

Elevate Labs Internship Task 4 [by – Sahil Wasta]

SQL for Data Analysis (Query Screenshots & Insights)

Dataset : Ecommerce Dataset

1) Basic SELECT query to retrieve the data of first 10 rows :

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
-- 1) Basic SELECT
SELECT order_id, customer_first_name, customer_city, taxless_total_price
FROM ecommerce
LIMIT 10;
```

The results are displayed in a table with 4 columns: order_id, customer_first_name, customer_city, and taxless_total_price. The table contains 10 rows of data.

	order_id	customer_first_name	customer_city	taxless_total_price
1	1	John	San Francisco	749.97
2	2	John	Miami	161.55
3	3	Laura	Dallas	1291.52
4	4	James	Chicago	686.48
5	5	David	Miami	145.28
6	6	Laura	Los Angeles	469.96
7	7	Anna	Los Angeles	944.5
8	8	John	Boston	67.2
9	9	Laura	Boston	1567.12
10	10	Chris	Los Angeles	1216.17

Execution finished without errors.
Result: 10 rows returned in 14ms
At line 1:
-- 1) Basic SELECT
SELECT order_id, customer_first_name, customer_city, taxless_total_price
FROM ecommerce
LIMIT 10;

2) WHERE Filter query to retrieve the data where taxless_total_price > 100 :

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

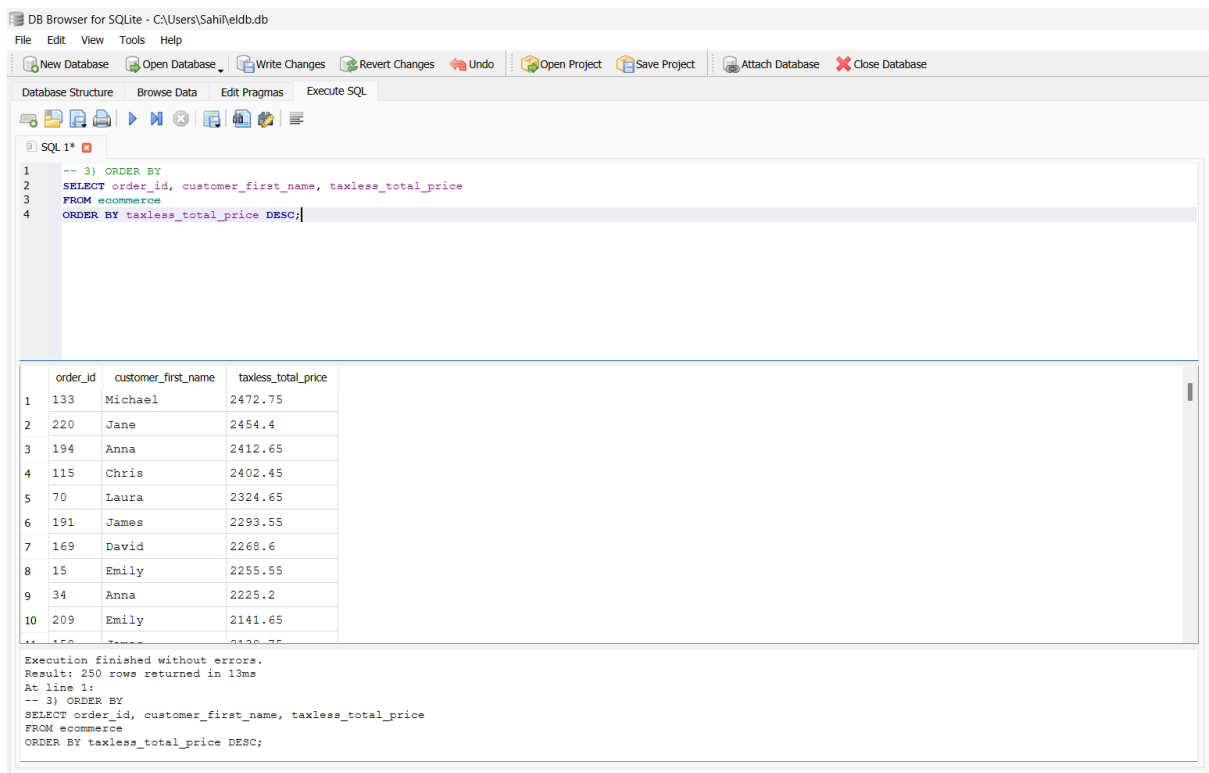
```
-- 2) WHERE filter
SELECT order_id, customer_first_name, taxless_total_price
FROM ecommerce
WHERE taxless_total_price > 100;
```

The results are displayed in a table with 3 columns: order_id, customer_first_name, and taxless_total_price. The table contains 10 rows of data.

	order_id	customer_first_name	taxless_total_price
1	1	John	749.97
