SQL for Data Analysis

- **Data source: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce**
- a python code used to load the the .csv file in the olist.db file

```
import pandas as pd
import sqlite3

# Create new or overwrite existing database
conn = sqlite3.connect('/content/olist.db')

# Load CSVs and create tables
customers = pd.read_csv('/content/olist_customers_dataset.csv')
order_payments = pd.read_csv('/content/olist_order_payments_dataset.csv')
order_items = pd.read_csv('/content/olist_order_items_dataset.csv')
geolocation = pd.read_csv('/content/olist_geolocation_dataset.csv')
order_reviews = pd.read_csv('/content/olist_order_reviews_dataset.csv')
orders = pd.read_csv('/content/olist_orders_dataset.csv')
products = pd.read_csv('/content/olist_products_dataset.csv')
sellers = pd.read_csv('/content/olist_sellers_dataset.csv')
product_category_name_translation =
pd.read_csv('/content/product_category_name_translation.csv')

customers.to_sql('customers', conn, if_exists='replace', index=False)
order_payments.to_sql('order_payments', conn, if_exists='replace', index=False)
order_items.to_sql('order_items', conn, if_exists='replace', index=False)
orders.to_sql('orders', conn, if_exists='replace', index=False)
products.to_sql('products', conn, if_exists='replace', index=False)
sellers.to_sql('sellers', conn, if_exists='replace', index=False)
product_category_name_translation.to_sql('product_category_name_translation', conn,
if_exists='replace', index=False)
geolocation.to_sql('geolocation', conn, if_exists='replace', index=False)
order_reviews.to_sql('order_reviews', conn, if_exists='replace', index=False)

conn.close()
```

1) SQL Queries (.sql File)

The SQL queries provided follow all requirements: SELECT, WHERE, ORDER BY, GROUP BY, JOINS (INNER, LEFT), subqueries, aggregates (SUM, AVG), views, and indexes.

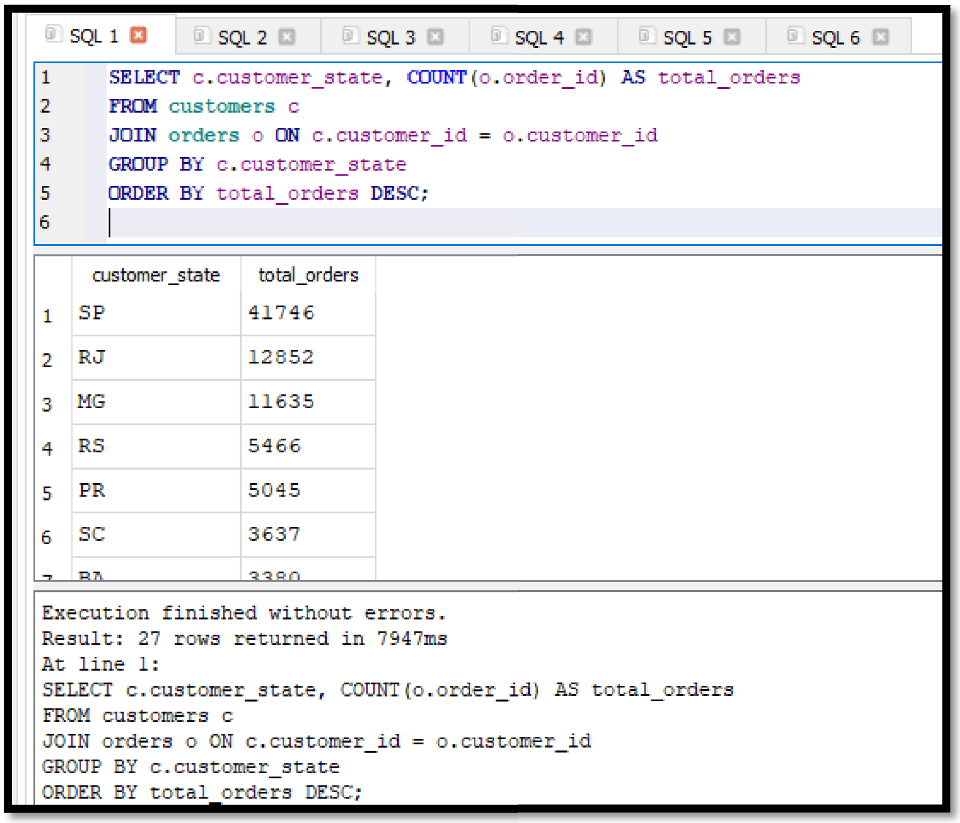
The script is saved as `olist_queries.sql` and can be run in SQLite-compatible tools like DB Browser for SQLite.

