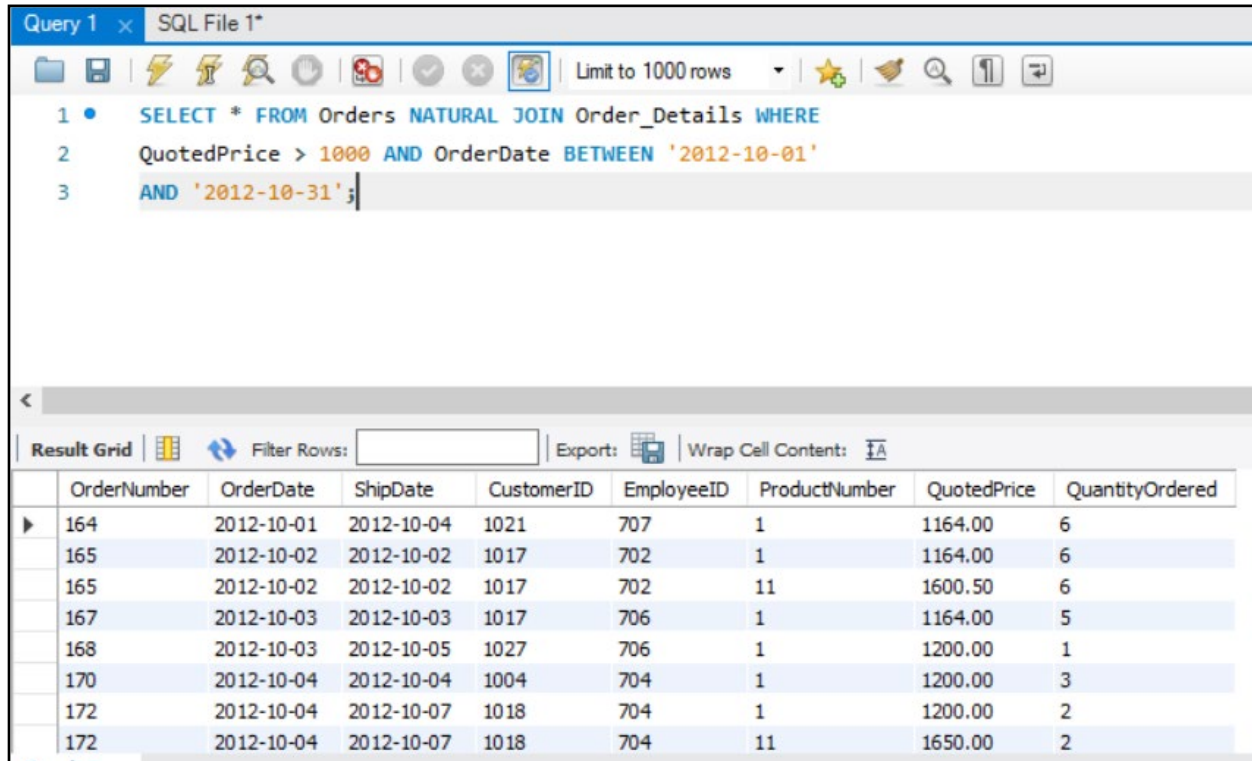


Task C10.2.1



The screenshot shows a SQL IDE window titled 'Query 1' and 'SQL File 1*'. The query editor contains the following SQL statement:

```
1 • SELECT * FROM Orders NATURAL JOIN Order_Details WHERE
2 QuotedPrice > 1000 AND OrderDate BETWEEN '2012-10-01'
3 AND '2012-10-31';
```

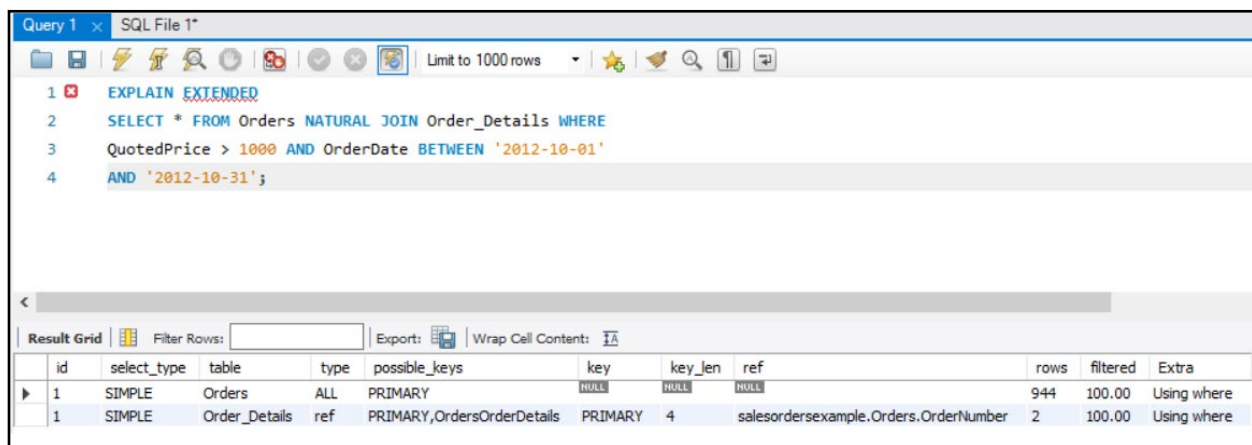
Below the query editor is the 'Result Grid' tab. It shows a table with 9 columns: OrderNumber, OrderDate, ShipDate, CustomerID, EmployeeID, ProductNumber, QuotedPrice, and QuantityOrdered. The table contains 8 rows of data.

