



**NAME : PRACHI PATEL**

**SAP ID: 60004200049**

**BRANCH : COMPUTER ENGINEERING(A2)**

**SUBJECT : Advanced DataBase Management System**

## **EXPERIMENT LIST**

<b>Experiment No.</b>	<b>Title</b>	<b>Page No.</b>
<b>1</b>	Case Study on Professional & Commercial Database	<b>1</b>
<b>2</b>	Optimize B and B+ tree	<b>5</b>
<b>3</b>	Simulate Query optimization	<b>22</b>
<b>4</b>	Query Monitoring	<b>40</b>
<b>5</b>	Fragmentation (Range , List , Hash & Key)	<b>46</b>
<b>6</b>	2 Phase & 3 Phase Commit Protocol	<b>53</b>
<b>7</b>	Query Execution XML Database	<b>63</b>
<b>8</b>	Data Handling using JSON	<b>69</b>
<b>9</b>	Study of Spatial and TemporalData	<b>76</b>
<b>10</b>	Case Study on Database Security Issues	<b>81</b>

## 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

D

Database used - Scylla DB

Scylla DB 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 Scylla DB 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 <sup>mix of</sup> reads and writes.

### Company - OLA

#### Features :-

- Data and metadata storage
- load 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 MySQL 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 data model 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.

MongoDB 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 <sup>Mongo</sup> DB.
- 3) On scaling point of view MongoDB clusters replica sets and uses smart routing instances whereas in <sup>Scylla</sup> DB we need to keep adding <sup>on</sup> nodes.
- 4) As mentioned earlier that <sup>it</sup> 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 not have performance 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 tools 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 light-weight 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. MongoDB Atlas can be used with any of leading cloud service providers.

#### Conclusion :-

We have adopted both the database and we have a clear view on choosing mongoDB as our data is not needed to be structured and needs more <sup>flexibility</sup> and mongoDB provides 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 Bt 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

## Intermediate 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 -  $O(\log n)$

Insertion Time complexity -  $O(\log n)$

Deletion Time complexity -  $O(\log n)$

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

Bt 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 Bt tree, records (data) can only be stored on the leaf node of a Bt tree while internal nodes can store key values. The leaf node of a Bt tree are linked together in form of singly linked lists to make the search queries more efficient. Bt 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 Bt tree are stored in main memory whereas leaf nodes are stored in secondary memory.

Advantages of Bt 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 Bt tree sequentially as well as directly.

	Root	Internal Node	Leaf Node
Max children	$m$	$m$	$n$
Min Blank 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 = BTTreeNode(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()
```

The screenshot shows a Python IDE interface with a code editor and a terminal window.

**Code Editor:**

```
main.py
58     return self.search_key(k, x.child[1])
59
60 else:
61     return self.search_key(k, self.root)
62
63 def main():
64     degree = int(input("Enter degree of the tree: "))
65     B = BTTree(degree)
66     number_of_nodes = int(input("Enter number of nodes: "))
67     for i in range(number_of_nodes):
68         node = int(input("Enter node element: "))
69         B.insert((i, 2 * i))
70
71 B.print_tree(B.root)
72 element_to_search = int(input("Enter element to search: "))
73 dt = datetime.now()
74 ts = datetime.timestamp(dt)
75 if B.Search_key(element_to_search) is not None:
76     print("\nFound!")
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
```

**Terminal Output:**

```
Enter degree of the tree: 15
Enter number of nodes: 25
Level 0 25:[(0, 0) (1, 2) (2, 4) (3, 6) (4, 8) (5, 10) (6, 12) (7, 14) (8, 16) (9, 18) (10, 20) (11, 22) (12, 24) (13, 26) (14, 28) (15, 30) (16, 32) (17, 34) (18, 36) (19, 38) (20, 40) (21, 42) (22, 44) (23, 46) (24, 48)]
Enter element to search: 12
found
time taken:  0.0009391307830810547

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

## 2) B+ TREE

```
# B+ tree 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)
```

The screenshot shows a Jupyter Notebook interface with the following details:

- Toolbar:** Includes icons for Run, Debug, Stop, Share, Save, and a Beautify button.
- File Menu:** Language (Python 3), Help, and Settings.
- Code Cell:** The code is named `main.py` and defines a `B+ tree` class. It includes imports for `math` and `datetime`, and a class definition with an `__init__` method that initializes `order`, `values`, `keys`, `nextkey`, `parent`, and `check_leaf`. A comment indicates the insertion point at the leaf node.
- Output Cell:** Shows the execution of the code. It first prints the number of nodes (15) and their values (10 through 14 and 2). It then prompts for a key and value to search for, receives the input '12' and '5', and outputs 'Not found'. Finally, it shows the time taken for the operation: 7.081031799316406e-05.
- Bottom Status:** Displays the message "...Program finished with exit code 0" and "Press ENTER to exit console."

### Conclusion :-

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

## EXPERIMENT - 3

### Simulate Query optimization

21/10/22

D  
24  
25

NAME - PRACHI PATEL

SAP ID - 60004200049

BRANCH - Computer

Engineering

Experiment - 3 :-

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

Theory :-

Query optimization is of great importance for the performance of a realtime / relational 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 intermediate 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 in numerical form for each step of a feasible plan for a given query and environment and then discards these values together to get a cost estimate for the plan or possible strategy.

## TABLE LEVEL OPTIMIZATION

### 1) SELECT QUERY

The screenshot shows the MySQL Workbench interface with a SQL editor containing the following query:

```

USE sakila;
select * from actor where upper(last_name) like "%GEN%";
```

The results pane displays the execution plan, which is a "Full Table Scan" for the "actor" table. The plan indicates a query cost of 20.25 and 200 rows.

### BEFORE OPTIMIZING

The screenshot shows the MySQL Workbench interface with the same SQL query as above. The results pane now displays "Query Statistics" for the query.

**Timing (as measured at client side):**  
Execution time: 0:00:0.000000000

**Timing (as measured by the server):**  
Execution time: 0:00:0.000495000  
Table lock wait time: 0:00:0.000003000

**Errors:**  
Had Errors: NO  
Warnings: 0

**Rows Processed:**  
Rows affected: 0  
Rows sent to client: 4  
Rows examined: 200

**Temporary Tables:**  
Temporary disk tables created: 0  
Temporary tables created: 0

**Joins per Type:**  
Full table scans (Select\_scan): 1  
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:**  
Sorts using (Sort::covg): 0  
Sort merge passes (Sort::merge\_passes): 0  
Sorts with ranges (Sort::range): 0  
Sorts with table scans (Sort::scan): 0

**Index Usage:**  
No Index used

**Other Info:**  
Event Id: 92  
Thread Id: 60

## AFTER OPTIMIZING

```

USE sakila;
select * from actor where upper(last_name) like '%GEN%';
optimize table actor;

```

**Query Statistics**

**Timing (as measured at client side):**  
Execution time: 0:00:0.0620000

**Timing (as measured by the server):**  
Execution time: 0:00:0.0573850  
Table lock wait time: 0:00:0.00003900

**Errors:**  
Had Errors: NO  
Warnings: 0

**Rows Processed:**  
Rows affected: 0  
Rows sent to clients: 0  
Rows examined: 0

**Temporary Tables:**  
Temporary tables created: 0  
Temporary tables created: 0

**Other Info:**  
Event Id: 106  
Thread Id: 60

**Index Usage:**  
At least one index was used

## 2) NESTED

```

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 file_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

1.2.26 nested loop  
100 rows  
1.8499999999999999 category  
Full Table Scan

10.41 nested loop  
62 rows  
10.41 film\_category  
Non-Unique Key Lookup  
fk\_film\_category\_category

85.14 nested loop  
478 rows  
72.88 nested loop  
4 rows  
72.88 inventory  
idx\_fk\_film\_id  
Non-Unique Key Lookup

670.11 nested loop  
1.67K rows  
584.97 nested loop  
3 rows  
584.97 rental  
idx\_fk\_inventory\_id  
Non-Unique Key Lookup

1483.11 nested loop  
599 rows  
477.73899999999995 customer  
PRIMARY  
Unique Key Lookup

## BEFORE OPTIMIZING

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following query:
 

```

1 • USE sakila;
2 • select * from actor where upper(last_name) like "%GEN%" ;
3 • SELECT concat(first_name, " ",last_name) AS name, email
4   from customer where customer_id IN
5   (select customer_id from rental where inventory_id in
6     (select inventory_id from inventory where film_id in
7       (select file_id from film_category join category using (category_id) where category.name = "Action")));
      
```
- Query Statistics:**
  - Timing (as measured at client side):** Execution time: 0:00:0.0000000
  - Timing (as measured by the server):** Execution time: 0:00:0.0051460
  - Rows Processed:** Rows affected: 0, Rows sent to client: 0, Rows examined: 2616
  - Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 1
- Execution Plan:** Shows the execution plan for the query.

## AFTER OPTIMIZING

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the same query as before, but with an additional line:
 

```

8 • optimize table customer , rental , inventory , category ;
      
```
- Query Statistics:**
  - Timing (as measured at client side):** Execution time: 0:00:0.6250000
  - Timing (as measured by the server):** Execution time: 0:00:0.6126820
  - Rows Processed:** Rows affected: 0, Rows sent to client: 0, Rows examined: 0
  - Temporary Tables:** Temporary disk tables created: 0
- Execution Plan:** Shows the execution plan for the query.

### 3) LEFT JOIN

MySQL Workbench Screenshot showing the execution plan for a left join query:

```

6   (select inventory_id from inventory where film_id in
7     (select film_id from film_category join category using (category_id) where category.name = "Action")));
8
9 •  select stf.first_name , stf.last_name , ad.address , ad.district, ad.postal_code, ad.city_id
10    from staff stf
11   left join address ad
12     on stf.address_id = ad.address_id;

```

The execution plan shows:

- Full Table Scan** on **staff** (3.2 rows)
- Unique Key Lookup** on **address** (0.7 rows)
- query\_block #1** (3.9 rows)
- nested loop**
- Result: 2 rows

### BEFORE OPTIMIZING

MySQL Workbench Screenshot showing the Query Statistics panel for the same query:

```

6   (select inventory_id from inventory where film_id in
7     (select film_id from film_category join category using (category_id) where category.name = "Action")));
8
9 •  select stf.first_name , stf.last_name , ad.address , ad.district, ad.postal_code, ad.city_id
10    from staff stf
11   left join address ad
12     on stf.address_id = ad.address_id;

```

Query Statistics:

- Timing (as measured at client side):** Execution time: 0:00:0.0000000
- Timing (as measured by the server):** Execution time: 0:00:0.0004150, Table lock wait time: 0:00:0.0000400
- Errors:** Had Errors: NO, Warnings: 0
- Row Processing:** Rows affected: 0, Rows sent to client: 2, Rows examined: 4
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 0
- Joins per Type:**
  - Full table scans (Select\_scan): 1
  - Joins using table scan (Select\_full\_join): 0
  - Joins using range search (Select\_full\_range\_join): 0
  - Joins using range checks (Select\_range\_check): 0
  - Joins using range (Select\_range): 0
- Sorting:**
  - Sorts needed (Sort\_scan): 0
  - Sort merge passes (Sort\_merge\_passes): 0
  - Sorts with ranges (Sort\_range): 0
  - Sorts with table scans (Sort\_scan): 0
- Index Usage:** No Index used
- Other Info:** Event Id: 170, Thread Id: 60

## AFTER OPTIMIZING

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Local instance MySQL80
Navigator Schemas SQL File 4*
8
9 •      select stf.first_name , stf.last_name , ad.address , ad.district , ad.postal_code , ad.city_id
10    from staff stf
11   left join address ad
12  on stf.address_id = ad.address_id;
13
14 •      optimize table staff , address

Query Statistics
Timing (as measured at client side):
Execution time: 0:00:0.1870000
Timing (as measured by the server):
Execution time: 0:00:0.1717930
Table lock wait time: 0:00:0.00011100

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 0
Rows examined: 0

Temporary Tables:
Temporary tables created: 0
Temporary tables created: 0

Index Usage:
At least one index was used

Other Info:
Event Id: 191
Thread Id: 60

Object Info Session Output

```

## 4) RIGHT JOIN

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Local instance MySQL80
Navigator Schemas SQL File 4* SQL File 4*
15
16   from film fml
17   right join film_actor fm_act
18  on fml.film_id = fm_act.film_id
19  group by fml.title
20
21

Visual Explain Display Info: Read + Eval cost Overview: View Source:
Query cost: 6154.92
query_block #1
GROUP
tmp table
6154.92 46K rows
ORDER
flesort
nested loop
656.38 5.48K rows
fm_act
idx_fk_film_id
6598.55 1 row
fm PRIMARY
Unique Key Lookup

Object Info Session Output

```

## BEFORE OPTIMIZING

MySQL Workbench - Local instance MySQL80

**SQL File 4\***

```

15   from film
16     right join film_actor fim_act
17   on fim.film_id = fim_act.film_id
18   group by fim.title
19   order by number_of_actors desc;
20
21

```

**Query Statistics**

**Timing (as measured at client side):**  
Execution time: 0:00:0.0000000

**Timing (as measured by the server):**  
Execution time: 0:00:0.0053390  
Table lock wait time: 0:00:0.00000300

**Errors:**  
Had Errors: NO  
Warnings: 0

**Rows Processed:**  
Rows affected: 0  
Rows sent to client: 997  
Rows examined: 1921

**Temporary Tables:**  
Temporary disk tables created: 0  
Temporary tables created: 1

**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): 1

**Index Usage:**  
At least one index was used

**Other Info:**  
Event Id: 328  
Thread Id: 60

**Joins per Type:**  
Full table scans (Select\_scan): 1  
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

**Object Info**   **Session**

actor 36   Result 37   Result 38   Result 39   Result 40\*   Result 41   Output

Apply   Revert   Context Help   Snippets

## AFTER OPTIMIZING

MySQL Workbench - Local instance MySQL80

**SQL File 4\***

```

15   from film
16     right join film_actor fim_act
17   on fim.film_id = fim_act.film_id
18   group by fim.title
19   order by number_of_actors desc;
20
21 • optimize table film , film_actor

```

**Query Statistics**

**Timing (as measured at client side):**  
Execution time: 0:00:0.2340000

**Timing (as measured by the server):**  
Execution time: 0:00:0.21942460  
Table lock wait time: 0:00:0.00005600

**Errors:**  
Had Errors: NO  
Warnings: 0

**Rows Processed:**  
Rows affected: 0  
Rows sent to client: 0  
Rows examined: 0

**Temporary Tables:**  
Temporary disk tables created: 0  
Temporary tables created: 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: 330  
Thread Id: 60

**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

**Object Info**   **Session**

actor 36   Result 37   Result 38   Result 39   Result 40   Result 41\*   Result 42   Output

Apply   Revert   Context Help   Snippets

## 5) INNER JOIN

MySQL Workbench - Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas SQL File 4\*

