Created Indicies:

```
project=# CREATE INDEX idx_ProtocolTable_flowId ON ProtocolTable (flowId);
CREATE INDEX idx_ProtocolTable_L7Protocol ON ProtocolTable (L7Protocol);
CREATE INDEX idx_FlowTable_flowDuration_timestamp ON FlowTable (flowDuration, timestamp);
CREATE INDEX idx_ProtocolNameTable_ProtocolName ON ProtocolNameTable (ProtocolName);
CREATE INDEX idx_flowId ON SubflowTable (flowId);
CREATE INDEX idx_flowId3 ON FlowTable (flowId);
CREATE INDEX idx_L7Protocol1 ON ProtocolNameTable (L7Protocol);
CREATE INDEX idx_timestamp ON FlowTable (timestamp);
CREATE INDEX idx_flowid2 ON ActivityTable (flowid);
CREATE INDEX idx_ActiveMax ON ActivityTable (ActiveMax);
CREATE INDEX idx_ProtocolTable_L7Protocol_flowID ON ProtocolTable (L7Protocol, flowID);
CREATE INDEX idx_IATTable_flowID ON IATTable (flowID);
CREATE INDEX idx_IATTable_flowIATMax ON IATTable (flowIATMax);
CREATE INDEX idx_ratio_downupratio ON RatioTable(downUpRatio);
CREATE INDEX idx_flows_flowDuration ON FlowTable(flowDuration);
```

CREATE INDEX idx_ProtocolTable_flowId ON ProtocolTable (flowId);

CREATE INDEX idx_ProtocolTable_L7Protocol ON ProtocolTable (L7Protocol);

CREATE INDEX idx_FlowTable_flowDuration_timestamp ON FlowTable (flowDuration, timestamp);

CREATE INDEX idx_ProtocolNameTable_ProtocolName ON ProtocolNameTable (ProtocolName);

CREATE INDEX idx_flowId ON SubflowTable (flowId);

CREATE INDEX idx_flowId3 ON FlowTable (flowId);

CREATE INDEX idx_L7Protocol1 ON ProtocolNameTable (L7Protocol);

CREATE INDEX idx_timestamp ON FlowTable (timestamp);

CREATE INDEX idx_flowid2 ON ActivityTable (flowid);

CREATE INDEX idx_ActiveMax ON ActivityTable (ActiveMax);

CREATE INDEX idx_ProtocolTable_L7Protocol_flowID ON ProtocolTable (L7Protocol, flowID);

CREATE INDEX idx_IATTable_flowID ON IATTable (flowID);

CREATE INDEX idx_IATTable_flowIATMax ON IATTable (flowIATMax);

CREATE INDEX idx_ratio_downupratio ON RatioTable(downUpRatio);

CREATE INDEX idx_flows_flowDuration ON FlowTable(flowDuration);

Size of database:

```
project=# SELECT pg_size_pretty(pg_database_size('project'));
 pg_size_pretty
----------------
 6194 MB
(1 row)
```

Query1:Find all the apps for the data is available

SQL Code:

SELECT ProtocolName

FROM ProtocolNameTable

ORDER BY ProtocolName;

Time without indexing:

```
Time: 1.953 ms
```

Time after indexing:

```
 SKINNY
Time: 0.862 ms
```

Query Plan

```
                              QUERY PLAN
-----------------------------------------------------------------------
 Sort  (cost=4.23..4.43 rows=78 width=8)
   Sort Key: protocolname
   ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=8)
(3 rows)
```

Query2: 2. Find apps using longest flows in flowDuration at a given timestamp.

SQL Code:

SELECT ProtocolNameTable.ProtocolName, FlowTable.flowDuration , FlowTable.timestamp

FROM FlowTable,ProtocolTable,ProtocolNameTable

WHERE FlowTable.flowId = ProtocolTable.flowId AND ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol

ORDER BY flowDuration DESC

LIMIT 10;

Time Without indexing:

```
Time: 6319.196 ms
```

Time after indexing:

```
Time: 16.995 ms
```

Query Plan

```
LIMIT 10;
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------
 Limit  (cost=528699.46..528699.48 rows=10 width=42)
   ->  Sort  (cost=528699.46..537642.70 rows=3577296 width=42)
         Sort Key: flowtable.flowduration
         ->  Hash Join  (cost=117296.92..451395.38 rows=3577296 width=42)
               Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
               ->  Hash Join  (cost=117294.16..402204.80 rows=3577296 width=41)
                     Hash Cond: (flowtable.flowid = protocoltable.flowid)
                     ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=42)
                     ->  Hash  (cost=55109.96..55109.96 rows=3577296 width=15)
                           ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
               ->  Hash  (cost=1.78..1.78 rows=78 width=15)
                     ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(12 rows)
```

Query3: Count number of flows for each protocol

SQL Code:

SELECT protocol, COUNT(*) AS numFlows

FROM FlowTable

GROUP BY protocol;

Time Without Indexing:

```
Time: 2002.616 ms
```

Time after indexing:

```
Time: 1358.120 ms
```

Query Plan

```
GROUP BY protocol;
                              QUERY PLAN
-----------------------------------------------------------------------
 HashAggregate  (cost=159842.44..159842.48 rows=3 width=6)
   ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=6)
(2 rows)
```

Query4: Calculate the average and standard deviation of the total length of forward packets for each protocol.

