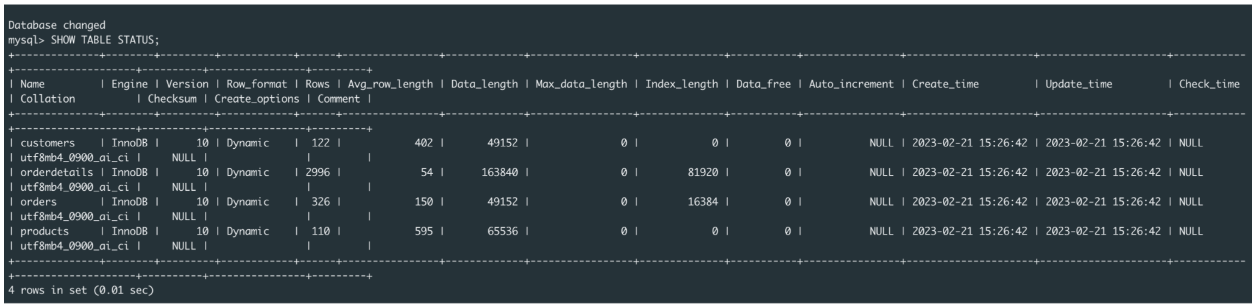
2

a) Initialization



```
Database changed
mysql> SHOW TABLE STATUS;
+-------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+------------
+--------------------+----------+----------------+---------+
| Name        | Engine | Version | Row_format | Rows | Avg_row_length | Data_length | Max_data_length | Index_length | Data_free | Auto_increment | Create_time         | Update_time         | Check_time
| Collation          | Checksum | Create_options | Comment |
+-------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+------------
+--------------------+----------+----------------+---------+
| customers   | InnoDB |      10 | Dynamic    |  122 |            402 |       49152 |               0 |            0 |         0 |           NULL | 2023-02-21 15:26:42 | 2023-02-21 15:26:42 | NULL
| utf8mb4_0900_ai_ci |     NULL |                |         |
| orderdetails | InnoDB |     10 | Dynamic    | 2996 |             54 |      163840 |               0 |        81920 |         0 |           NULL | 2023-02-21 15:26:42 | 2023-02-21 15:26:42 | NULL
| utf8mb4_0900_ai_ci |     NULL |                |         |
| orders      | InnoDB |      10 | Dynamic    |  326 |            150 |       49152 |               0 |        16384 |         0 |           NULL | 2023-02-21 15:26:42 | 2023-02-21 15:26:42 | NULL
| utf8mb4_0900_ai_ci |     NULL |                |         |
| products    | InnoDB |      10 | Dynamic    |  110 |            595 |       65536 |               0 |            0 |         0 |           NULL | 2023-02-21 15:26:42 | 2023-02-21 15:26:42 | NULL
| utf8mb4_0900_ai_ci |     NULL |                |         |
+-------------+--------+---------+------------+------+----------------+-------------+-----------------+--------------+-----------+----------------+---------------------+---------------------+------------
+--------------------+----------+----------------+---------+
4 rows in set (0.01 sec)
```

b) Deliverables:

- Script file: assignment1_load.py
- Data file: assignment1_load.sql

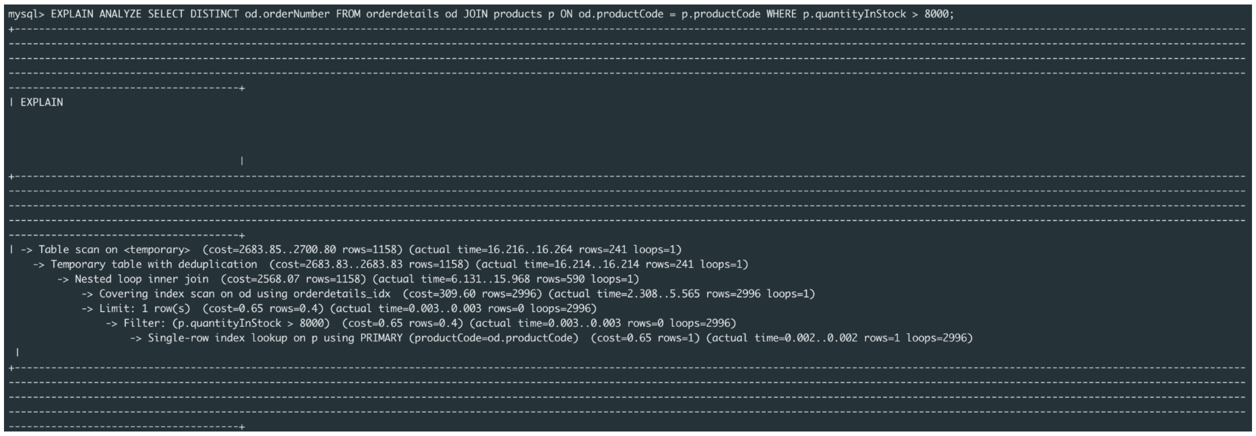
3) Deliverables:

- assignment1_queries.sql

4)

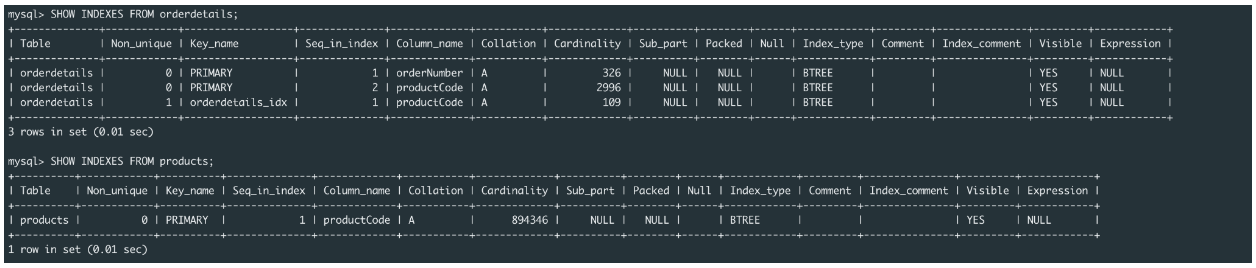
Output of:

```sql
SELECT DISTINCT od.orderNumber
FROM orderdetails od
JOIN products p ON od.productCode = p.productCode
WHERE p.quantityInStock > 8000;
```



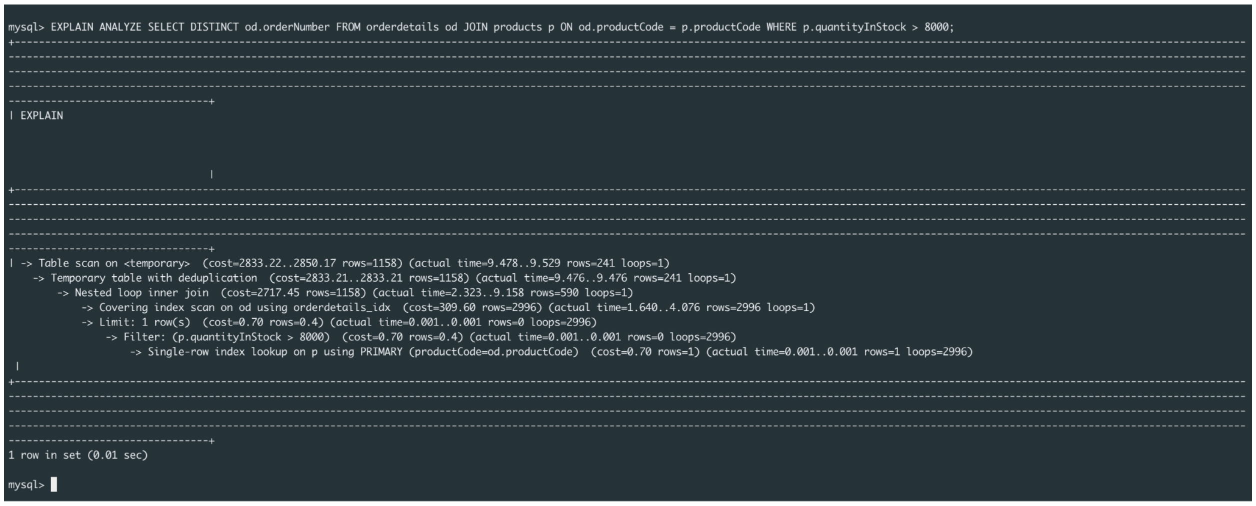
```
mysql> EXPLAIN ANALYZE SELECT DISTINCT od.orderNumber FROM orderdetails od JOIN products p ON od.productCode = p.productCode WHERE p.quantityInStock > 8000;
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
| EXPLAIN



                                     |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
| -> Table scan on <temporary>  (cost=2683.85..2700.80 rows=1158) (actual time=16.216..16.264 rows=241 loops=1)
    -> Temporary table with deduplication  (cost=2683.83..2683.83 rows=1158) (actual time=16.214..16.214 rows=241 loops=1)
        -> Nested loop inner join  (cost=2568.07 rows=1158) (actual time=6.131..15.968 rows=590 loops=1)
            -> Covering index scan on od using orderdetails_idx  (cost=309.60 rows=2996) (actual time=2.308..5.565 rows=2996 loops=1)
            -> Limit: 1 row(s)  (cost=0.65 rows=0.4) (actual time=0.003..0.003 rows=0 loops=2996)
                -> Filter: (p.quantityInStock > 8000)  (cost=0.65 rows=0.4) (actual time=0.003..0.003 rows=0 loops=2996)
                    -> Single-row index lookup on p using PRIMARY (productCode=od.productCode)  (cost=0.65 rows=1) (actual time=0.002..0.002 rows=1 loops=2996)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
```