```

15   from film fim
16     inner join film_actor fim_act
17       on fim.film_id = fim_act.film_id
18     group by fim.title
19     order by number_of_actors desc;
20
21

```

Visual Explain Display Info Read + Eval cost Overview View Source

Query cost: 1473.18  
query\_block #1

GROUP ORDER  
tmp\_table,filesort

1473.18 48K rows

110.1B 1000 rows  
Full Index Scan  
film idx\_title

1363.0 5 rows  
Non-Unique Key Lookup  
film\_actor idx\_fk\_film\_id

actor 30 Result 31 Result 32 Result 33 X Read Only Context Help Snippets

Object Info Session Output

## BEFORE OPTIMIZING

MySQL Workbench - Local instance MySQL80 X

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas SQL File 4\* SQL File 4\*

```

15   from film fim
16     inner join film_actor fim_act
17       on fim.film_id = fim_act.film_id
18     group by fim.title
19     order by number_of_actors desc;
20
21

```

Query Statistics

**Timing (as measured at client side):**  
Execution time: 0:00:0.00000000

**Timing (as measured by the server):**  
Execution time: 0:00:0.00371900  
Table lock wait time: 0:00:0.00000300

**Joins per Type:**  
Full table scans (Select\_scan): 1  
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

**Errors:**  
Had Errors: NO  
Warnings: 0

**Rows Processed:**  
Rows affected: 0  
Rows inserted: 0  
Rows updated: 0  
Rows examined: 749

**Temporary Tables:**  
Temporary tables created: 0  
Temporary tables used: 0

**Index Usage:**  
At least one index was used

**Other Info:**  
Event Id: 288  
Thread Id: 60

actor 30 Result 31 Result 32 Result 33 Result 34 X Apply Revert Context Help Snippets

Object Info Session Output

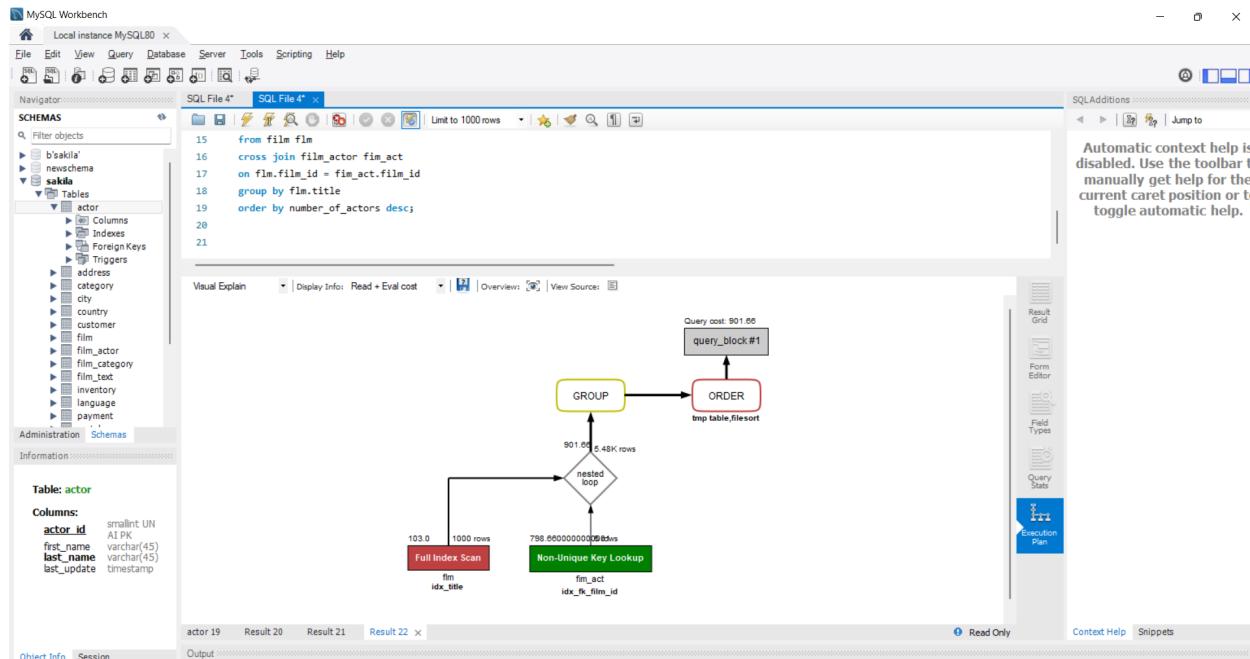
## AFTER OPTIMIZING

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Local instance MySQL80 ×
Navigator: Schemas
SCHEMAS
Filter objects
b'sakila'
newschema
sakila
Tables
Table: actor
Columns:
actor_id smallint UN
first_name varchar(45)
last_name varchar(45)
last_update timestamp
Rows: 199
Temporary Tables: 0
Created_tmp_tables: 0
Created_tmp_disk_tables: 0
Sorts: 0
Temporary table scans: 0
Rows affected: 0
Rows sent to client: 0
Rows examined: 0
Rows matched: 0
Other Info:
Event Id: 253
Thread Id: 60
Information: SQL File 4" ×
15   from film
16     inner join film_actor fim_act
17   on film.film_id = fim_act.film_id
18   group by fim.title
19   order by number_of_actors desc;
20
21 • optimize table film , film_actor
Query Statistics
Timing (as measured at client side):
Execution time: 0:00:0.1880000
Timing (as measured by the server):
Execution time: 0:00:0.17838100
Table lock wait time: 0:00:0.00000500
Errors:
Had Errors: NO
Warnings: 0
Row Processing:
Rows affected: 0
Rows sent to client: 0
Rows examined: 0
Temporary Tables:
Temporary tables created: 0
Temporary tables created: 0
Index Usage:
At least one index was used
Other Info:
Event Id: 253
Thread Id: 60
Object Info Session Output

```

## 6) CROSS JOIN



## BEFORE OPTIMIZING

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor Content:**

```
15 from film
16 cross join film_actor fim_act
17 on fim.film_id = fim_act.film_id
18 group by fim.title
19 order by number_of_actors desc;
20
21
```
- Query Statistics:**
  - Timing (as measured at client side):** Execution time: 0:00:0.1950000
  - Timing (as measured by the server):** Execution time: 0:00:0.0058150
  - Errors:** Had Errors: NO Warnings: 0
  - Rows Processed:** Rows affected: 0 Rows sent to client: 997 Rows examined: 749
  - Temporary Tables:** Temporary disk tables created: 0 Temporary tables created: 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): 1
  - Index Usage:** At least one index was used
  - Other Info:** Event Id: 222 Thread Id: 60
- Right Panel:** Shows the "Query Stats" tab selected, with a note: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

## AFTER OPTIMIZING

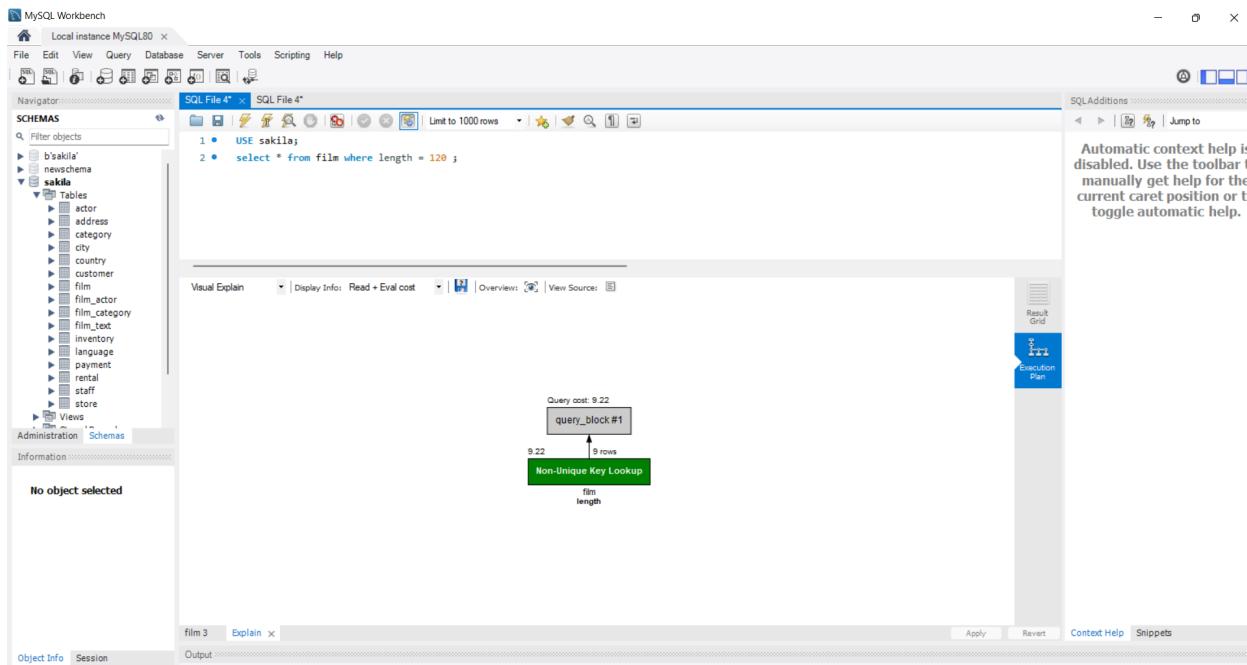
The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor Content:**

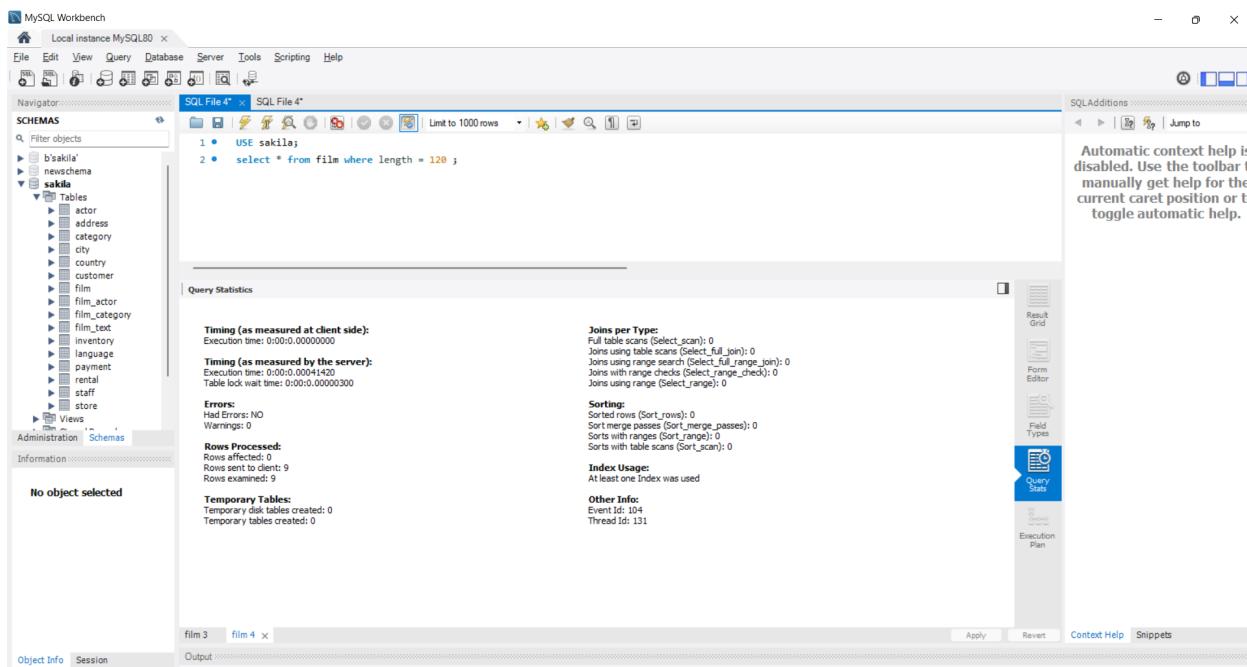
```
15 from film
16 cross join film_actor fim_act
17 on fim.film_id = fim_act.film_id
18 group by fim.title
19 order by number_of_actors desc;
20
21 • optimize table film , film_actor
--
```
- Query Statistics:**
  - Timing (as measured at client side):** Execution time: 0:00:0.1880000
  - Timing (as measured by the server):** Execution time: 0:00:0.0922000
  - Errors:** Had Errors: NO Warnings: 0
  - Rows Processed:** Rows affected: 0 Rows sent to client: 0 Rows examined: 0
  - Temporary Tables:** Temporary disk tables created: 0 Temporary tables created: 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: 245 Thread Id: 60
- Right Panel:** Shows the "Query Stats" tab selected, with a note: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

## INDEX LEVEL OPTIMIZATION

## 1) SIMPLE QUERY



## BEFORE INDEXING



## AFTER INDEXING

MySQL Workbench Local instance MySQL80 X

**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: 0:00:0.0000000

**Timing (as measured by the server):**  
Execution time: 0:00:0.0003602  
Table lock wait time: 0:00:0.0000200

**Errors:**  
No 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

**Index Usage:**  
At least one index was used

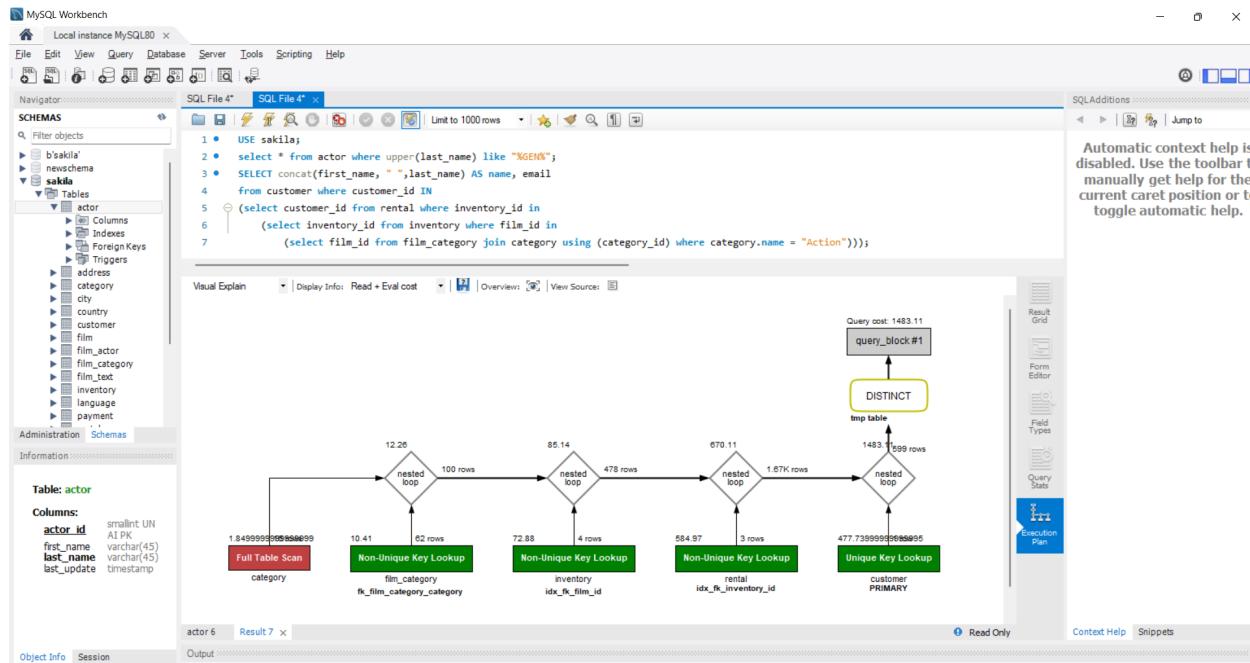
**Other Info:**  
Event Id: 143  
Thread Id: 131