SQL Code:

SELECT protocol, AVG(totalLengthOfFwdPackets) AS avgFwdLength, STDDEV(totalLengthOfFwdPackets) AS stdFwdLength

FROM LengthTable

JOIN FlowTable ON LengthTable.flowId = FlowTable.flowId

GROUP BY protocol;

Time Without Indexing:

```
Time: 9656.276 ms
```

Time after indexing:

```
Time: 9513.011 ms
```

Query Plan

```
                                    QUERY PLAN
-----------------------------------------------------------------------------------
 HashAggregate  (cost=438959.52..438959.57 rows=3 width=12)
   ->  Hash Join  (cost=155167.16..412129.80 rows=3577296 width=12)
         Hash Cond: (flowtable.flowid = lengthtable.flowid)
         ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=14)
         ->  Hash  (cost=92982.96..92982.96 rows=3577296 width=14)
               ->  Seq Scan on lengthtable  (cost=0.00..92982.96 rows=3577296 width=14)
(6 rows)
```

Query5: Identify top 20 apps with a high number of FIN flags

SELECT ProtocolNameTable.ProtocolName, SUM(FINFlagCount)

FROM TCPFlagTable , ProtocolTable , ProtocolNameTable

WHERE TCPFlagTable.flowId = ProtocolTable.flowId AND ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol

GROUP BY ProtocolNameTable.ProtocolName

ORDER BY SUM(FINFlagCount) DESC

LIMIT 20;

Time without indexing:

```
Time: 6196.789 ms
```

Time after indexing:

```
Time: 6172.827 ms
```

Query Plan

```
LIMIT 20;
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Limit  (cost=372033.91..372033.96 rows=20 width=13)
   ->  Sort  (cost=372033.91..372034.10 rows=78 width=13)
         Sort Key: (sum(tcpflagtable.finflagcount))
         ->  HashAggregate  (cost=372030.85..372031.83 rows=78 width=13)
               ->  Hash Join  (cost=117296.92..354144.38 rows=3577296 width=13)
                     Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
                     ->  Hash Join  (cost=117294.16..304953.80 rows=3577296 width=12)
                           Hash Cond: (tcpflagtable.flowid = protocoltable.flowid)
                           ->  Seq Scan on tcpflagtable  (cost=0.00..72652.96 rows=3577296 width=13)
                           ->  Hash  (cost=55109.96..55109.96 rows=3577296 width=15)
                                 ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
                     ->  Hash  (cost=1.78..1.78 rows=78 width=15)
                           ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(13 rows)
```

Query6: Apps having flows with a high number of subflows at a given timestamp.

SELECT ProtocolNameTable.ProtocolName, SubflowTable.SubflowFwdPackets, SubflowTable.SubFlowBwdPackets , FlowTable.timestamp

FROM SubflowTable,ProtocolTable,ProtocolNameTable,FlowTable

WHERE SubflowTable.flowId = FlowTable.flowId AND FlowTable.flowId = ProtocolTable.flowId AND ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol AND SubflowTable.SubflowFwdPackets + SubflowTable.SubFlowBwdPackets > 10000;

Time without indexing:

```
Time: 4349.807 ms
```

Time after indexing:

```
Time: 4115.459 ms
```

Query Plan

```
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Hash Join  (cost=269865.61..536655.18 rows=1192432 width=47)
   Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
   ->  Hash Join  (cost=269862.86..520256.48 rows=1192432 width=46)
         Hash Cond: (flowtable.flowid = subflowtable.flowid)
         ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=34)
         ->  Hash  (cost=245641.46..245641.46 rows=1192432 width=36)
               ->  Hash Join  (cost=105382.84..245641.46 rows=1192432 width=36)
                     Hash Cond: (protocoltable.flowid = subflowtable.flowid)
                     ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
                     ->  Hash  (cost=83490.44..83490.44 rows=1192432 width=21)
                           ->  Seq Scan on subflowtable  (cost=0.00..83490.44 rows=1192432 width=21)
                                 Filter: ((subflowfwdpackets + subflowbwdpackets) > 10000::numeric)
   ->  Hash  (cost=1.78..1.78 rows=78 width=15)
         ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(14 rows)
```

Query7: Calculate the average and standard deviation of the flow inter-arrival time for each protocol

SELECT protocol, AVG(flowIATMean) AS avgIAT, STDDEV(flowIATMean) AS stdIAT

FROM IATTable

JOIN FlowTable ON IATTable.flowId = FlowTable.flowId

GROUP BY protocol;

Time without indexing:

```
Time: 6077.188 ms
```

Time after indexing:

```
Time: 5338.290 ms
```

Query Plan

```
                                    QUERY PLAN
-----------------------------------------------------------------------------------
 HashAggregate  (cost=452518.52..452518.57 rows=3 width=14)
   ->  Hash Join  (cost=168726.16..425688.80 rows=3577296 width=14)
         Hash Cond: (flowtable.flowid = iattable.flowid)
         ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=14)
         ->  Hash  (cost=106541.96..106541.96 rows=3577296 width=16)
               ->  Seq Scan on iattable  (cost=0.00..106541.96 rows=3577296 width=16)
(6 rows)
```

Query8: Time used by apps

SELECT ProtocolNameTable.ProtocolName , SUM(FlowTable.flowDuration) AS total_time_used

FROM FlowTable,ProtocolTable,ProtocolNameTable

WHERE FlowTable.flowId = ProtocolTable.flowId AND ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol

GROUP BY ProtocolNameTable.ProtocolName

ORDER BY SUM(FlowTable.flowDuration) DESC LIMIT 10;

Time without indexing:

```
Time: 6383.733 ms
```

Time after indexing:

```
Time: 6251.721 ms
```

Query Plan

```
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Limit  (cost=441336.52..441336.54 rows=10 width=16)
   ->  Sort  (cost=441336.52..441336.71 rows=78 width=16)
         Sort Key: (sum(flowtable.flowduration))
         ->  HashAggregate  (cost=441333.85..441334.83 rows=78 width=16)
               ->  Hash Join  (cost=117296.92..423447.38 rows=3577296 width=16)
                     Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
                     ->  Hash Join  (cost=117294.16..374256.80 rows=3577296 width=15)
                           Hash Cond: (flowtable.flowid = protocoltable.flowid)
                           ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=16)
                           ->  Hash  (cost=55109.96..55109.96 rows=3577296 width=15)
                                 ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
                     ->  Hash  (cost=1.78..1.78 rows=78 width=15)
                           ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(13 rows)
```

