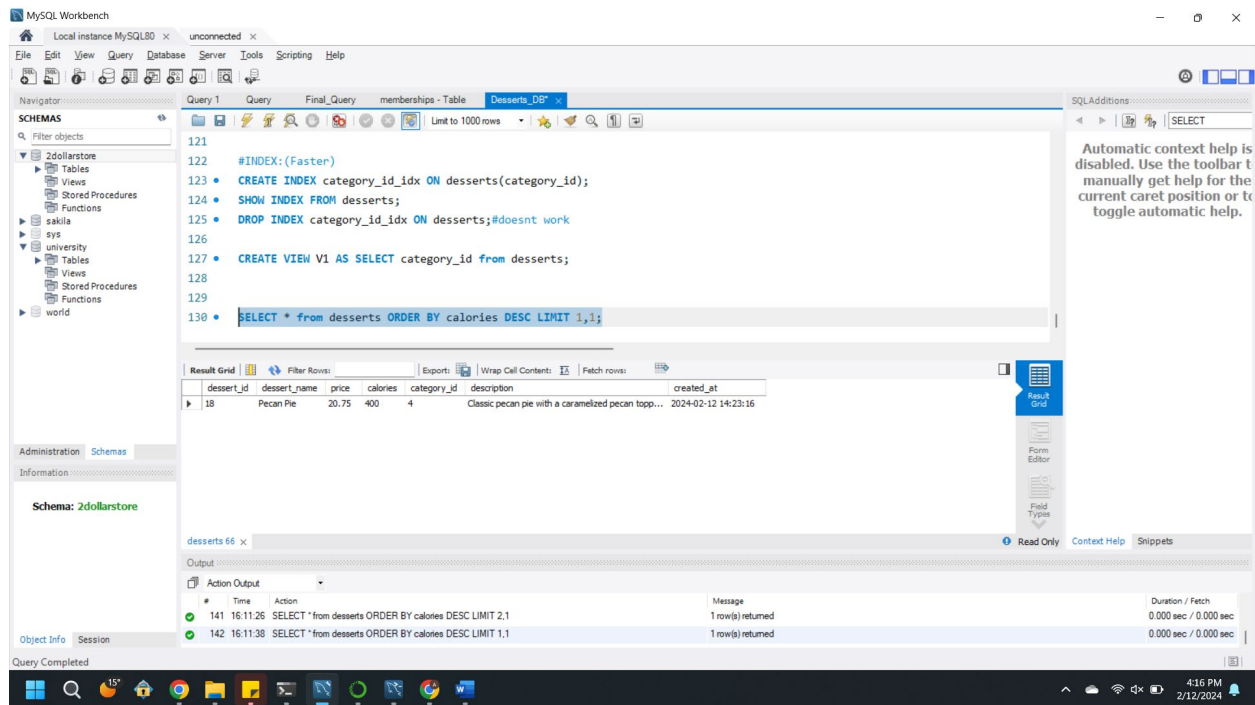


Homework2_AparnaBharathiSuresh

Question 1:

Display the 2nd highest calorific dessert.

Screenshot 1:



Query 1:

SELECT * from desserts ORDER BY calories DESC LIMIT 1,1;

Question 2:

Find the desserts that have more than one word in their name. Display the dessert name.

Screenshot 2:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with '2dollarstore' selected. The main query editor contains the following SQL code:

```
124 • CREATE INDEX category_id_idx ON desserts(category_id);
125 • SHOW INDEX FROM desserts;
126 • DROP INDEX category_id_idx ON desserts;#doesn't work
127
128 • CREATE VIEW V1 AS SELECT category_id FROM desserts;
129
130
131 • SELECT * FROM desserts ORDER BY calories DESC LIMIT 1,1;
132
133 • SELECT dessert_name FROM desserts WHERE CHAR_LENGTH(dessert_name) - CHAR_LENGTH(REPLACE(dessert_name, ' ', '')) > 0;
```

The 'Result Grid' shows the results of the last query, displaying a list of dessert names:

dessert_name
Chocolate Cake
Vanilla Ice Cream
Oatmeal Cookies
Apple Pie
Almond Croissant
Fruit Tart
Peanut Butter Cookies
Strawberry Shortcake
Lemon Sorbet
Red Velvet Cake

The 'Output' pane at the bottom shows the execution of the query, indicating that 19 rows were returned.

Query 2:

```
SELECT dessert_name FROM desserts WHERE CHAR_LENGTH(dessert_name) - CHAR_LENGTH(REPLACE(dessert_name, ' ', '')) > 0;
```

Question 3:

Find the reviews that were given in the month of November 2023.