2) Screenshots of Output with query

Screenshots of SQL query & outputs captured from database tool DB Browser for SQLite.

1. Basic queries – [SELECT, WHERE, ORDER BY, GROUP BY]

```
SELECT c.customer_state, COUNT(o.order_id) AS total_orders
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY total_orders DESC;
```



```sql
SELECT c.customer_state, COUNT(o.order_id) AS total_orders
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY total_orders DESC;
```

	customer_state	total_orders
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637

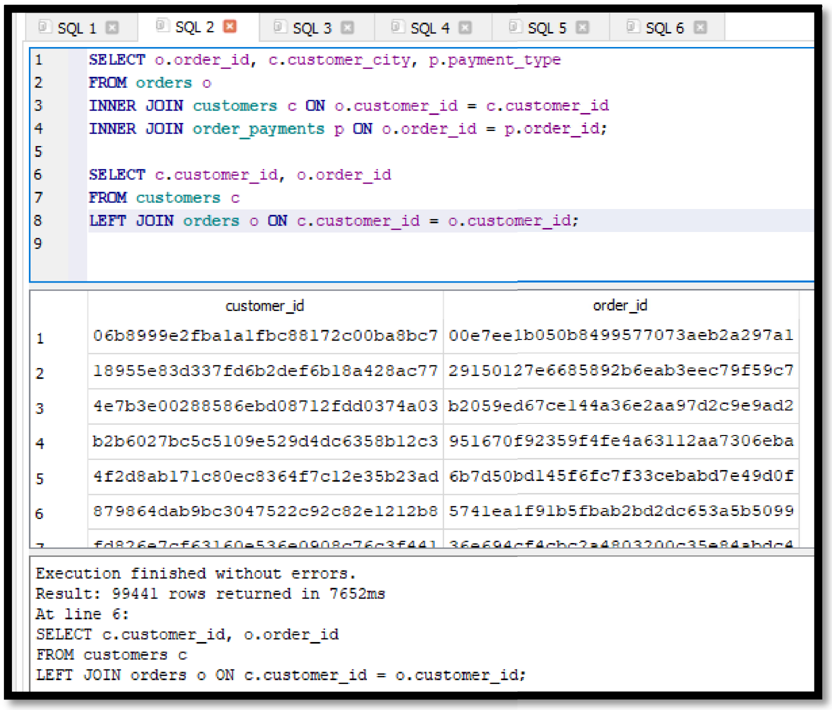
```
Execution finished without errors.
Result: 27 rows returned in 7947ms
At line 1:
SELECT c.customer_state, COUNT(o.order_id) AS total_orders
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY total_orders DESC;
```

2. JOIN query – [INNER, LEFT]

(Note : SQLite does not support RIGHT JOIN)

SELECT o.order_id, c.customer_city, p.payment_type
FROM orders o
INNER JOIN customers c ON o.customer_id = c.customer_id
INNER JOIN order_payments p ON o.order_id = p.order_id;

SELECT c.customer_id, o.order_id
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;