Query9: Find the top 10 most active Apps in terms of maximum active time

SELECT ProtocolNameTable.ProtocolName, ActivityTable.ActiveMax

FROM ActivityTable,ProtocolTable,ProtocolNameTable

WHERE ActivityTable.flowid = ProtocolTable.flowid AND ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol

ORDER BY ActiveMax DESC

LIMIT 10;

Time without indexing:

```
Time: 5834.684 ms
```

Time after indexing:

```
Time: 10.530 ms
```

Query Plan

```
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Limit  (cost=443545.46..443545.48 rows=10 width=13)
   ->  Sort  (cost=443545.46..452488.70 rows=3577296 width=13)
         Sort Key: activitytable.activemax
         ->  Hash Join  (cost=117296.92..366241.38 rows=3577296 width=13)
               Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
               ->  Hash Join  (cost=117294.16..317050.80 rows=3577296 width=12)
                     Hash Cond: (activitytable.flowid = protocoltable.flowid)
                     ->  Seq Scan on activitytable  (cost=0.00..84749.96 rows=3577296 width=13)
                     ->  Hash  (cost=55109.96..55109.96 rows=3577296 width=15)
                           ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
               ->  Hash  (cost=1.78..1.78 rows=78 width=15)
                     ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(12 rows)
```

Query10: Potential Threats for Non UDP,TCP Protocols

SELECT * FROM FlowTable

WHERE protocol NOT IN (6, 17);

Time without indexing:

```
Time: 894.207 ms
```

Time after indexing:

```
Time: 870.078 ms
```

Query Plan

```
                              QUERY PLAN
-----------------------------------------------------------------
 Seq Scan on flowtable  (cost=0.00..150899.20 rows=5123 width=88)
   Filter: (protocol <> ALL ('{6,17}'::numeric[]))
(2 rows)
```

Query11: Potential Threats which are FTP , Telnet , SNMP

SELECT *

FROM FlowTable,ProtocolNameTable,ProtocolTable

WHERE FlowTable.flowId = ProtocolTable.flowId AND ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol AND ProtocolName IN ('FTP', 'Telnet', 'SNMP');

Time without indexing:

```
Time: 1932.731 ms
```

Time after indexing:

```
Time: 908.897 ms
```

Query Plan

```
project=# explain SELECT *
FROM FlowTable,ProtocolNameTable,ProtocolTable
WHERE FlowTable.flowId = ProtocolTable.flowId AND ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol AND ProtocolName IN ('FTP', 'Telnet', 'SNMP');
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Hash Join  (cost=72563.66..341484.22 rows=137588 width=118)
   Hash Cond: (flowtable.flowid = protocoltable.flowid)
   ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=88)
   ->  Hash  (cost=69902.81..69902.81 rows=137588 width=30)
         ->  Hash Join  (cost=2.11..69902.81 rows=137588 width=30)
               Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
               ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
               ->  Hash  (cost=2.07..2.07 rows=3 width=15)
                     ->  Seq Scan on protocolnametable  (cost=0.00..2.07 rows=3 width=15)
                           Filter: ((protocolname)::text = ANY ('{FTP,Telnet,SNMP}'::text[]))
(10 rows)
```

Query12: To optimize resource allocation, we could analyze network usage and identify areas where resources can be reallocated to improve performance and reduce costs.

SELECT sourcePort, destinationPort, COUNT(*) AS numFlows FROM FlowTable

GROUP BY sourcePort, destinationPort

ORDER BY numFlows DESC

LIMIT 10;

Time without indexing:

```
Time: 15687.571 ms
```

Time after indexing:

```
Time: 15455.314 ms
```

Query Plan

```
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Limit  (cost=762741.47..762741.49 rows=10 width=15)
   ->  Sort  (cost=762741.47..763635.79 rows=357730 width=15)
         Sort Key: (count(*))
         ->  GroupAggregate  (cost=714766.47..755011.05 rows=357730 width=15)
               ->  Sort  (cost=714766.47..723709.71 rows=3577296 width=15)
                     Sort Key: sourceport, destinationport
                     ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=15)
(7 rows)
```

Query13: query to find the source/destination pairs that have the longest average flow duration (network optimization)

SELECT sourceIp, destinationIp, AVG(flowDuration) AS avgDuration FROM FlowTable

GROUP BY sourceIp, destinationIp

ORDER BY avgDuration DESC

LIMIT 10;

Time without indexing:

```
Time: 61256.143 ms
```

Time after indexing:

```
Time: 15455.314 ms
```

Query Plan

```
LIMIT 10;
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------
 Limit  (cost=872781.47..872781.49 rows=10 width=33)
   ->  Sort  (cost=872781.47..873675.79 rows=357730 width=33)
         Sort Key: (avg(flowduration))
         ->  GroupAggregate  (cost=824806.47..865051.05 rows=357730 width=33)
               ->  Sort  (cost=824806.47..833749.71 rows=3577296 width=33)
                     Sort Key: sourceip, destinationip
                     ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=33)
(7 rows)
```

Query14: A lower average idle time could indicate a more active network flow, while a higher average idle time could suggest a less active or stalled flow.