Screenshot 3:

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
128 CREATE VIEW V1 AS SELECT category_id FROM desserts;
129
130
131 SELECT * FROM desserts ORDER BY calories DESC LIMIT 1,1;
132
133 SELECT dessert_name FROM desserts WHERE CHAR_LENGTH(dessert_name) - CHAR_LENGTH(REPLACE(dessert_name, ' ', ''))>0;
134
135 SELECT * FROM reviews;
136
137 SELECT * FROM reviews WHERE review_date LIKE '2023-11-____';
```

The result grid displays the following data:

review_id	dessert_id	customer_name	rating	comment	review_date
7	3	Bob Johnson	5	Absolutely fantastic!	2023-11-05
8	5	Eva White	4	Tasty pie, loved the blueberries.	2023-11-10
9	1	Mary Davis	4	Moist and rich, loved it!	2023-11-15
10	4	John Doe	3	Could use more pecans.	2023-11-20
11	6	Charlie Brown	0	Great croissant, but a bit pricey.	2023-11-25

The output pane shows the execution of the query, indicating that 5 rows were returned.

Query 3:

```
SELECT * FROM reviews WHERE review_date LIKE '2023-11-____';
```

Question 4:

Calculate the total price for each category, considering only desserts with a price. Display the category name and total price (use Total_Price as the column alias) and order the results by total price in descending order.

Screenshot 4:

The screenshot shows the MySQL Workbench interface. The 'Query Editor' window displays a SQL query (Query 1) that calculates the total price for each category. The query is as follows:

```
134 SELECT * FROM Categories;
135
136
137 SELECT * FROM desserts WHERE (description LIKE '%lemon%' OR description LIKE '%frosting%' OR description LIKE '%ice cream%') AND price>20;
138
139 SELECT dessert_id, customer_name, comment FROM Reviews WHERE comment LIKE '%delicious%' OR comment LIKE '%fantastic%';
140
141 Select c.category_name AS Category_Name, SUM(d.price) AS Total_Price FROM Categories c JOIN desserts d ON c.category_id= d.category_id WHERE d.
142
143 Select c.category_name, SUM(d.price) AS Total_Price FROM Categories c, desserts d where c.category_id=d.category_id AND d.price IS NOT NULL GR
144
145 SELECT * FROM reviews WHERE review_date LIKE '2023-11-__';
```

The 'Result Grid' window shows the results of the query, displaying the category name and total price for each category:

category_name	Total_Price
Cakes	130.24
Pies	86.25
Cookies	72.25
Ice Cream	52.75
Pastries	15.00

The 'Output' window shows the execution of the query, indicating that 5 rows were returned.

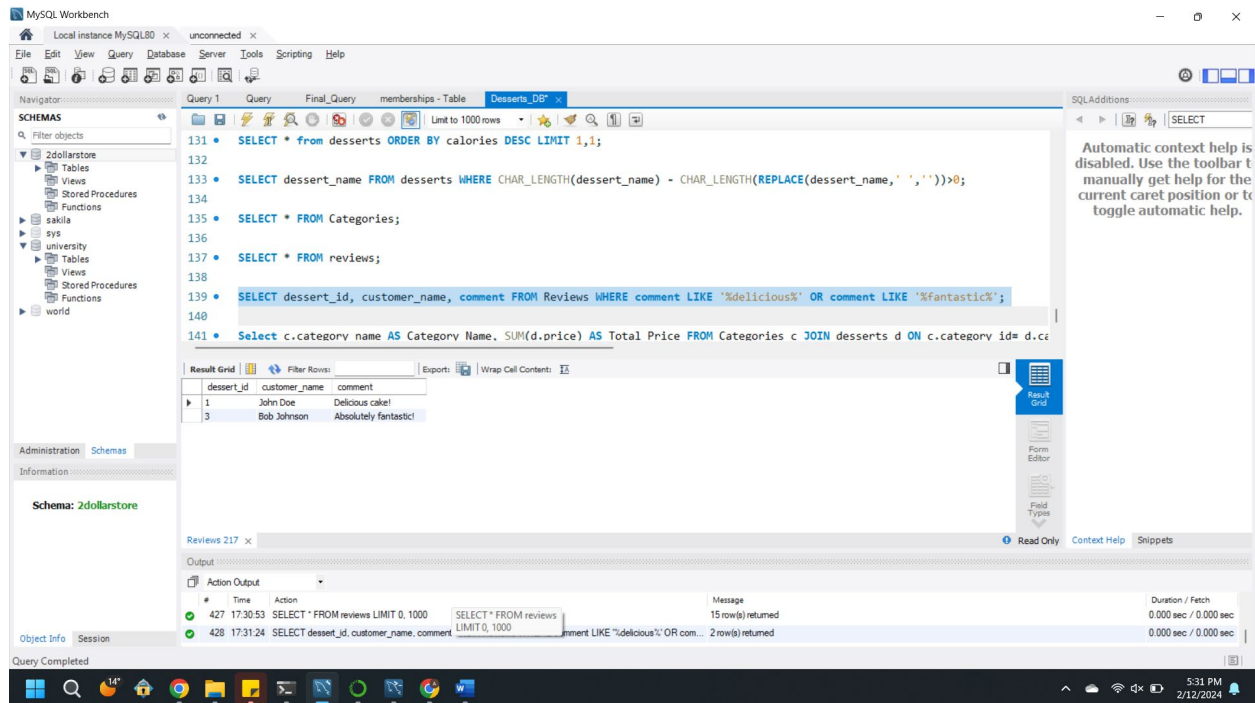
Query 4:

Select c.category_name, SUM(d.price) AS Total_Price FROM Categories c, desserts d where c.category_id=d.category_id AND d.price IS NOT NULL GROUP BY c.category_id ORDER BY Total_Price DESC;

Question 5:

Find the desserts that have reviews with comments containing the word 'delicious' or 'fantastic'. Display the dessert id, customer name, and comment.

Screenshot 5:



Query 5:

```
SELECT dessert_id, customer_name, comment FROM Reviews WHERE comment LIKE '%delicious%' OR comment LIKE '%fantastic%';
```

Question 6:

Display the desserts where the description has one of the words 'Lemon', 'frosting' or 'ice cream' and the price is more than \$20.

Screenshot 6:

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying a SQL query. The 'Result Grid' shows the results of the query, which is a SELECT statement filtering desserts by price and description. The 'Output' pane at the bottom shows the execution log.