We can see that the productCode column is already indexed on both the products and orderdetails tables from the figure below. However, MySQL has decided to do a complete table scan instead of using the index.



```
mysql> SHOW INDEXES FROM orderdetails;
+--------------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table        | Non_unique | Key_name          | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+--------------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| orderdetails |          0 | PRIMARY           |            1 | orderNumber | A         |         326 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| orderdetails |          0 | PRIMARY           |            2 | productCode | A         |        2996 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
| orderdetails |          1 | orderdetails_idx  |            1 | productCode | A         |         109 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+--------------+------------+-------------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
3 rows in set (0.01 sec)

mysql> SHOW INDEXES FROM products;
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| Table    | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
| products |          0 | PRIMARY  |            1 | productCode | A         |      894346 |     NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+------------+
1 row in set (0.01 sec)
```

I tried creating an index on the quantityInStock column to see if the execution could be improved, but the query plan remained the same. It appears that the query is already optimized for the current size of the orderdetails table.



```
mysql> EXPLAIN ANALYZE SELECT DISTINCT od.orderNumber FROM orderdetails od JOIN products p ON od.productCode = p.productCode WHERE p.quantityInStock > 8000;
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------+
| EXPLAIN



                                 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------+
| -> Table scan on <temporary>  (cost=2833.22..2850.17 rows=1158) (actual time=9.478..9.529 rows=241 loops=1)
    -> Temporary table with deduplication  (cost=2833.21..2833.21 rows=1158) (actual time=9.476..9.476 rows=241 loops=1)
        -> Nested loop inner join  (cost=2717.45 rows=1158) (actual time=2.323..9.158 rows=590 loops=1)
            -> Covering index scan on od using orderdetails_idx  (cost=309.60 rows=2996) (actual time=1.640..4.076 rows=2996 loops=1)
            -> Limit: 1 row(s)  (cost=0.70 rows=0.4) (actual time=0.001..0.001 rows=0 loops=2996)
                -> Filter: (p.quantityInStock > 8000)  (cost=0.70 rows=0.4) (actual time=0.001..0.001 rows=0 loops=2996)
                    -> Single-row index lookup on p using PRIMARY (productCode=od.productCode)  (cost=0.70 rows=1) (actual time=0.001..0.001 rows=1 loops=2996)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------+
1 row in set (0.01 sec)

mysql> █
```

<u>Based on the Slack communication:</u>
In order to see impact on performance, I tried the following query given by the professor: "**Return the product Name and Vendor of products with at least 9,500 items in stock**"

Commands for improving performance:

- ALTER TABLE products ADD INDEX idx_stock_quantity (quantityInStock);

MySQL plans before and after

- Initial query plan is performing a table scan to retrieve all the rows from the products table, and then filtering the rows based on the quantityInStock condition.
- Final query plan, on the other hand, is using the newly created index to retrieve only the rows that match the quantityInStock condition.