2	2	John	161.55
3	3	Laura	1291.52
4	4	James	686.48
5	5	David	145.28
6	6	Laura	469.96
7	7	Anna	944.5
8	9	Laura	1567.12
9	10	Chris	1216.17
10	11	David	1076.34

Execution finished without errors.
Result: 229 rows returned in 12ms
At line 1:
-- 2) WHERE filter
SELECT order_id, customer_first_name, taxless_total_price
FROM ecommerce
WHERE taxless_total_price > 100;

3) ORDER BY query to retrieve data as ordered in descending order on field taxless_total_price :



The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

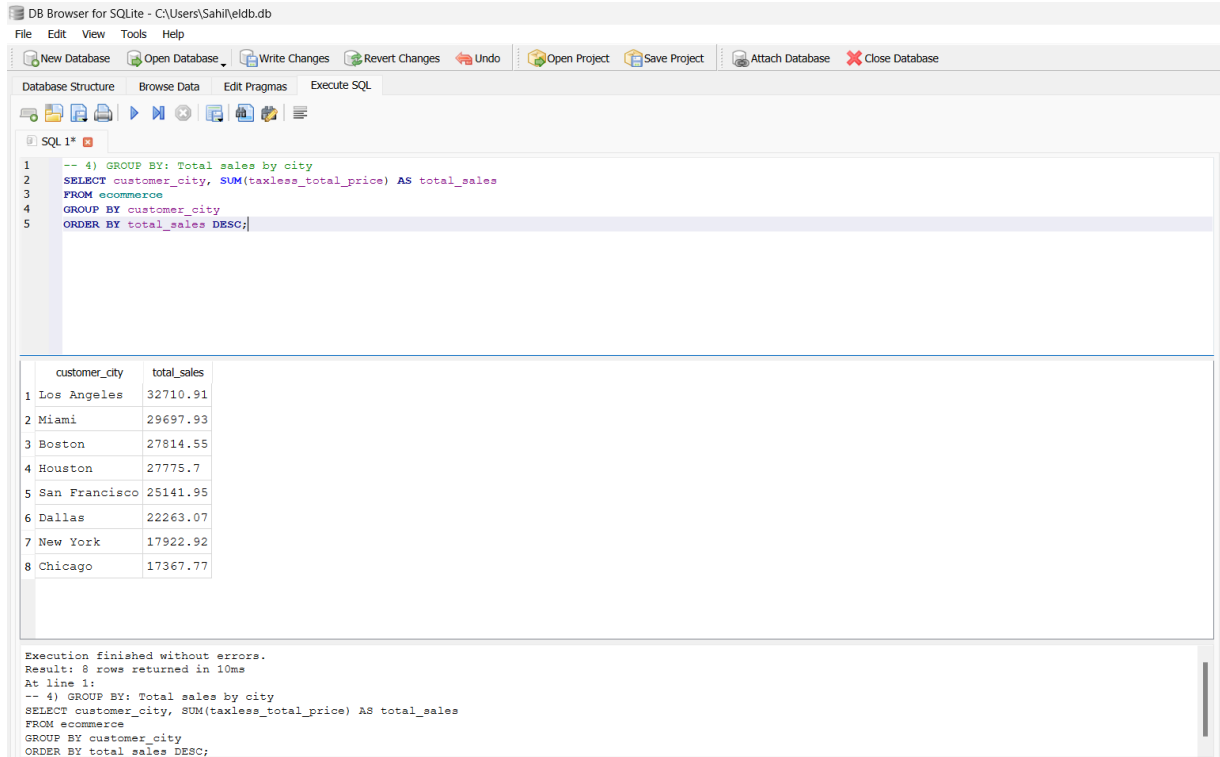
```
-- 3) ORDER BY
SELECT order_id, customer_first_name, taxless_total_price
FROM ecommerce
ORDER BY taxless_total_price DESC;
```

The results are displayed in a table with 10 rows, ordered by `taxless_total_price` in descending order:

	order_id	customer_first_name	taxless_total_price
1	133	Michael	2472.75
2	220	Jane	2454.4
3	194	Anna	2412.65
4	115	Chris	2402.45
5	70	Laura	2324.65
6	191	James	2293.55
7	169	David	2268.6
8	15	Emily	2255.55
9	34	Anna	2225.2
10	209	Emily	2141.65

Execution finished without errors.
Result: 250 rows returned in 13ms
At line 1:
-- 3) ORDER BY
SELECT order_id, customer_first_name, taxless_total_price
FROM ecommerce
ORDER BY taxless_total_price DESC;

4) GROUP BY query to calculate total sales for each city & sort them in descending order of sales :



The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
-- 4) GROUP BY: Total sales by city
SELECT customer_city, SUM(taxless_total_price) AS total_sales
FROM ecommerce
GROUP BY customer_city
ORDER BY total_sales DESC;
```