```
Query 1
126 • category_id_idx ON desserts;#doesn't work
127
128 • AS SELECT category_id FROM desserts;
129
130
131 • desserts ORDER BY calories DESC LIMIT 1,1;
132
133 • :_name FROM desserts WHERE CHAR_LENGTH(dessert_name) - CHAR_LENGTH(REPLACE(dessert_name, ' ', ''))>0;
134
135 • Categories;
136
137 • desserts WHERE (description LIKE '%Lemon%' OR description LIKE '%frosting%' OR description LIKE '%ice cream%') AND price>20;
138
```

dessert_id	dessert_name	price	calories	category_id	description	created_at
1	Chocolate Cake	25.99	350	1	Rich and moist chocolate cake with frosting.	2024-02-12 17:27:47
11	Red Velvet Cake	27.50	380	1	Velvety red cake with cream cheese frosting.	2024-02-12 17:27:47

Schema: 2dollarstore

Output

#	Time	Action	Message	Duration / Fetch
433	16:05:32	SELECT * FROM desserts WHERE description LIKE '%Lemon%' OR description LIKE '%frosting%' OR description LIKE '%ice cream%' AND price>20;	4 row(s) returned	0.000 sec / 0.000 sec
434	16:08:38	SELECT * FROM desserts WHERE (description LIKE '%Lemon%' OR description LIKE '%frosting%' OR description LIKE '%ice cream%') AND price>20;	2 row(s) returned	0.000 sec / 0.000 sec

Query Completed

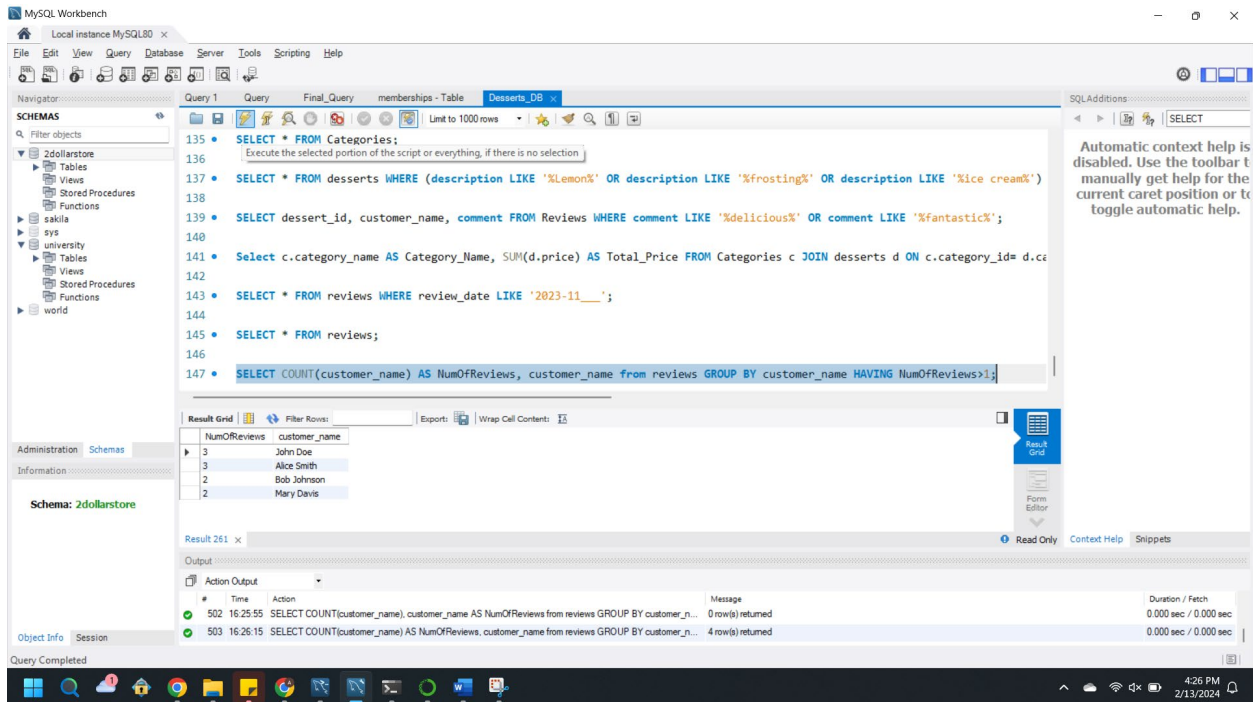
Query 6:

```
SELECT * FROM desserts WHERE (description LIKE '%Lemon%' OR description LIKE '%frosting%' OR description LIKE '%ice cream%') AND price>20;
```

Question 7:

Display the number of reviews given by each customer, if the number of reviews is more than 1.

Screenshot 7:



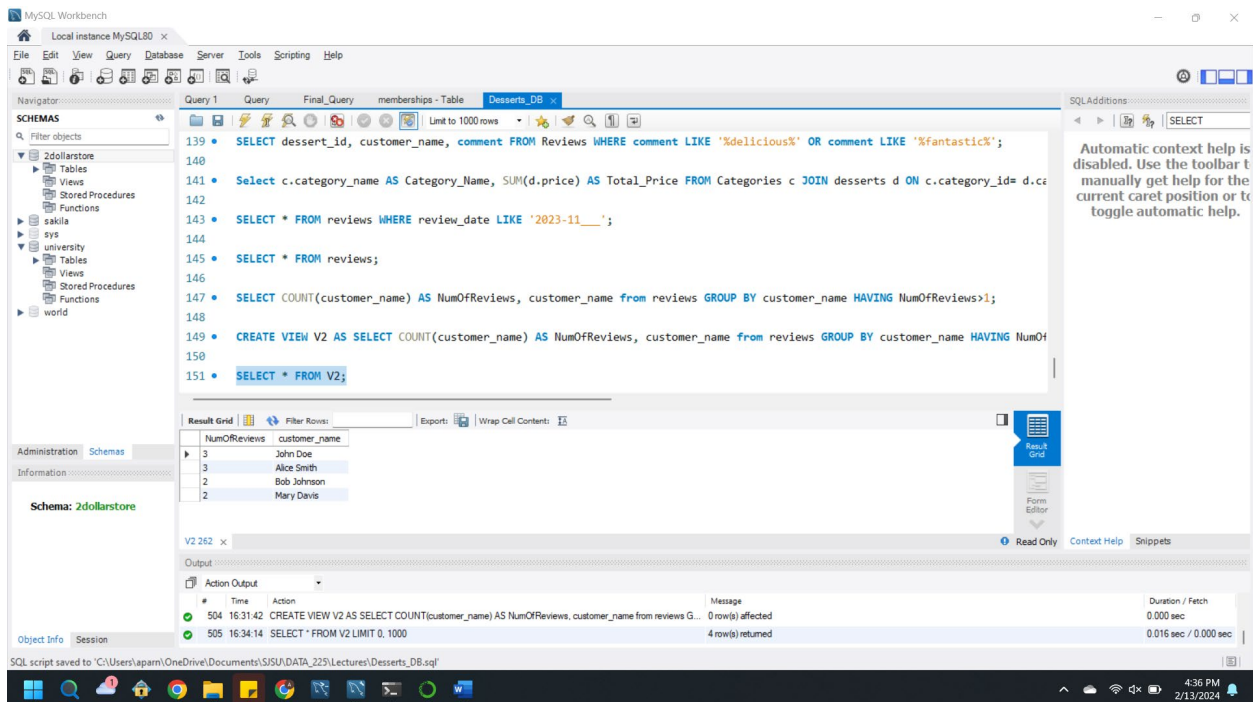
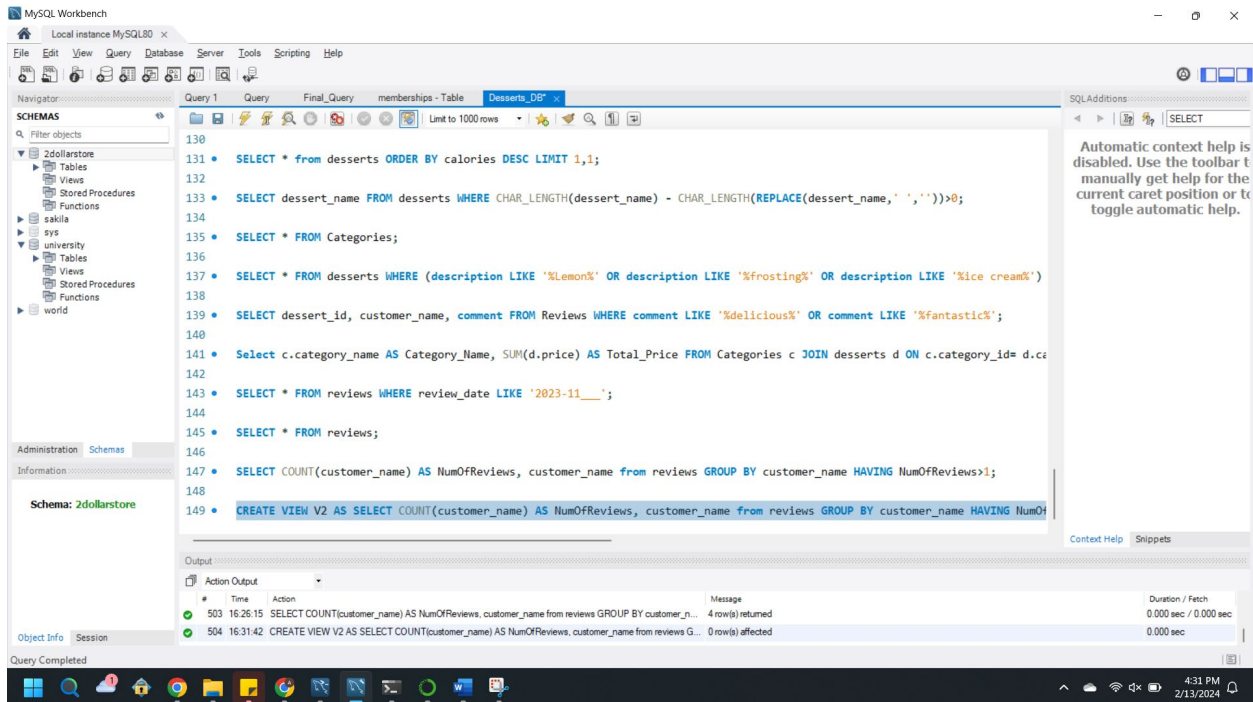
Query 7:

```
SELECT COUNT(customer_name) AS NumOfReviews, customer_name from reviews GROUP BY customer_name HAVING NumOfReviews>1;
```


Question 8:

Create a view for question 7

Screenshot 8:



Query 8:

```
CREATE VIEW V2 AS SELECT COUNT(customer_name) AS NumOfReviews, customer_name from reviews  
GROUP BY customer_name HAVING NumOfReviews>1;
```

```
SELECT * FROM V2;
```

Question 9:

Set the price of all cookies to \$13.50.

Screenshot 9:

Before Update

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
142  
143 SELECT * FROM reviews WHERE review_date LIKE '2023-11-__';  
144  
145 SELECT * FROM reviews;  
146  
147 SELECT COUNT(customer_name) AS NumOfReviews, customer_name from reviews GROUP BY customer_name HAVING NumOfReviews>1;  
148  
149 CREATE VIEW V2 AS SELECT COUNT(customer_name) AS NumOfReviews, customer_name from reviews GROUP BY customer_name HAVING NumOfReviews>1;  
150  
151 SELECT * FROM desserts;  
152 SELECT * FROM desserts WHERE dessert_name LIKE '%Cookies%';  
153  
154 UPDATE desserts SET price=13.50 WHERE dessert_name LIKE '%Cookies%';
```

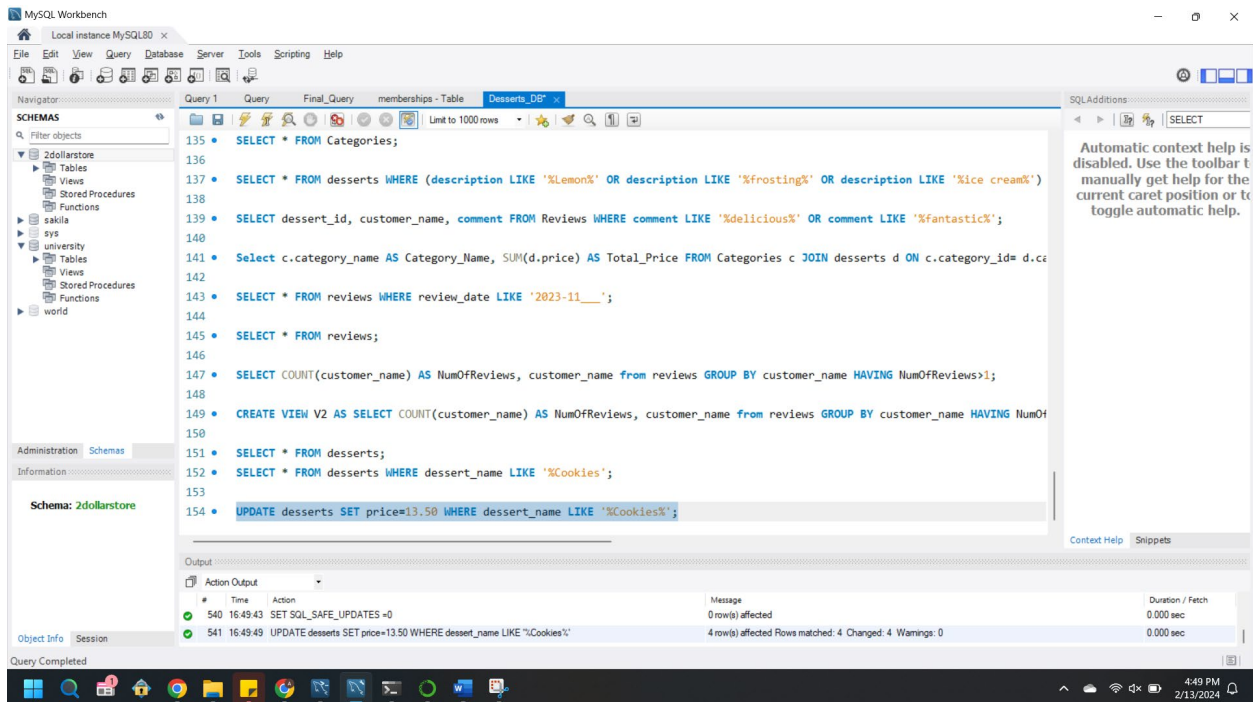
The result grid shows the following data:

dessert_id	dessert_name	price	calories	category_id	description	created_at
3	Oatmeal Cookies	8.75	180	2	Homemade oatmeal cookies with raisins and nuts.	2024-02-13 16:46:57
8	Peanut Butter Cookies	9.50	180	2	Soft peanut butter cookies.	2024-02-13 16:46:57
12	Double Chocolate Cookies	8.75	200	2	Double chocolate chip cookies for chocolate lovers.	2024-02-13 16:46:57
19	Snickerdoodle Cookies	8.75	180	2	Cinnamon-sugar coated snickerdoodle cookies.	2024-02-13 16:46:57

The output pane shows the following messages:

```
535 16:47:45 SELECT * FROM desserts WHERE dessert_name LIKE '%Cookies%' LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec  
536 16:48:04 SELECT * FROM desserts WHERE dessert_name LIKE '%Cookies%' LIMIT 0, 1000 4 row(s) returned 0.000 sec / 0.000 sec
```

Update:



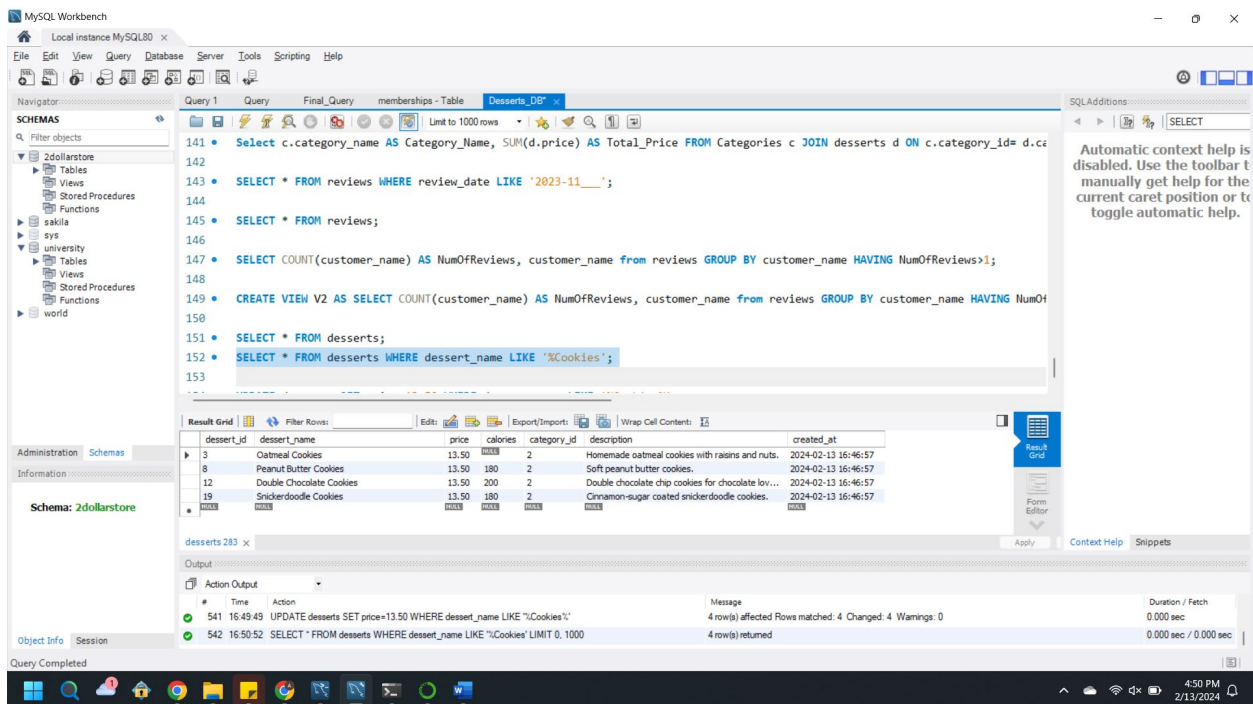
MySQL Workbench interface showing a query execution. The query is:

```
135 • SELECT * FROM Categories;
136
137 • SELECT * FROM desserts WHERE (description LIKE '%lemon%' OR description LIKE '%frosting%' OR description LIKE '%ice cream%');
138
139 • SELECT dessert_id, customer_name, comment FROM Reviews WHERE comment LIKE '%delicious%' OR comment LIKE '%fantastic%';
140
141 • Select c.category_name AS Category_Name, SUM(d.price) AS Total_Price FROM Categories c JOIN desserts d ON c.category_id=d.c
142
143 • SELECT * FROM reviews WHERE review_date LIKE '2023-11-__';
144
145 • SELECT * FROM reviews;
146
147 • SELECT COUNT(customer_name) AS NumOfReviews, customer_name from reviews GROUP BY customer_name HAVING NumOfReviews>1;
148
149 • CREATE VIEW V2 AS SELECT COUNT(customer_name) AS NumOfReviews, customer_name from reviews GROUP BY customer_name HAVING NumOf
150
151 • SELECT * FROM desserts;
152 • SELECT * FROM desserts WHERE dessert_name LIKE '%Cookies';
153
154 • UPDATE desserts SET price=13.50 WHERE dessert_name LIKE '%Cookies';
```

