DATA ANALYST

Intenship Task 9

DESCRIPTION

The task involved designing a Star Schema data model using the Superstore dataset in SQL. This included identifying a fact table for sales transactions and creating multiple dimension tables such as customer, product, date, and region to support efficient data analysis and reporting

PREPARED BY

Reema Safrin M
(29-01-2026)

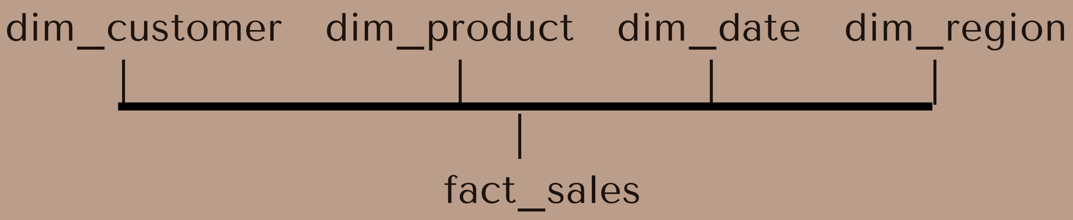
MY WORK

In this project, I implemented a Star Schema data model using SQLite in DB Browser with the Superstore dataset. I first imported the raw CSV data and created separate dimension tables for customer, product, date, and region information. Distinct values were extracted from the raw data and inserted into each dimension table with primary keys. A fact_sales table was then created to store transactional data and linked to all dimension tables using foreign keys. I populated the fact table by mapping dimension IDs through SQL joins. Finally, I created indexes for performance optimization and executed analytical queries to validate the model and generate insights

