INFO20003 Week 8 Lab Solutions

PLEASE NOTE: Your exact query cost may differ from those displayed in these solutions. It will depend on many factors (disk characteristics, database parameters, etc). When optimising queries, it is the *change in the cost*, not the cost itself, that is an indication of query optimisation.

Section 3: Measuring query performance

◆ **Task 3.1** Run the following queries, and for each one, record the number of rows returned, the total cost of the query, and which indexes were used (if any).

	Number of rows returned	Cost of query	Indexes used
Query 1	256	5545.35	None
Query 2	383	5545.35	None
Query 3	10	1742.19	*clients* table: *clients_country_fk* *customers* table: None
Query 4	620	5545.35	None

Section 4: Practical query optimisation using indexes

◆ **Task 4.9** Look back at the first two queries from Task 3.1. Using what you have learned in this section, create one or more indexes that will improve the execution cost and plan of Queries 1 and 2. Then run the queries and fill in the following table.

Queries 1 and 2 both have a WHERE condition on the *cust_state_province* column. This column is a candidate for the creation of an index:

```
CREATE INDEX cust_state_country_idx
ON clients (cust_state_country);
```

This improves the query cost as follows:

	Original cost of query (Task 3.1)	New cost of query	Indexes used
Query 1	5545.35	115.71	*cust_state_country_idx*
Query 2	5545.35	212.10	*cust_state_country_idx*

◆ **Task 4.10** How much does the unique index improve the cost of Query 3 from Task 3.1? Fill in the table:

	Original cost of query (Task 3.1)	**New cost of query**	**Indexes used**
Query 3	1742.19	1212.45	*clients* table: *clients_country_fk* *customers* table: *country_name_idx*

◆ **Task 4.11** Can the performance of Query 4 be improved any further? If so, how? If not, why not?

The query is asking how many rows are present for each (*country_id*, *cust_state_province*, *cust_city*) combination. The obvious way to do this is by reading the entire table, i.e. performing a full table scan.

In theory, you could also accomplish this by reading the entirety of an index over these three columns, i.e. a full index scan. This would be markedly cheaper given the number of columns in the *clients* table. Unfortunately, MySQL 8.0 does not support full index scans for GROUP BY when aggregate functions other than MIN or MAX are used. Since this query uses a COUNT function, we will not be able to optimise it any further.