**Explain film 5 X**

Object Info Session Output

Context Help Snippets

## 2) NESTED



## BEFORE INDEXING

```

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")));

```

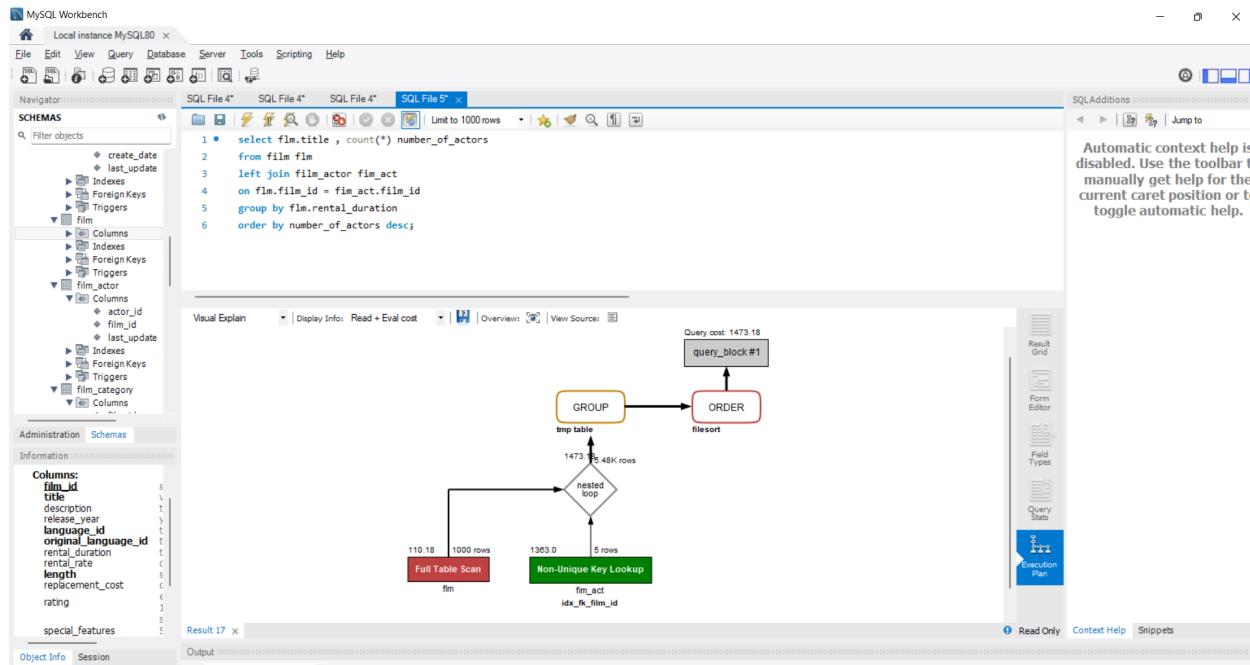
## AFTER INDEXING

```

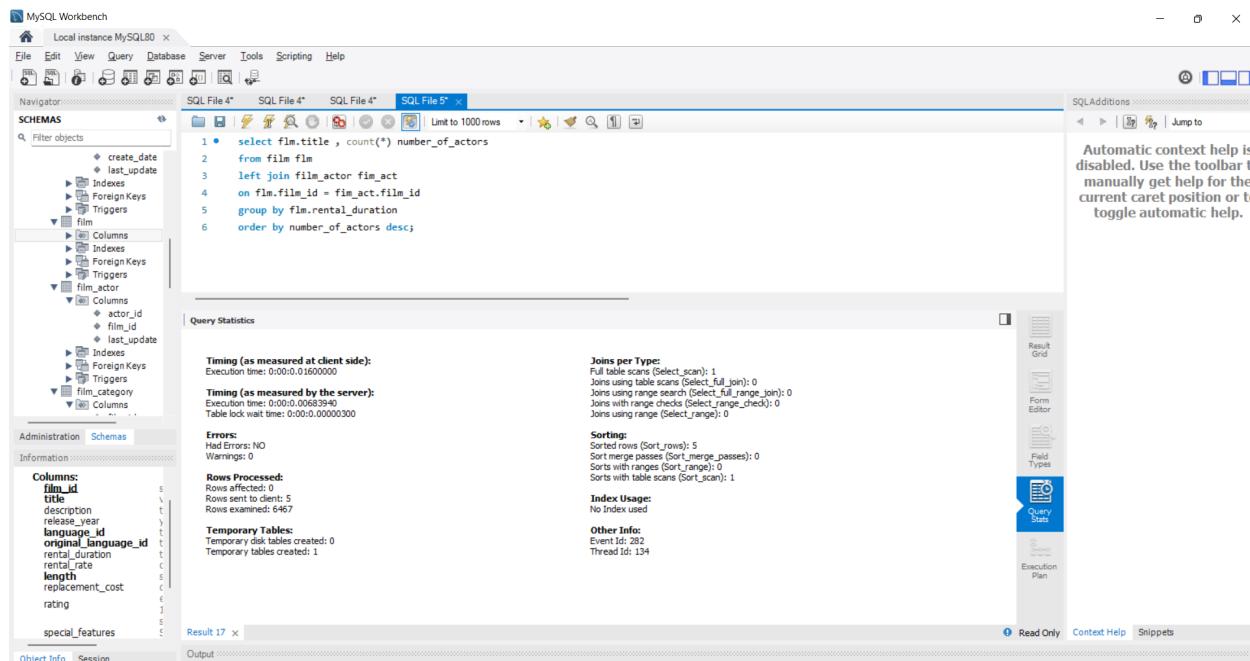
create index first_name on customer(first_name);
select concat(first_name, " ",last_name) AS name, email
from customer where customer_id IN
(select customer_id from rental where inventory_id in
(select inventory_id from inventory where film_id in
(select film_id from film_category join category using (category_id) where category.name = "Action")));

```

### 3) LEFT JOIN



### BEFORE INDEXING



## AFTER INDEXING

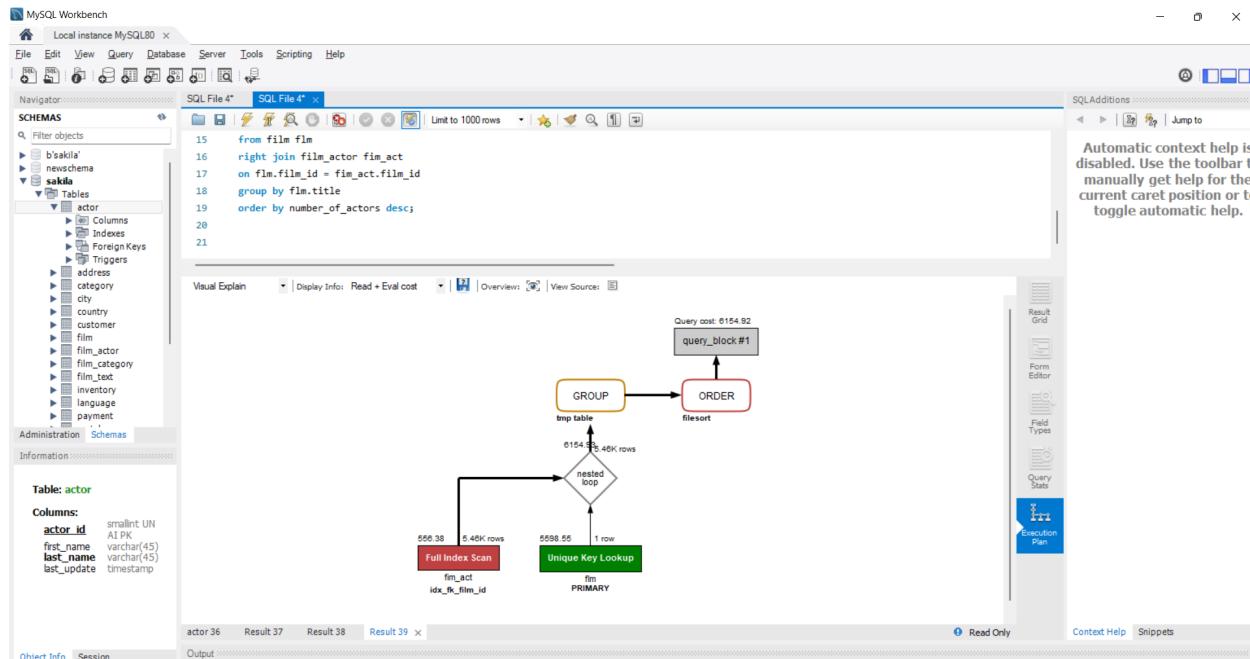
The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- SQL File 4\*** contains the SQL code for creating an index and performing a query:
 

```

1 • create index rental_duration on film(rental_duration);
2 • select film.title , count(*) number_of_actors
3   from film
4   left join film_actor fim_act
5     on film.film_id = fim_act.film_id
6   group by fim.rental_duration
7   order by number_of_actors desc;
      
```
- Query Statistics:**
  - Timing (as measured at client side):** Execution time: 0:00:0.0150000
  - Timing (as measured by the server):** Execution time: 0:00:0.00662890
  - Table lock wait time:** 0:00:0.000000000
- Errors:** Had Errors: NO, Warnings: 0
- Rows Processed:** Rows affected: 0, Rows sent to client: 5, Rows examined: 6467
- Temporary Tables:** Temporary disk tables created: 0, Temporary tables created: 1
- Index Usage:** No Index used
- Other Info:** Event Id: 279, Thread Id: 134

## 4) RIGHT JOIN



## BEFORE INDEXING

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Local instance MySQL80 ×
Navigator: Schemas
SCHEMAS
customer
  Columns
    customer_id
    store_id
    first_name
    last_name
    email
    address_id
    active
    create_date
    last_update
  Indexes
  Foreign Keys
  Triggers
film
  Columns
    film_id
    title
    description
    release_year
    language_id
    original_language_id
    rental_duration
    rental_rate
    length
    replacement_cost
    rating
    special_features
  Indexes
  Foreign Keys
  Triggers
film_actor
Administration Schemas
Information:
Columns:
  film_id
  title
  description
  release_year
  language_id
  original_language_id
  rental_duration
  rental_rate
  length
  replacement_cost
  rating
  special_features
Object Info Session
SQL File 4* SQL File 4* SQL File 4* SQL File 5* ×
1 • select film.title , count(*) number_of_actors
   from film
   right join film_actor film_act
   on film.film_id = film_act.film_id
   group by film.title
   order by number_of_actors desc;
Query Statistics
Timing (as measured at client side):
Execution time: 0:00:0.0000000
Timing (as measured by the server):
Execution time: 0:00:0.0056290
Table lock wait time: 0:00:0.0000000
Joins per Type:
Full table scans (Select_scan): 1
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): 997
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 1
Index Usage:
At least one index was used
Other Info:
Event Id: 244
Thread Id: 134
Result 14 ×
Output:
Context Help Snippets
Read Only

```

## AFTER INDEXING

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Local instance MySQL80 ×
Navigator: Schemas
SCHEMAS
customer
  Columns
    customer_id
    store_id
    first_name
    last_name
    email
    address_id
    active
    create_date
    last_update
  Indexes
  Foreign Keys
  Triggers
film
  Columns
    film_id
    title
    description
    release_year
    language_id
    original_language_id
    rental_duration
    rental_rate
    length
    replacement_cost
    rating
    special_features
  Indexes
  Foreign Keys
  Triggers
film_actor
Administration Schemas
Information:
Columns:
  film_id
  title
  description
  release_year
  language_id
  original_language_id
  rental_duration
  rental_rate
  length
  replacement_cost
  rating
  special_features
Object Info Session
SQL File 4* SQL File 4* SQL File 4* SQL File 5* ×
1 • create index rental_rate on film(rental_rate);
2 • select film.title , count(*) number_of_actors
   from film
   right join film_actor film_act
   on film.film_id = film_act.film_id
   group by film.title
   order by number_of_actors desc;
Query Statistics
Timing (as measured at client side):
Execution time: 0:00:0.0150000
Timing (as measured by the server):
Execution time: 0:00:0.00503120
Table lock wait time: 0:00:0.0000000
Joins per Type:
Full table scans (Select_scan): 1
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): 997
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 1
Index Usage:
At least one index was used
Other Info:
Event Id: 244
Thread Id: 134
Result 13 ×
Output:
Context Help Snippets
Apply Revert

```

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 multi-block 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.

C

C

## EXPERIMENT - 4

### Query Monitoring

4/11/22

NAME - PRACHI PATEL

24  
25

SAP ID - 60004200049

BRANCH - Computer Engineering

Experiment - 4 :-

Implementation of  
AIM - 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 beginning 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 in 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 referenced
- 7) key\_len - length of key used
- 8) used\_ref - Mentions any sort of references used in query while comparing columns or not.

The above mentioned <sup>words</sup> are some column names of EXPLAIN keyword results.



