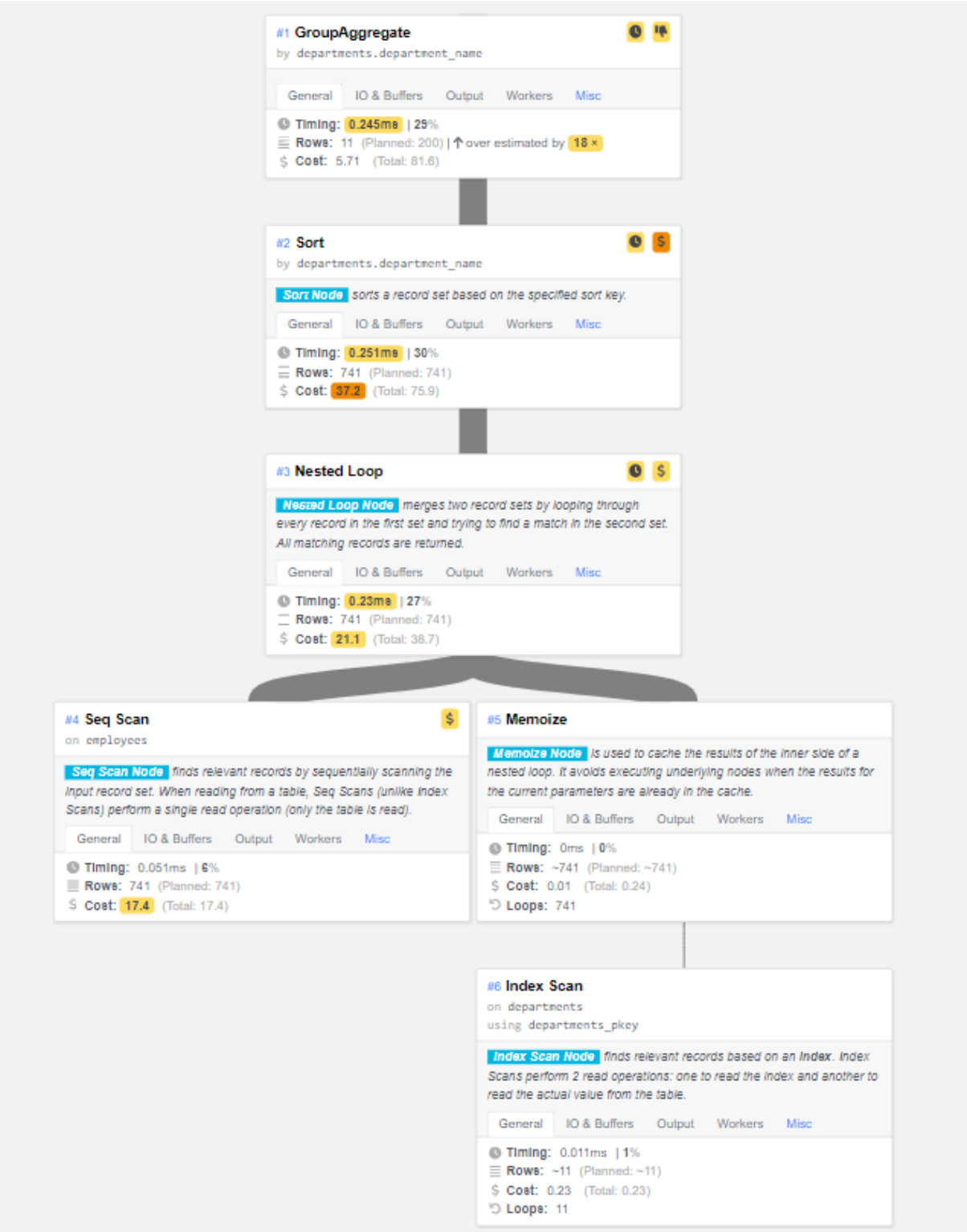
Lab Indexes – Part 2

Question 1 Part a



The above screenshot is the visualisation of this query:

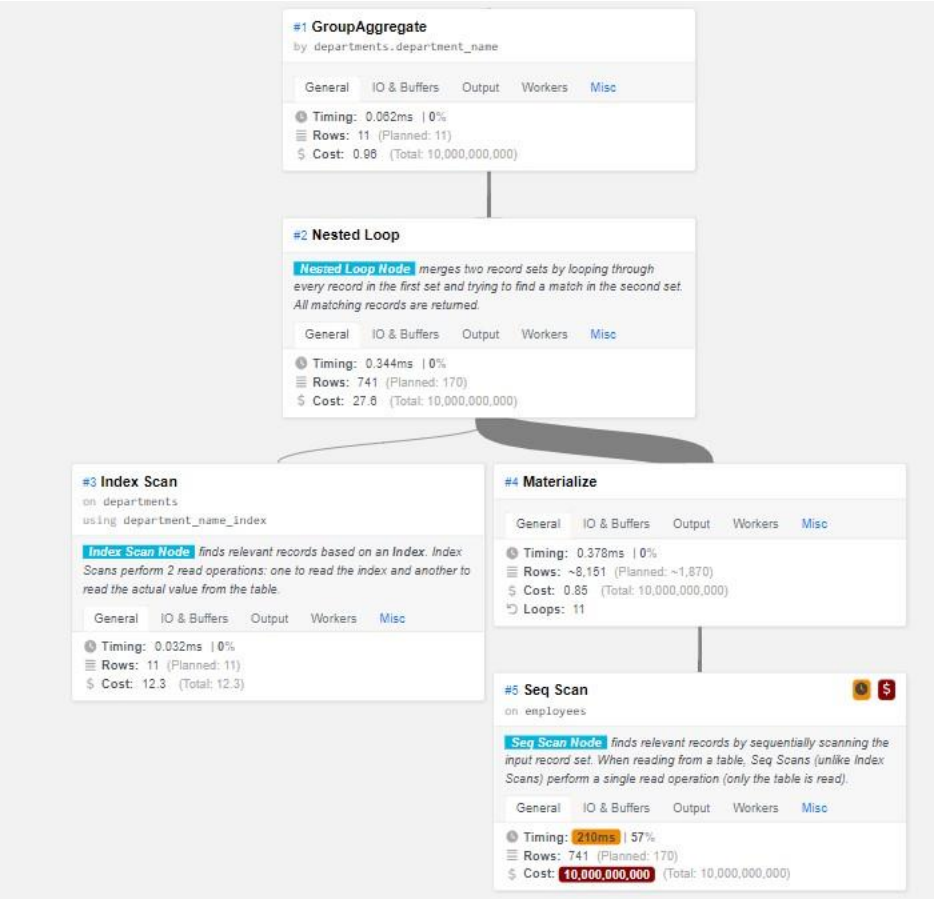
```
EXPLAIN ANALYSE
SELECT COUNT (DISTINCT employee_id), department_name
FROM employees
JOIN departments USING (department_id)
GROUP BY department_name;
```

This is what the query returns without the EXPLAIN ANALYSE

count	department_name
74	Accounting
60	Administration
57	Executive
70	Finance
88	Human Resources
61	IT
65	Marketing
63	Public Relations
80	Purchasing
62	Sales
61	Shipping

This query here counts the number of employee ids that are in each department. The EXPLAIN ANALYSE shows how long the group aggregate, sort, nested loop, seq scan, memoize and index scan will take to run. It also shows the cost of each of them as well as the rows this query went through. In this case, the query went through 741 rows, which is the amount of count numbers added up. The group aggregate only had to sort through 11 rows as it sorted the 11 rows at the end that are displayed in the screenshot. The cost is high on the sort and the time on the group aggregate was the most.

Question 1 Part b



This is the new visualisation of the query once the index was created.

```
CREATE UNIQUE INDEX department_name_index on departments ( department_name );
```

This is the index creation. The timing on the query was slower than the other one without the index. The execution time was 370ms which is slower than the other one. The seq scan at node 5 was the slowest, with a time of 210ms. The seq scan cost was 10,000,000,000. This indexed query performed worse than the above.

Question 2 Part a

This is the query below that is going to be visualised.

```
EXPLAIN ANALYSE
SELECT COUNT (department_id), city
FROM departments
JOIN locations USING (location_id)
GROUP BY city;
```

The result of this query is:

count	city
1	Oxford
1	Southlake
1	London
1	South San Francisco
1	Munich
1	Toronto
5	Seattle

The departments in each city are counted.