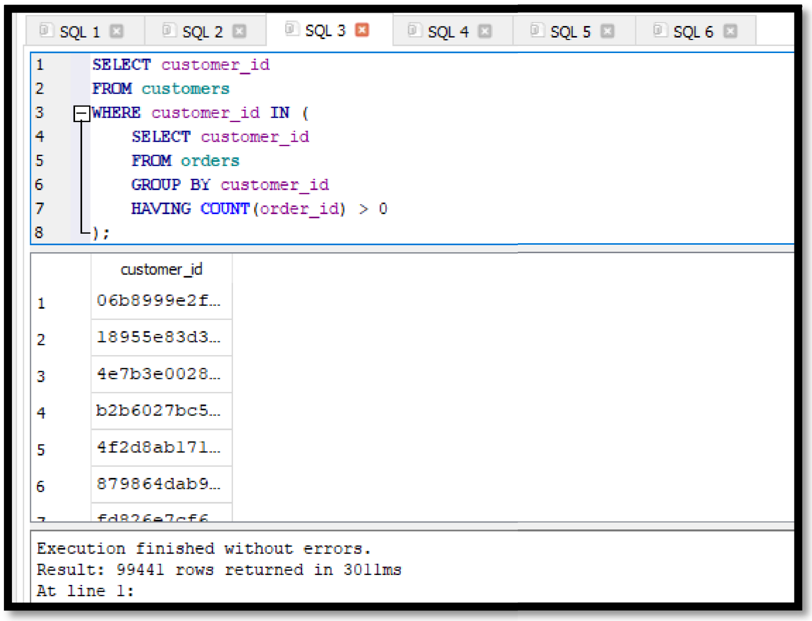
```sql
SELECT o.order_id, c.customer_city, p.payment_type
FROM orders o
INNER JOIN customers c ON o.customer_id = c.customer_id
INNER JOIN order_payments p ON o.order_id = p.order_id;

SELECT c.customer_id, o.order_id
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;
```

	customer_id	order_id
1	06b8999e2fba1a1fbc88172c00ba8bc7	00e7ee1b050b8499577073aeb2a297a1
2	18955e83d337fd6b2def6b18a428ac77	29150127e6685892b6eab3eec79f59c7
3	4e7b3e00288586ebd08712fdd0374a03	b2059ed67ce144a36e2aa97d2c9e9ad2
4	b2b6027bc5c5109e529d4dc6358b12c3	951670f92359f4fe4a63112aa7306eba
5	4f2d8ab171c80ec8364f7c12e35b23ad	6b7d50bd145f6fc7f33cebabd7e49d0f
6	879864dab9bc3047522c92c82e1212b8	5741ea1f91b5fbab2bd2dc653a5b5099

```
Execution finished without errors.
Result: 99441 rows returned in 7652ms
At line 6:
SELECT c.customer_id, o.order_id
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;
```

3. Subquery

```
SELECT customer_id
FROM customers
WHERE customer_id IN (
   SELECT customer_id
   FROM orders
   GROUP BY customer_id
   HAVING COUNT(order_id) > 0
);
```



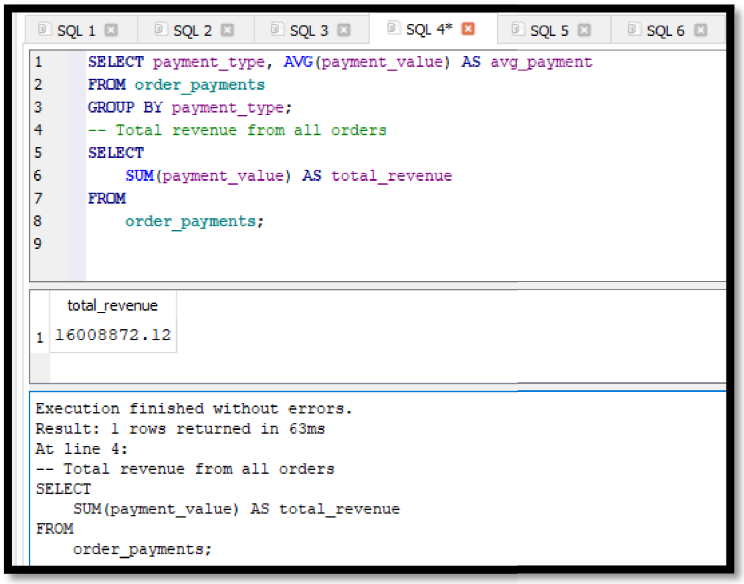
```
SELECT customer_id
FROM customers
WHERE customer_id IN (
    SELECT customer_id
    FROM orders
    GROUP BY customer_id
    HAVING COUNT(order_id) > 0
);
```

```
     customer_id
1    06b8999e2f...
2    18955e83d3...
3    4e7b3e0028...
4    b2b6027bc5...
5    4f2d8ab171...
6    879864dab9...
7    fd826e7cf6
```

```
Execution finished without errors.
Result: 99441 rows returned in 3011ms
At line 1:
```