SELECT ProtocolNameTable.ProtocolName , AVG(ActivityTable.IdleMean) AS avg_idle_time

FROM ActivityTable,FlowTable,ProtocolNameTable,ProtocolTable

WHERE FlowTable.flowDuration > 10 AND ActivityTable.flowId = FlowTable.flowID AND FlowTable.flowID = ProtocolTable.flowID AND ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol

GROUP BY ProtocolNameTable.ProtocolName

ORDER BY AVG(ActivityTable.IdleMean) DESC

LIMIT 10;

Time without indexing:

```
Time: 9666.231 ms
```

Time after indexing:

```
Time: 4760.895 ms
```

Query Plan

```
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Limit  (cost=693928.49..693928.51 rows=10 width=16)
   ->  Sort  (cost=693928.49..693928.68 rows=78 width=16)
         Sort Key: (avg(activitytable.idlemean))
         ->  HashAggregate  (cost=693925.83..693926.80 rows=78 width=16)
               ->  Hash Join  (cost=323990.40..676922.11 rows=3400743 width=16)
                     Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
                     ->  Hash Join  (cost=323987.65..630159.14 rows=3400743 width=15)
                           Hash Cond: (activitytable.flowid = protocoltable.flowid)
                           ->  Hash Join  (cost=206693.49..396029.98 rows=3400743 width=24)
                                 Hash Cond: (activitytable.flowid = flowtable.flowid)
                                 ->  Seq Scan on activitytable  (cost=0.00..84749.96 rows=3577296 width=16)
                                 ->  Hash  (cost=150899.20..150899.20 rows=3400743 width=8)
                                       ->  Seq Scan on flowtable  (cost=0.00..150899.20 rows=3400743 width=8)
                                             Filter: (flowduration > 10::numeric)
                           ->  Hash  (cost=55109.96..55109.96 rows=3577296 width=15)
                                 ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
                     ->  Hash  (cost=1.78..1.78 rows=78 width=15)
                           ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(18 rows)
```

Query15: Query to find the average packet length for each protocol

SELECT ProtocolNameTable.ProtocolName, AVG(PacketSizeTable.averagePacketSize) AS avg_packet_length

FROM PacketSizeTable

JOIN ProtocolTable ON PacketSizeTable.flowId = ProtocolTable.flowID

JOIN ProtocolNameTable ON ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol

GROUP BY ProtocolNameTable.ProtocolName;

Time without indexing:

```
Time: 5701.019 ms
```

Time after indexing:

```
Time: 3494.241 ms
```

Query Plan

```
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------
 HashAggregate  (cost=379315.85..379316.83 rows=78 width=16)
   ->  Hash Join  (cost=117296.92..361429.38 rows=3577296 width=16)
         Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
         ->  Hash Join  (cost=117294.16..312238.80 rows=3577296 width=15)
               Hash Cond: (packetsizetable.flowid = protocoltable.flowid)
               ->  Seq Scan on packetsizetable  (cost=0.00..79937.96 rows=3577296 width=16)
               ->  Hash  (cost=55109.96..55109.96 rows=3577296 width=15)
                     ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
         ->  Hash  (cost=1.78..1.78 rows=78 width=15)
               ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(10 rows)
```

Query16: This query can be used to identify flows that have unusually long inter-arrival times between packets, which could be indicative of certain types of applications or activities.

SELECT ProtocolNameTable.ProtocolName, IATTable.flowIATMax, FlowTable.timestamp

FROM ProtocolNameTable

INNER JOIN ProtocolTable ON ProtocolNameTable.L7Protocol = ProtocolTable.L7Protocol

INNER JOIN FlowTable ON ProtocolTable.flowID = FlowTable.flowID

INNER JOIN IATTable ON FlowTable.flowID = IATTable.flowID

ORDER BY IATTable.flowIATMax DESC

LIMIT 10;

Time without indexing:

```
Time: 10888.795 ms
```

Time after indexing:

```
Time: 3494.241 ms
Time: 4.600 ms
```

Query Plan

```
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 Limit  (cost=819418.30..819418.32 rows=10 width=42)
   ->  Sort  (cost=819418.30..828361.54 rows=3577296 width=42)
         Sort Key: iattable.flowiatmax
         ->  Hash Join  (cost=408016.76..742114.22 rows=3577296 width=42)
               Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
               ->  Hash Join  (cost=408014.00..692923.64 rows=3577296 width=41)
                     Hash Cond: (flowtable.flowid = protocoltable.flowid)
                     ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=34)
                     ->  Hash  (cost=338842.80..338842.80 rows=3577296 width=31)
                           ->  Hash Join  (cost=117294.16..338842.80 rows=3577296 width=31)
                                 Hash Cond: (iattable.flowid = protocoltable.flowid)
                                 ->  Seq Scan on iattable  (cost=0.00..106541.96 rows=3577296 width=16)
                                 ->  Hash  (cost=55109.96..55109.96 rows=3577296 width=15)
                                       ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
               ->  Hash  (cost=1.78..1.78 rows=78 width=15)
                     ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(16 rows)
```

Query17: This table can be useful for analyzing network traffic patterns and identifying potential network congestion or bandwidth issues.

For example, a high down/up ratio for a particular flow could indicate that the flow is consuming more downlink bandwidth than uplink bandwidth, which could lead to network congestion or other performance issues. Network administrators can use this information to identify and resolve such issues.

SELECT ProtocolNameTable.ProtocolName, FlowTable.TimeStamp, RatioTable.downUpRatio

FROM FlowTable

JOIN ProtocolTable ON FlowTable.flowId = ProtocolTable.flowId

JOIN ProtocolNameTable ON ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol

JOIN RatioTable ON FlowTable.flowId = RatioTable.flowId

WHERE RatioTable.downUpRatio > 200

ORDER BY RatioTable.downUpRatio DESC;