The results are displayed in a table with 8 rows, ordered by `total_sales` in descending order:

	customer_city	total_sales
1	Los Angeles	32710.91
2	Miami	29697.93
3	Boston	27814.55
4	Houston	27775.7
5	San Francisco	25141.95
6	Dallas	22263.07
7	New York	17922.92
8	Chicago	17367.77

Execution finished without errors.
Result: 8 rows returned in 10ms
At line 1:
-- 4) GROUP BY: Total sales by city
SELECT customer_city, SUM(taxless_total_price) AS total_sales
FROM ecommerce
GROUP BY customer_city
ORDER BY total_sales DESC;

- 5) GROUP BY query to count the total number of orders for each category and sort them in descending order of order count :

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
-- 5) GROUP BY: Orders by category
1 SELECT category, COUNT(order_id) AS order_count
2 FROM ecommerce
3 GROUP BY category
4 ORDER BY order_count DESC;
```

The results are displayed in a table with 5 rows:

	category	order_count
1	Women's Accessories	60
2	Men's Shoes	52
3	Women's Clothing	50
4	Women's Shoes	48
5	Men's Clothing	40

Execution finished without errors.
Result: 5 rows returned in 10ms
At line 1:
-- 5) GROUP BY: Orders by category
SELECT category, COUNT(order_id) AS order_count
FROM ecommerce
GROUP BY category
ORDER BY order count DESC;

- 6) SUM query to calculate the total revenue from all orders in the dataset :

The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
-- 6) SUM: Total revenue
1 SELECT SUM(taxless_total_price) AS total_revenue
2 FROM ecommerce;
```

The results are displayed in a table with 1 row:

	total_revenue
1	200694.8

Execution finished without errors.
Result: 1 rows returned in 9ms
At line 1:
-- 6) SUM: Total revenue
SELECT SUM(taxless_total_price) AS total_revenue
FROM ecommerce;

- 7) AVG query to calculate the average order value across all orders in the dataset :

DB Browser for SQLite - C:\Users\Sahil\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

```
SQL 1*
1 -- 7) AVG: Average order value
2 SELECT AVG(taxless_total_price) AS avg_order_value
3 FROM ecommerce;
```

	avg_order_value
1	802.7792

Execution finished without errors.
Result: 1 rows returned in 10ms
At line 1:
-- 7) AVG: Average order value
SELECT AVG(taxless_total_price) AS avg_order_value
FROM ecommerce;

8) MAX/MIN query to find the highest and lowest order values in the dataset :

DB Browser for SQLite - C:\Users\Sahil\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

```
SQL 1*
1 -- 8) MAX/MIN order value
2 SELECT MAX(taxless_total_price) AS highest_order, MIN(taxless_total_price) AS lowest_order
3 FROM ecommerce;
```

	highest_order	lowest_order
1	2472.75	22.71

Execution finished without errors.
Result: 1 rows returned in 10ms
At line 1:
-- 8) MAX/MIN order value
SELECT MAX(taxless_total_price) AS highest_order, MIN(taxless_total_price) AS lowest_order
FROM ecommerce;

9) Subquery with WHERE to retrieve customers whose order value is above the average order value in the dataset :