The output shows the execution of the UPDATE statement:

#	Time	Action	Message	Duration / Fetch
540	16:49:43	SET SQL_SAFE_UPDATES=0	0 row(s) affected	0.000 sec
541	16:49:49	UPDATE desserts SET price=13.50 WHERE dessert_name LIKE '%Cookies%'	4 row(s) affected Rows matched: 4 Changed: 4 Warnings: 0	0.000 sec

After Update:



MySQL Workbench interface showing the result grid of the query. The result grid displays the updated price of 'Cookies' as 13.50.

dessert_id	dessert_name	price	calories	category_id	description	created_at
3	Oatmeal Cookies	13.50	180	2	Homemade oatmeal cookies with raisins and nuts.	2024-02-13 16:46:57
8	Peanut Butter Cookies	13.50	180	2	Soft peanut butter cookies.	2024-02-13 16:46:57
12	Double Chocolate Cookies	13.50	200	2	Double chocolate chip cookies for chocolate lov...	2024-02-13 16:46:57
19	Snickerdoodle Cookies	13.50	180	2	Cinnamon-sugar coated snickerdoodle cookies.	2024-02-13 16:46:57

The output shows the execution of the UPDATE statement:

#	Time	Action	Message	Duration / Fetch
541	16:49:49	UPDATE desserts SET price=13.50 WHERE dessert_name LIKE '%Cookies%'	4 row(s) affected Rows matched: 4 Changed: 4 Warnings: 0	0.000 sec
542	16:50:52	SELECT * FROM desserts WHERE dessert_name LIKE '%Cookies' LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Query 9:

```
UPDATE desserts SET price=13.50 WHERE dessert_name LIKE '%Cookies%';
```

Question 10:

Measure the performance from the SQL workbench on different DDL and DML queries performed in this homework so far.

Answer 10:

By analyzing the time taken by each query, I see that the DML queries are faster than DDL queries.

DDL queries:

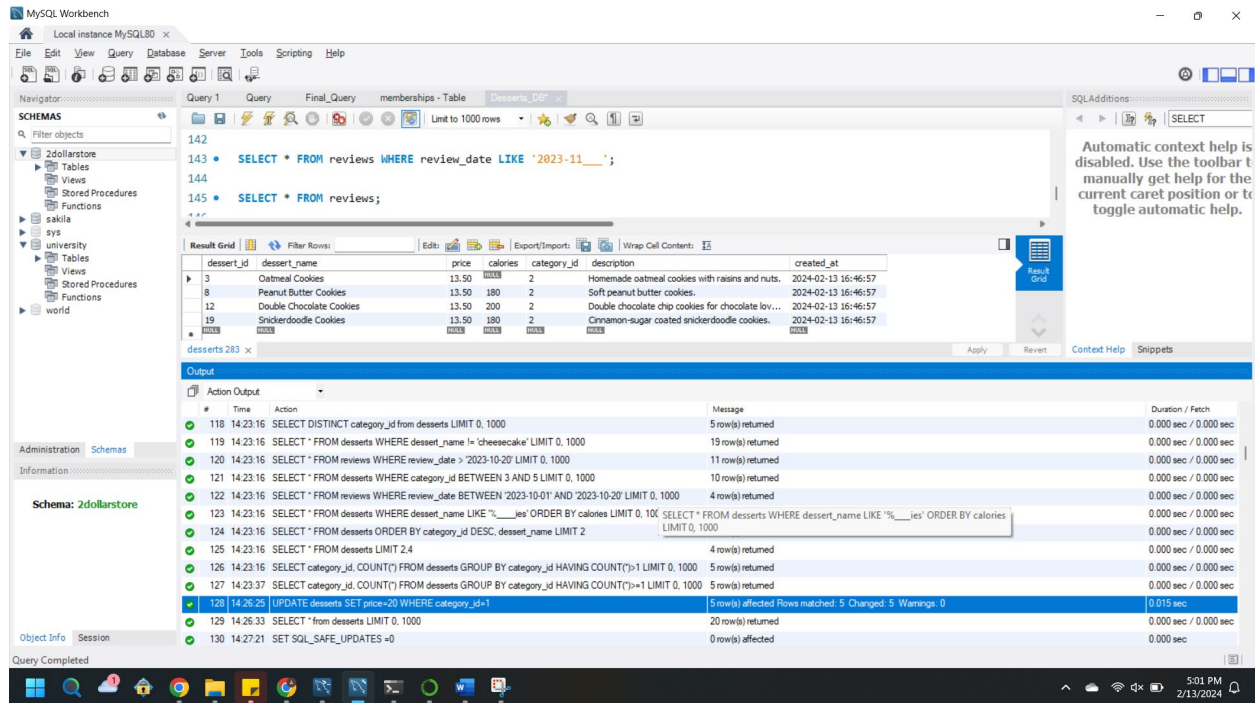
CREATE database, table, index and DROP table took more time than other queries. On average it took 0.020 seconds.