Time without indexing:

```
Time: 505.804 ms
```

Time after indexing:

```
Time: 3.013 ms
```

Query Plan

```
ORDER BY RatioTable.downUpRatio DESC;
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Sort  (cost=67597.48..67597.50 rows=5 width=42)
   Sort Key: ratiotable.downupratio
   ->  Nested Loop  (cost=0.00..67597.42 rows=5 width=42)
         ->  Nested Loop  (cost=0.00..67586.02 rows=5 width=41)
               ->  Nested Loop  (cost=0.00..67541.87 rows=5 width=50)
                     ->  Seq Scan on ratiotable  (cost=0.00..67492.20 rows=5 width=16)
                           Filter: (downupratio > 200::double precision)
                     ->  Index Scan using flowtable_pkey on flowtable  (cost=0.00..9.92 rows=1 width=34)
                           Index Cond: (flowtable.flowid = ratiotable.flowid)
               ->  Index Scan using protocoltable_pkey on protocoltable  (cost=0.00..8.82 rows=1 width=15)
                     Index Cond: (protocoltable.flowid = flowtable.flowid)
         ->  Index Scan using protocolnametable_pkey on protocolnametable  (cost=0.00..2.27 rows=1 width=15)
               Index Cond: (protocolnametable.l7protocol = protocoltable.l7protocol)
(13 rows)
```

Query18: For a particular time range which app uses time.

SELECT ProtocolNameTable.ProtocolName, SUM(FlowTable.flowDuration) AS count

FROM FlowTable

INNER JOIN ProtocolTable ON FlowTable.flowID = ProtocolTable.flowID

INNER JOIN ProtocolNameTable ON ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol

WHERE timestamp >= '2017-04-27 11:00:00' AND timestamp < '2017-04-27 12:00:00'

GROUP BY ProtocolNameTable.ProtocolName

ORDER BY count DESC;

Time without indexing:

```
Time: 3571.243 ms
```

Time after indexing:

```
Time: 998.529 ms
```

Query Plan

```
                                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=277631.26..277631.45 rows=78 width=16)
   Sort Key: (sum(flowtable.flowduration))
   ->  HashAggregate  (cost=277627.83..277628.80 rows=78 width=16)
         ->  Hash Join  (cost=161677.95..277100.73 rows=105420 width=16)
               Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
               ->  Hash Join  (cost=161675.19..275648.45 rows=105420 width=15)
                     Hash Cond: (protocoltable.flowid = flowtable.flowid)
                     ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
                     ->  Hash  (cost=159842.44..159842.44 rows=105420 width=16)
                           ->  Seq Scan on flowtable  (cost=0.00..159842.44 rows=105420 width=16)
                                 Filter: ((("timestamp")::text >= '2017-04-27 11:00:00'::text) AND (("timestamp")::text < '2017-04-27 12:00:00'::text))
               ->  Hash  (cost=1.78..1.78 rows=78 width=15)
                     ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(13 rows)
```

Query 19 :Top 10 flows with longest avgDuration

SELECT *

FROM FlowTable

ORDER BY flowDuration DESC

LIMIT 10;

Time without indexing:

```
Time: 940.877 ms
```

Time after indexing:

```
Time: 2.234 ms
```

Query Plan

```
                                QUERY PLAN
-------------------------------------------------------------------------------
 Limit  (cost=219260.04..219260.07 rows=10 width=88)
   ->  Sort  (cost=219260.04..228203.28 rows=3577296 width=88)
         Sort Key: flowduration
         ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=88)
(4 rows)
```

Query20:

```
WITH RECURSIVE paths AS (
  SELECT
    FlowTable.flowId,
    FlowTable.sourceIp,
    FlowTable.destinationIp,
    FlowTable.timestamp,
    ARRAY[FlowTable.flowId] AS flowIds
  FROM
    FlowTable
  WHERE
    FlowTable.sourceIp = '192.168.180.37'
    AND FlowTable.destinationIp = '10.200.7.7'

  UNION ALL

  SELECT
    f.flowId,
    p.sourceIp,
    f.destinationIp,
    f.timestamp,
    p.flowIds || f.flowId
  FROM
    paths p
    INNER JOIN FlowTable f ON p.destinationIp = f.sourceIp
    AND p.timestamp = f.timestamp
    AND array_length(p.flowIds,1) < 100
  WHERE
     NOT (f.flowId = ANY(p.flowIds))
```

```
)
SELECT
  flowIds
FROM
  paths
WHERE
  destinationIp = '10.200.7.7'
LIMIT 1;
```

Time without indexing:

```
Time: 180.263 ms
```

Time after indexing:

```
Time: 2.660 ms
```

Query Plan

```
LIMIT 1;
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Limit  (cost=3196545.25..3196549.77 rows=1 width=32)
   CTE paths
     ->  Recursive Union  (cost=0.00..3196545.25 rows=12642 width=136)
           ->  Seq Scan on flowtable  (cost=0.00..159842.44 rows=2512 width=59)
                 Filter: (((sourceip)::text = '192.168.180.37'::text) AND ((destinationip)::text = '10.200.7.7'::text))
           ->  Hash Join  (cost=234043.40..303645.00 rows=1013 width=136)
                 Hash Cond: (((p.destinationip)::text = (f.sourceip)::text) AND ((p."timestamp")::text = (f."timestamp")::text))
                 Join Filter: (f.flowid <> ALL (p.flowids))
                 ->  WorkTable Scan on paths p  (cost=0.00..628.00 rows=8373 width=226)
                       Filter: (array_length(flowids, 1) < 100)
                 ->  Hash  (cost=141955.96..141955.96 rows=3577296 width=59)
                       ->  Seq Scan on flowtable f  (cost=0.00..141955.96 rows=3577296 width=59)
   ->  CTE Scan on paths  (cost=0.00..284.44 rows=63 width=32)
         Filter: ((destinationip)::text = '10.200.7.7'::text)
(14 rows)
```