4. Aggregate function

```
SELECT payment_type, AVG(payment_value) AS avg_payment
FROM order_payments
GROUP BY payment_type;
-- Total revenue from all orders
SELECT
   SUM(payment_value) AS total_revenue
FROM
   order_payments;
```



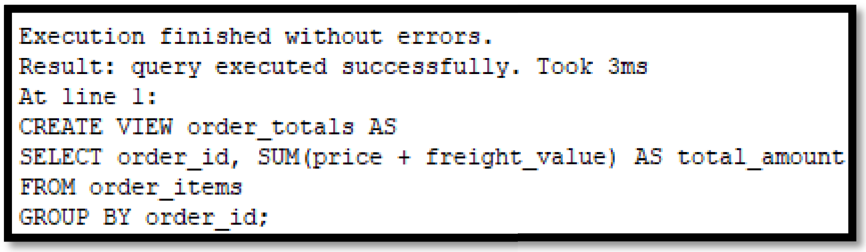
```
SELECT payment_type, AVG(payment_value) AS avg_payment
FROM order_payments
GROUP BY payment_type;
-- Total revenue from all orders
SELECT
    SUM(payment_value) AS total_revenue
FROM
    order_payments;
```

```
   total_revenue
1  16008872.12
```

```
Execution finished without errors.
Result: 1 rows returned in 63ms
At line 4:
-- Total revenue from all orders
SELECT
    SUM(payment_value) AS total_revenue
FROM
    order_payments;
```

5. Views for analysis

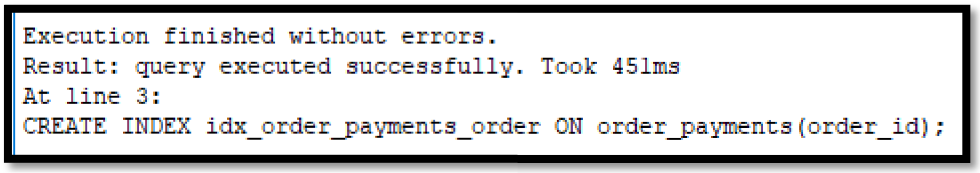
CREATE VIEW order_totals AS
SELECT order_id, SUM(price + freight_value) AS total_amount
FROM order_items
GROUP BY order_id;