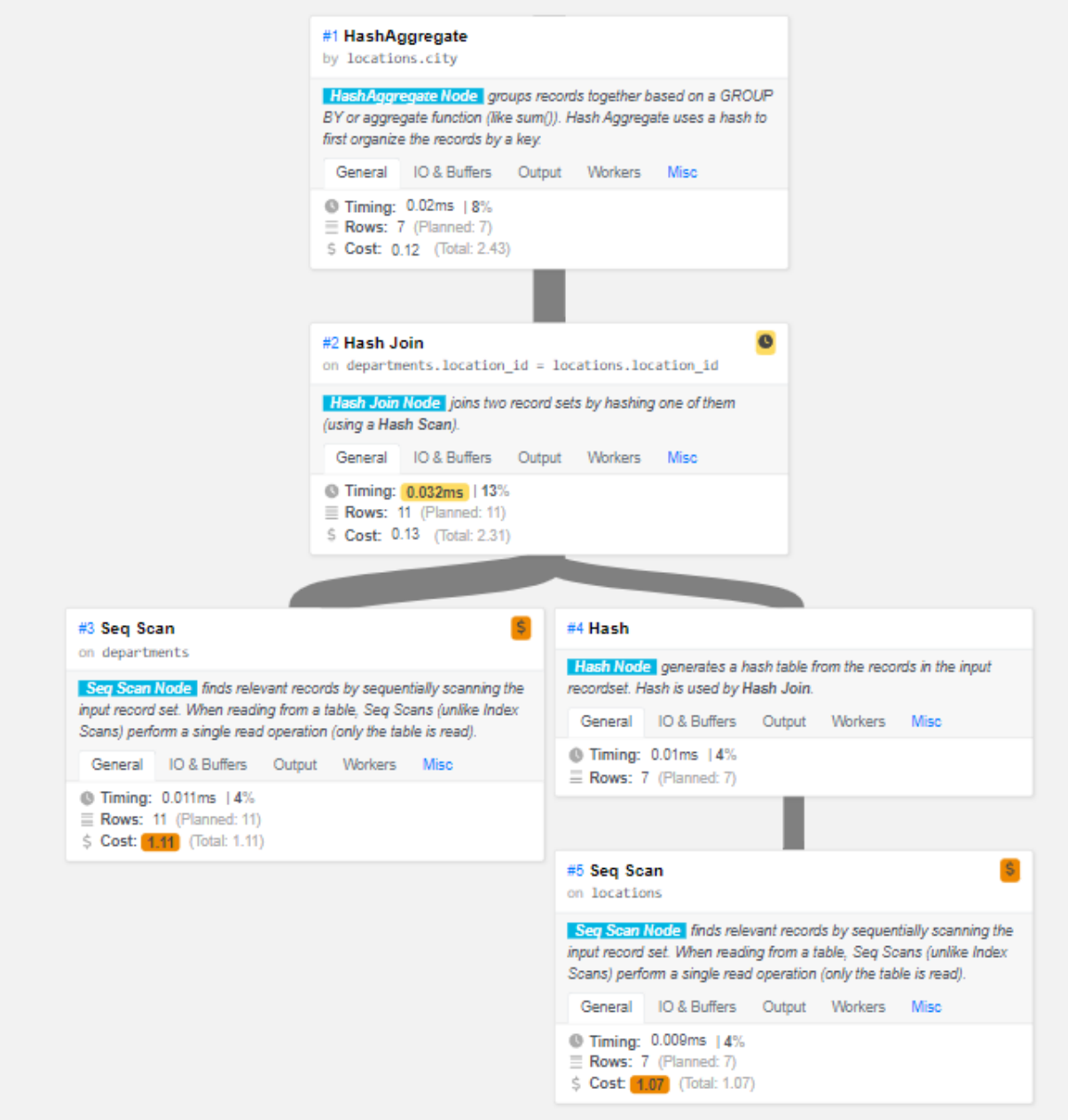
The node that took the longest to execute was the hash join. The node that ran through the most rows was the hash join, the hash and the seq scan. The cost was the highest for the seq scan at 12.4. The sequence of the query was from the hash aggregate to the hash join, then the seq scan followed by the hash and finally the seq scan. Because there were two tables involved, the seq scan was used twice, more time was spent on the 5th node seq scan which was the departments table. The 3rd node seq scan was done on table locations, which has the highest cost.