Query21: protocols having avg(initial window) in desc

```
SELECT
  ProtocolNameTable.ProtocolName,
  AVG(WindowTable.InitWinbytesforward + WindowTable.InitWinbytesbackward) AS AvgInitWinBytes
FROM
  ProtocolTable
  INNER JOIN ProtocolNameTable ON ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol
  INNER JOIN WindowTable ON ProtocolTable.flowId = WindowTable.flowId
```

```
GROUP BY
  ProtocolNameTable.ProtocolName
ORDER BY
  AvgInitWinBytes DESC
LIMIT 10;
```

Time without indexing:

```
Time: 8026.389 ms
```

Time after indexing:

```
Time: 4756.887 ms
```

Query Plan

```
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Limit  (cost=372049.71..372049.74 rows=10 width=22)
   ->  Sort  (cost=372049.71..372049.91 rows=78 width=22)
         Sort Key: (avg((windowtable.initwinbytesforward + windowtable.initwinbytesbackward)))
         ->  HashAggregate  (cost=372046.85..372048.03 rows=78 width=22)
               ->  Hash Join  (cost=117296.92..354160.38 rows=3577296 width=22)
                     Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
                     ->  Hash Join  (cost=117294.16..304969.80 rows=3577296 width=21)
                           Hash Cond: (windowtable.flowid = protocoltable.flowid)
                           ->  Seq Scan on windowtable  (cost=0.00..65682.96 rows=3577296 width=22)
                           ->  Hash  (cost=55109.96..55109.96 rows=3577296 width=15)
                                 ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
                     ->  Hash  (cost=1.78..1.78 rows=78 width=15)
                           ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(13 rows)
```

Query22: protocols having avg(initial window) negative

```
SELECT
  ProtocolNameTable.ProtocolName,
  AVG(WindowTable.InitWinbytesforward + WindowTable.InitWinbytesbackward) AS AvgInitWinBytes
FROM
  ProtocolTable
  INNER JOIN ProtocolNameTable ON ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol
  INNER JOIN WindowTable ON ProtocolTable.flowId = WindowTable.flowId
GROUP BY
  ProtocolNameTable.ProtocolName
having AVG(WindowTable.InitWinbytesforward + WindowTable.InitWinbytesbackward) < 0;
```

Time without indexing:

```
Time: 7939.998 ms
project=#
```

Time after indexing:

```
Time: 4664.274 ms
```

Query Plan

```
having avg((windowtable.initwinbytesforward + windowtable.initwinbytesbackward)) < 0;
                                        QUERY PLAN
-----------------------------------------------------------------------------------------------
 HashAggregate  (cost=380990.09..380991.85 rows=78 width=22)
   Filter: (avg((windowtable.initwinbytesforward + windowtable.initwinbytesbackward)) < 0::numeric)
   ->  Hash Join  (cost=117296.92..354160.38 rows=3577296 width=22)
         Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
         ->  Hash Join  (cost=117294.16..304969.80 rows=3577296 width=21)
               Hash Cond: (windowtable.flowid = protocoltable.flowid)
               ->  Seq Scan on windowtable  (cost=0.00..65682.96 rows=3577296 width=22)
               ->  Hash  (cost=55109.96..55109.96 rows=3577296 width=15)
                     ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
         ->  Hash  (cost=1.78..1.78 rows=78 width=15)
               ->  Seq Scan on protocolnametable  (cost=0.00..1.78 rows=78 width=15)
(11 rows)
```

Query23: daily activeness in 6 days

```
SELECT
  ProtocolNameTable.ProtocolName AS protocol_name,
  date_trunc('day', to_timestamp(FlowTable.timestamp, 'YYYY-MM-DD HH24:MI:SSOF')) AS day,
  SUM(FlowTable.flowDuration) AS time_used
FROM
  FlowTable
  JOIN ProtocolTable ON FlowTable.flowId = ProtocolTable.flowId
  JOIN ProtocolNameTable ON ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol
WHERE
  ProtocolNameTable.ProtocolName = 'FACEBOOK'
  AND to_timestamp(FlowTable.timestamp, 'YYYY-MM-DD HH24:MI:SSOF') BETWEEN '2017-04-26 03:03:25+05:30'  AND '2017-05-15 11:31:48+05:30'
GROUP BY
  ProtocolNameTable.ProtocolName,
```

```
  date_trunc('day', to_timestamp(FlowTable.timestamp, 'YYYY-MM-DD HH24:MI:SSOF'))
ORDER BY
  day ASC;
```

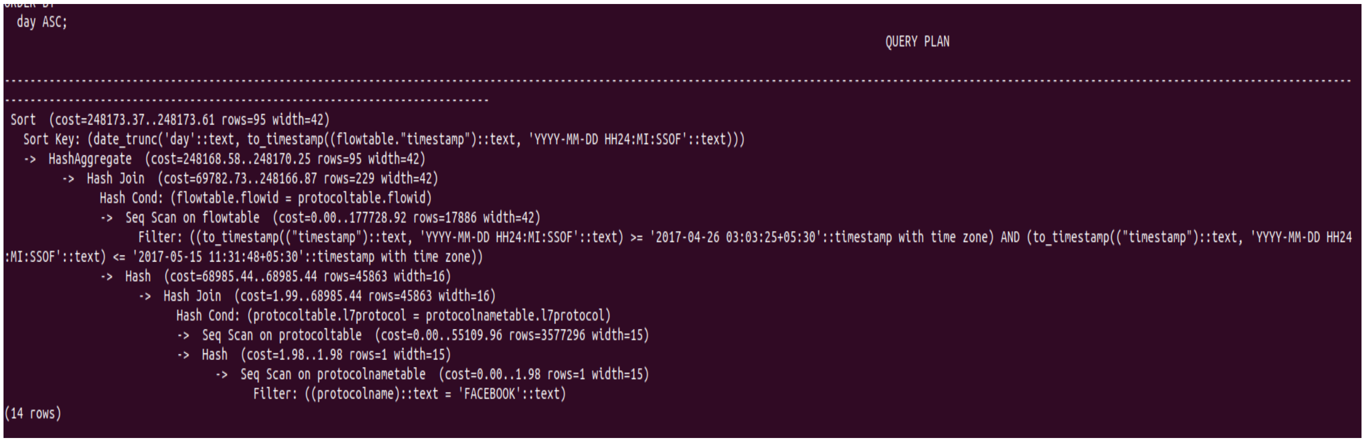
Time without indexing:

```
Time: 6786.600 ms
```