	OrderNumber	OrderDate	ShipDate	CustomerID	EmployeeID	ProductNumber	QuotedPrice	QuantityOrdered
▶	164	2012-10-01	2012-10-04	1021	707	1	1164.00	6
	165	2012-10-02	2012-10-02	1017	702	1	1164.00	6
	165	2012-10-02	2012-10-02	1017	702	11	1600.50	6
	167	2012-10-03	2012-10-03	1017	706	1	1164.00	5
	168	2012-10-03	2012-10-05	1027	706	1	1200.00	1
	170	2012-10-04	2012-10-04	1004	704	1	1200.00	3
	172	2012-10-04	2012-10-07	1018	704	1	1200.00	2
	172	2012-10-04	2012-10-07	1018	704	11	1650.00	2

Figure 1 SELECT query without EXPLAIN EXTENDED

Running SELECT query without prefixing EXPLAIN EXPANDED: Yes this is what I was expecting from given query as in this query we are first doing NATURE JOIN which joins tables with same column names in our case that column is OrderNumber and that's why it has joined rows in both tables with having same values for OrderNumber field.

Other Conditions are also fulfilled in this output as for each row quoted price is more than 1000 and order date is between 2012-10-01 and 2012-10-31.



The screenshot shows the same SQL IDE window, but the query editor now includes the `EXPLAIN EXTENDED` command at the beginning of the query:

```
1 [x] EXPLAIN EXTENDED
2 SELECT * FROM Orders NATURAL JOIN Order_Details WHERE
3 QuotedPrice > 1000 AND OrderDate BETWEEN '2012-10-01'
4 AND '2012-10-31';
```

The 'Result Grid' tab now displays the output of the `EXPLAIN` command. It shows a table with 12 columns: id, select_type, table, type, possible_keys, key, key_len, ref, rows, filtered, and Extra. The table contains 2 rows of data.

	id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	Orders	ALL	PRIMARY	<u>HULL</u>	<u>HULL</u>	<u>HULL</u>	944	100.00	Using where
	1	SIMPLE	Order_Details	ref	PRIMARY,OrdersOrderDetails	PRIMARY	4	salesordersexample.Orders.OrderNumber	2	100.00	Using where

Figure 2 with Explain Extended

Running query with EXPLAIN EXPANDED prefix: from above output it is clearly seen that database engine is going through every row in orders table first and then matching it with index in Order_Details table in this case index is Primary key in both tables. Select_type is UNION.

Length of key used to join table is 4 which happens to be a primary key.

Here database is examining 100 percent rows in both table according to WHERE clause which is computationally very expensive.

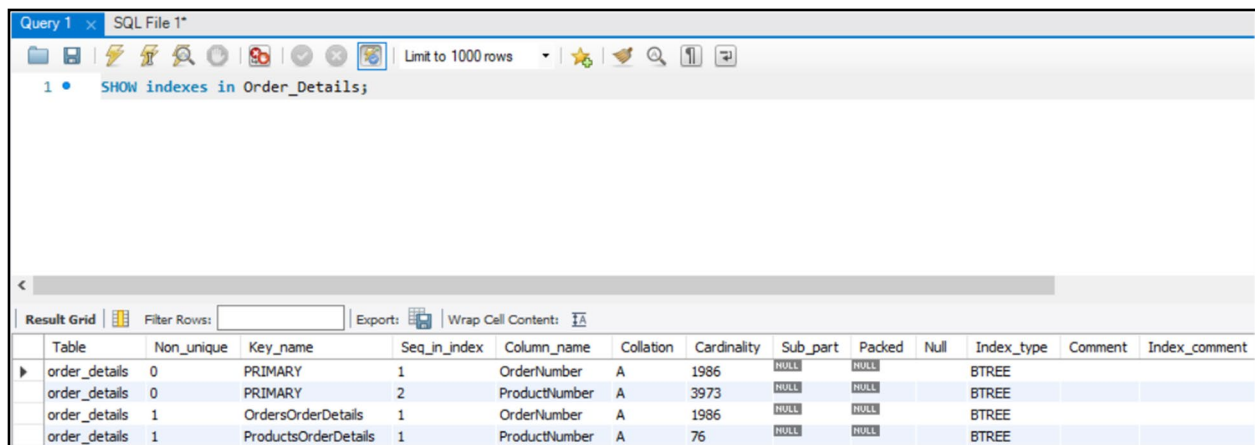


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment
order_details	0	PRIMARY	1	OrderNumber	A	1986	NULL	NULL		BTREE		
order_details	0	PRIMARY	2	ProductNumber	A	3973	NULL	NULL		BTREE		
order_details	1	OrdersOrderDetails	1	OrderNumber	A	1986	NULL	NULL		BTREE		
order_details	1	ProductsOrderDetails	1	ProductNumber	A	76	NULL	NULL		BTREE		

Figure 3 SHOW indexes

Currently there are four indexes associated with order_Details table with two columns relative to Orders and Product tables.