The EXPLAIN has some limitations though. They are,

- 1) It does not provide any information about how triggers, stored functions 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

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `sakila`.
- Query Editor:** Contains the following SQL code:
 

```

1 • USE sakila;
2 • EXPLAIN
3 •
4 select * from film where length = 120 ;
5
6
    
```
- Result Grid:** Displays the results of the EXPLAIN command, showing the execution plan for the query. The output is:
 

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key_len	Ref	Rows	Filtered	Extra
1	SIMPLE	film	NULL	ref	length,`length`	length	3	const	9	100.00	NULL
- SQL Additions:** A note states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

## 2) NESTED

The screenshot shows the MySQL Workbench interface with the following details:

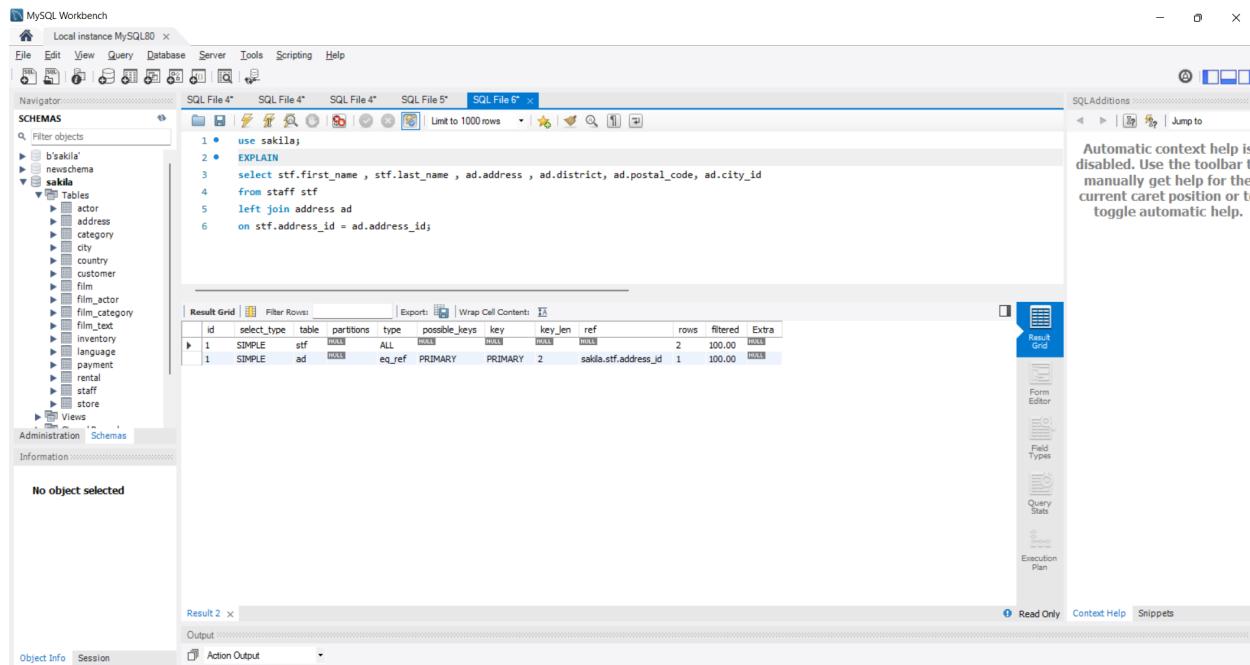
- Schemas:** The current schema is `sakila`.
- Query Editor:** Contains the following SQL code:
 

```

1 • USE sakila;
2 • EXPLAIN
3 •
4 select concat(first_name, " ", last_name) AS name, email
5   from customer where customer_id IN
6   (
7     (select customer_id from rental where inventory_id in
8      (select inventory_id from inventory where film_id in
9        (select film_id from film_category join category using (category_id) where category.name = "Action")));
    
```
- Result Grid:** Displays the results of the EXPLAIN command, showing the execution plan for the nested query. The output is:
 

ID	Select Type	Table	Partitions	Type	Possible Keys	Key	Key_len	Ref	Rows	Filtered	Extra
1	SIMPLE	category	NULL	ALL	PRIMARY	category	NULL	NULL	16	10.00	Using temporary; Using filesort
1	SIMPLE	film_category	NULL	ref	PRIMARY,`fk_film_category_category`	fk_film_category_category	1	sakila.category.category_id	62	100.00	Using index
1	SIMPLE	inventory	NULL	ref	PRIMARY,`idx_fk_film_id`	idx_fk_film_id	2	sakila.film_category.film_id	4	100.00	Using index
1	SIMPLE	rental	NULL	ref	idx_fk_inventory_id,`idx_fk_customer_id`	idx_fk_inventory_id	3	sakila.inventory.inventory_id	3	100.00	Using index
1	SIMPLE	customer	NULL	eq_ref	PRIMARY,customer_id,customer_id,cus_id	PRIMARY	2	sakila.rental.customer_id	1	100.00	End of file
- SQL Additions:** A note states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

### 3) LEFT JOIN



The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```

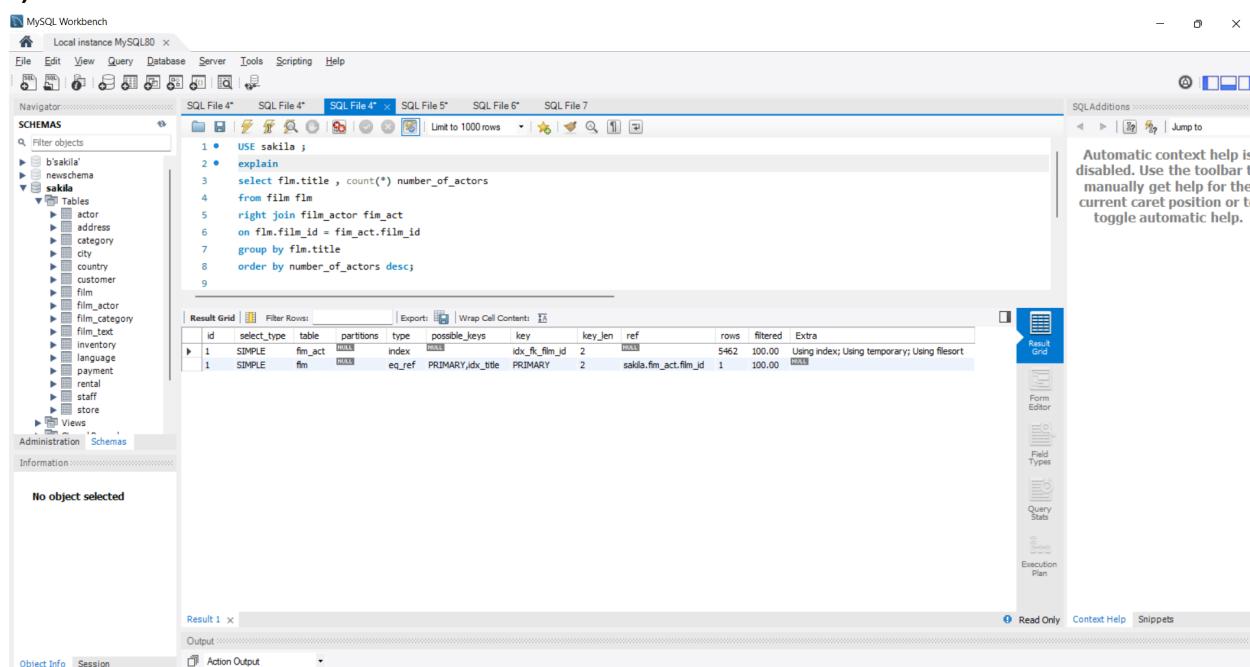
1 • use sakila;
2 • EXPLAIN
3 select stf.first_name , stf.last_name , ad.address , ad.district , ad.postal_code , ad.city_id
4 from staff stf
5 left join address ad
6 on stf.address_id = ad.address_id;
    
```

The Result Grid shows the following data:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	stf	NULL	ALL	NULL	NULL	NULL	NULL	2	100.00	NULL
1	2	SIMPLE	ad	NULL	eq_ref	PRIMARY	PRIMARY	2	sakila.staff.address_id	1	100.00	NULL

The SQL Additions panel displays a note: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

### 4) RIGHT JOIN



The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```

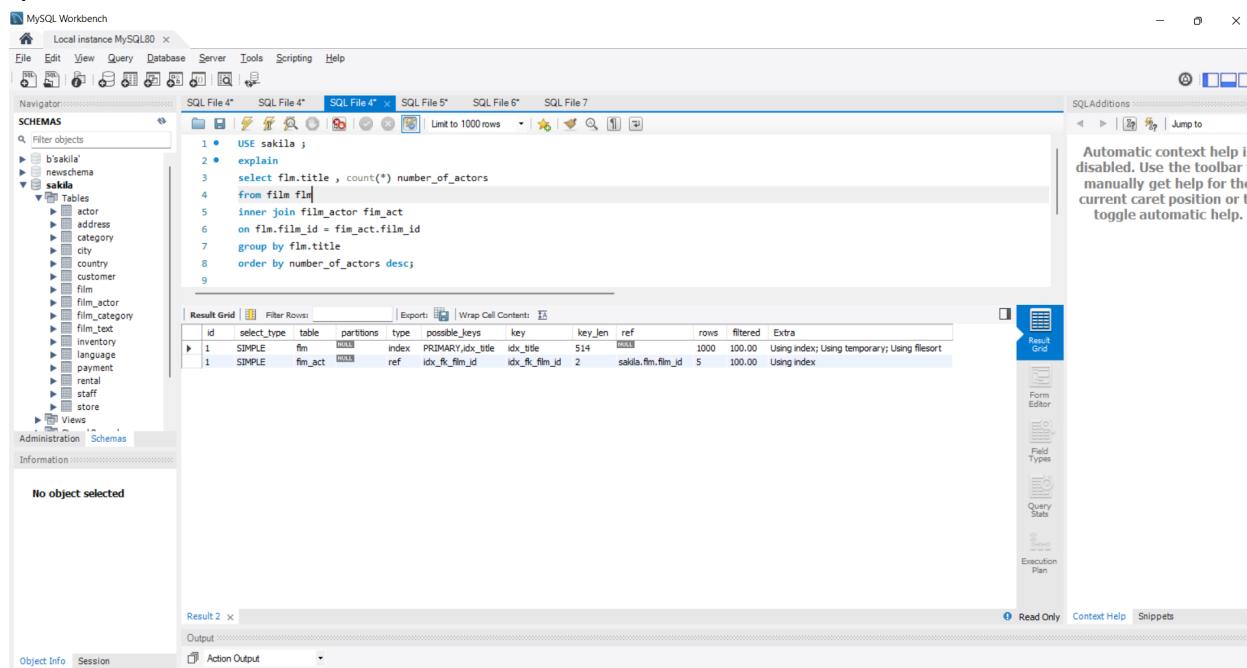
1 • USE sakila ;
2 • explain
3 select fm.title , count(*) number_of_actors
4 from film fm
5 right join film_actor fm_act
6 on fm.film_id = fm_act.film_id
7 group by fm.title
8 order by number_of_actors desc;
9
    
```

The Result Grid shows the following data:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	fm_act	NULL	index	NULL	idx_Rc_film_id	2	NULL	5462	100.00	Using index; Using temporary; Using filesort
1	2	SIMPLE	fm	NULL	eq_ref	PRIMARY	idx_fk_title	PRIMARY	sakila.film_actor.film_id	1	100.00	NULL

The SQL Additions panel displays a note: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

## 5) INNER JOIN



The screenshot shows the MySQL Workbench interface with the Sakila database selected. A query window displays the following SQL code:

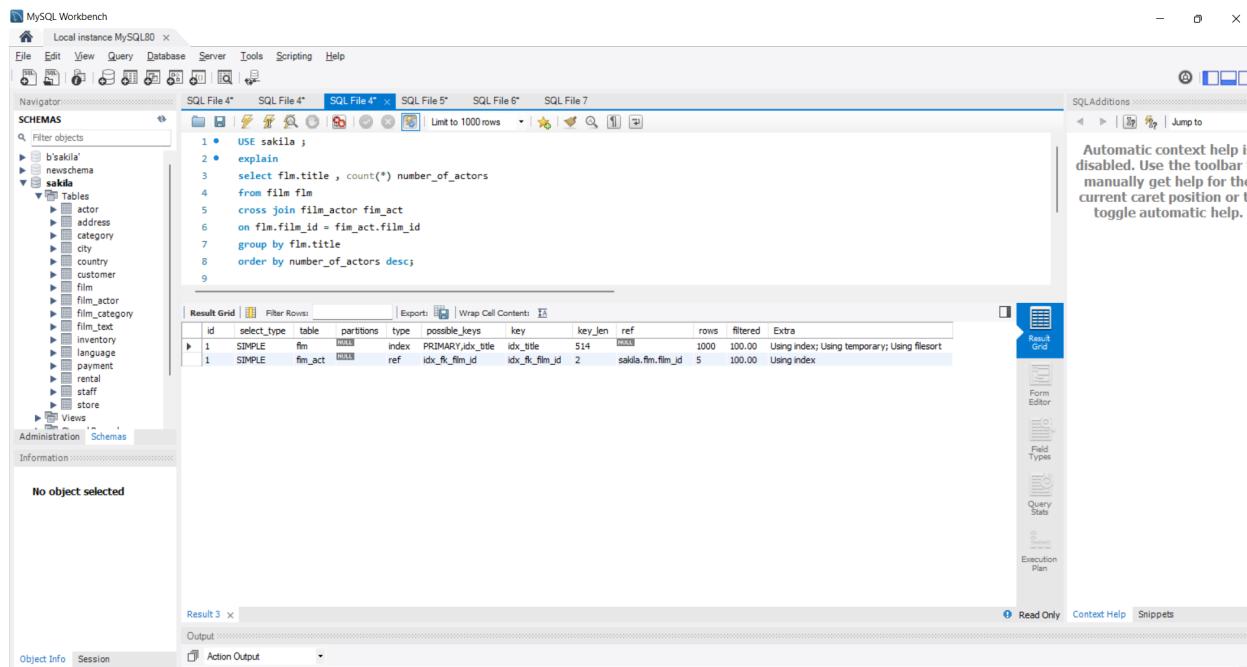
```

1 • USE sakila ;
2 • explain
3 select film.title , count(*) number_of_actors
4 from film fin
5 inner join film_actor fin_act
6 on fin.film_id = fin_act.film_id
7 group by fin.title
8 order by number_of_actors desc;
9

```

The results grid shows the execution plan for the query, indicating a simple index scan on the film table and a ref scan on the film\_actor table.

## 6) CROSS JOIN



The screenshot shows the MySQL Workbench interface with the Sakila database selected. A query window displays the following SQL code:

```

1 • USE sakila ;
2 • explain
3 select film.title , count(*) number_of_actors
4 from film fin
5 cross join film_actor fin_act
6 on fin.film_id = fin_act.film_id
7 group by fin.title
8 order by number_of_actors desc;
9

```

The results grid shows the execution plan for the query, indicating a simple index scan on the film table and a ref scan on the film\_actor table.

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.