Time after indexing:

```
Time: 3946.018 ms
```

Query Plan



```
  day ASC;
                                                                                                                                         QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------
 Sort  (cost=248173.37..248173.61 rows=95 width=42)
   Sort Key: (date_trunc('day'::text, to_timestamp((flowtable."timestamp")::text, 'YYYY-MM-DD HH24:MI:SSOF'::text)))
   ->  HashAggregate  (cost=248168.58..248170.25 rows=95 width=42)
         ->  Hash Join  (cost=69782.73..248166.87 rows=229 width=42)
               Hash Cond: (flowtable.flowid = protocoltable.flowid)
               ->  Seq Scan on flowtable  (cost=0.00..177728.92 rows=17886 width=42)
                     Filter: ((to_timestamp(("timestamp")::text, 'YYYY-MM-DD HH24:MI:SSOF'::text) >= '2017-04-26 03:03:25+05:30'::timestamp with time zone) AND (to_timestamp(("timestamp")::text, 'YYYY-MM-DD HH24
:MI:SSOF'::text) <= '2017-05-15 11:31:48+05:30'::timestamp with time zone))
               ->  Hash  (cost=68985.44..68985.44 rows=45863 width=16)
                     ->  Hash Join  (cost=1.99..68985.44 rows=45863 width=16)
                           Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
                           ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
                           ->  Hash  (cost=1.98..1.98 rows=1 width=15)
                                 ->  Seq Scan on protocolnametable  (cost=0.00..1.98 rows=1 width=15)
                                       Filter: ((protocolname)::text = 'FACEBOOK'::text)
(14 rows)
```

Query 24 :hourly activeness in a given day

```
  SELECT
    ProtocolNameTable.ProtocolName AS protocol_name,
    date_trunc('hour', to_timestamp(FlowTable.timestamp, 'YYYY-MM-DD HH24:MI:SSOF')) AS hour,
    SUM(FlowTable.flowDuration) AS time_used
  FROM
    FlowTable
    JOIN ProtocolTable ON FlowTable.flowId = ProtocolTable.flowId
    JOIN ProtocolNameTable ON ProtocolTable.L7Protocol = ProtocolNameTable.L7Protocol
  WHERE
    ProtocolNameTable.ProtocolName = 'FACEBOOK'
```

```
  AND to_timestamp(FlowTable.timestamp, 'YYYY-MM-DD HH24:MI:SSOF')::date = '2017-04-26'::date
GROUP BY
  ProtocolNameTable.ProtocolName,
  hour
ORDER BY
  hour ASC;
```

Time without indexing:

```
Time: 3700.112 ms
```

Time after indexing:

```
Time: 1943.932 ms
```

Query Plan

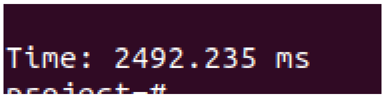
```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=239230.13..239230.37 rows=95 width=42)
   Sort Key: (date_trunc('hour'::text, to_timestamp((flowtable."timestamp")::text, 'YYYY-MM-DD HH24:MI:SSOF'::text)))
   ->  HashAggregate  (cost=239225.35..239227.01 rows=95 width=42)
         ->  Hash Join  (cost=69782.73..239223.63 rows=229 width=42)
               Hash Cond: (flowtable.flowid = protocoltable.flowid)
               ->  Seq Scan on flowtable  (cost=0.00..168785.68 rows=17886 width=42)
                     Filter: ((to_timestamp(("timestamp")::text, 'YYYY-MM-DD HH24:MI:SSOF'::text))::date = '2017-04-26'::date)
               ->  Hash  (cost=68985.44..68985.44 rows=45863 width=16)
                     ->  Hash Join  (cost=1.99..68985.44 rows=45863 width=16)
                           Hash Cond: (protocoltable.l7protocol = protocolnametable.l7protocol)
                           ->  Seq Scan on protocoltable  (cost=0.00..55109.96 rows=3577296 width=15)
                           ->  Hash  (cost=1.98..1.98 rows=1 width=15)
                                 ->  Seq Scan on protocolnametable  (cost=0.00..1.98 rows=1 width=15)
                                       Filter: ((protocolname)::text = 'FACEBOOK'::text)
(14 rows)
```

Query25: Possible ip address reachable from a particular ip address at a given time-stamp

```
WITH RECURSIVE ReachableNodes AS (
  SELECT DISTINCT destinationIp, timestamp
  FROM FlowTable
  WHERE sourceIp = '192.168.180.37'
  AND timestamp = '2017-04-26 11:12:09+05:30'
  UNION
  SELECT DISTINCT FlowTable.destinationIp, FlowTable.timestamp
  FROM FlowTable
```

```
  JOIN ReachableNodes ON ReachableNodes.destinationIp = FlowTable.sourceIp
  AND ReachableNodes.timestamp = FlowTable.timestamp
)
SELECT timestamp, ARRAY_AGG(DISTINCT destinationIp) AS reachable_nodes
FROM ReachableNodes
GROUP BY timestamp
ORDER BY timestamp;
```

Time without indexing:

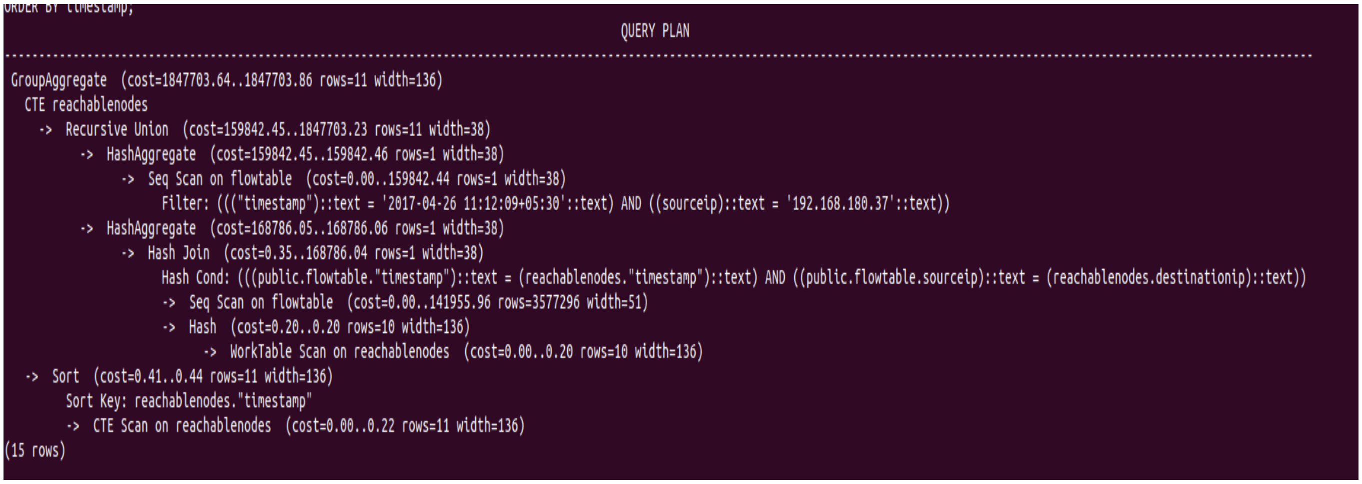


```
Time: 2492.235 ms
```

Time after indexing:

```
Time: 3.163 ms
```

Query Plan



```
ORDER BY timestamp;
                                                                              QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=1847703.64..1847703.86 rows=11 width=136)
   CTE reachablenodes
     ->  Recursive Union  (cost=159842.45..1847703.23 rows=11 width=38)
           ->  HashAggregate  (cost=159842.45..159842.46 rows=1 width=38)
                 ->  Seq Scan on flowtable  (cost=0.00..159842.44 rows=1 width=38)
                       Filter: ((("timestamp")::text = '2017-04-26 11:12:09+05:30'::text) AND ((sourceip)::text = '192.168.180.37'::text))
           ->  HashAggregate  (cost=168786.05..168786.06 rows=1 width=38)
                 ->  Hash Join  (cost=0.35..168786.04 rows=1 width=38)
                       Hash Cond: (((public.flowtable."timestamp")::text = (reachablenodes."timestamp")::text) AND ((public.flowtable.sourceip)::text = (reachablenodes.destinationip)::text))
                       ->  Seq Scan on flowtable  (cost=0.00..141955.96 rows=3577296 width=51)
                       ->  Hash  (cost=0.20..0.20 rows=10 width=136)
                             ->  WorkTable Scan on reachablenodes  (cost=0.00..0.20 rows=10 width=136)
   ->  Sort  (cost=0.41..0.44 rows=11 width=136)
         Sort Key: reachablenodes."timestamp"
         ->  CTE Scan on reachablenodes  (cost=0.00..0.22 rows=11 width=136)
(15 rows)
```

Query26: In a given TimeStamp which source ip has highest traffic

```
  SELECT
  sourceIp,
  SUM(fwdIATTotal + BwdIATTotal) AS totalTraffic
FROM
  FlowTable
```

```
  JOIN IATTable ON FlowTable.flowId = IATTable.flowId
WHERE
  timestamp = '2017-04-26 11:12:09+05:30'
GROUP BY
  sourceIp
ORDER BY
  totalTraffic DESC
LIMIT
  10;
```

Time without indexing:

```
Time: 757.603 ms
```

Time after indexing:

```
Time: 10.285 ms
```

Query Plan

```
  10;
                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Limit  (cost=152529.81..152529.82 rows=1 width=27)
   ->  Sort  (cost=152529.81..152529.82 rows=1 width=27)
         Sort Key: (sum((iattable.fwdiattotal + iattable.bwdiattotal)))
         ->  HashAggregate  (cost=152529.79..152529.80 rows=1 width=27)
               ->  Nested Loop  (cost=0.00..152528.86 rows=185 width=27)
                     ->  Seq Scan on flowtable  (cost=0.00..150899.20 rows=185 width=21)
                           Filter: (("timestamp")::text = '2017-04-26 11:12:09+05:30'::text)
                     ->  Index Scan using iattable_pkey on iattable  (cost=0.00..8.80 rows=1 width=22)
                           Index Cond: (iattable.flowid = flowtable.flowid)
(9 rows)
```

Dropping indicies:

```
DROP INDEX idx_ProtocolTable_flowId;
DROP INDEX idx_ProtocolTable_L7Protocol;
DROP INDEX idx_FlowTable_flowDuration_timestamp;
DROP INDEX idx_ProtocolNameTable_ProtocolName;
DROP INDEX idx_flowId;
```

```
DROP INDEX idx_flowId3;

DROP INDEX idx_L7Protocol1;

DROP INDEX idx_timestamp;

DROP INDEX idx_flowid2;

DROP INDEX idx_ActiveMax;

DROP INDEX idx_ProtocolTable_L7Protocol_flowID;

DROP INDEX idx_IATTable_flowID;

DROP INDEX idx_IATTable_flowIATMax;

DROP INDEX idx_ratio_downupratio;

DROP INDEX idx_flows_flowDuration;
```