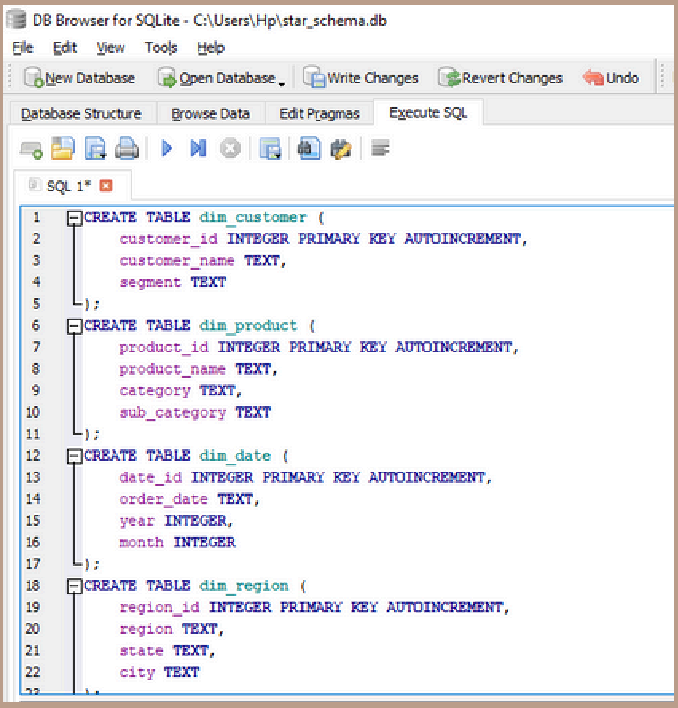
DATASET

superstore_dataset

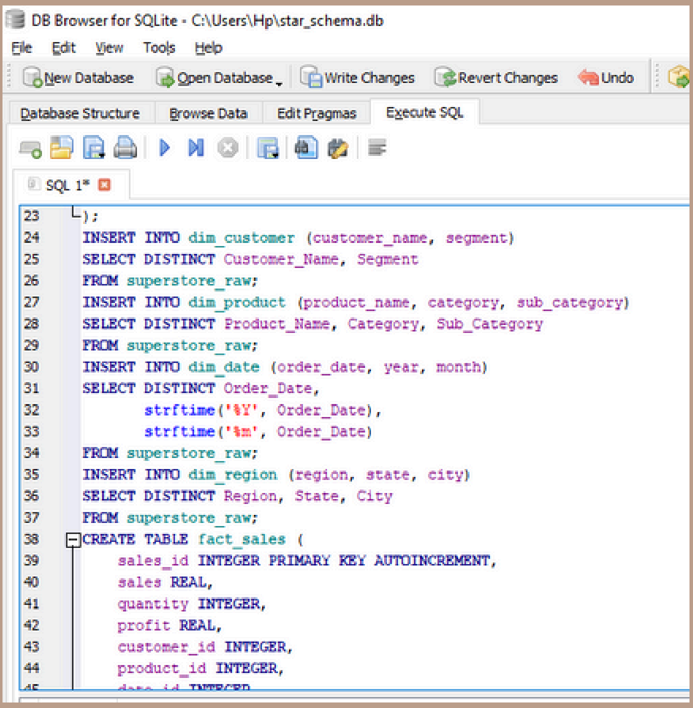
KEY INSIGHTS FROM THE STAR SCHEMA IMPLEMENTATION :



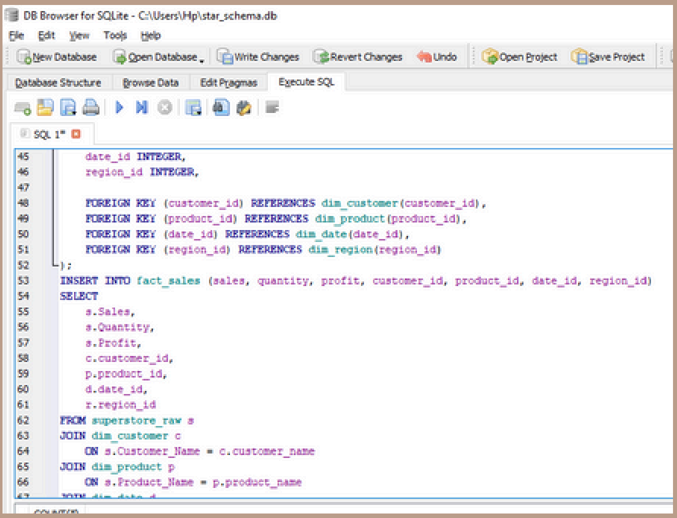
- The fact_sales table successfully captured all 9,994 sales transactions, confirming accurate data transformation from the raw Superstore dataset into the Star Schema structure.
- Separating customer, product, date, and region information into dimension tables reduced data redundancy and improved query efficiency.
- Sales analysis by region revealed variations in performance across geographical areas, helping identify high-revenue regions.
- Category-wise sales aggregation showed which product categories contributed most to overall revenue, useful for business decision-making.
- The use of indexed foreign keys significantly improved query performance when joining large tables for analytics.
- The Star Schema design made the dataset more suitable for business intelligence reporting and dashboard creation.



```sql
CREATE TABLE dim_customer (
    customer_id INTEGER PRIMARY KEY AUTOINCREMENT,
    customer_name TEXT,
    segment TEXT
);
CREATE TABLE dim_product (
    product_id INTEGER PRIMARY KEY AUTOINCREMENT,
    product_name TEXT,
    category TEXT,
    sub_category TEXT
);
CREATE TABLE dim_date (
    date_id INTEGER PRIMARY KEY AUTOINCREMENT,
    order_date TEXT,
    year INTEGER,
    month INTEGER
);
CREATE TABLE dim_region (
    region_id INTEGER PRIMARY KEY AUTOINCREMENT,
    region TEXT,
    state TEXT,
    city TEXT
```