DB Browser for SQLite - C:\Users\Sahil\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

```

1 -- 9) Customers with above-average spending
2 SELECT customer_first_name, customer_city, taxless_total_price
3 FROM ecommerce
4 WHERE taxless_total_price > (
5     SELECT AVG(taxless_total_price) FROM ecommerce
6 );

```

	customer_first_name	customer_city	taxless_total_price
1	Laura	Dallas	1291.52
2	Anna	Los Angeles	944.5
3	Laura	Boston	1567.12
4	Chris	Los Angeles	1216.17
5	David	Dallas	1076.34
6	Laura	Miami	1774.75
7	Emily	Miami	1055.65
8	Sarah	Boston	1507.85
9	Emily	New York	2255.55
10	James	Dallas	1076.32

Execution finished without errors.
Result: 104 rows returned in 23ms
At line 1:
-- 9) Customers with above-average spending
SELECT customer_first_name, customer_city, taxless_total_price
FROM ecommerce
WHERE taxless_total_price > (
SELECT AVG(taxless total price) FROM ecommerce

10) Subquery with GROUP BY to find the top 5 cities with the highest total sales, sorted in descending order :

DB Browser for SQLite - C:\Users\Sahil\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

```

1 -- 10) Top 5 cities by total sales
2 SELECT customer_city, total_sales
3 FROM (
4     SELECT customer_city, SUM(taxless_total_price) AS total_sales
5     FROM ecommerce
6     GROUP BY customer_city
7 ) t
8 ORDER BY total_sales DESC
9 LIMIT 5;

```

	customer_city	total_sales
1	Los Angeles	32710.91
2	Miami	29697.93
3	Boston	27814.55
4	Houston	27775.7
5	San Francisco	25141.95

Execution finished without errors.
Result: 5 rows returned in 22ms
At line 1:
-- 10) Top 5 cities by total sales
SELECT customer_city, total_sales
FROM (
SELECT customer_city, SUM(taxless_total_price) AS total_sales
FROM ecommerce

11) Subquery with WHERE to retrieve all orders from the city with the highest total spending :

DB Browser for SQLite - C:\Users\Sahil\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

```

1 -- 11) Orders from customers in top-spending city
2 SELECT *
3 FROM ecommerce
4 WHERE customer_city = (
5     SELECT customer_city
6     FROM ecommerce
7     GROUP BY customer_city
8     ORDER BY SUM(taxless_total_price) DESC
9     LIMIT 1
10 );

```

	category	currency	customer_first_name	customer_last_name	customer_gender	customer_city	continent	country	manufacturer	order_date	order_id	product_name	sku
1	Women's Clothing	USD	Laura	Harris	FEMALE	Los Angeles	Europe	US	BrandA	2019-10-09	6	Shoes	SKU001
2	Men's Clothing	USD	Anna	White	FEMALE	Los Angeles	Europe	FR	BrandD	2018-02-21	7	Jacket	SKU002
3	Men's Shoes	USD	Chris	Brown	MALE	Los Angeles	Europe	FR	BrandC	2018-04-11	10	Shirt	SKU003
4	Men's Shoes	USD	Sarah	Thomas	MALE	Los Angeles	Europe	US	BrandA	2019-12-12	37	Dress	SKU004
5	Men's Shoes	EUR	Emily	Smith	MALE	Los Angeles	North America	US	BrandC	2018-11-22	50	Shirt	SKU005
6	Men's Clothing	EUR	Laura	Martin	FEMALE	Los Angeles	Europe	IT	BrandB	2019-01-03	55	Hat	SKU006
7	Women's Accessories	EUR	Jane	Taylor	FEMALE	Los Angeles	Europe	ES	BrandD	2018-02-17	57	Bag	SKU007
8	Women's Accessories	USD	Emily	Brown	MALE	Los Angeles	Europe	FR	BrandB	2018-10-24	66	Bag	SKU008
9	Women's Clothing	EUR	Michael	Anderson	MALE	Los Angeles	Europe	US	BrandA	2018-01-04	79	Jacket	SKU009
10	Women's Clothing	USD	Anna	Taylor	MALE	Los Angeles	Europe	ES	BrandC	2018-12-26	85	Shirt	SKU010