C

C

## EXPERIMENT - 5

### Fragmentation (Range , List , Hash & Key)

4/11/22

P  
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 - Partitioning in MySQL is used to split the rows of table into separate 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 within 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 Partitioning. 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 Partitioning:-

It is similar to Hash Partitioning where the hash partitioning 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

```

1  -- create database expadbms;
2  -- use expadbms;
3 • CREATE TABLE Sales ( cust_id INT NOT NULL, name VARCHAR(40),
4   store_id VARCHAR(20) NOT NULL, bill_no INT NOT NULL,
5   bill_date DATE PRIMARY KEY NOT NULL, amount DECIMAL(8,2) NOT NULL)
6 • PARTITION BY RANGE (year(bill_date))( 
7   PARTITION p0 VALUES LESS THAN (2016),
8   PARTITION p1 VALUES LESS THAN (2017),
9   PARTITION p2 VALUES LESS THAN (2018),
10  PARTITION p3 VALUES LESS THAN (2020));
11 • INSERT INTO Sales VALUES
12  (1, 'Mike', 'S001', 101, '2015-01-02', 125.56),
13  (2, 'Robert', 'S003', 103, '2015-01-25', 476.50),
14  (3, 'Peter', 'S012', 122, '2016-02-15', 335.00),
15  (4, 'Joseph', 'S345', 121, '2016-03-26', 787.00),
16  (5, 'Harry', 'S234', 132, '2017-04-19', 678.00),
17  (6, 'Stephen', 'S743', 111, '2017-05-31', 864.00),
18  (7, 'Jacson', 'S234', 115, '2018-06-11', 762.00),
19  (8, 'Smith', 'S012', 125, '2019-07-24', 300.00),
20  (9, 'Adam', 'S456', 119, '2019-08-02', 492.20);

```

Result Grid						
	cust_id	name	store_id	bill_no	bill_date	amount
▶	3	Peter	S012	122	2016-02-15	335.00
▶	4	Joseph	S345	121	2016-01-20	2016-02-15
▶	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

```

11 • SELECT * FROM Sales;
12 • SELECT TABLE_NAME, PARTITION_NAME, TABLE_ROWS, AVG_ROW_LENGTH, DATA_LENGTH
13   FROM INFORMATION_SCHEMA.PARTITIONS
14   WHERE TABLE_SCHEMA = 'expadbms' AND TABLE_NAME = 'Sales';
15 • ALTER TABLE Sales TRUNCATE PARTITION p0;
16 • SELECT PARTITION_NAME, TABLE_ROWS
17   FROM INFORMATION_SCHEMA.PARTITIONS
18   WHERE TABLE_SCHEMA = 'expadbms' AND TABLE_NAME = 'Sales';
19 • CREATE TABLE Stores (
20   cust_name VARCHAR(40),
21   bill_no VARCHAR(20) NOT NULL,
22   store_id INT PRIMARY KEY NOT NULL,
23   bill_date DATE NOT NULL,
24   amount DECIMAL(8,2) NOT NULL
25 )

```

Result Grid				
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

PARTITION_NAME	TABLE_ROWS
p0	0
p1	2
p2	2
p3	3

## 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),

```

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

```

1 CREATE TABLE StoresHashNew (
2     cust_name VARCHAR(40),
3     bill_no VARCHAR(20) NOT NULL,
4     store_id INT PRIMARY KEY NOT NULL,
5     bill_date DATE NOT NULL,
6     amount DECIMAL(8,2) NOT NULL
7 )
8 PARTITION BY HASH(store_id)
9 PARTITIONS 4;
10 INSERT INTO StoresHashNew VALUES
11 ('Mike', 'S001', 101, '2015-01-02', 125.56),
12 ('Robert', 'S003', 103, '2015-01-25', 476.50),
13 ('Peter', 'S012', 104, '2016-02-15', 335.00),
14 ('Joseph', 'S345', 105, '2016-03-26', 787.00),
15 ('Harry', 'S234', 106, '2017-04-19', 678.00),
16 ('Stephen', 'S743', 107, '2017-05-31', 864.00),
17 ('Jacson', 'S234', 108, '2018-06-11', 762.00);
18 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

Result Grid | Form Editor | Field Types

## 5) RANGE COLUMN PARTITIONING

```

69 CREATE TABLE test_part (A INT, B CHAR(5), C INT, D INT)
70 PARTITION BY RANGE COLUMNS(A, B, C)
71 (PARTITION p0 VALUES LESS THAN (50, 'test1', 100),
72 PARTITION p1 VALUES LESS THAN (100, 'test2', 200),
73 PARTITION p2 VALUES LESS THAN (150, 'test3', 300),
74 PARTITION p3 VALUES LESS THAN (MAXVALUE, MAXVALUE, MAXVALUE));
75 INSERT INTO test_part VALUES
76 ( 45,'N', 101, 125.56),
77 ( 47,'R', 103, 476.50),
78 ( 74,'P', 104, 335.00),
79 ( 190,'J', 105, 787.00);
80 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

Result Grid | Form Editor | Field Types

test\_part 13 x Read Only

## 6) LIST COLUMN PARTITIONING

```

31
32 • CREATE TABLE AgentDetail (
33     agent_id VARCHAR(10),
34     agent_name VARCHAR(40),
35     city VARCHAR(10))
36     PARTITION BY LIST COLUMNS(agent_id) (
37         PARTITION pNewYork VALUES IN('A1', 'A2', 'A3'),
38         PARTITION pTexas VALUES IN('B1', 'B2', 'B3'),
39         PARTITION pCalifornia VALUES IN ('C1', 'C2', 'C3'));
40
41 •     INSERT INTO AgentDetail VALUES
42     ( 'A1','Mike', 101 ),
43     ( 'A2','Robert', 103),
44     ( 'B1','Peterson', 104),
45     ( 'C1','Jack', 105);
46 •     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

AgentDetail 14 x Read Only

## 7) KEY PARTITIONING

```

8 • CREATE TABLE AgentDetailKey (
9     agent_name VARCHAR(40) ,
10    agent_id INT NOT NULL PRIMARY KEY
11 )
12     PARTITION BY KEY()
13     PARTITIONS 2;
14 •     INSERT INTO AgentDetailKey VALUES
15     ( 'Mike', 101 ),
16     ( 'Robert', 103),
17     ( 'Peterson', 104),
18     ( 'Jack', 105);
19 •     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

AgentDetailKey 15 x 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 partitioning 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

P  
24  
25

NAME - PRACHI PATEL

SAP ID - 60004200049

BRANCH - Computer  
Engineering

#### Experiment - 6 - 2 PC & 3 PC

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 coordinator and writes a "PREPARE" log in its stable storage and sends them and writes "PREPARE" message to all its subordinates.

Step 2 - After receiving the prepare message, the subordinates decide whether they are capable of committing the transaction. If it can commit the transaction, 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 receiving 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 receive acknowledgement back to the coordinator stating that all the sub transactions had written a COMMIT log in their local database and send an acknowledgement back to the coordinator of successful completion of executing transaction.

Step 3 - Upon receiving 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 uninterrupted, 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 measures the sites regarding the <sup>failure.</sup>

1) Phase 2 or reasuring phase in 2PC protocol

It involves the step that if the coordinator receives an ABORT reply from one of the subordinates, then the coordinator will send "PREPARE ABORT" message to all the subordinate. Now, if all the subordinates send a "READY" message to commit the sub-transactions, then after receiving 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 :

```

PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python server.py
Server is running!
Coordinator : PREPARE
Subordinator 0 ('127.0.0.1', 58974) : READY
Subordinator 1 ('127.0.0.1', 58976) : READY
Subordinator 2 ('127.0.0.1', 58977) : READY
Coordinator : COMMIT
Subordinator 0 ('127.0.0.1', 58979) : SUCCESS
Subordinator 1 ('127.0.0.1', 58980) : SUCCESS
Subordinator 2 ('127.0.0.1', 58981) : SUCCESS
TRANSACTION COMPLETED!
The final log is: PREPARE COMMIT COMPLETE
Coordinator : COMPLETE
Subordinator 0 ('127.0.0.1', 58982) : OK
Subordinator 1 ('127.0.0.1', 58983) : OK
Subordinator 2 ('127.0.0.1', 58984) : OK
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> |

PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client.py
Coordinator: PREPARE
System Status: READY
Coordinator: COMMIT
System Status: SUCCESS
Coordinator: COMPLETE
System Status: OK
END OF TRANSACTION
final log is: READY SUCCESS OK COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> |

PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client2.py
Coordinator: PREPARE
System Status: READY
Coordinator: COMMIT
System Status: SUCCESS
Coordinator: COMPLETE
System Status: OK
END OF TRANSACTION
final log is: READY SUCCESS OK COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> |

PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client3.py
Coordinator: PREPARE
System Status: READY
Coordinator: COMMIT
System Status: SUCCESS
Coordinator: COMPLETE
System Status: OK
END OF TRANSACTION
final log is: READY SUCCESS OK COMPLETE
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> |


PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python server.py
Server is running!
Coordinator : PREPARE
Subordinator 0 ('127.0.0.1', 59013) : READY
Subordinator 1 ('127.0.0.1', 59014) : ABORT
Subordinator 2 ('127.0.0.1', 59015) : READY
TRANSACTION ABORTED!
The final log is: PREPARE ABORT
Coordinator : ABORT
Subordinator 0 ('127.0.0.1', 59016) : OK
Subordinator 1 ('127.0.0.1', 59019) : OK
Subordinator 2 ('127.0.0.1', 59020) : OK
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> |

PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client.py
Coordinator: PREPARE
System Status: READY
Coordinator: ABORT
TRANSACTION ABORTED!
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> |


PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client2.py
Coordinator: PREPARE
System Status: ABORT
Coordinator: ABORT
TRANSACTION ABORTED!
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> |

PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> python client3.py
Coordinator: PREPARE
System Status: READY
Coordinator: ABORT
TRANSACTION ABORTED!
PS C:\Users\savla\Documents\SEM 5 PRACS\ADBMS\2c_3pc> |

```

## 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 :

```

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> |

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> |

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> |


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> |

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> |

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

P  
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 Generalized 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 issues arise, including how to organize, manipulate and query the XML data.

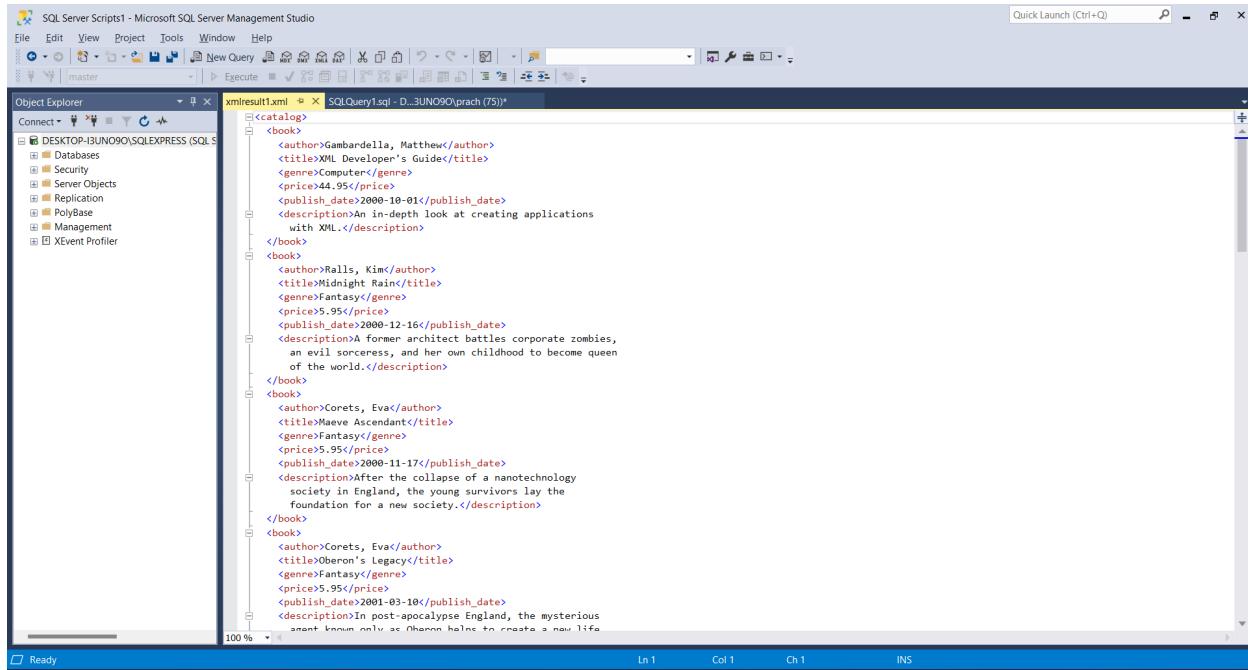
#### Querying XML data :-

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



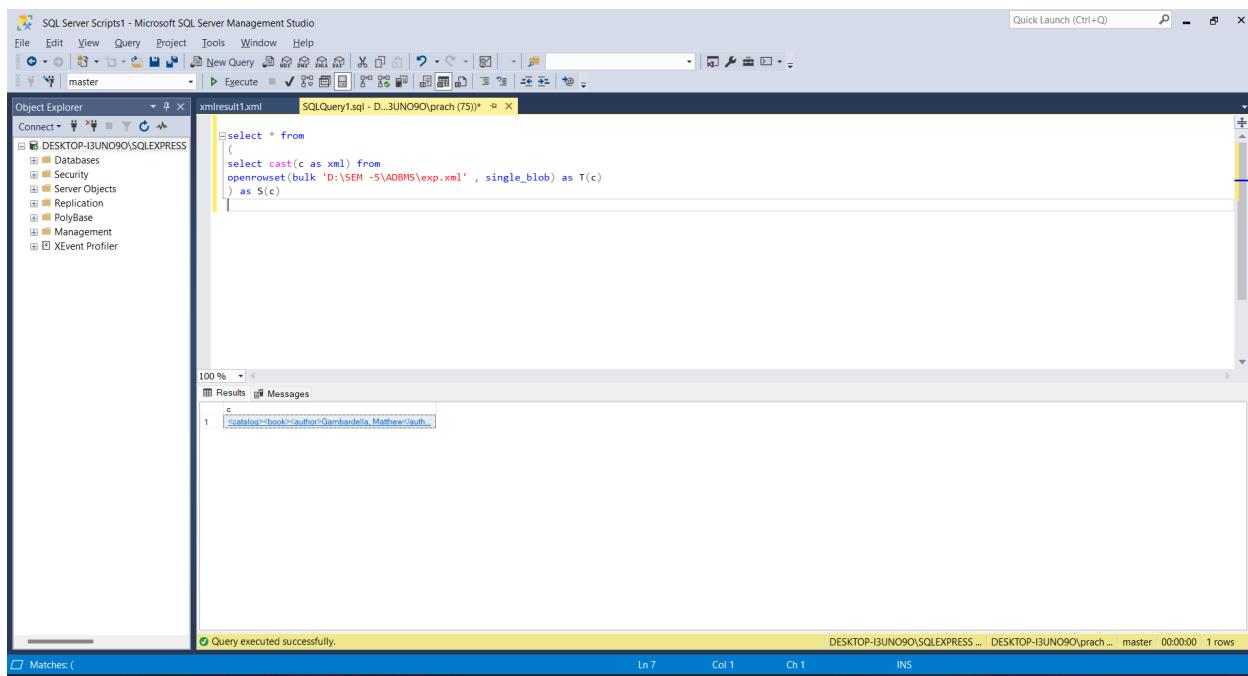
The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQL Server Scripts1 - Microsoft SQL Server Management Studio". The main window displays an XML result set titled "xmlext1.xml". The XML structure represents a catalog of books, with each book having an author, title, genre, price, publish date, and a detailed description. The XML code is as follows:

```

<catalog>
  <book>
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications with XML.</description>
  </book>
  <book>
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-12-16</publish_date>
    <description>A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.</description>
  </book>
  <book>
    <author>Corets, Eva</author>
    <title>Maeve Ascendant</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-11-17</publish_date>
    <description>After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.</description>
  </book>
  <book>
    <author>Corets, Eva</author>
    <title>Oberon's Legacy</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2001-03-10</publish_date>
    <description>In post-apocalypse England, the mysterious agent known only as Oberon hails to create a new life</description>
  </book>