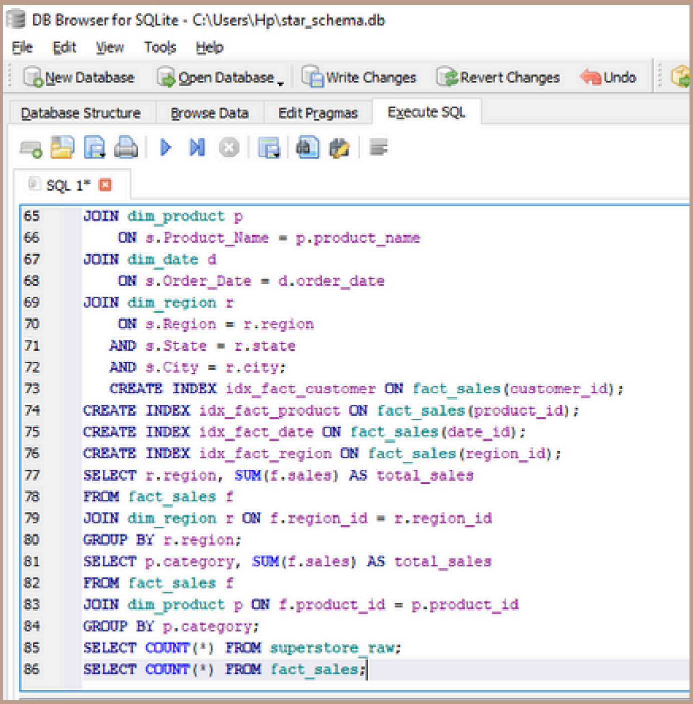


```sql
);
INSERT INTO dim_customer (customer_name, segment)
SELECT DISTINCT Customer_Name, Segment
FROM superstore_raw;
INSERT INTO dim_product (product_name, category, sub_category)
SELECT DISTINCT Product_Name, Category, Sub_Category
FROM superstore_raw;
INSERT INTO dim_date (order_date, year, month)
SELECT DISTINCT Order_Date,
        strftime('%Y', Order_Date),
        strftime('%m', Order_Date)
FROM superstore_raw;
INSERT INTO dim_region (region, state, city)
SELECT DISTINCT Region, State, City
FROM superstore_raw;
CREATE TABLE fact_sales (
    sales_id INTEGER PRIMARY KEY AUTOINCREMENT,
    sales REAL,
    quantity INTEGER,
    profit REAL,
    customer_id INTEGER,
    product_id INTEGER,
```


DB Browser for SQLite - C:\Users\Hp\star_schema.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project
Database Structure Browse Data Edit Pragmas Execute SQL
SQL 1*