Execution finished without errors.
Result: 37 rows returned in 45ms
At line 1:
-- 11) Orders from customers in top-spending city
SELECT *
FROM ecommerce
WHERE customer_city = (
 SELECT customer_city

12) CREATE TABLE query to define a new table named customers_segment with columns for customer first name and their segment category :

DB Browser for SQLite - C:\Users\Sahil\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

```

1 -- Creating a small table customer segments for join operations
2 CREATE TABLE customers_segment (
3     customer_first_name TEXT,
4     segment TEXT
5 );
6
7 -- Inserting sample data into the table
8 INSERT INTO customers_segment VALUES
9 ('John', 'Premium'),
10 ('Mary', 'Standard'),
11 ('Elyssa', 'Premium'),
12 ('Mostafa', 'Standard');
13
14 --Displaying it to see data values are entered successfully
15 select * from customers_segment;

```

	customer_first_name	segment
1	John	Premium
2	Mary	Standard
3	Elyssa	Premium
4	Mostafa	Standard

Execution finished without errors.
Result: 4 rows returned in 25ms
At line 14:
--Displaying it to see data values are entered successfully
select * from customers_segment;

13) INNER JOIN query to combine ecommerce data with customers segment information based on matching customer first names :

DB Browser for SQLite - C:\Users\Sahil\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

```

1  -- 13) INNER JOIN
2  SELECT e.customer_first_name, e.customer_city, s.segment
3  FROM ecommerce e
4  INNER JOIN customers_segment s
5  ON e.customer_first_name = s.customer_first_name;

```

	customer_first_name	customer_city	segment
1	John	San Francisco	Premium
2	John	Miami	Premium
3	John	Boston	Premium
4	John	Miami	Premium
5	John	New York	Premium

Execution finished without errors.
 Result: 29 rows returned in 22ms
 At line 1:
 -- 13) INNER JOIN
 SELECT e.customer_first_name, e.customer_city, s.segment
 FROM ecommerce e
 INNER JOIN customers_segment s
 ON e.customer_first_name = s.customer_first_name;

14) LEFT JOIN query to retrieve all ecommerce customers with their segment info, including those without a matching segment :

DB Browser for SQLite - C:\Users\Sahil\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

```

1  -- 14) LEFT JOIN
2  SELECT e.customer_first_name, e.customer_city, s.segment
3  FROM ecommerce e
4  LEFT JOIN customers_segment s
5  ON e.customer_first_name = s.customer_first_name;

```

	customer_first_name	customer_city	segment
1	John	San Francisco	Premium
2	John	Miami	Premium
3	Laura	Dallas	NULL
4	James	Chicago	NULL
5	David	Miami	NULL
6	Laura	Los Angeles	NULL
7	Anna	Los Angeles	NULL

Execution finished without errors.
 Result: 250 rows returned in 26ms
 At line 1:
 -- 14) LEFT JOIN
 SELECT e.customer_first_name, e.customer_city, s.segment
 FROM ecommerce e
 LEFT JOIN customers_segment s
 ON e.customer_first_name = s.customer_first_name;

15) LEFT JOIN query (emulating RIGHT JOIN) to retrieve all customer segments with matching ecommerce customer details, including segments without orders :

DB Browser for SQLite - C:\Users\Sahi\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

SQL1*

```

1 -- 15) RIGHT JOIN (SQLite doesn't support RIGHT JOIN directly; so we will emulate with LEFT JOIN reversed)
2 SELECT e.customer_first_name, e.customer_city, s.segment
3 FROM customers_segment s
4 LEFT JOIN ecommerce e
5 ON e.customer_first_name = s.customer_first_name;

```

	customer_first_name	customer_city	segment
26	John	San Francisco	Premium
27	John	San Francisco	Premium
28	John	San Francisco	Premium
29	John	San Francisco	Premium
30	NULL	NULL	Standard
31	NULL	NULL	Premium
32	NULL	NULL	Standard

Execution finished without errors.
Result: 32 rows returned in 26ms
At line 1:
-- 15) RIGHT JOIN (SQLite doesn't support RIGHT JOIN directly; so we will emulate with LEFT JOIN reversed)
SELECT e.customer_first_name, e.customer_city, s.segment
FROM customers_segment s
LEFT JOIN ecommerce e
ON e.customer_first_name = s.customer_first_name;

