Extra Task

To test how indexes work in MySQL, I expanded my location table and populated it with many entries. Currently, the table contains over 700,000 rows of data.

	total_rows
▶	774450

Subsequently, I created a clone table and indexed it.

```
CREATE TABLE location_clone AS SELECT * FROM location;
CREATE INDEX idx_country ON location_clone (country);
```

Following this, I performed a simple select query to count the number of rows where 'Argentina' is present as the country.

```
SELECT COUNT(*) AS argentina_count
FROM location
WHERE country = 'Argentina';

SELECT COUNT(*) AS argentina_count
FROM location_clone
WHERE country = 'Argentina';
```

Here, we observe a significant difference between the indexed and non-indexed tables, favouring the indexed one.

Try #1:

22:17:18	SELECT COUNT(*) AS argentina_count FROM location WHERE country = 'Argenti...	1 row(s) returned	0.453 sec / 0.000 sec
22:17:18	SELECT COUNT(*) AS argentina_count FROM location_clone WHERE country = '...	1 row(s) returned	0.063 sec / 0.000 sec

Try #2:

22:18:53	SELECT COUNT(*) AS argentina_count FROM location WHERE country = 'Argenti...	1 row(s) returned	0.312 sec / 0.000 sec
22:18:53	SELECT COUNT(*) AS argentina_count FROM location_clone WHERE country = '...	1 row(s) returned	0.031 sec / 0.000 sec

Try #3:

22:19:43	SELECT COUNT(*) AS argentina_count FROM location WHERE country = 'Argenti...	1 row(s) returned	0.328 sec / 0.000 sec
22:19:44	SELECT COUNT(*) AS argentina_count FROM location_clone WHERE country = '...	1 row(s) returned	0.016 sec / 0.000 sec

As we observe with each execution of this SELECT query, the time decreases consistently for the indexed table, whereas it remains relatively constant for the non-indexed table.

Explanation:
The improved performance of the indexed table is due to the presence of indexes, which allow the database engine to swiftly locate specific rows based on the indexed column(s). When a query filters rows based on indexed columns, the database engine can efficiently narrow down the search space, resulting in faster retrieval times. In contrast, without

indexes, the database engine must scan through the entire table sequentially to find matching rows, regardless of the filtering criteria. This sequential scan becomes increasingly time-consuming as the size of the table grows, leading to relatively consistent query execution times irrespective of the filtering condition.