |     1041 | IN_PASSENGER_VEHICLE | LOW        | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-06 | 03:31:00   |     9680 | IN_TRAIN             | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-06 | 07:19:00   |     1146 | IN_PASSENGER_VEHICLE | LOW        | Malad Station                                      |
| 2022-06-06 | 07:29:00   |     1338 | IN_PASSENGER_VEHICLE | LOW        | Vile Parle                                         |
| 2022-06-06 | 07:49:00   |     9571 | IN_TRAIN             | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-06 | 02:38:00   |      911 | MOTORCYCLING         | LOW        | Vile Parle Station                                 |
| 2022-06-07 | 02:49:00   |     1034 | MOTORCYCLING         | LOW        | McDonald's                                         |
| 2022-06-07 | 03:03:00   |     9641 | IN_TRAIN             | LOW        | Union Bank Of India                                |
| 2022-06-07 | 07:21:00   |     1305 | MOTORCYCLING         | LOW        | Rekha Steel Centre                                 |
| 2022-06-07 | 07:35:00   |     1231 | IN_PASSENGER_VEHICLE | LOW        | Bikaji Retail Store Malad East                     |
| 2022-06-07 | 07:53:00   |     9442 | IN_TRAIN             | LOW        | Surbhi Sweets and Snacks                           |
| 2022-06-07 | 02:18:00   |      928 | MOTORCYCLING         | LOW        | Union Bank Of India                                |
| 2022-06-08 | 02:43:00   |     9966 | IN_TRAIN             | LOW        | Bikaji Retail Store Malad East                     |
| 2022-06-08 | 06:30:00   |     1141 | MOTORCYCLING         | LOW        | M R Boutique                                       |
| 2022-06-08 | 06:54:00   |     1159 | WALKING              | HIGH       | Ashoka General Hospital                            |
| 2022-06-08 | 07:11:00   |     9111 | IN_TRAIN             | MEDIUM     | Union Bank Of India                                |
| 2022-06-09 | 08:50:00   |      544 | MOTORCYCLING         | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-10 | 09:30:00   |      359 | MOTORCYCLING         | LOW        | Ashoka General Hospital                            |
| 2022-06-10 | 09:42:00   |      530 | IN_PASSENGER_VEHICLE | LOW        | Malad                                              |
| 2022-06-14 | 04:03:00   |      287 | MOTORCYCLING         | LOW        | Vile Parle                                         |
| 2022-06-15 | 04:16:00   |     1030 | CYCLING              | LOW        | Monica Hall                                        |
| 2022-06-15 | 04:39:00   |     9661 | IN_TRAIN             | HIGH       | Bandra Station                                     |
| 2022-06-15 | 08:10:00   |     1133 | MOTORCYCLING         | LOW        | Malad Station                                      |
| 2022-06-15 | 08:27:00   |      838 | IN_PASSENGER_VEHICLE | LOW        | Oberoi Mall                                        |
| 2022-06-15 | 08:38:00   |      122 | WALKING              | LOW        | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-15 | 09:00:00   |     5003 | IN_TRAIN             | LOW        | Malad Station                                      |
| 2022-06-15 | 05:25:00   |    15342 | IN_TRAIN             | LOW        | Ashoka General Hospital                            |
+------------+------------+----------+----------------------+------------+----------------------------------------------------+
29 rows in set (0.00 sec)
```

B)SPATIAL QUERY ON DISTANCE

```
mysql> select *from 2022_JUNE WHERE DISTANCE > 2000;
+------------+------------+----------+-------------+------------+-----------------------------------------------+
| START_DATE | START_TIME | DISTANCE | MODE        | CONFIDENCE | PLACES                                        |
+------------+------------+----------+-------------+------------+-----------------------------------------------+
| 2022-06-06 | 03:31:00   |     9680 | IN_TRAIN    | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-06 | 07:49:00   |     9571 | IN_TRAIN    | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-07 | 03:03:00   |     9641 | IN_TRAIN    | LOW        | Union Bank Of India                           |
| 2022-06-07 | 07:53:00   |     9442 | IN_TRAIN    | LOW        | Surbhi Sweets and Snacks                      |
| 2022-06-08 | 02:43:00   |     9966 | IN_TRAIN    | LOW        | Bikaji Retail Store Malad East                |
| 2022-06-08 | 07:11:00   |     9111 | IN_TRAIN    | MEDIUM     | Union Bank Of India                           |
| 2022-06-11 | 11:50:00   |     4413 | MOTORCYCLING | LOW        | Surbhi Sweets and Snacks                      |
| 2022-06-15 | 04:39:00   |     9661 | IN_TRAIN    | HIGH       | Bandra Station                                |
| 2022-06-15 | 09:00:00   |     5003 | IN_TRAIN    | LOW        | Malad Station                                 |
| 2022-06-15 | 05:25:00   |    15342 | IN_TRAIN    | LOW        | Ashoka General Hospital                       |
+------------+------------+----------+-------------+------------+-----------------------------------------------+
10 rows in set (0.00 sec)

mysql>
```

C) SPATIAL QUERY ON PLACES

```
mysql> select *from 2022_JUNE WHERE PLACES = "SVKM's Dwarkadas J. Sanghvi College of Engineering";
+------------+------------+----------+---------------------+------------+-----------------------------------------------+
| START_DATE | START_TIME | DISTANCE | MODE                | CONFIDENCE | PLACES                                        |
+------------+------------+----------+---------------------+------------+-----------------------------------------------+
| 2022-06-06 | 03:12:00   |     1041 | IN_PASSENGER_VEHICLE | LOW        | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-06 | 03:31:00   |     9680 | IN_TRAIN            | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-06 | 07:49:00   |     9571 | IN_TRAIN            | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-09 | 11:56:00   |      230 | MOTORCYCLING        | LOW        | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-09 | 08:50:00   |      544 | MOTORCYCLING        | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-11 | 11:50:00   |      176 | WALKING             | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-15 | 08:38:00   |      122 | WALKING             | LOW        | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-15 | 10:22:00   |      770 | WALKING             | LOW        | SVKM's Dwarkadas J. Sanghvi College of Engineering |
+------------+------------+----------+---------------------+------------+-----------------------------------------------+
8 rows in set (0.00 sec)
```

D)QUERY ON CONFIDENCE