Question 2 Part b

Below is the index that was created for this query.

```
CREATE UNIQUE INDEX city_name_index on locations ( city );
```

The index was created on city names in the location table. This created the visualised EXPLAIN ANALYSE as below



The highest cost was 1.11 which is substantially lower than the highest cost in part a, which was 12.4. The highest cost is on the seq scan on the departments table, and the other seq scan, on the locations table, has a cost of 1.07. The longest time was 0.014ms on the hash join node, which is higher than the longest time without the index, which was 0.01ms, but overall, the time taken with the index was higher, 0.248ms compared to the query without the index, which was 0.079ms.

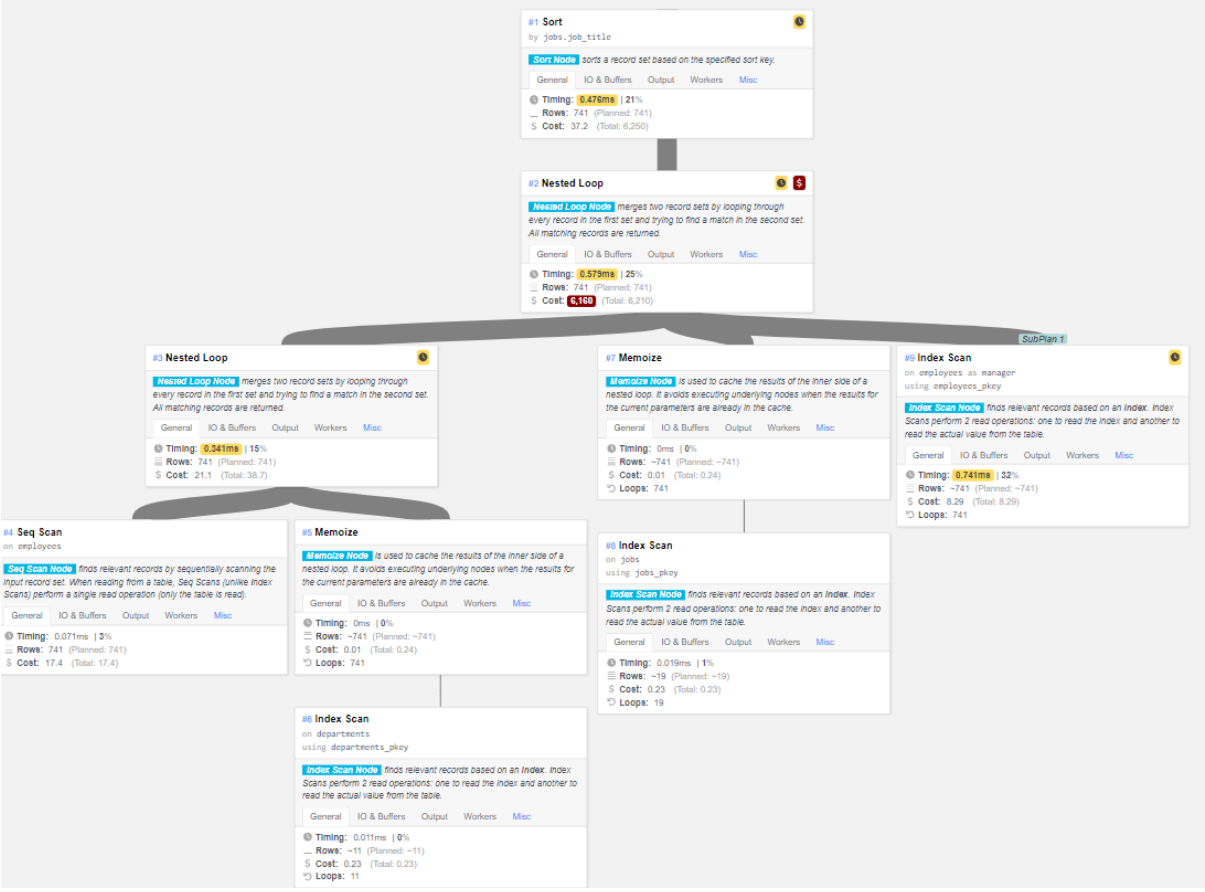
Question 3 Part a

This is the query that is going to be visualised.

```
SELECT employee_id, first_name, last_name, email, phone_number, salary, hire_date,
job_title, department_name,
(SELECT CONCAT(first_name, ' ', last_name) as m_name FROM employees manager WHERE manager.employee_id = employees.manager_id)
FROM departments
JOIN employees USING (department_id)
JOIN jobs USING (job_id)
ORDER BY job_title;
```