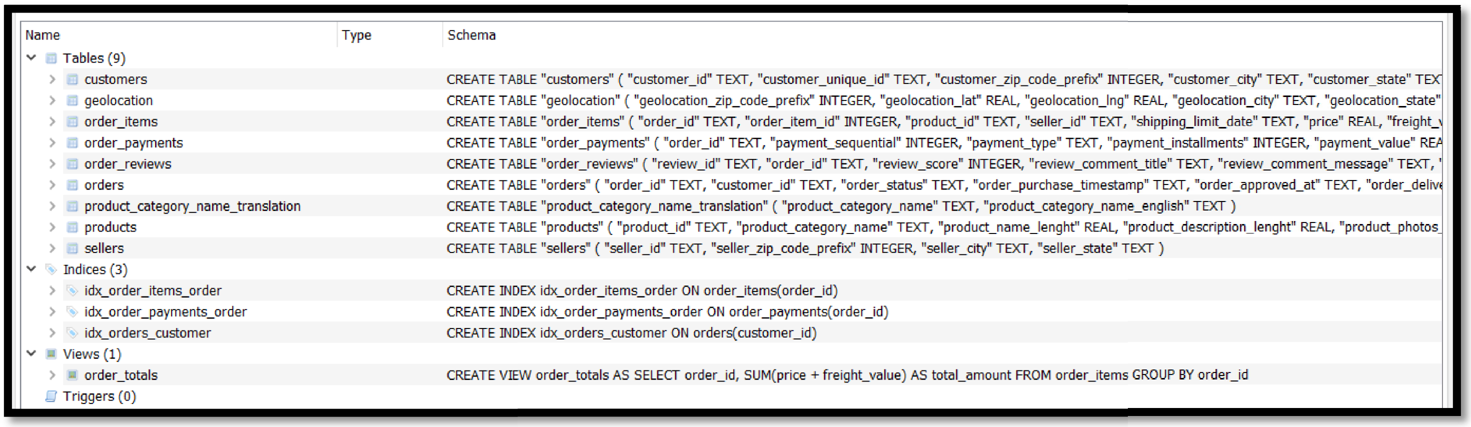
```
Execution finished without errors.
Result: query executed successfully. Took 3ms
At line 1:
CREATE VIEW order_totals AS
SELECT order_id, SUM(price + freight_value) AS total_amount
FROM order_items
GROUP BY order_id;
```

6. Index effectiveness

CREATE INDEX idx_orders_customer ON orders(customer_id);
CREATE INDEX idx_order_items_order ON order_items(order_id);
CREATE INDEX idx_order_payments_order ON order_payments(order_id);



```
Execution finished without errors.
Result: query executed successfully. Took 451ms
At line 3:
CREATE INDEX idx_order_payments_order ON order_payments(order_id);
```



Name	Type	Schema
Tables (9)		
customers		CREATE TABLE "customers" ("customer_id" TEXT, "customer_unique_id" TEXT, "customer_zip_code_prefix" INTEGER, "customer_city" TEXT, "customer_state" TEX
geolocation		CREATE TABLE "geolocation" ("geolocation_zip_code_prefix" INTEGER, "geolocation_lat" REAL, "geolocation_lng" REAL, "geolocation_city" TEXT, "geolocation_state"
order_items		CREATE TABLE "order_items" ("order_id" TEXT, "order_item_id" INTEGER, "product_id" TEXT, "seller_id" TEXT, "shipping_limit_date" TEXT, "price" REAL, "freight_
order_payments		CREATE TABLE "order_payments" ("order_id" TEXT, "payment_sequential" INTEGER, "payment_type" TEXT, "payment_installments" INTEGER, "payment_value" REA
order_reviews		CREATE TABLE "order_reviews" ("review_id" TEXT, "order_id" TEXT, "review_score" INTEGER, "review_comment_title" TEXT, "review_comment_message" TEXT, "
orders		CREATE TABLE "orders" ("order_id" TEXT, "customer_id" TEXT, "order_status" TEXT, "order_purchase_timestamp" TEXT, "order_approved_at" TEXT, "order_delive
product_category_name_translation		CREATE TABLE "product_category_name_translation" ("product_category_name" TEXT, "product_category_name_english" TEXT)
products		CREATE TABLE "products" ("product_id" TEXT, "product_category_name" TEXT, "product_name_lenght" REAL, "product_description_lenght" REAL, "product_photos_
sellers		CREATE TABLE "sellers" ("seller_id" TEXT, "seller_zip_code_prefix" INTEGER, "seller_city" TEXT, "seller_state" TEXT)
Indices (3)		
idx_order_items_order		CREATE INDEX idx_order_items_order ON order_items(order_id)
idx_order_payments_order		CREATE INDEX idx_order_payments_order ON order_payments(order_id)
idx_orders_customer		CREATE INDEX idx_orders_customer ON orders(customer_id)
Views (1)		
order_totals		CREATE VIEW order_totals AS SELECT order_id, SUM(price + freight_value) AS total_amount FROM order_items GROUP BY order_id
Triggers (0)		