```

## 1) SELECT



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the connection is to DESKTOP-I3UNO90\SQLEXPRESS. In the center pane, a query window titled 'xmlext1.xml' contains the following T-SQL code:

```

select * from
(
    select cast(c as xml) from
    openrowset(bulk 'D:\SEM -5\ADBSMS\exp.xml' , single_blob) as T(c)
) as S(c)

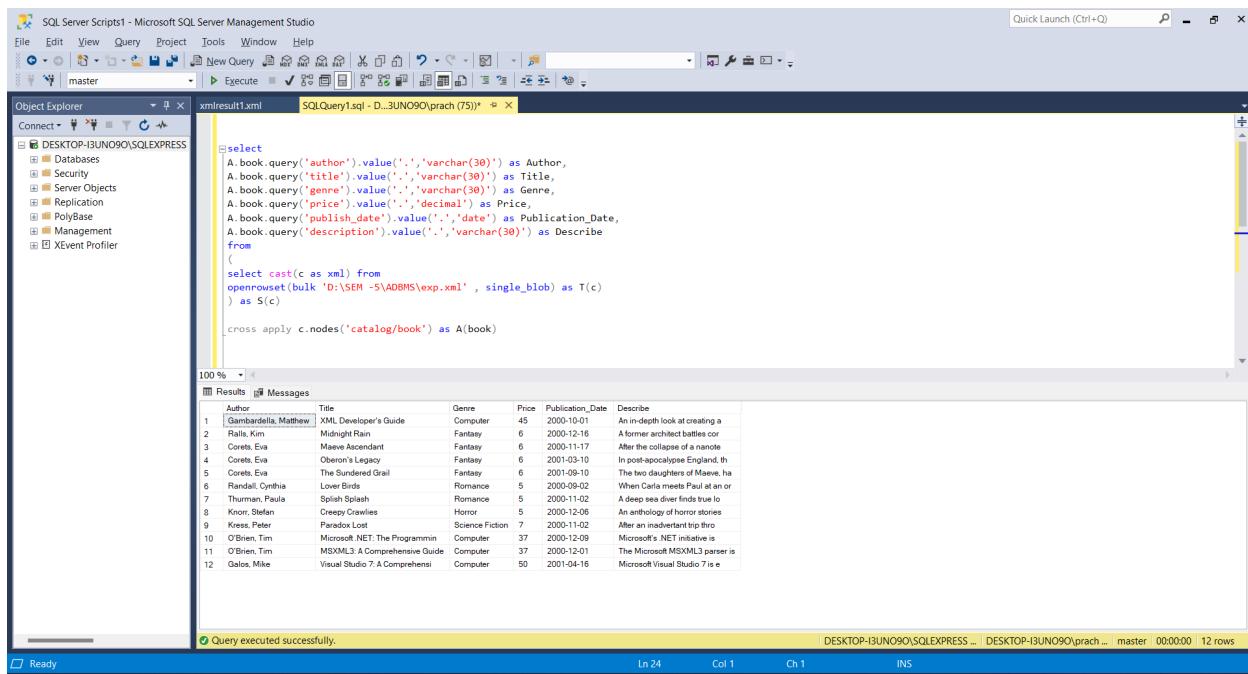
```

In the results pane, there is one row of data:

c
catalog>book>author>Gambardella, Matthew</auth...

At the bottom, a message indicates: "Query executed successfully."

## 2) DISPLAY DATA IN FORM OF TABLE



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the connection is to DESKTOP-I3UNO90\SQLEXPRESS. In the center pane, a query window titled 'xmlext1.xml' contains the following T-SQL code:

```

select
    A.book.query('author').value('.','varchar(30)') as Author,
    A.book.query('title').value('.','varchar(30)') as Title,
    A.book.query('genre').value('.','varchar(30)') as Genre,
    A.book.query('price').value('.','decimal') as Price,
    A.book.query('publish_date').value('.','date') as Publication_Date,
    A.book.query('description').value('.','varchar(30)') as Describe
from
(
    select cast(c as xml) from
    openrowset(bulk 'D:\SEM -5\ADBSMS\exp.xml' , single_blob) as T(c)
) as S(c)

cross apply c.nodes('catalog/book') as A(book)

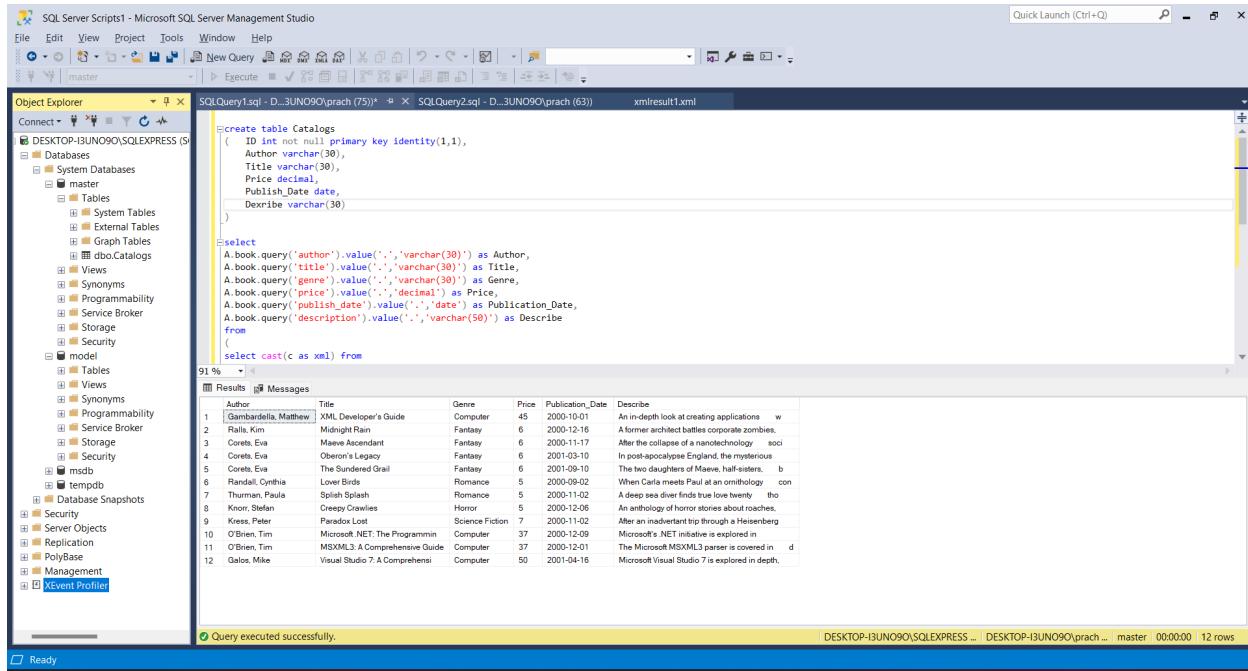
```

In the results pane, the data is displayed in a table format:

Author	Title	Genre	Price	Publication_Date	Describe
Gambardella, Matthew	XML Developer's Guide	Computer	45	2000-10-01	An in-depth look at creating a
Ralls, Kim	Midnight Rain	Fantasy	6	2000-12-16	A former archivist battles cor
Coret, Eva	Maeve Ascendant	Fantasy	6	2000-11-17	After the collapse of a nanote
Coret, Eva	Oberon's Legacy	Fantasy	6	2001-03-10	In post-apocalyptic England, th
Coret, Eva	The Sundered Grail	Fantasy	6	2001-09-10	The two daughters of Maeve, ha
Randall, Cynthia	Lover Birds	Romance	5	2000-09-02	When Carla meets Paul at an or
Thurman, Paula	Splish Splash	Romance	5	2000-11-02	A deep sea diver finds true lo
Knorr, Stefan	Other Crawlers	Horror	5	2000-09-22	An anthropomorphic species
Kress, Peter	Pandorum	Science Fiction	5	2003-11-02	After an interdimensional tra
O'Brien, Tim	Microsoft .NET: The Programming	Computer	37	2000-12-09	Microsoft's .NET initiative is
O'Brien, Tim	MSXML3: A Comprehensive Guide	Computer	37	2000-12-01	The Microsoft MSXML3 parser is
Galos, Mike	Visual Studio 7: A Comprehensive	Computer	50	2001-04-16	Microsoft Visual Studio 7 is e

At the bottom, a message indicates: "Query executed successfully."

### 3) CREATE TABLE



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the master database is selected. In the center pane, a script is being run to create a table named 'Catalogs' from an XML file. The results pane displays the 12 rows of data that were inserted into the table.

```

CREATE TABLE Catalogs
(
    ID int not null primary key identity(1,1),
    Author varchar(30),
    Title varchar(30),
    Price decimal,
    Publish_Date date,
    Describe varchar(30)
)

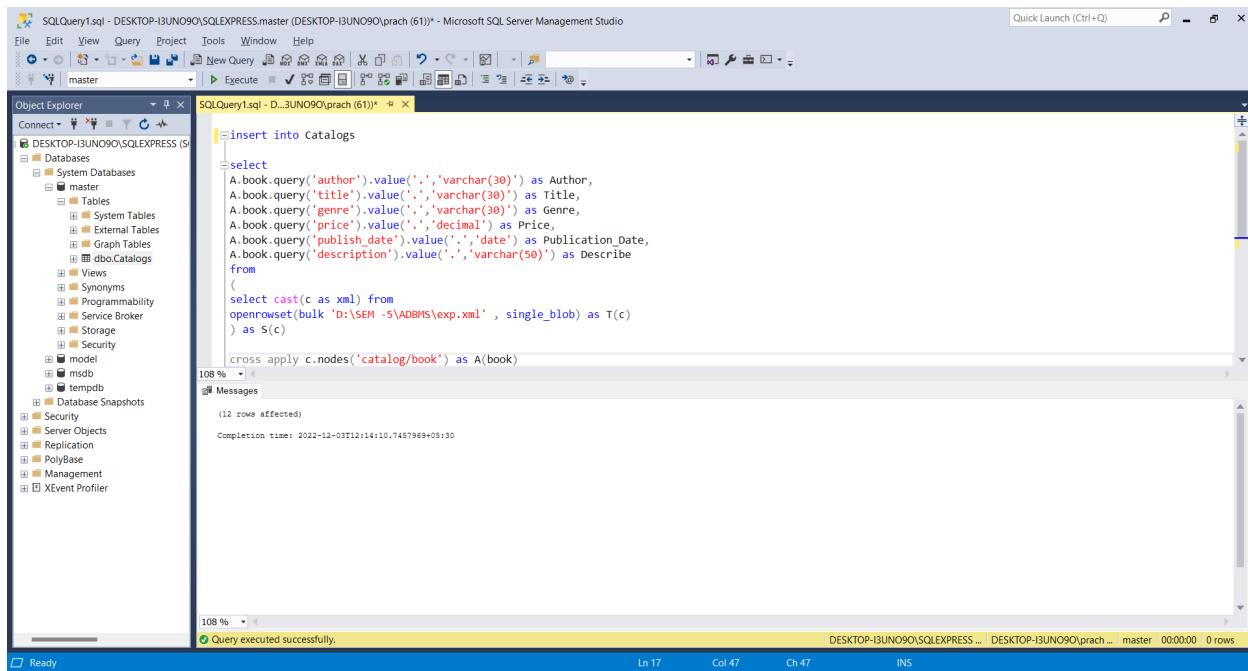
SELECT
    A.book.query('author').value('.', 'varchar(30)') AS Author,
    A.book.query('title').value('.', 'varchar(30)') AS Title,
    A.book.query('genre').value('.', 'varchar(30)') AS Genre,
    A.book.query('price').value('.', 'decimal') AS Price,
    A.book.query('publish_date').value('.', 'date') AS Publication_Date,
    A.book.query('description').value('.', 'varchar(50)') AS Describe
FROM
    (
        SELECT cast(c AS XML) FROM openrowset(bulk 'D:\SEM -5\ADBMS\exp.xml', single_blob) AS T(c)
    ) AS S(c)
CROSS APPLY S(c).nodes('catalog/book') AS A(book)

```

Author	Title	Genre	Price	Publication Date	Describe
Gambardella, Matthew	XML Developer's Guide	Computer	45	2000-10-01	An in-depth look at creating applications
Ralls, Kim	Midnight Rain	Fantasy	6	2000-12-16	A former architect battles corporate zombies.
Cores, Eva	Maeve Ascendant	Fantasy	6	2000-11-17	After the collapse of a nanotechnology
Cores, Eva	Oberon's Legacy	Fantasy	6	2001-03-10	In post-apocalypse England, the mysterious
Cores, Eva	The Sundered Grail	Fantasy	6	2001-09-10	The two daughters of Maeve, half-sisters.
Randall, Cynthia	Lover Birds	Romance	5	2000-09-02	When Carla meets Paul at an ornithology
Thurman, Paula	Splash Splash	Romance	5	2000-11-02	A deep sea diver finds true love twenty
Knor, Stefan	Creepy Crawlies	Horror	5	2000-12-06	An anthology of horror stories about roaches.
Kress, Peter	Paradox Lost	Science Fiction	7	2000-11-02	After an inadvertent trip through a Heisenberg
O'Brien, Tim	Microsoft .NET: The Programmer's Guide	Computer	37	2000-12-09	Microsoft's .NET initiative is explored in
O'Brien, Tim	MSXML3: A Comprehensive Guide	Computer	37	2000-12-01	The Microsoft MSXML3 parser is covered in
Dalos, Mike	Visual Studio 7: A Comprehensive Guide	Computer	50	2001-04-16	Microsoft Visual Studio 7 is explored in depth.

Query executed successfully.

### 4) INSERT DATA IN TABLE



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the master database is selected. In the center pane, a script is being run to insert data into the 'Catalogs' table from an XML file. The results pane shows the completion message indicating 12 rows affected.

```

INSERT INTO Catalogs
SELECT
    A.book.query('author').value('.', 'varchar(30)') AS Author,
    A.book.query('title').value('.', 'varchar(30)') AS Title,
    A.book.query('genre').value('.', 'varchar(30)') AS Genre,
    A.book.query('price').value('.', 'decimal') AS Price,
    A.book.query('publish_date').value('.', 'date') AS Publication_Date,
    A.book.query('description').value('.', 'varchar(50)') AS Describe
FROM
    (
        SELECT cast(c AS XML) FROM openrowset(bulk 'D:\SEM -5\ADBMS\exp.xml', single_blob) AS T(c)
    ) AS S(c)
CROSS APPLY S(c).nodes('catalog/book') AS A(book)

```

(12 rows affected)

Completion time: 2022-12-03T12:14:10.7457969+05:30

Query executed successfully.