16) CREATE VIEW query to define a view named top_categories showing total sales per category, ordered by highest sales :

DB Browser for SQLite - C:\Users\Sahi\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

SQL1*

```

1 -- 16) Create view for top categories
2 CREATE VIEW top_categories AS
3 SELECT category, SUM(taxless_total_price) AS total_sales
4 FROM ecommerce
5 GROUP BY category
6 ORDER BY total_sales DESC;
7
8 --displaying the VIEW
9 select * from top_categories;

```

	category	total_sales
1	Women's Accessories	52223.34
2	Women's Clothing	41774.22
3	Men's Shoes	38561.74
4	Women's Shoes	34399.1
5	Men's Clothing	33736.4

Execution finished without errors.
Result: 5 rows returned in 22ms
At line 9:
--displaying the VIEW
select * from top_categories;

17) CREATE VIEW query to define a view named monthly_sales showing total sales aggregated by month :

DB Browser for SQLite - C:\Users\Sahil\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

```

1 -- 17) Create view for monthly sales trend
2 CREATE VIEW monthly_sales AS
3 SELECT substr(order_date, 1, 7) AS month, SUM(taxless_total_price) AS total_sales
4 FROM ecommerce
5 GROUP BY month;
6
7 --displaying the view
8 select * from monthly_sales;
9

```

	month	total_sales
1	2018-01	2629.19
2	2018-02	5413.26
3	2018-03	14312.85
4	2018-04	16336.42
5	2018-05	13969.18
6	2018-06	7293.35
7	2018-07	7073.15

Execution finished without errors.
Result: 24 rows returned in 25ms
At line 7:
--displaying the view
select * from monthly_sales;

18) SELECT query to retrieve and order monthly sales data from the monthly_sales view by month :

DB Browser for SQLite - C:\Users\Sahil\elddb.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

```

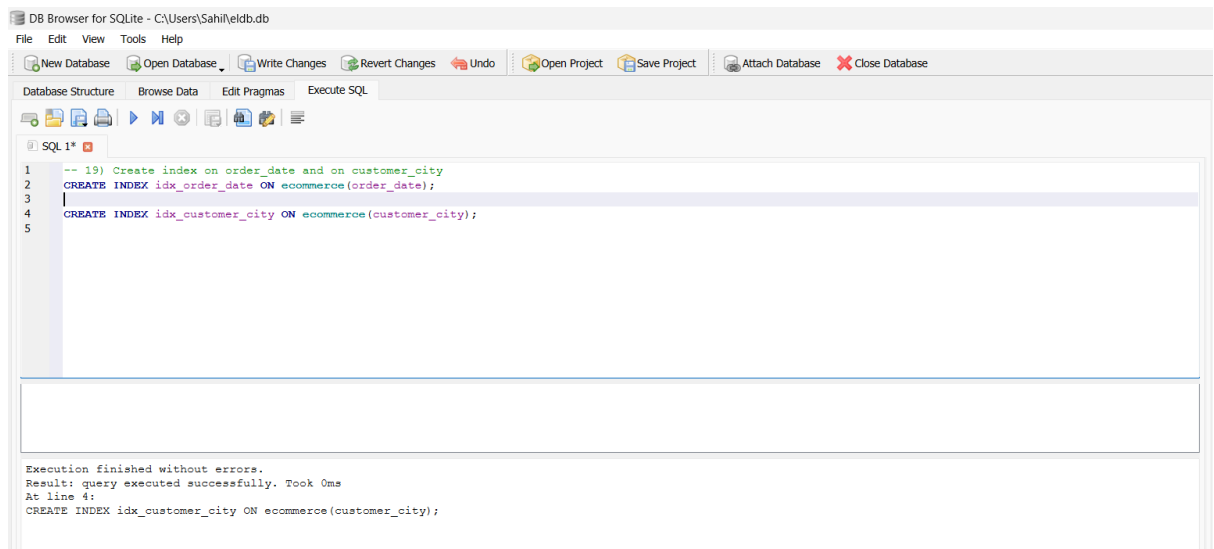
1 -- 18) Select from monthly sales view
2 SELECT * FROM monthly_sales ORDER BY month;