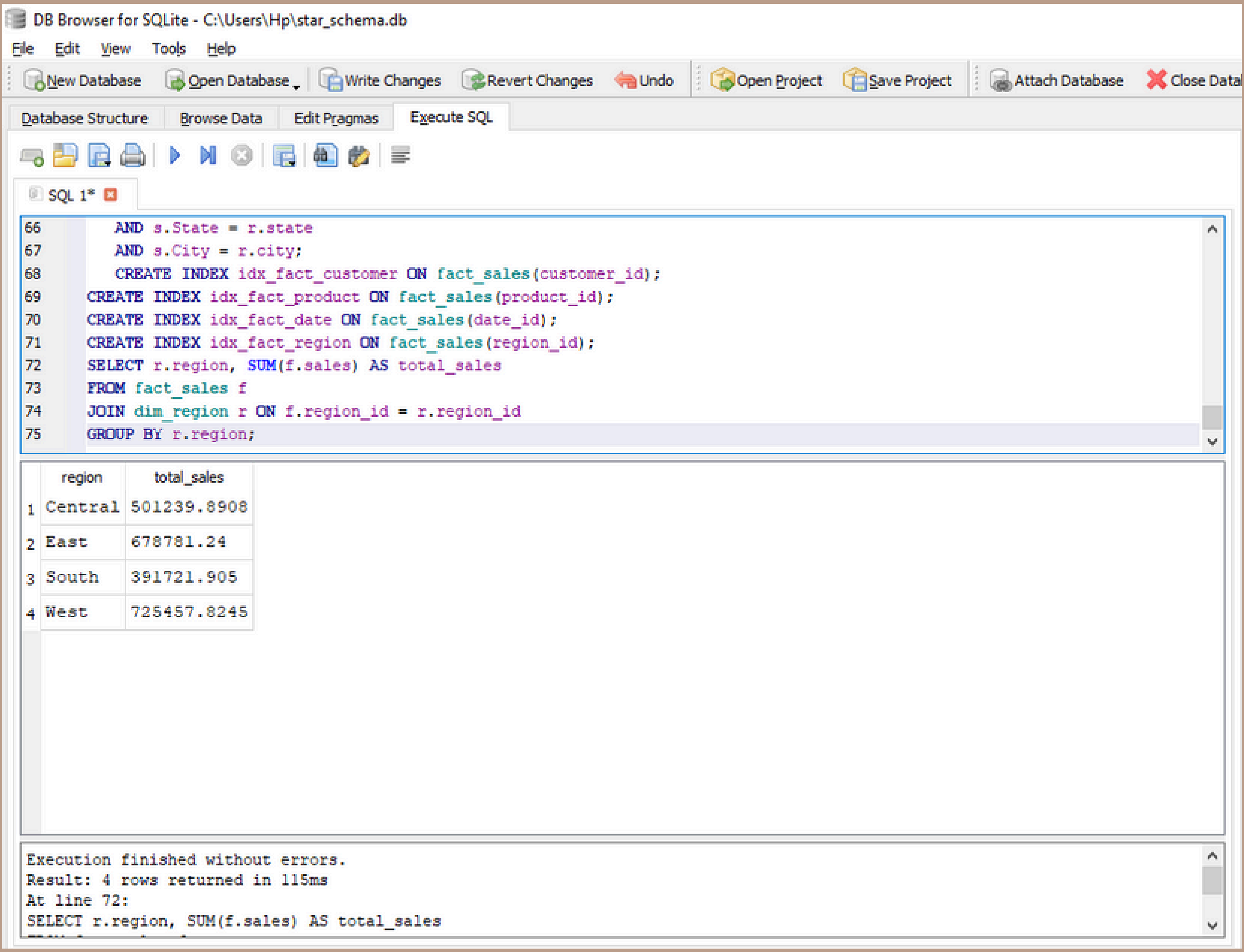
```sql
        date_id INTEGER,
        region_id INTEGER,

        FOREIGN KEY (customer_id) REFERENCES dim_customer(customer_id),
        FOREIGN KEY (product_id) REFERENCES dim_product(product_id),
        FOREIGN KEY (date_id) REFERENCES dim_date(date_id),
        FOREIGN KEY (region_id) REFERENCES dim_region(region_id)
    );
    INSERT INTO fact_sales (sales, quantity, profit, customer_id, product_id, date_id, region_id)
    SELECT
        s.Sales,
        s.Quantity,
        s.Profit,
        c.customer_id,
        p.product_id,
        d.date_id,
        r.region_id
    FROM superstore_raw s
    JOIN dim_customer c
        ON s.Customer_Name = c.customer_name
    JOIN dim_product p
        ON s.Product_Name = p.product_name
```