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the master database and its tables. The central Results grid displays the output of the following SQL query:

```
select * from Catalogs
```

The results show 12 rows of data from the Catalogs table:

ID	Author	Title	Genre	Price	Publication_Date	Describe
1	Gambardella, Matthew	XML Developer's Guide	Computer	45	2000-10-01	An in-depth look at creating applications w
2	Ralls, Kim	Midnight Rain	Fantasy	6	2000-12-16	A former architect battles corporate zombies.
3	Coetz, Eva	Maeve Ascendant	Fantasy	6	2000-11-17	After the collapse of a nanotechnology soci
4	Coetz, Eva	Oberon's Legacy	Fantasy	6	2001-03-10	In post-apocalypse England, the mysterious
5	Coetz, Eva	The Sundered Grail	Fantasy	6	2001-09-10	The two daughters of Maeve, half-sisters, b
6	Randall, Cynthia	Lover Birds	Romance	5	2000-09-02	When Carla meets Paul at an ornithology con
7	Thurman, Paula	Splish Splash	Romance	5	2000-11-02	A deep sea diver finds true love twenty tho
8	Knoer, Stefan	Creepy Crawlies	Horror	5	2000-12-06	An anthology of horror stories about roaches.
9	Kress, Peter	Paradox Lost	Science Fiction	7	2000-11-02	After an inadvertent trip through a Heisenberg
10	O'Brien, Tim	Microsoft .NET: The Programming	Computer	37	2000-12-09	Microsoft's .NET initiative is explored in
11	O'Brien, Tim	MSXML3.0 A Comprehensive Guide	Computer	37	2000-12-01	The Microsoft MSXML3 parser is covered in d
12	Galos, Mike	Visual Studio 7: A Comprehensi	Computer	50	2001-04-16	Microsoft Visual Studio 7 is explored in depth,

At the bottom of the Results grid, a message indicates: "Query executed successfully." Below the Results grid, the status bar shows: "DESKTOP-I3JUN090\SQLEXPRESS ... DESKTOP-I3JUN090\prach ... master | 00:00:00 | 12 rows".

## 5) DELETE DATA FROM TABLE

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure, including the master database and its tables. The central Results grid displays the output of the following SQL query:

```
DELETE FROM Catalogs WHERE Price = 37
```

The results show 10 rows of data from the Catalogs table:

ID	Author	Title	Genre	Price	Publication_Date	Describe
1	Gambardella, Matthew	XML Developer's Guide	Computer	45	2000-10-01	An in-depth look at creating applications w
2	Ralls, Kim	Midnight Rain	Fantasy	6	2000-12-16	A former architect battles corporate zombies.
3	Coetz, Eva	Maeve Ascendant	Fantasy	6	2000-11-17	After the collapse of a nanotechnology soci
4	Coetz, Eva	Oberon's Legacy	Fantasy	6	2001-03-10	In post-apocalypse England, the mysterious
5	Coetz, Eva	The Sundered Grail	Fantasy	6	2001-09-10	The two daughters of Maeve, half-sisters, b
6	Randall, Cynthia	Lover Birds	Romance	5	2000-09-02	When Carla meets Paul at an ornithology con
7	Thurman, Paula	Splish Splash	Romance	5	2000-11-02	A deep sea diver finds true love twenty tho
8	Knoer, Stefan	Creepy Crawlies	Horror	5	2000-12-06	An anthology of horror stories about roaches.
9	Kress, Peter	Paradox Lost	Science Fiction	7	2000-11-02	After an inadvertent trip through a Heisenberg
10	Galos, Mike	Visual Studio 7: A Comprehensi	Computer	50	2001-04-16	Microsoft Visual Studio 7 is explored in depth,

At the bottom of the Results grid, a message indicates: "Query executed successfully." Below the Results grid, the status bar shows: "DESKTOP-I3JUN090\SQLEXPRESS ... DESKTOP-I3JUN090\prach ... master | 00:00:00 | 10 rows".

CONCLUSION :-

After 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

f  
24  
25

NAME - PRACHI PATEL

SAP ID - 60004200049

BRANCH - Computer  
Engineering

#### 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 associated 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.

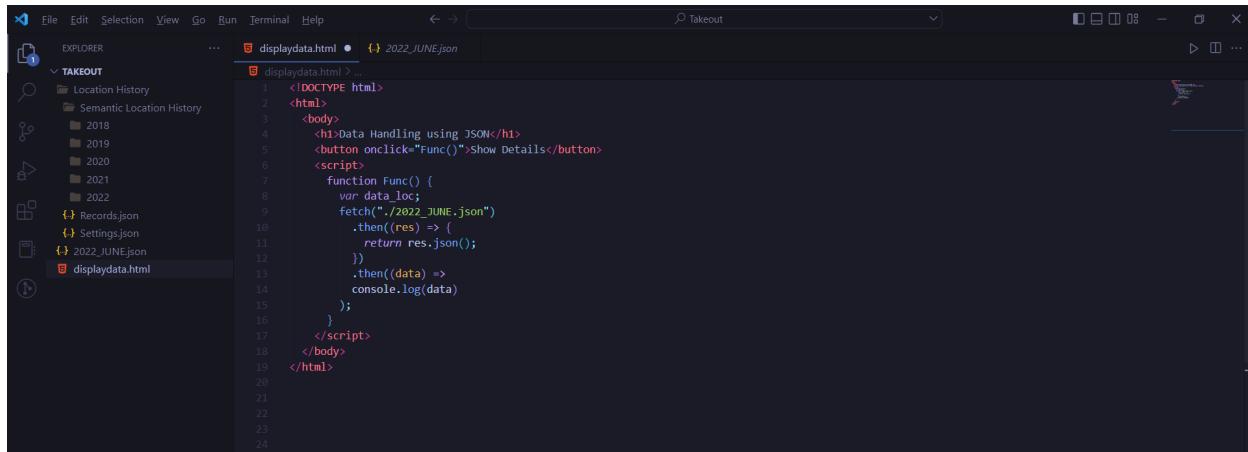
For this purpose we need two methods.

- 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.

## **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**

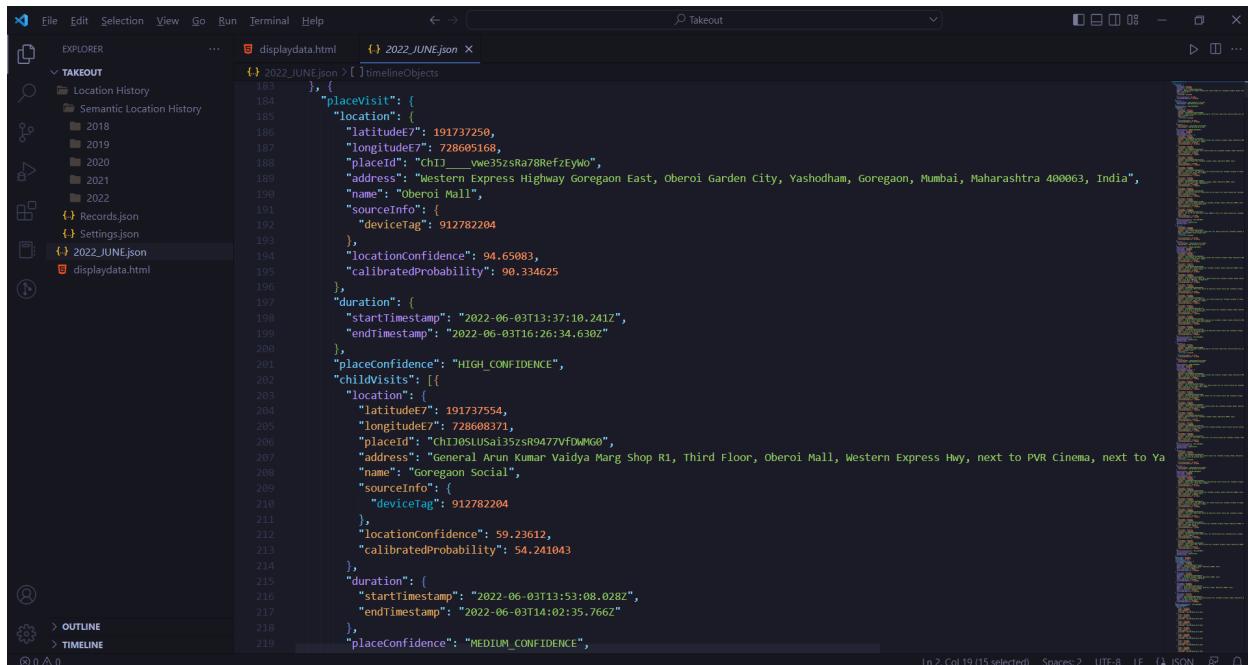


```

<!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)
    };
}</script>
</body>
</html>

```

### **2) DATA :**

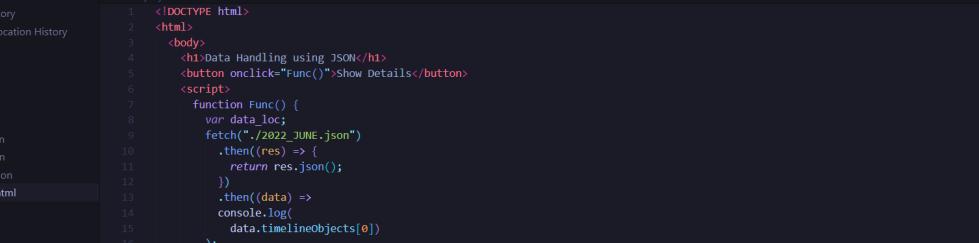


```

2022_JUNE.json > [ timelineObjects
  183  },
  184   "placeVisit": {
  185     "location": {
  186       "latitudeE7": 191737250,
  187       "longitudeE7": 728605168,
  188       "placeId": "ChIJ_wve35zsRa78RefzEylo",
  189       "address": "Western Express Highway Goregaon East, Oberoi Garden City, Yashodham, Goregaon, Mumbai, Maharashtra 400063, India",
  190       "name": "Oberoi Mall",
  191       "sourceInfo": {
  192         "deviceTag": 912782204
  193       },
  194       "locationConfidence": 94.65083,
  195       "calibratedProbability": 98.334625
  196     },
  197     "duration": {
  198       "startTimestamp": "2022-06-03T13:37:10.241Z",
  199       "endTimestamp": "2022-06-03T16:26:34.630Z"
  200     },
  201     "placeConfidence": "HIGH_CONFIDENCE",
  202     "childVisits": [
  203       "location": {
  204         "latitudeE7": 191737554,
  205         "longitudeE7": 728608371,
  206         "placeId": "ChIJ0SLUSa135zsR9477vFDwMG0",
  207         "address": "General Arun Kumar Vaidya Marg Shop R1, Third Floor, Oberoi Mall, Western Express Hwy, next to PVR Cinema, next to Ya",
  208         "name": "Goregaon Social",
  209         "sourceInfo": {
  210           "deviceTag": 912782204
  211         },
  212         "locationConfidence": 59.23612,
  213         "calibratedProbability": 54.241043
  214       },
  215       "duration": {
  216         "startTimestamp": "2022-06-03T13:53:08.028Z",
  217         "endTimestamp": "2022-06-03T14:02:35.766Z"
  218     },
  219     "placeConfidence": "MEDIUM_CONFIDENCE",

```

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



The screenshot shows a code editor interface with the following details:

- File Explorer:** On the left, there's a sidebar titled "EXPLORER" under "TAKEOUT". It lists several JSON files: "Location History", "Semantic Location History", "2018", "2019", "2020", "2021", "2022", "Records.json", "Settings.json", "2022\_JUNE.json", and "displaydata.html".
- Search Bar:** At the top center, it says "displaydata.html" and "2022\_JUNE.json".
- Code Editor:** The main area contains the following code:

```
<!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>
```
- Status Bar:** At the bottom right, it shows "Ln 38, Col 1" and "Spaces:4" and "CRLF" and "HTML".

## **OUTPUT :**

localhost:8000/displaydata.html

localhost:8000/displaydata.html

Netflix Watch The Midnigh... GitHub - labrijasaad... Twitter Sentiment A... Twitter Sentiment A... Home - Netflix

# Data Handling using JSON

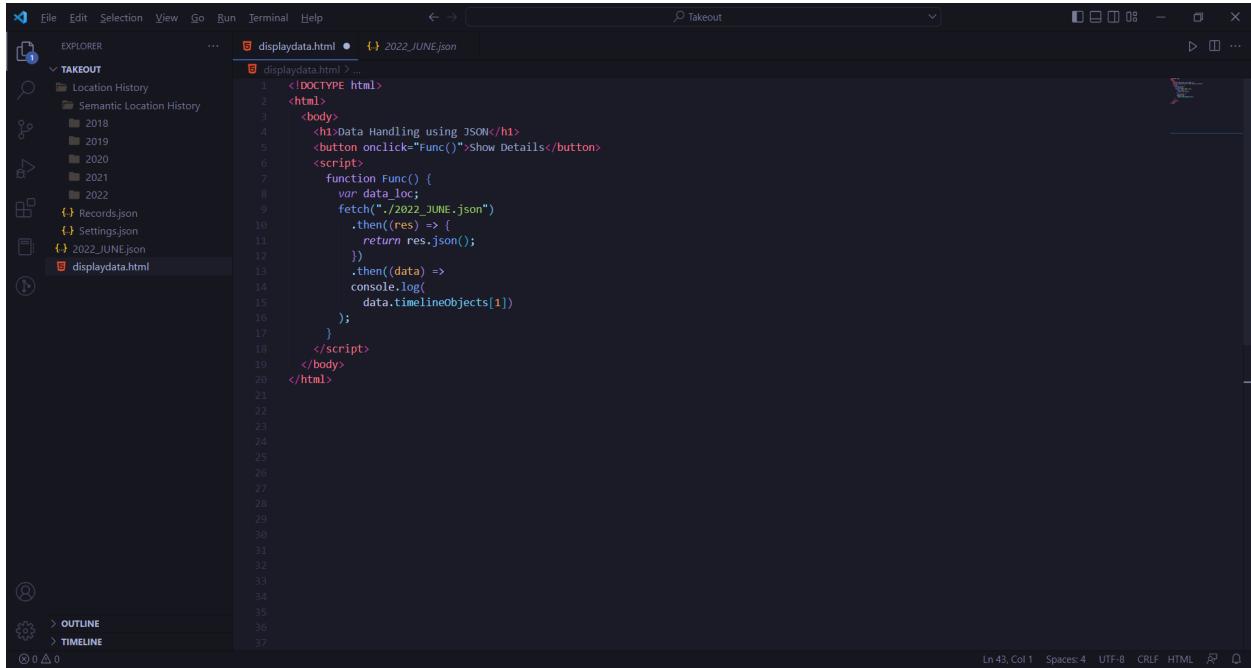
Show Details

```
⚠ DevTools failed to load source map: Could not load content for chrome-extension://feogkgkdfchfphceifdbepaoicaho/sourceMap/chrome/scripts/iframe_form_check.map: System error: net::ERR_BLOCKED_BY_CLIENT
```

displaydata.html:14

```
* {activitySegment: {}} ↴
  * activitySegment
    * activities: (15) [{} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {}]
      * activityType: "IN_PASSENGER_VEHICLE"
      * confidence: "LOW"
      * distance: 1665
    * duration:
      * endTimeStamp: "2022-06-03T13:37:10.241Z"
      * startTimeStamp: "2022-06-03T12:45:39.015Z"
    * endLocation:
      * latitudeE7: 191740278
      * longitudeE7: 728600495
    * sourceInfo: {deviceTag: 912782204}
    * timestamp: "2022-06-03T13:38:35.646Z"
  * parkingEvent:
    * location: {latitudeE7: 191739130, longitudeE7: 728600935, accuracyMetres: 77}
    * locationSource: "NOT FROM RAW LOCATION"
    * method: "END_OF_ACTIVITY_SEGMENT"
    * timestamp: "2022-06-03T13:38:35.646Z"
  * startLocation:
    * latitudeE7: 191817040
    * longitudeE7: 728534531
  * sourceInfo: {deviceTag: 912782204}
  * timestamp: "2022-06-03T13:38:35.646Z"
  * waypointPath:
    * confidence: 1
    * distanceMeters: 1998.8746174180279
    * waypoints: (20) [{} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {} , {}]
      * source: "PREDIRED"
      * travelMode: "DRIVE"
    * waypoints: (3) [{} , {} , {} ]
    * timestamp: "2022-06-03T13:38:35.646Z"
  * timestamp: "2022-06-03T13:38:35.646Z"
```