This is part of the results of it

employee_id	first_name	last_name	email	phone_number	salary
113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	6900.00
786	firstname786	lastname786	786@mail.com	01 968786	20786.00
844	firstname844	lastname844	844@mail.com	01 968844	20844.00
544	firstname544	lastname544	544@mail.com	01 968544	20544.00
753	firstname753	lastname753	753@mail.com	01 968753	20753.00
537	firstname537	lastname537	537@mail.com	01 968537	20537.00



The execution time for this is 2.32ms. The second node, nested loop, has a cost of 6,160. The most time is on the ninth node, which is 0.741ms. the first node, the sort has a time of 0.476ms, the second node is timed at 0.579ms and the third node is at 0.341ms. The nested loop has taken up all the cost and the time was taken up on the index scan.

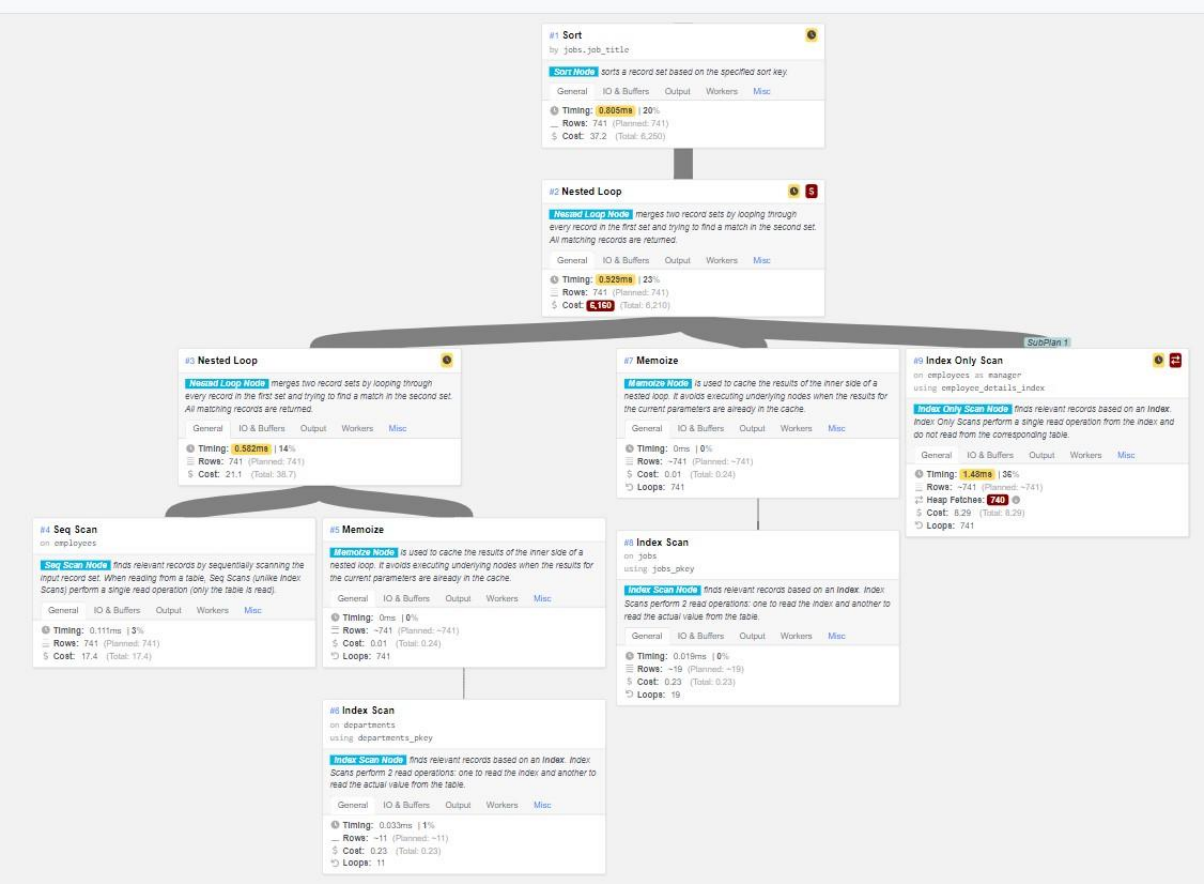
Question 3 Part b

<u>The first index</u>

This is the index created:

```
CREATE INDEX employee_details_index on employees ( employee_id, first_name, last_name );
```

