DB Lab 8

Task 1

Before create index

```
1  explain analyze
2  select id from customer
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on customer (cost=0.00..41.00 rows=1000 width=4) (actual time=0.009..0.170 rows=1000 loops...
2	Planning Time: 1.708 ms
3	Execution Time: 0.194 ms

After creating the index

	QUERY PLAN text
1	Seq Scan on customer (cost=0.00..41.00 rows=1000 width=4) (actual time=0.009..0.103 rows=1000 loops...
2	Planning Time: 0.050 ms
3	Execution Time: 0.127 ms

According to pictures we can sy that queries works faster after index creation.

Task 2

	store_id smallint	city_id smallint	max_profit numeric
1	1	300	243.10
2	2	576	271.08

Explain analyze

	QUERY PLAN text
1	GroupAggregate (cost=855.06..864.06 rows=2 width=36) (actual time=2.865..2.867 rows=2 loops=1)
2	Group Key: inventory.store_id, address.city_id
3	-> Incremental Sort (cost=855.06..864.03 rows=2 width=36) (actual time=2.862..2.864 rows=2 loops=1)
4	Sort Key: inventory.store_id, address.city_id
5	Presorted Key: inventory.store_id
6	Full-sort Groups: 1 Sort Method: quicksort Average Memory: 25kB Peak Memory: 25kB
7	-> Nested Loop (cost=846.15..863.94 rows=2 width=36) (actual time=2.783..2.859 rows=2 loops=1)
8	Join Filter: (store.store_id = inventory.store_id)
9	Rows Removed by Join Filter: 2
10	-> GroupAggregate (cost=845.11..846.50 rows=2 width=34) (actual time=2.757..2.771 rows=2 loops=1)
11	Group Key: inventory.store_id
12	-> Sort (cost=845.11..845.56 rows=182 width=8) (actual time=2.738..2.745 rows=182 loops=1)
13	Sort Key: inventory.store_id
14	Sort Method: quicksort Memory: 34kB
15	-> Nested Loop (cost=329.50..838.28 rows=182 width=8) (actual time=2.020..2.714 rows=182 loops=1)
16	-> Hash Join (cost=329.21..781.86 rows=182 width=10) (actual time=2.013..2.474 rows=182 loops=1)
17	Hash Cond: (rental.rental_id = payment.rental_id)
18	-> Seq Scan on rental (cost=0.00..310.44 rows=16044 width=8) (actual time=0.008..0.662 rows=16044 loops=1)
19	-> Hash (cost=326.94..326.94 rows=182 width=10) (actual time=0.931..0.931 rows=182 loops=1)
20	Buckets: 1024 Batches: 1 Memory Usage: 16kB
21	-> Seq Scan on payment (cost=0.00..326.94 rows=182 width=10) (actual time=0.893..0.912 rows=182 loops=1)
22	Filter: ((payment_date >= '2007-05-01 00:00:00'::timestamp without time zone) AND (payment_date <= '2007-05-31 00:00:00'::timestamp without time zone))
23	Rows Removed by Filter: 14414
24	-> Index Scan using inventory_pkey on inventory (cost=0.28..0.31 rows=1 width=6) (actual time=0.001..0.001 rows=1 loops=182)
25	Index Cond: (inventory_id = rental.inventory_id)
26	-> Materialize (cost=1.04..17.37 rows=2 width=6) (actual time=0.012..0.043 rows=2 loops=2)
27	-> Hash Join (cost=1.04..17.36 rows=2 width=6) (actual time=0.022..0.083 rows=2 loops=1)
28	Hash Cond: (address.address_id = store.address_id)
29	-> Seq Scan on address (cost=0.00..14.03 rows=603 width=6) (actual time=0.009..0.040 rows=603 loops=1)
30	-> Hash (cost=1.02..1.02 rows=2 width=6) (actual time=0.007..0.007 rows=2 loops=1)
31	Buckets: 1024 Batches: 1 Memory Usage: 9kB
32	-> Seq Scan on store (cost=0.00..1.02 rows=2 width=6) (actual time=0.005..0.005 rows=2 loops=1)
33	Planning Time: 0.503 ms
34	Execution Time: 2.924 ms

Joints are working too long, so it is better to create indexes to improve its speed.