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



The screenshot shows the Visual Studio Code interface. The left sidebar has a tree view under 'EXPLORER' labeled 'TAKEOUT' containing files like 'Location History', 'Semantic Location History', 'Records.json', 'Settings.json', '2022\_JUNE.json', and 'displaydata.html'. The main editor area contains the following code:

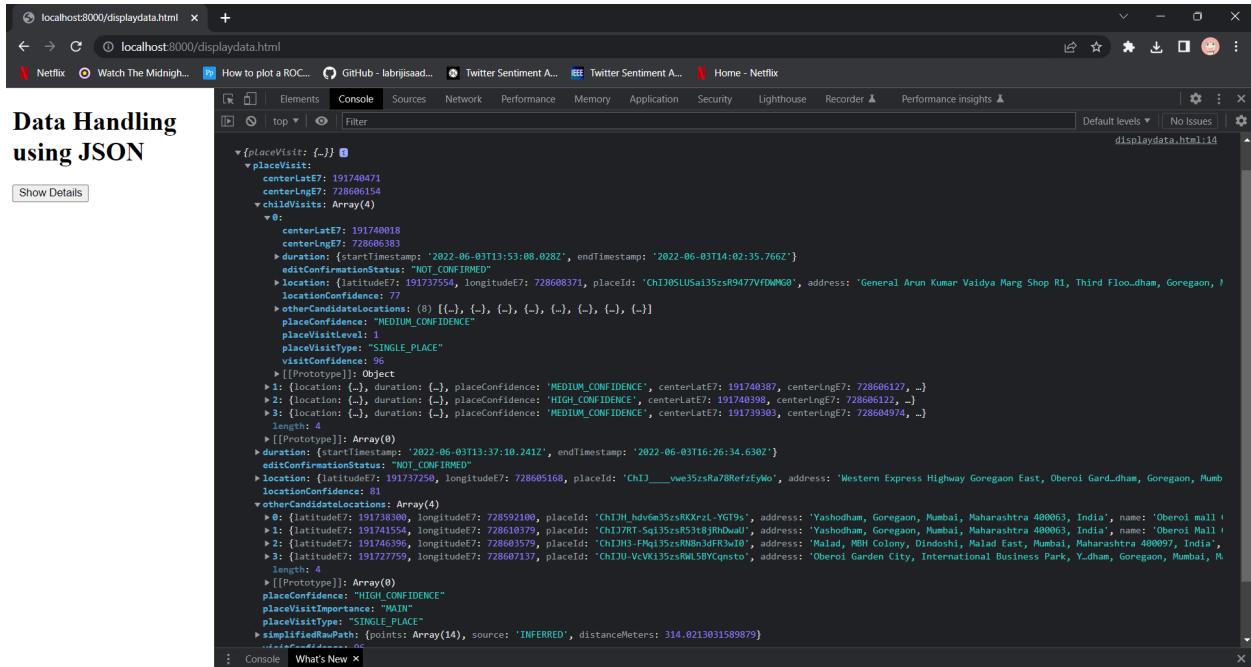
```

<!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[1])
}
</script>
</body>
</html>

```

The status bar at the bottom indicates 'Line 43 Col 1 Spaces: 4 UTF-8 CR LF HTML'.

## OUTPUT:



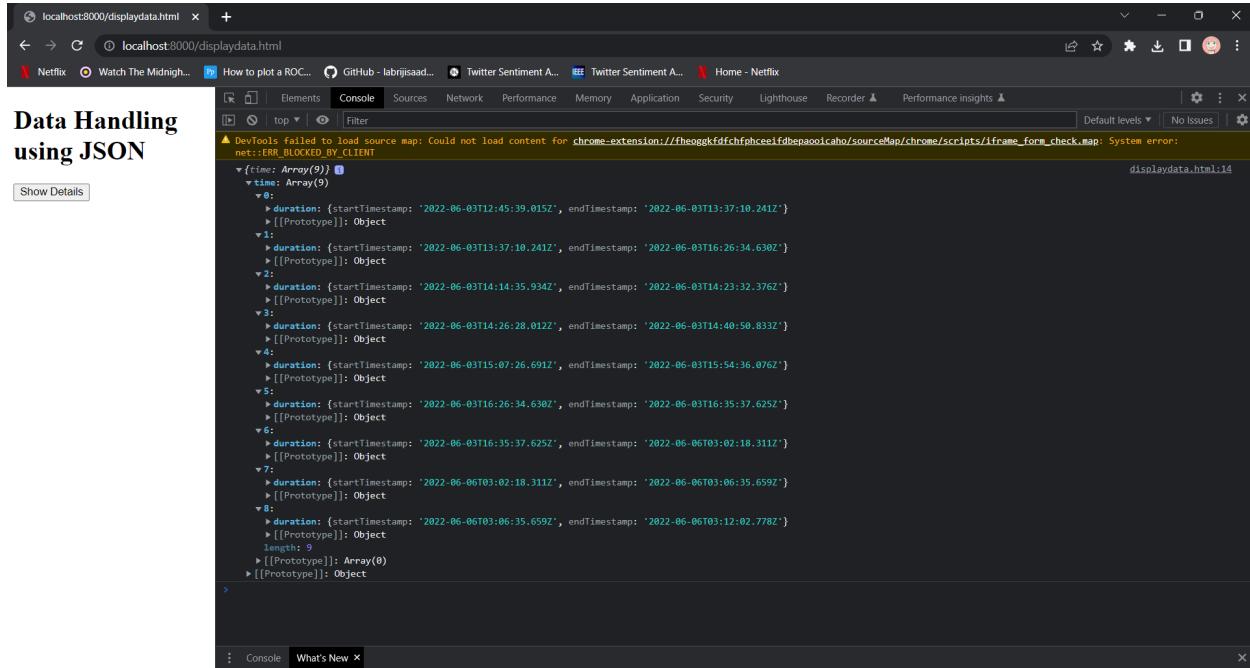
The screenshot shows a browser window with the URL 'localhost:8000/displaydata.html'. The browser's developer tools are open, specifically the 'Console' tab. The output shows the expanded JSON object from the previous code execution:

```

{
  "placeVisit": {
    "placeVisit": {
      "centerLatE7": 191740471,
      "centerLangE7": 728666154,
      "childVisits": [
        {
          "centerLatE7": 191740018,
          "centerLangE7": 728606383,
          "duration": {
            "startTimestamp": "2022-06-03T13:53:08.028Z",
            "endTimestamp": "2022-06-03T14:02:35.766Z"
          },
          "editConfirmationStatus": "NOT CONFIRMED",
          "location": {
            "latitudeE7": 191737554,
            "longitudeE7": 728688371,
            "placeId": "ChIJ05lUSAi35zsR9477VfDMPG0",
            "address": "General Arun Kumar Valida Marg Shop R1, Third Floor,dham, Goregaon, locationConfidence: 77
          },
          "otherCandidateLocations": [
            {
              "placeConfidence": "MEDIUM_CONFIDENCE"
            }
          ],
          "placeVisitLevel": 1,
          "placeVisitType": "SINGLE_PLACE",
          "visitConfidence": 96
        }
      ]
    }
  }
}

```

## 5) TEMPORAL QUERY



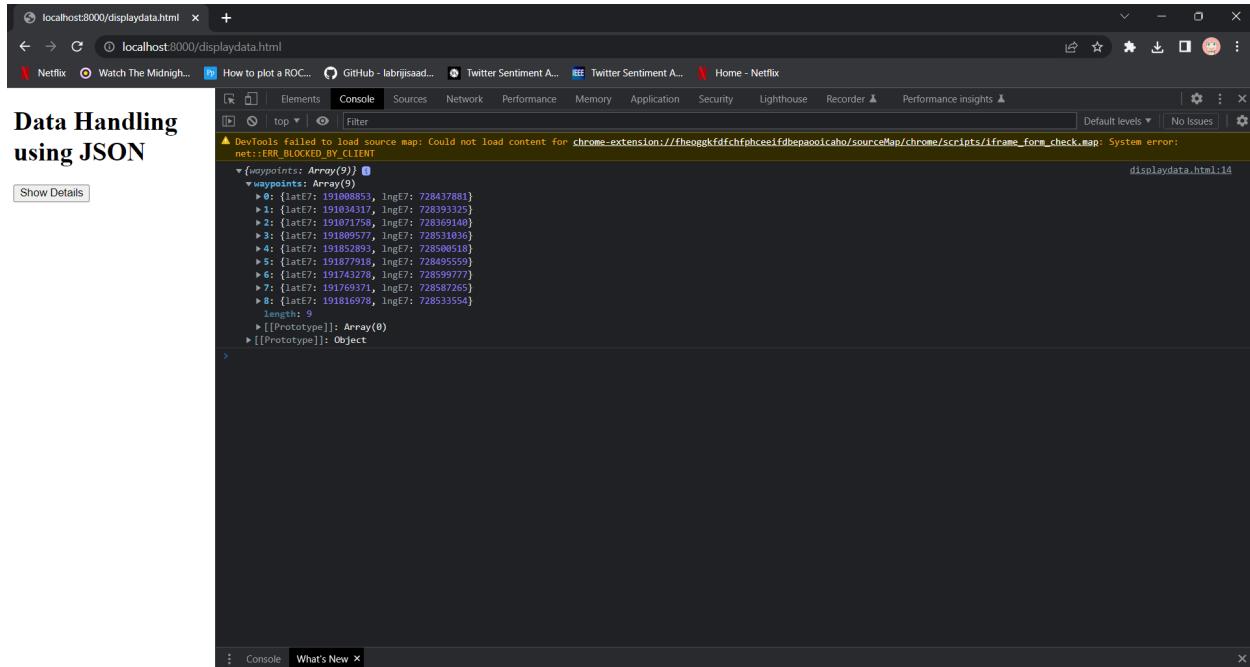
The screenshot shows a browser developer tools console with the URL `localhost:8000/displaydata.html`. The console output displays a JSON object representing a series of time intervals (timepoints) with their start and end timestamps.

```

{
  "timepoints": [
    {
      "startTimestamp": "2022-06-03T12:45:39.015Z",
      "endTimestamp": "2022-06-03T13:37:10.241Z"
    },
    {
      "startTimestamp": "2022-06-03T13:37:10.241Z",
      "endTimestamp": "2022-06-03T16:26:34.630Z"
    },
    {
      "startTimestamp": "2022-06-03T14:14:35.934Z",
      "endTimestamp": "2022-06-03T14:23:32.376Z"
    },
    {
      "startTimestamp": "2022-06-03T14:26:28.012Z",
      "endTimestamp": "2022-06-03T14:40:50.833Z"
    },
    {
      "startTimestamp": "2022-06-03T15:07:26.691Z",
      "endTimestamp": "2022-06-03T15:54:36.076Z"
    },
    {
      "startTimestamp": "2022-06-03T16:26:34.630Z",
      "endTimestamp": "2022-06-03T16:35:37.625Z"
    },
    {
      "startTimestamp": "2022-06-03T16:35:37.625Z",
      "endTimestamp": "2022-06-06T03:02:18.311Z"
    },
    {
      "startTimestamp": "2022-06-06T03:02:18.311Z",
      "endTimestamp": "2022-06-06T03:06:35.659Z"
    },
    {
      "startTimestamp": "2022-06-06T03:06:35.659Z",
      "endTimestamp": "2022-06-06T03:12:02.778Z"
    }
  ]
}

```

## 6) SPATIAL QUERY ON COORDINATES



The screenshot shows a browser developer tools console with the URL `localhost:8000/displaydata.html`. The console output displays a JSON object representing a series of waypoints, each with a latitude (latE) and longitude (lngE).

```

{
  "waypoints": [
    {
      "latE": 191088853,
      "lngE": 728437881
    },
    {
      "latE": 191034317,
      "lngE": 728393325
    },
    {
      "latE": 191071758,
      "lngE": 728369140
    },
    {
      "latE": 1918899577,
      "lngE": 728511036
    },
    {
      "latE": 191852893,
      "lngE": 728580518
    },
    {
      "latE": 191877918,
      "lngE": 728495559
    },
    {
      "latE": 191743278,
      "lngE": 728599777
    },
    {
      "latE": 191769371,
      "lngE": 728587265
    },
    {
      "latE": 191816978,
      "lngE": 728533554
    }
  ]
}

```

CONCLUSION :-

After serializing and deserializing, JSON data obtained from ~~grocery~~ 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. Customers expect information delivered based on, among other things, where they are. For example, a person touring the historic German city of Regensburg could receive 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 critical when a database management system has built-in temporal support.

#### Managing Spatial & Temporal data :-

A spatial database is a data management system for the collection, storage, manipulation and output of spatially related

information. A theme refers to data describing a particular topic (e.g - 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:**

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
2022-06-07	03:43:00	9966	WALKING	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	11:28:00	501	WALKING	LOW	Vile Parle
2022-06-11	11:53:00	322	WALKING	MEDIUM	McDonald's
2022-06-11	11:59:00	176	WALKING	MEDIUM	SVKM's Dwarkadas J. Sanghvi College of Engineering
2022-06-11	11:59:00	4413	MOTORCYCLING	LOW	Surbhi Sweets and Snacks
2022-06-14	12:38:00	886	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	666	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**

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

U1 11/22

NAME - PRACHI PATEL

f  
24  
25

SAP ID - 60004200049

BRANCH - Computer  
EngineeringExperiment - 10 :-Aim - Case study on Database Security Issues.

Patient 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 restrictions

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



FOR EDUCATIONAL USE

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 <sup>polarity</sup> of selectable nodes.
- 5) The method of claim 1, wherein at least one data source is to be used in maintaining 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 maintaining the data store.
- 7) The method of claim 1, wherein the generating the identity fingerprint comprises comprising the set of data.
- 8) The method of claim 1, wherein the generating <sup>the</sup> 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 <sup>of data</sup> of 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 in any one or more of - cellular activity and plan data.
- 18) The medium of claim 15, wherein the generating identity fingerprint comprises of comprising 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 more modes specify the accessible disclose the security fingerprints 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.