Execution time before and after

Before:

- The actual time value for the table scan operation is 554.405 ms

After:

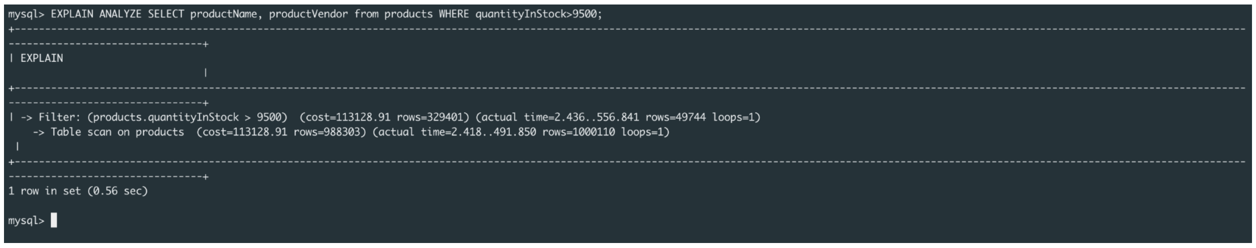
- The actual time value after is 469.828 ms to execute.

Results discussion

- Before adding the index, the query performed a full table scan, which is a very expensive operation for large tables. After adding the index, the query plan shows that an index range scan was used to find the matching rows.

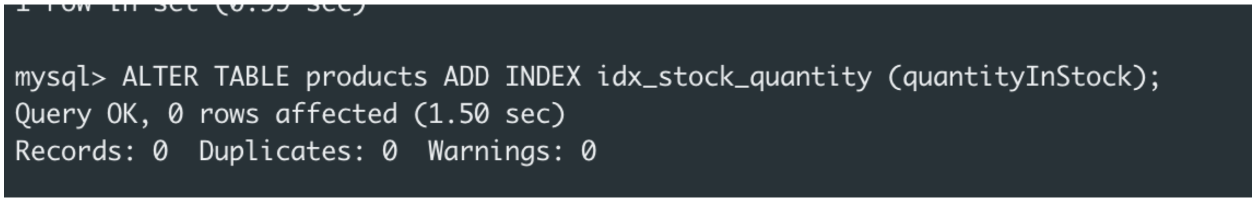
  The index allowed the database engine to locate the relevant rows much more quickly, without having to scan the entire table.

Before optimizing the code:



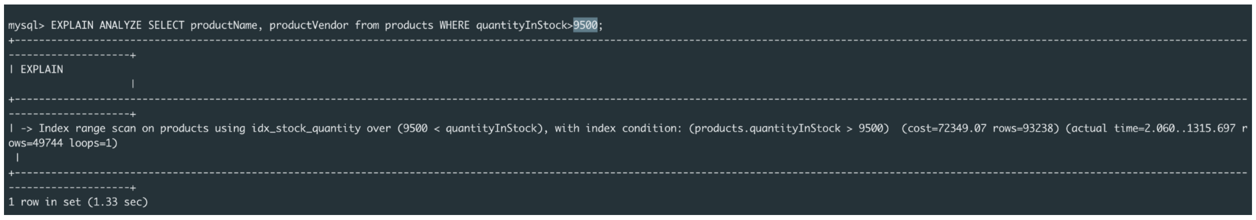
```
mysql> EXPLAIN ANALYZE SELECT productName, productVendor from products WHERE quantityInStock>9500;
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------+
| EXPLAIN
                                |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------+
| -> Filter: (products.quantityInStock > 9500)  (cost=113128.91 rows=329401) (actual time=2.436..556.841 rows=49744 loops=1)
    -> Table scan on products  (cost=113128.91 rows=988303) (actual time=2.418..491.850 rows=1000110 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------+
1 row in set (0.56 sec)

mysql> █
```

Added quantity in stock as an index



```
mysql> ALTER TABLE products ADD INDEX idx_stock_quantity (quantityInStock);
Query OK, 0 rows affected (1.50 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Result after optimizing:



```
mysql> EXPLAIN ANALYZE SELECT productName, productVendor from products WHERE quantityInStock>9500;
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------+
| EXPLAIN
                    |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------+
| -> Index range scan on products using idx_stock_quantity over (9500 < quantityInStock), with index condition: (products.quantityInStock > 9500)  (cost=72349.07 rows=93238) (actual time=2.060..1315.697 r
ows=49744 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------+
1 row in set (1.33 sec)
```