The visualisation that was created from the EXPLAIN ANALYSE was as below



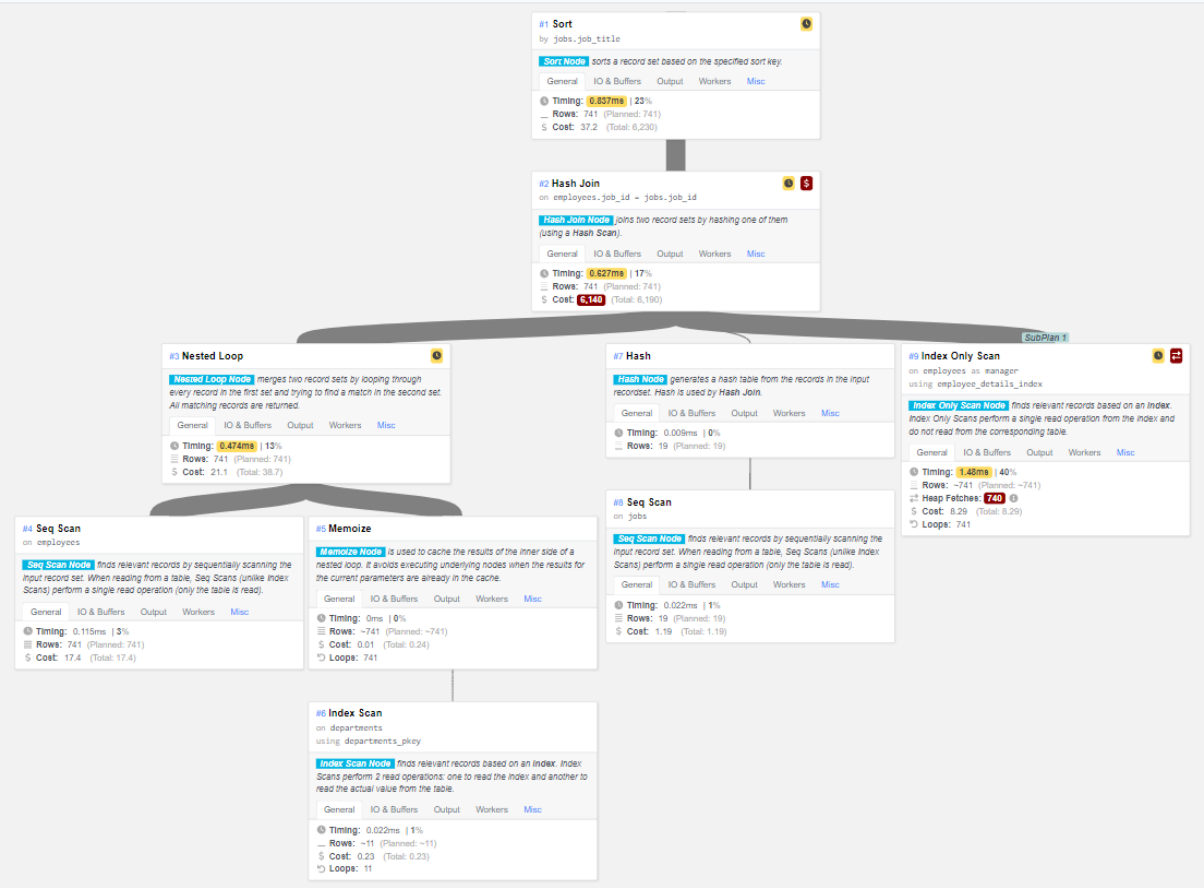
The execution time is 4.09ms, which is more than the execution time of the last part. Here the nested loop has used up the same cost, 6160, and the time on the nested loop at node two is 0.929ms. The last part had a faster time on the nested loop. The first node, the sort is 0.805ms at the first part had at time of 0.476ms, so it is slower. The second node, nested loop, has a time 0.929ms, the first part had a time there of 0.579ms. The third node, the nested loop is at a time of 0.582ms the first part had a time of 0.341ms there. The ninth node is at a time of 1.48ms and it has heap fetches of 740. This is not the case for the first one.