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying a query: `SELECT * FROM reviews WHERE review_date LIKE '2023-11-';`. The 'Result Grid' shows a table with columns: `dessert_id`, `dessert_name`, `price`, `calories`, `category_id`, `description`, and `created_at`. The table contains 5 rows of data for different cookies. The 'Output' tab is also visible, showing a log of executed queries and their execution times. The log includes the following queries and their durations:

#	Time	Action	Message	Duration / Fetch
100	14:10:17	SELECT * FROM desserts ORDER BY category_id DESC, dessert_name LIMIT 2	2 row(s) returned	0.000 sec / 0.000 sec
101	14:14:54	SELECT * FROM desserts LIMIT 4,2	2 row(s) returned	0.000 sec / 0.000 sec
102	14:15:14	SELECT * FROM desserts LIMIT 2,4	4 row(s) returned	0.000 sec / 0.000 sec
103	14:22:14	SELECT category_id, COUNT(*) FROM desserts GROUP BY category_id HAVING COUNT(*)>2 LIMIT 0, 1000	Error Code: 1054. Unknown column 'COUNT' in 'having clause'	0.000 sec
104	14:22:56	SELECT category_id, COUNT(*) FROM desserts GROUP BY category_id HAVING COUNT(*)>2 LIMIT 0, 1000	4 row(s) returned	0.016 sec / 0.000 sec
105	14:23:16	DROP DATABASE IF EXISTS 'desserts_db'	3 row(s) affected	0.078 sec
106	14:23:16	CREATE DATABASE 'desserts_db'	1 row(s) affected	0.000 sec
107	14:23:16	USE desserts_db	0 row(s) affected	0.000 sec
108	14:23:16	CREATE TABLE IF NOT EXISTS Categories (category_id INT AUTO_INCREMENT PRIMARY KEY, cat...	0 row(s) affected	0.016 sec
109	14:23:16	CREATE TABLE IF NOT EXISTS Desserts (dessert_id INT AUTO_INCREMENT PRIMARY KEY, dessert...	0 row(s) affected	0.015 sec
110	14:23:16	CREATE TABLE IF NOT EXISTS Reviews (review_id INT AUTO_INCREMENT PRIMARY KEY, dessert...	0 row(s) affected	0.031 sec
111	14:23:16	INSERT INTO Categories (category_name) VALUES ('Cakes'), ('Cookies'), ('Ice Cream'), ('Pies'), (...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.000 sec
112	14:23:16	INSERT INTO Desserts (dessert_name, price, calories, category_id, description) VALUES ('Chocolate Cake'...	20 row(s) affected Records: 20 Duplicates: 0 Warnings: 0	0.000 sec

DML queries:

UPDATE query took 0.015 seconds, SELECT and INSERT query took 0 seconds.



The screenshot displays the MySQL Workbench interface for a local instance of MySQL 8.0. The 'Query' tab is active, showing a query window with the following SQL statements:

```
142
143 * SELECT * FROM reviews WHERE review_date LIKE '2023-11-__';
144
145 * SELECT * FROM reviews;
```

The 'Result Grid' shows a table with columns: dessert_id, dessert_name, price, calories, category_id, description, and created_at. The table contains 5 rows of data.

The 'Output' tab is active, showing a log of SQL statements and their execution times. The log includes the following statements and their execution times:

#	Time	Action	Message	Duration / Fetch
118	14:23:16	SELECT DISTINCT category_id from desserts LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
119	14:23:16	SELECT * FROM desserts WHERE dessert_name != 'cheesecake' LIMIT 0, 1000	19 row(s) returned	0.000 sec / 0.000 sec
120	14:23:16	SELECT * FROM reviews WHERE review_date > '2023-10-20' LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
121	14:23:16	SELECT * FROM desserts WHERE category_id BETWEEN 3 AND 5 LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
122	14:23:16	SELECT * FROM reviews WHERE review_date BETWEEN '2023-10-01' AND '2023-10-20' LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
123	14:23:16	SELECT * FROM desserts WHERE dessert_name LIKE '%__ies' ORDER BY calories LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
124	14:23:16	SELECT * FROM desserts ORDER BY category_id DESC, dessert_name LIMIT 2	2 row(s) returned	0.000 sec / 0.000 sec
125	14:23:16	SELECT * FROM desserts LIMIT 2,4	4 row(s) returned	0.000 sec / 0.000 sec
126	14:23:16	SELECT category_id, COUNT(*) FROM desserts GROUP BY category_id HAVING COUNT(*)>1 LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
127	14:23:37	SELECT category_id, COUNT(*) FROM desserts GROUP BY category_id HAVING COUNT(*)>=1 LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
128	14:26:29	UPDATE desserts SET price=20 WHERE category_id=1	5 row(s) affected Rows matched: 5 Changed: 5 Warnings: 0	0.015 sec
129	14:26:33	SELECT * from desserts LIMIT 0, 1000	20 row(s) returned	0.000 sec / 0.000 sec
130	14:27:21	SET SQL_SAFE_UPDATES=0	0 row(s) affected	0.000 sec

The 'Query Completed' status is shown at the bottom of the interface.