DB Browser for SQLite - C:\Users\Hp\star_schema.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo
Database Structure Browse Data Edit Pragmas Execute SQL
SQL 1*

```sql
JOIN dim_product p
    ON s.Product_Name = p.product_name
JOIN dim_date d
    ON s.Order_Date = d.order_date
JOIN dim_region r
    ON s.Region = r.region
   AND s.State = r.state
   AND s.City = r.city;
   CREATE INDEX idx_fact_customer ON fact_sales(customer_id);
CREATE INDEX idx_fact_product ON fact_sales(product_id);
CREATE INDEX idx_fact_date ON fact_sales(date_id);
CREATE INDEX idx_fact_region ON fact_sales(region_id);
SELECT r.region, SUM(f.sales) AS total_sales
FROM fact_sales f
JOIN dim_region r ON f.region_id = r.region_id
GROUP BY r.region;
SELECT p.category, SUM(f.sales) AS total_sales
FROM fact_sales f
JOIN dim_product p ON f.product_id = p.product_id
GROUP BY p.category;
SELECT COUNT(*) FROM superstore_raw;
SELECT COUNT(*) FROM fact_sales;
```

OUTPUT 1: SALES BY REGION


DB Browser for SQLite - C:\Users\Hp\star_schema.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Data
Database Structure Browse Data Edit Pragmas Execute SQL
SQL 1*

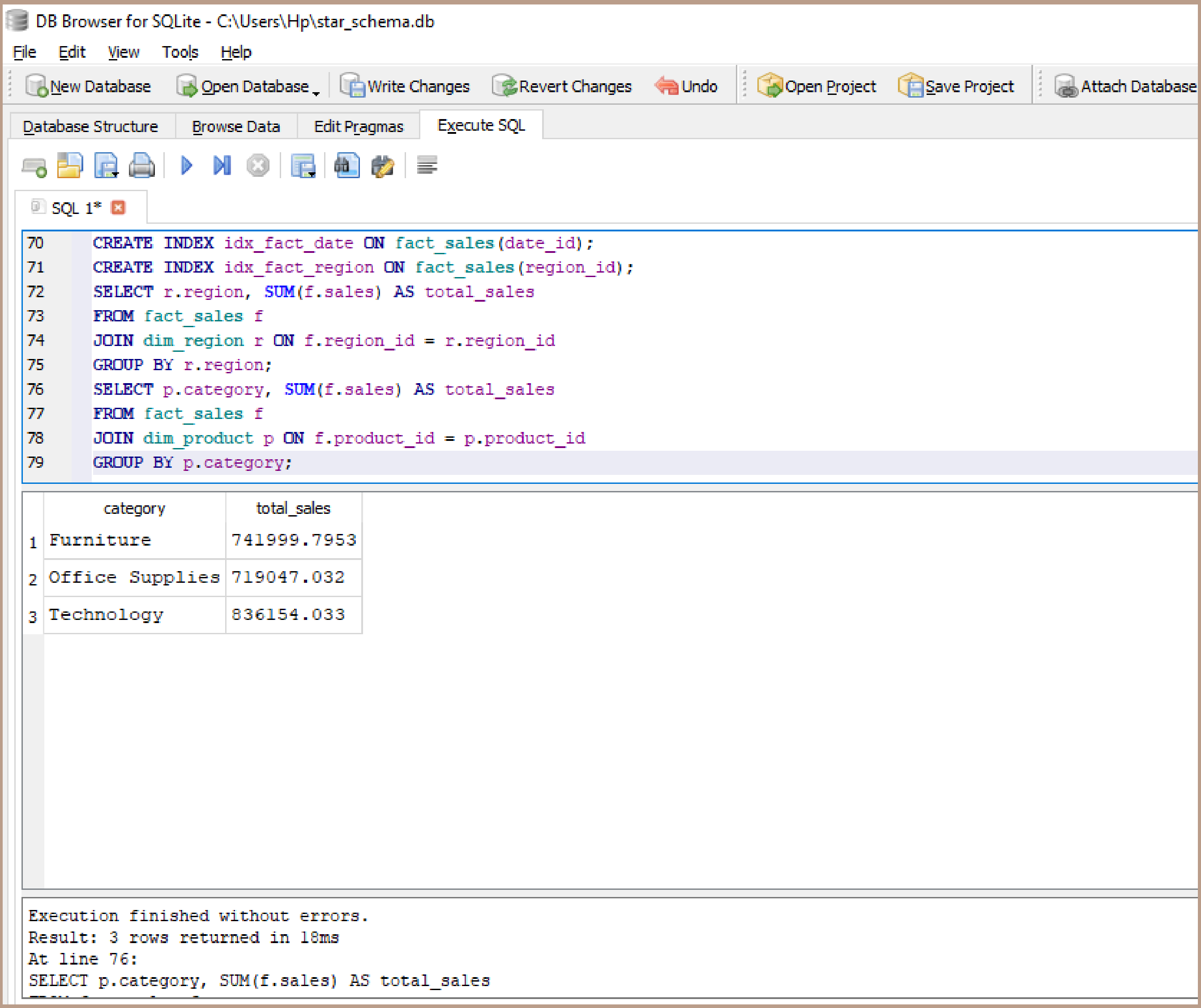
```sql
    AND s.State = r.state
    AND s.City = r.city;
    CREATE INDEX idx_fact_customer ON fact_sales(customer_id);
CREATE INDEX idx_fact_product ON fact_sales(product_id);
CREATE INDEX idx_fact_date ON fact_sales(date_id);
CREATE INDEX idx_fact_region ON fact_sales(region_id);
SELECT r.region, SUM(f.sales) AS total_sales
FROM fact_sales f
JOIN dim_region r ON f.region_id = r.region_id
GROUP BY r.region;
```