The second index

The index that was created is as below

```
CREATE INDEX job_title_index ON jobs (job_title);
```

And this is the visualised data from this index:

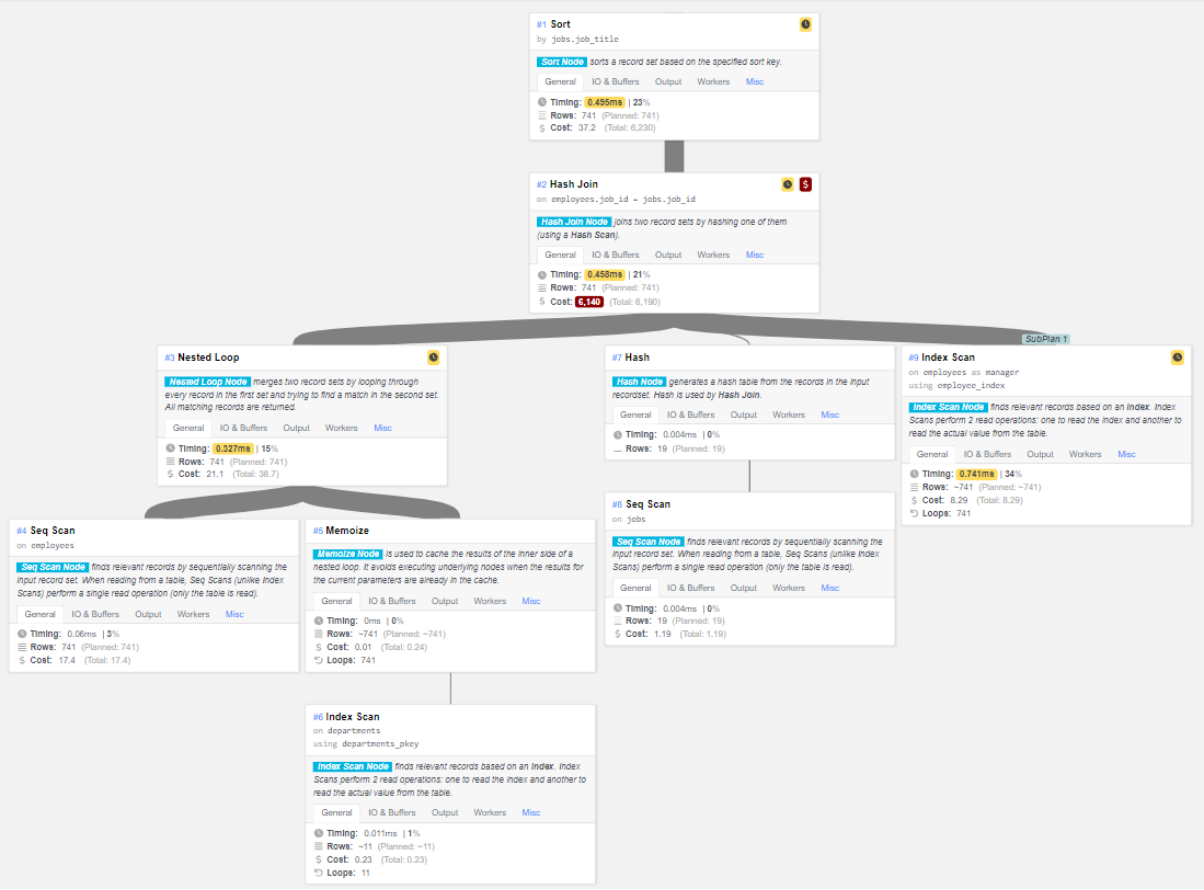


The execution time on this is 3.7ms. this is faster than the last index which was 4.09ms.The first node, the sort here is 0.837ms and the last index this node was 0.805ms. The cost on this for the second node on the hash join is 6,140 and the last index this node had 6,160. The time on last index was slower too, which is 0.627ms here. The third node is faster here too with 0.474ms and the last index this node as 0.582ms. The ninth nodes in this index and the last index are the same, with the same cost and timing.

The third index

```
--Third Index
CREATE INDEX employee_index ON employees ( employee_id, email, phone_number );
```

The visualised query:



The execution time here is 2.18ms. This is faster than the last index which was an execution time of 3.7ms. The time on this sort on this first node is 0.495ms, which is faster than the last index first node. The second node, the hash join has a time 0.458ms which is faster than the last index at this node, the time 0.627ms. The costs here are the same. The third node, the nested loop, is at time 0.327ms, which is faster than other index which had a time of 0.474ms. Here the ninth node is the index scan, the time is 0.741ms. The time on the previous index here was 1.48ms. The index before had 740 heap fetches, which is not here, making it faster.