```
mysql> select *from 2022_JUNE WHERE CONFIDENCE="MEDIUM";
+------------+------------+----------+-------------+------------+-----------------------------------------------+
| START_DATE | START_TIME | DISTANCE | MODE        | CONFIDENCE | PLACES                                        |
+------------+------------+----------+-------------+------------+-----------------------------------------------+
| 2022-06-06 | 03:31:00   |     9680 | IN_TRAIN    | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-06 | 07:49:00   |     9571 | IN_TRAIN    | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-08 | 07:11:00   |     9111 | IN_TRAIN    | MEDIUM     | Union Bank Of India                           |
| 2022-06-09 | 13:06:00   |      250 | MOTORCYCLING | MEDIUM     | Rekha Steel Centre                            |
| 2022-06-09 | 08:50:00   |      544 | MOTORCYCLING | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-11 | 11:38:00   |      322 | WALKING     | MEDIUM     | McDonald's                                    |
| 2022-06-11 | 11:50:00   |      176 | WALKING     | MEDIUM     | SVKM's Dwarkadas J. Sanghvi College of Engineering |
| 2022-06-14 | 12:38:00   |      806 | MOTORCYCLING | MEDIUM     | McDonald's                                    |
| 2022-06-14 | 12:51:00   |      557 | MOTORCYCLING | MEDIUM     | Union Bank Of India                           |
+------------+------------+----------+-------------+------------+-----------------------------------------------+
9 rows in set (0.00 sec)
```

CONCLUSION :-

After implementing data handling on spatial & temporal database various data management technique can be studied for spatial and temporal data

EXPERIMENT - 10

Case Study on Database Security Issues

01/11/22

NAME - PRACHI PATEL
SAP ID - 60004200049
BRANCH - Computer Engineering

24
25

Experiment - 10 :-

AIM - Case study on Database Security Issues.
Patent is related to - Multi-factor profile and security fingerprint analysis

Description -
In multi factor identity fingerprinting, at least a subset of a user's profile stored online becomes bound to that user. The information system may authenticate or verify a user's identity, query of user's profile. Thus a large set of security checks may be monitored. This information may be analyzed to identify patterns of security attacks / threat monitoring or for identity management.

Claims :-
Based on at least a profile associated with one or more users, retrieving at least one record from a data store of records relating to the historical actions of the one or more users and creating an identity fingerprint. The identity fingerprint has a plurality of selectable modes (a first mode associated with a patient, a second mode with family and third mode with caretaker). Each having different sets of access restrictions

1) A method to generate an identity fingerprint
2) The method of claim 1, wherein the profile further comprises a

a specification of access restrictions exposing at least a part of set of data for querying.

3) The method of claim 1, wherein the profile enumerates at least one data field comprising an identifier and an indicator of whether user has enabled that field by a query applied to identity fingerprint.

4) The method of claim 1, wherein the profiles specifies at least one of the plurality of selectable nodes.

5) The method of claim 1, wherein at least one data source is to be used in maintaing the data store.

6) The method of claim 5, wherein one or more users have enabled at least one data source to be used in maintaing the data store.

7) The method of claim 1, wherein the generating the identity fingerprint comprises compressing the set of data.

8) The method of claim 1, wherein the generating the identity fingerprint comprises encrypting the set of data.

9) The method of claim 1, wherein the identity fingerprint includes summary of the historical activities of the one or more users.

10) The method of claim 1, wherein the second or third mode allows access to a subset of set of data.

11) A system for creating identity fingerprint includes : a memory & processor, The memory ~~constraints~~ contains instructions that when executed by processor causes the following operations to be executed,

12) The system of claim 11, wherein the profile further consist of specification of access restrictions associated with set of data exposing some part of data for query.

13) The system of claim 11, wherein the identity fingerprint comprises a plurality of selectable modes with different sets of access restrictions

14) The system of claim 13, wherein the profile specifies the polarity of selectable modes.

15) A non-transitory computer-readable medium having stored thereon computer-executable instructions configured to program a computing device to perform operations comprising -

16) The medium of claim 15, wherein the profile specifies what data source is used to maintain the data store.

17) The medium of claim 16, wherein at least one data source is to be used in maintaining the data store is any one or more of - cellular activity and plan data.

18) The medium of claim 15, wherein the generating identity fingerprint comprises of compressing set of data.

Conclusion :-

So, it is clear that a security fingerprint architecture has been revealed. A behavioural factor of variables that store a history of events connected to one or more individuals make up security fingerprint. One or modes specify the accessible disclose the security fingerprint's data. A variety of primitive actions are supported by security fingerprints, enabling the execution of set operations. Security fingerprints may be used alone or in conjunction with outside data sources for authentication, advertising and other purposes.