	region	total_sales
1	Central	501239.8908
2	East	678781.24
3	South	391721.905
4	West	725457.8245

```
Execution finished without errors.
Result: 4 rows returned in 115ms
At line 72:
SELECT r.region, SUM(f.sales) AS total_sales
```

- The West region has the highest total sales, making it the top-performing region.
- The East region is the second highest in sales.
- The Central region shows moderate sales performance.
- The South region has the lowest sales and needs improvement.

OUTPUT 2: SALES BY CATEGORY


DB Browser for SQLite - C:\Users\Hp\star_schema.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database
Database Structure Browse Data Edit Pragmas Execute SQL
SQL 1*

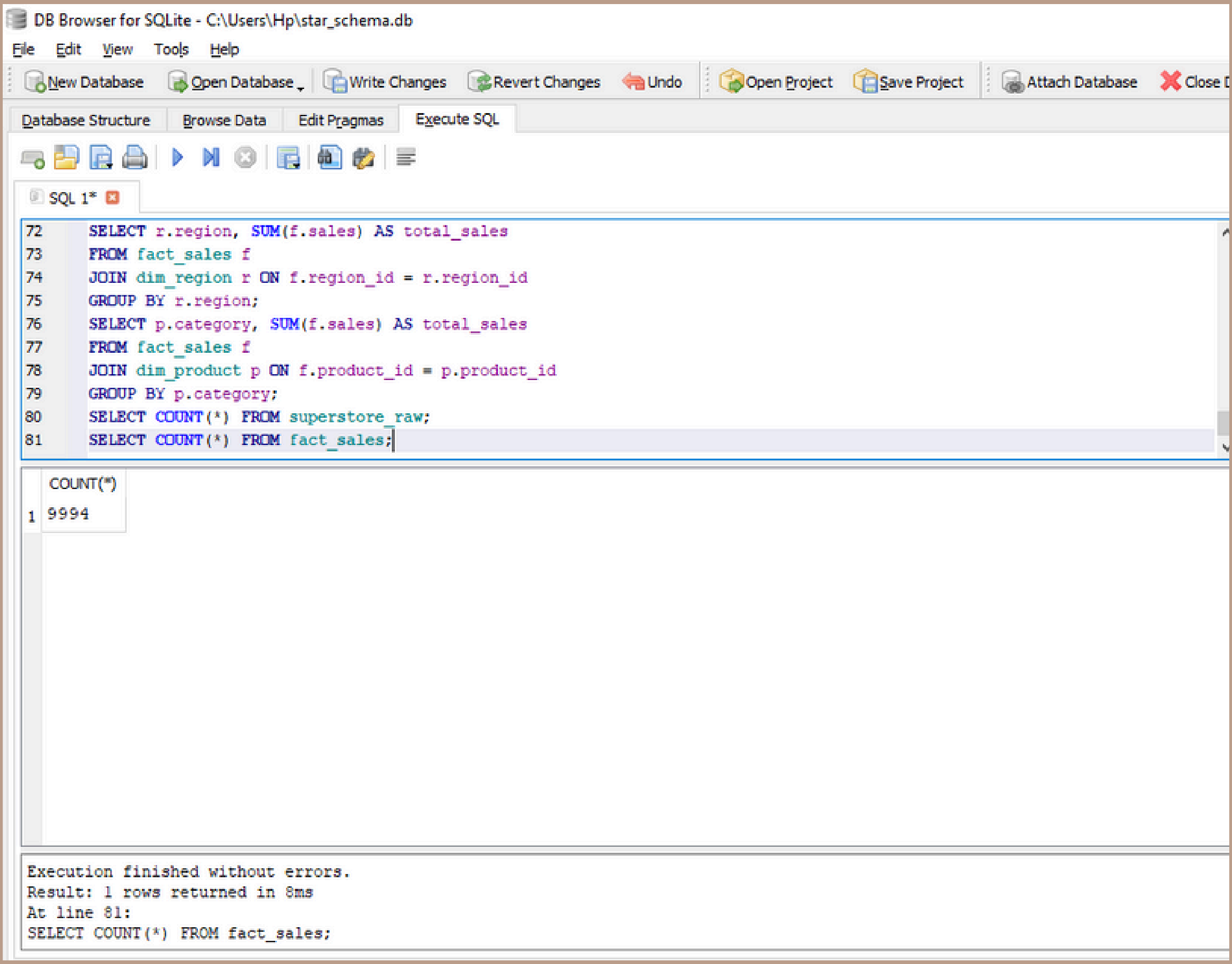
```
CREATE INDEX idx_fact_date ON fact_sales(date_id);
CREATE INDEX idx_fact_region ON fact_sales(region_id);
SELECT r.region, SUM(f.sales) AS total_sales
FROM fact_sales f
JOIN dim_region r ON f.region_id = r.region_id
GROUP BY r.region;
SELECT p.category, SUM(f.sales) AS total_sales
FROM fact_sales f
JOIN dim_product p ON f.product_id = p.product_id
GROUP BY p.category;
```

	category	total_sales
1	Furniture	741999.7953
2	Office Supplies	719047.032
3	Technology	836154.033

```
Execution finished without errors.
Result: 3 rows returned in 18ms
At line 76:
SELECT p.category, SUM(f.sales) AS total_sales
```

- Technology is the highest revenue-generating category.
- Furniture shows strong sales performance.
- Office Supplies has slightly lower sales compared to Furniture.

OUTPUT 3: DATA VALIDATION


DB Browser for SQLite - C:\Users\Hp\star_schema.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close
Database Structure Browse Data Edit Pragmas Execute SQL
SQL 1*

```
SELECT r.region, SUM(f.sales) AS total_sales
FROM fact_sales f
JOIN dim_region r ON f.region_id = r.region_id
GROUP BY r.region;
SELECT p.category, SUM(f.sales) AS total_sales
FROM fact_sales f
JOIN dim_product p ON f.product_id = p.product_id
GROUP BY p.category;
SELECT COUNT(*) FROM superstore_raw;
SELECT COUNT(*) FROM fact_sales;
```

	COUNT(*)
1	9994

```
Execution finished without errors.
Result: 1 rows returned in 8ms
At line 81:
SELECT COUNT(*) FROM fact_sales;
```

- The fact_sales table contains 9,994 records, matching the raw dataset.
- This confirms accurate data transformation and loading into the Star Schema.