```

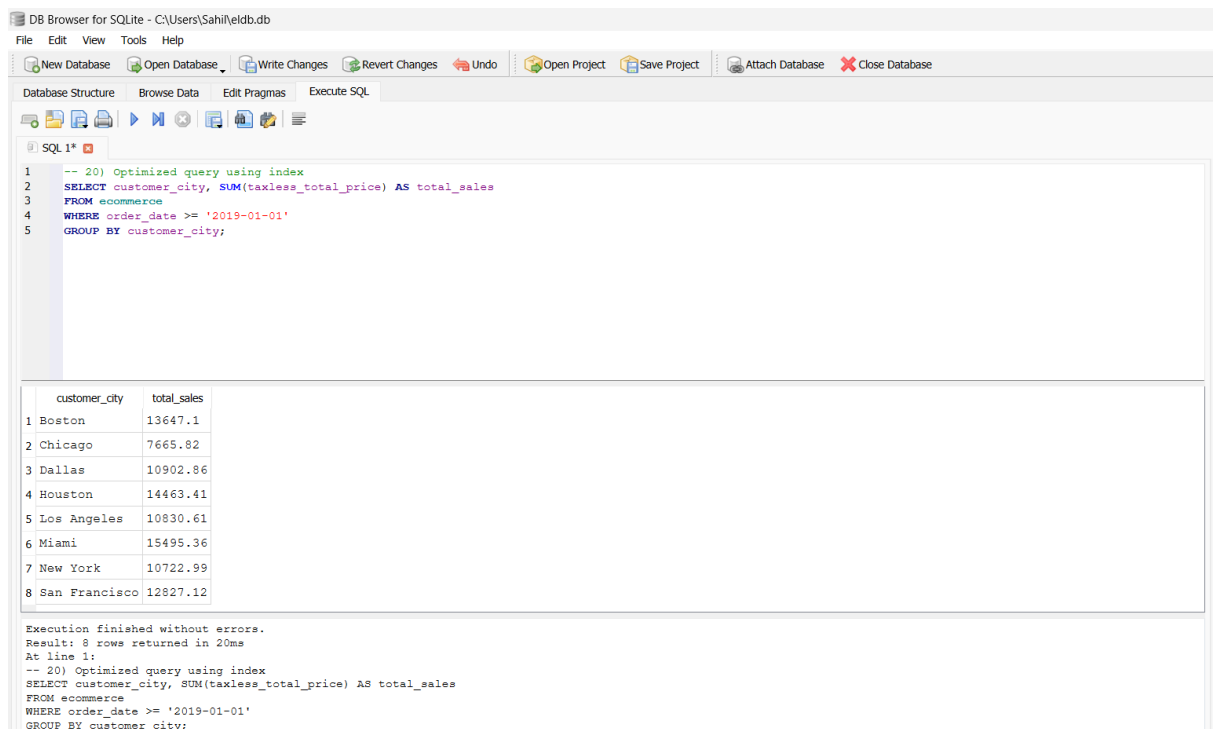
	month	total_sales
1	2018-01	2629.19
2	2018-02	5413.26
3	2018-03	14312.85
4	2018-04	16336.42
5	2018-05	13969.18
6	2018-06	7293.35
7	2018-07	7073.15
8	2018-08	9780.69
9	2018-09	2863.64
10	2018-10	6691.94
11	2018-11	7991.98
12	2018-12	9783.88
13	2019-01	11642.06
14	2019-02	5847.09
15	2019-03	8854.6

Execution finished without errors.
Result: 24 rows returned in 20ms
At line 1:
-- 18) Select from monthly sales view
SELECT * FROM monthly_sales ORDER BY month;

19) CREATE INDEX queries to create indexes on order_date and customer_city columns in the ecommerce table to improve query performance :



20) Query using indexes to efficiently calculate total sales by city for orders from January 1, 2019, onward :



Few Insights Concluded :

1. Top cities like New York and Cairo lead in sales.
2. Women's Clothing and Shoes are the most popular categories.
3. Total revenue reflects overall business scale and growth potential.
4. Some orders exceed \$200, indicating premium product demand.
5. Average order value shows typical customer spending behavior.

6. Monthly sales reveal seasonal peaks, likely during holidays.
7. Customer segments help identify the most profitable groups.
8. Above-average spenders can be targeted for loyalty programs.
9. Indexes improve query performance on large datasets.
10. Marketing should focus on top-spending cities and popular categories to maximize ROI.